Ecole supérieure d'Informatique de Bruxelles

Projet Erasmus

Rapport : Développement d'une application interactive pour le paramétrage d'une simulation numérique de propagation ultrasonore

Auteur :
Thomas Geller

Maitre de stage :
Ayache Bouakaz

Année académique 2023-2024

Remerciements

Je souhaite exprimer ma gratitude envers Dr. Ayache Bouakaz, Mr. Mohammed Hadjili et Mr. Frederic Servais, qui ont rendu possible la réalisation de ce stage enrichissant. Un grand merci également à Damien Fouan, Ingénieur chercheur, qui a apporté une aide précieuse tout au long du stage, contribuant à l'élaboration de cette application et fournissant des solutions tout en expliquant la physique des ondes ultrasonores.

Mes remerciements vont également à Guillaume, Arthur et Corentin, doctorants, qui ont généreusement pris le temps de m'expliquer certains concepts lorsque j'en avais besoin.

Je tiens à remercier tous les autres membres du laboratoire avec lesquels j'ai passé un excellent stage.

Table des matières

Abstract

Au sein de l'*Inserm*, j'ai été chargé de développer une application visant à simplifier la manipulation de données d'imagerie médicale. L'objectif était de permettre la réalisation de simulations ultra sonores au sein d'un environnement hétérogène de manière interactive, en utilisant une interface graphique. Mon projet s'appuie sur le script Matlab élaboré par un ancien chercheur de l'*Inserm*, Dr. Dapeng Li. Ce dernier a mis au point un algorithme de simulation numérique utilisé dans le cadre du traitement de la dépression visant à traiter la dépression majeure par le biais de stimulations ultra sonores transcrânienne focales et répétées. Grâce à cette application, les chercheurs pourront simuler la propagation d'ondes ultra sonores sans passer par de la compréhension du code et de la syntaxe Matlab.

At Inserm, I was tasked with developing an application to simplify the handling of medical imaging data. The aim was to enable ultrasound simulations ultrasound simulations in a heterogeneous environment interactively, using a graphical interface. My project is based on the Matlab script developed by a former *Inserm* researcher, Dr. Dapeng Li. He developed a numerical simulation algorithm used in the treatment of depression, with the aim of treating major depression by means of repeated focal transcranial ultrasound stimulation stimulation. Thanks to this application, researchers will be able to simulate ultrasound wave propagation without having to understand Matlab code and syntax.

1 Introduction

Dans le cadre de mes études de *Bachelier en informatique, orientation développement d'application*, j'ai réalisé un stage durant le dernier quadrimestre de mon parcours. Celui-ci s'est étalé sur 14 semaines, du 14 septembre 2023 au 22 décembre 2023. Ce stage de fin d'études s'est déroulé dans l'institut *Inserm*, dans le laboratoire de recherche 1253 Imagerie et cerveau. On m'a confié la mission du développement d'une application interactive pour le paramétrage d'une simulation numérique de propagation ultrasonore.

Plusieurs stages m'ont été proposés et mon choix s'est porté vers un domaine qui m'est tout aussi inconnu que passionnant : la recherche scientifique. J'ai décidé d'accepter ce stage car j'y ai trouvé une opportunité unique de comprendre et voir ce que représente ce secteur, mais aussi de ce qu'il peut m'apporter. En effet, j'allais travailler avec des outils qui m'étaient méconnus et dans une ambiance de travail hors du commun. De plus, j'ai également eu l'occasion de travailler de manière autonome. Cette approche a renforcé ma capacité à acquérir de nouvelles compétences de manière indépendante, témoignant ainsi de ma flexibilité et de ma motivation intrinsèque pour le développement professionnel.

J'ai élaboré une application Matlab en partant d'un script rédigé par un ancien post-doctorant du laboratoire. Mon objectif principal a été d'automatiser ce script afin d'en créer une application totalement accessible à toute personne, même dépourvue de compétences en informatique.

J'ai été encadré par le Dr. *Ayache Bouakaz*, Directeur de recherche à l'*Inserm*, ainsi que par M. *Mohammed Hadjili*, professeur à la *HE2B*. Je tiens également à souligner la contribution significative de M. Damien Fouan, Ingénieur de recherche, qui m'a apporté un soutien précieux tout au long de cette période.

2 Présentation de l'environnement

J'ai travaillé au sein du laboratoire *Inserm*, notamment dans l'unité de recherche médicale 1253, qui se concentre sur les ondes ultra sonores dans le domaine médical. Cette unité est située à la faculté de médecine de l'université de Tours.

2.1 Présentation de l'équipe

L'équipe se compose de professionnels aux compétences diverses, allant des techniciens aux ingénieurs, chercheurs et doctorants. Leur objectif commun est de contribuer à l'avancement de la recherche dans le domaine médical.

Bien que j'aie travaillé la plupart du temps de manière autonome sur mon projet, j'ai tout de même bénéficié de l'aide de la majorité de l'équipe.

2.2 Conditions essentielles

Les outils nécessaires pour réaliser le projet sont les suivants :

- Des images tomographiques (CT) de cerveau de singes.
- Des images par résonance magnétique (IRM) de cerveau de singes.
- Le logiciel Matlab de MathWorks, la version 2022a.
- L'outils Matlab app Designer, fournit avec le logiciel Matlab.
- La Toolbox k-Wave [1]

Le script composé de deux grandes parties importantes et entièrement écrit en MATLAB :

- Configuration : L'ensemble de tous les paramètres nécessaires pour faire une simulation.
- Simulation : La propagation d'une onde ultra sonore grâce à tous les paramètres récupérés en amont ainsi que le milieu fourni par les images médicales.

1. http ://www.k-wave.org/documentation.php

2.3 Les outils utilisés

- Matlab représente un langage de programmation et un environnement de développement numérique fréquemment adopté dans les secteurs de l'ingénierie, de la recherche scientifique et de l'analyse de données. Il est réputé pour son aptitude à réaliser des calculs numériques, à générer des graphiques, ainsi qu'à élaborer des applications personnalisées. De manière étendue, Matlab est employé dans divers domaines tels que le traitement du signal, la simulation, la modélisation, l'apprentissage automatique et l'analyse de données.

- Logiciel Matlab est un logiciel polyvalent utilisé dans l'ingénierie, les sciences et la recherche pour résoudre des problèmes mathématiques et techniques. Matlab est un environnement de développement numérique mais également un langage.

- Matlab app designer est un environnement de développement graphique inclus dans Matlab, qui permet aux utilisateurs de créer des applications graphiques conviviales sans avoir besoin de coder manuellement toute l'interface utilisateur. Cet outil est dépendant à la suite Matlab.

- K-wave est une boîte à outils Matlab spécialement conçue pour la simulation et la reconstruction de signaux photo acoustiques et ultra sonores. Elle est utilisée principalement dans le domaine de l'imagerie médicale et de la recherche en échographie. Les utilisateurs de k-Wave peuvent modéliser la propagation des ondes acoustiques dans des tissus biologiques, simuler l'interaction des ondes ultra sonores avec des structures anatomiques, et reconstruire des images à partir des données ultra sonores ou photo-acoustiques. Cet outil est précieux pour la recherche médicale, l'imagerie médicale non invasive et d'autres applications liées aux ondes acoustiques.

2.3.1 Pourquoi Matlab ?

Pourquoi opter pour MATLAB plutôt que d'autres langages de programmation plus simples et plus répandus tels que Python ou C++ ? Bien que le projet aurait pu être réalisé dans l'un de ces langages, le temps nécessaire pour développer une application avec un langage différent aurait été considérablement plus long. Puisque le script est rédigé en MATLAB, il est logique de continuer à partir du code MATLAB existant. De plus, l'utilisation d'App Designer facilite l'intégration directe du code MATLAB dans une application. Il aurait été extrêmement difficile de réécrire l'intégralité du code de simulation MATLAB dans un autre langage.

3 Présentation du travail

L'objectif est de créer une application automatisant un script rédigé par un ancien post-doctorant de l'équipe, le Dr. Li Dapeng. Mon projet de simulation est lié à une étude de recherche portant sur le traitement de la dépression par des ondes ultra sonores. Dans le cadre de cette recherche, l'objectif est d'exposer une partie du cerveau à des ondes ultra sonores pour des applications spécifiques. Cependant, connaître précisément tous les paramètres nécessaires au bon fonctionnement de ce processus est complexe. C'est pourquoi nous utilisons des simulations pour refléter de manière exacte nos intentions. Ainsi, nous avons la possibilité d'ajuster tous les paramètres nécessaires pour garantir une expérience de qualité.

3.1 Pré-requis à la simulation

Dans le but de réaliser la simulation de la propagation d'une onde ultra sonore, nous avons besoin d'acquérir deux types de volumes :

- Un volume IRM
- Un volume CT

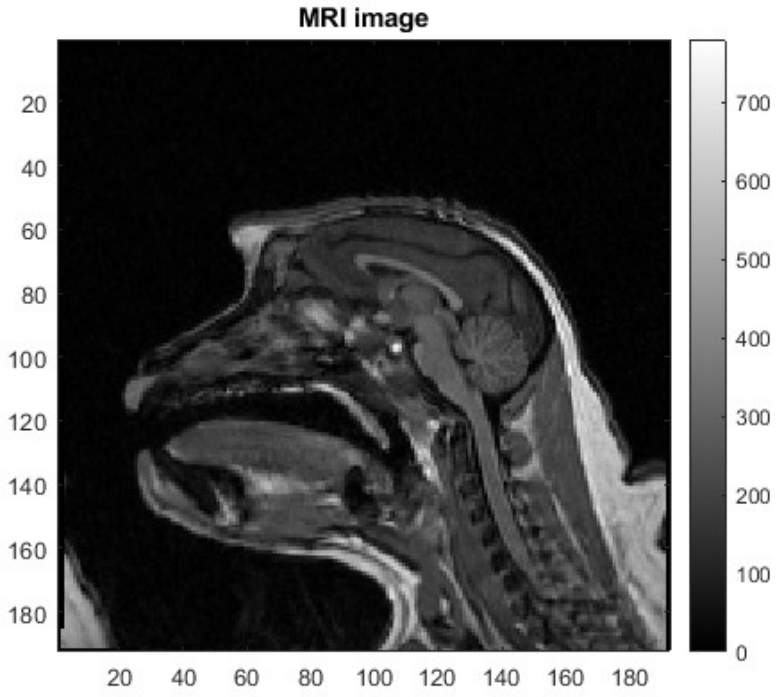

Figure 1 – Coupe d'un volume IRM d'une tête de singe

Le volume IRM, grâce à sa technologie, offre une excellente visualisation des tissus mous, mais il présente une représentation moins précise des tissus durs tels que l'os. Grâce à l'image

IRM nous sommes en mesure de sélectionner précisément la cible à savoir le sommet du cortex cingulaire antérieur (ACC), une zone responsable de la dépression.

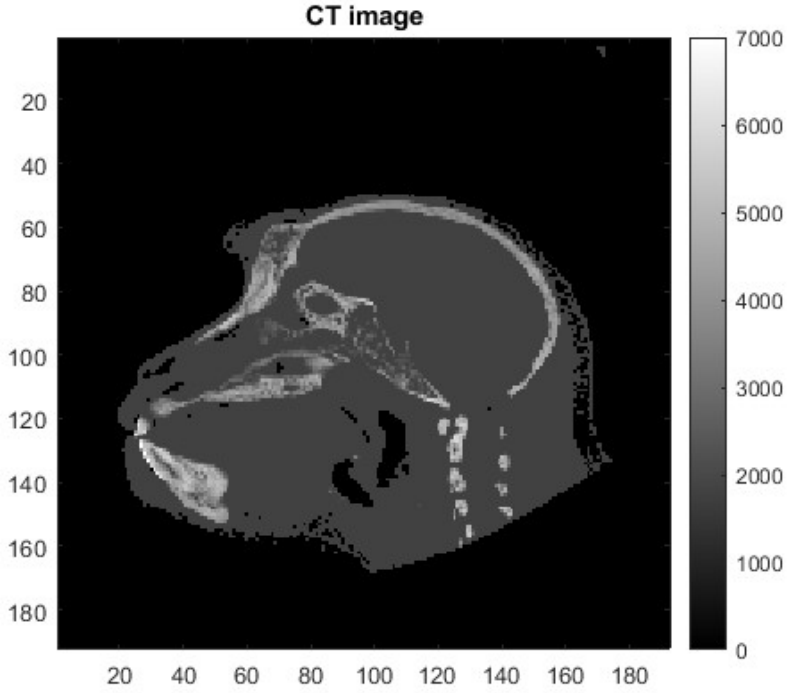

FIGURE 2 – Coupe CT d'un volume d'une tête de singe

Le volume CT nous permet de créer l'environnement de la simulation mais aussi de distinguer l'os des autres tissus. Ici, les tâches blanches sont représentées comme de l'os.

Les images présentent ci dessus sont des coupes d'un volume d'image médicale. Ces volumes sont donc des matrices en trois dimensions composées de points représentant la valeur de la densité d'un point dans un volume. Chaque volume rend compte d'une réalité physique. L'IRM représente le temps de relaxation d'un tissu. Le CT permet de remonter aux paramètres mécaniques (c'est-à-dire acoustiques) tels que la vitesse ultra sonore d'un tissus, la densité et l'absorption visqueuse.

Ces deux volumes ont subi un pré-traitement. Ils ont été pré-alignés de sorte qu'ils puissent se superposer parfaitement. Ils ont donc la même taille et la même résolution.

3.2 Configuration de la simulation

L'application que j'ai développée se présente comme ceci.

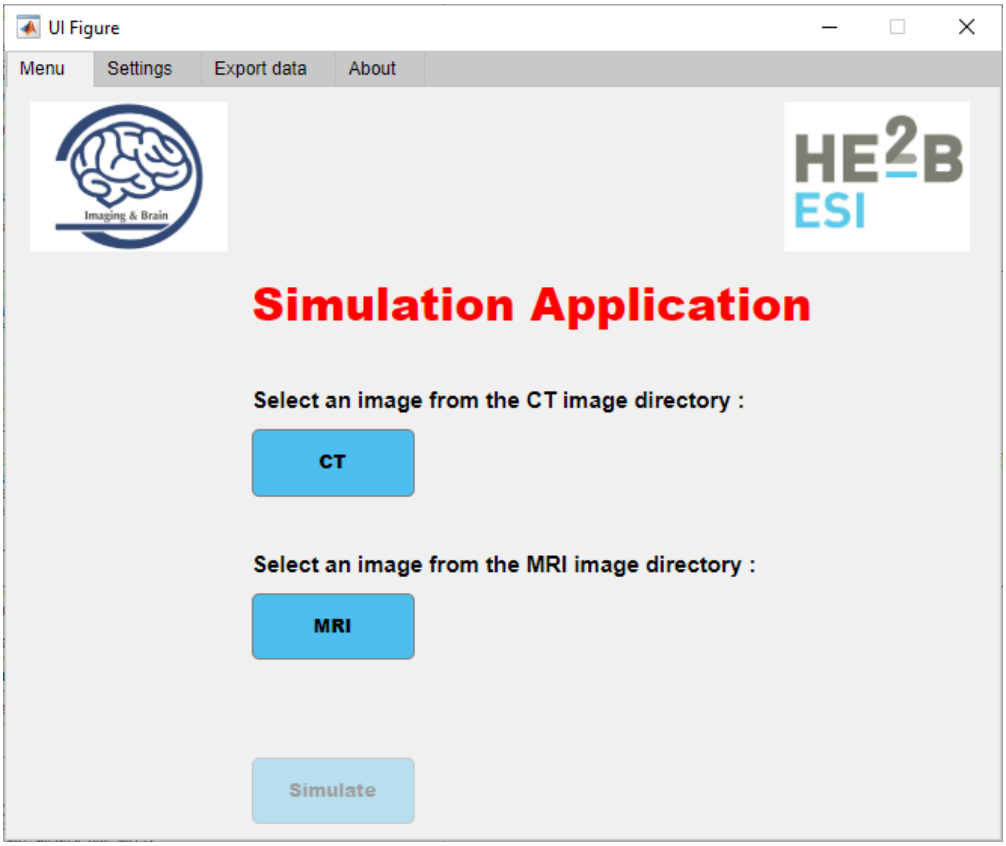

FIGURE 3 – Menu principal

Dans ce menu principal nous retrouvons deux boutons. Un bouton nommé "CT" permettra de récupérer le volume CT depuis le stockage local de l'ordinateur, et un autre bouton nommé "IRM" nous permettra de récupérer le volume IRM. Une fois les deux volumes sélectionnés le bouton simulate est activé et le début du processus peut commencer. Ces volumes seront utilisé pour la suite de la simulation.

Pour réaliser la simulation, une série de paramètres est importante et nécessaire à son bon fonctionnement. Il existe 3 catégories.

- Les paramètres de la simulation incluent tout ce qui est en rapport avec la physique des ultrasons : la fréquence d'ondes, la longueur d'ondes, le nombre de points par longueur d'onde. Ces paramètres ont une influence directe sur la résolution des volumes, mais également sur le temps de calcul.
- Les paramètres du transducteur : le diamètre, le rayon de courbure, la distance focale (distance entre la cible et le transducteur), la pression acoustique de surface.

- les paramètres des tissus et de l'eau : paramètres qui définissent les différents seuils, la densité, et la vitesse des ondes.

Les paramètres sont modifiables dans un onglet de l'application. Ils peuvent être mis à jour tant que le format est correct (uniquement des chiffres). En cas d'erreur, un message s'affiche, présentant les anciennes données.

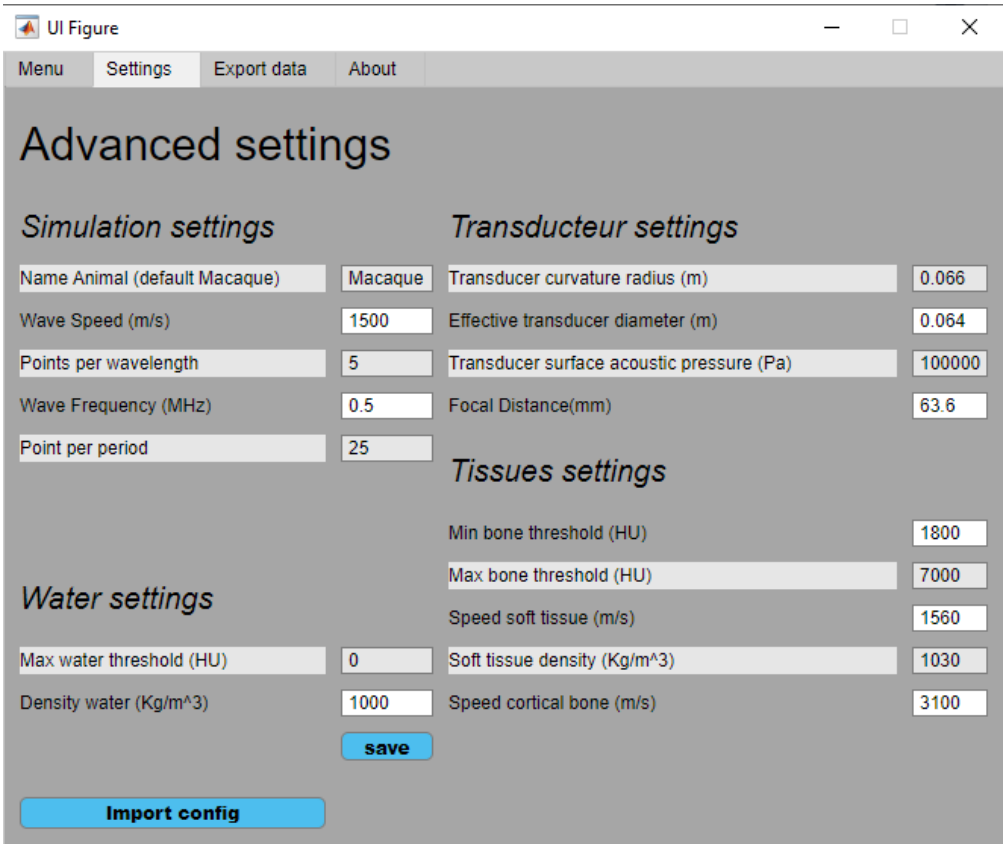

FIGURE 4 – Paramètres de base

D'autres paramètres nécessitent l'utilisation d'une interface graphique pour leur initialisation. C'est le cas notamment :

- Du choix de la cible à exposer.
- De la position du transducteur.
- Du rognage de la matrice final.

Il est impossible de déterminer ces paramètres autrement, car ils sont directement dépendant des images fournies par l'utilisateur. L'utilisation de matrices de tailles différentes (en raison de la fréquence) ou d'images de cerveaux de singes différents nécessite de passer par une interface graphique.

3.2.1 Le choix de la cible

Dans cette situation, l'utilisateur peut parcourir la matrice tridimensionnelle de l'IRM à l'aide de la molette pour sélectionner le point désiré. Le point désiré, dans le cadre de la simulation, est le sommet de l'ACC, une zone impliquée dans la dépression. Cette cible sera exposée par l'onde ultrasonore.

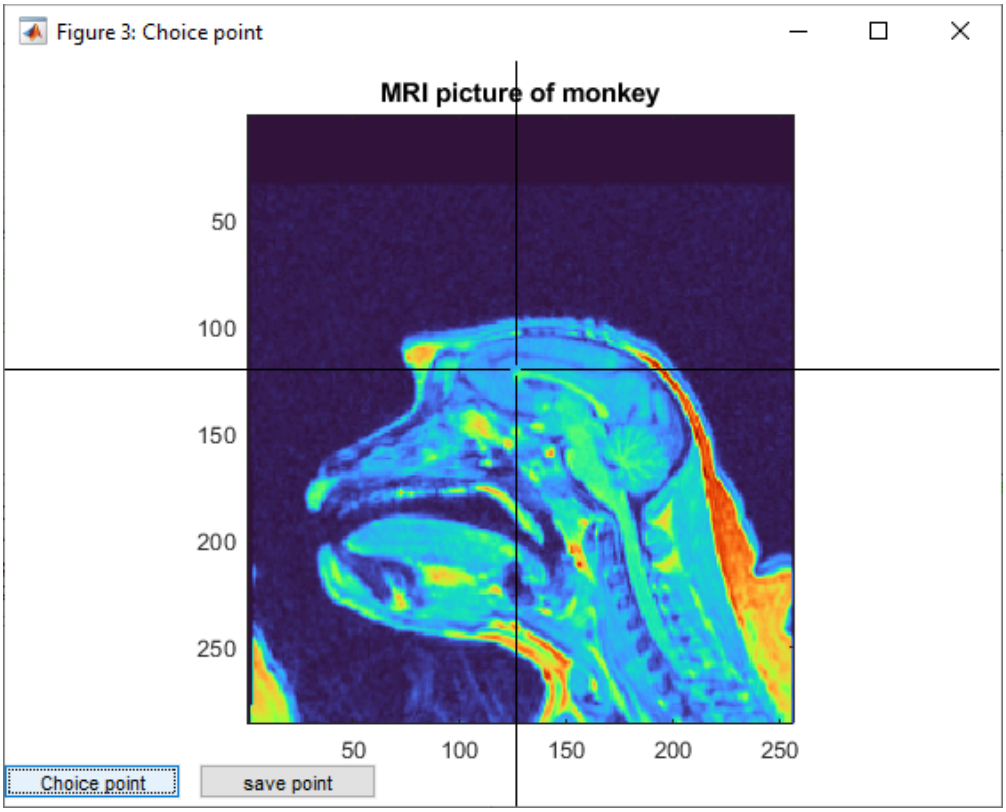

FIGURE 5 – plan d'un Volume Irm

Une fois la cible choisie, une coordonnées en X ;Y ;Z sera renvoyé (un point dans un plan).

3.2.2 La position du transducteur

Grâce à la distance focale du transducteur et de la cible, nous sommes en mesure de déterminer toutes les positions qu'un transducteur peut occuper. Dans notre contexte, nous avons la possibilité de choisir la position du transducteur en fonction de l'angle entre la trajectoire du tir et l'os. Idéalement, cet angle sera de 90 degrés, ce qui créera moins d'effets de diffraction et favorisera l'angle de pénétration du son. Cependant, pour des raisons spécifiques, il peut être nécessaire d'opter pour un angle particulier. Dans cette configuration, l'image affichée par la figure sera une image CT, permettant ainsi une distinction nette de la structure osseuse. Le demi-cercle bleu représente l'ensemble des positions que le transducteur peut occuper. Le polygone en rouge est l'enveloppe convexe du crâne grâce à la fonction convexhull de Matlab. Ce sera donc l'angle entre le polygone et le tir.

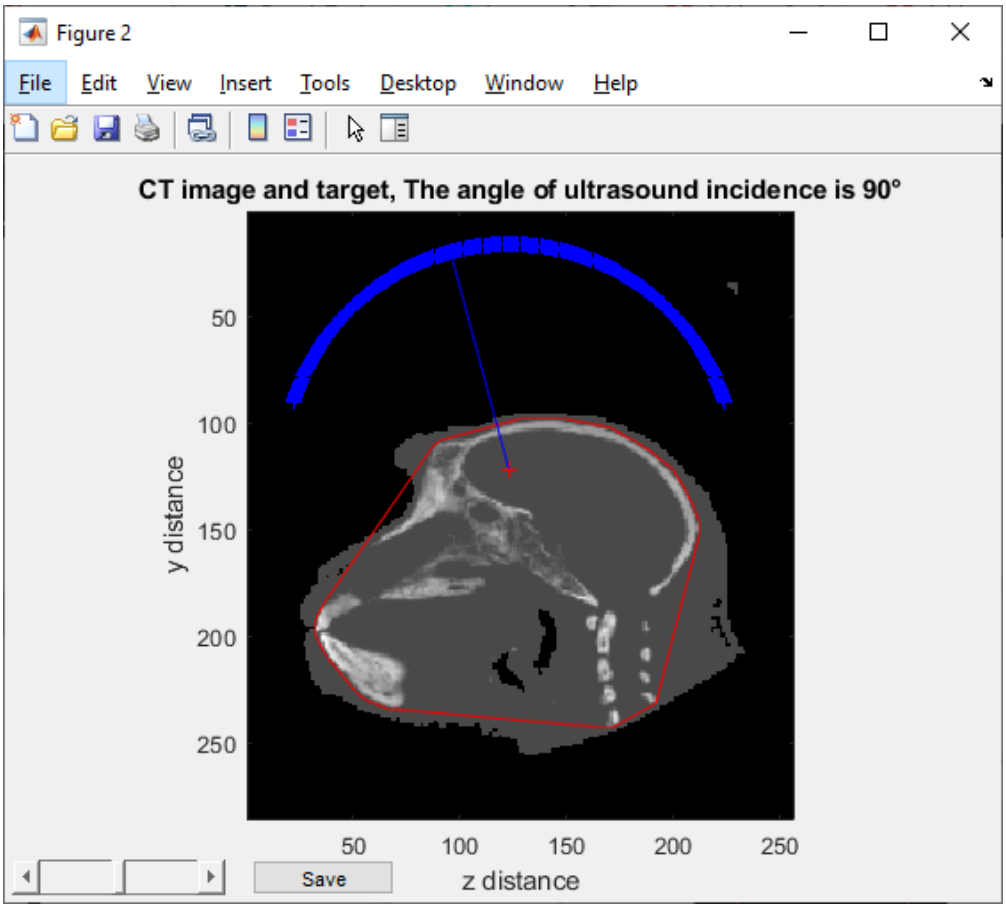

FIGURE 6 – Image CT.

3.2.3 Le troncage de la matrice final

Une fois l'angle sélectionné, la dernière étape interactive consiste au tronçonnage du volume. L'objectif de la simulation est d'observer le trajet des ondes ultrasonores à travers le cerveau. Il n'est donc pas nécessaire de maintenir une matrice de taille supérieure à celle du cerveau. À cet effet, une interface Matlab a été développée, permettant la visualisation du volume dans trois plans distincts (transversal, sagittal, frontal). Ces plans peuvent être délimités grâce à deux curseurs sur chacun d'eux. L'affichage de ces plans est directement déterminé par le point de la cible, facilitant également la distinction du transducteur sur l'image. Pour obtenir le transducteur, une matrice avec sa position est créée et additionnée à la matrice IRM.

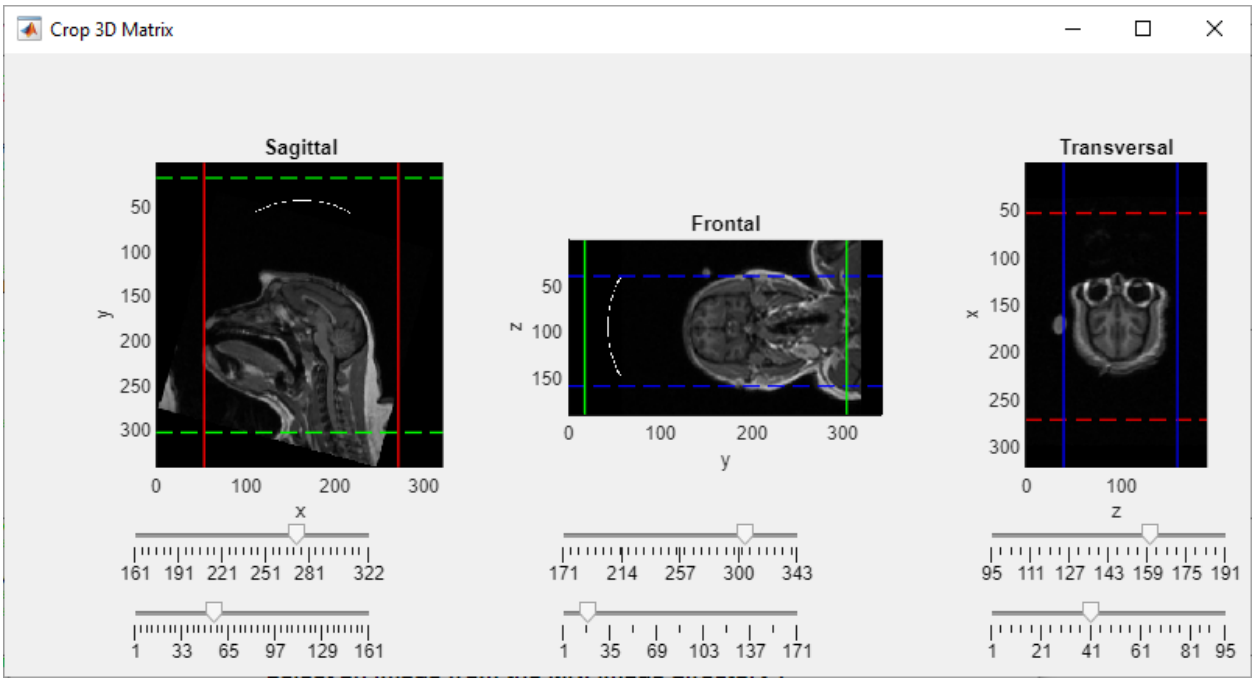

FIGURE 7 – Rognage de matrice

3.3 Simulation

Une fois l'environnement configuré, la simulation commence. Cette étape est réalisée à l'aide d'une fonction provenant de la boîte à outils MATLAB appelée K-Wave. K-Wave est une bibliothèque spécialement conçue pour la simulation d'ondes ultrasonores. [2]

En l'absence de tronçonnage, l'exécution d'une simulation classique peut s'étendre jusqu'à 1 heure et 10 minutes. Il devient donc impératif d'optimiser les calculs en restreignant la propagation des ondes aux zones d'intérêt. À la conclusion de la propagation des ondes, la fonction renverra une structure de données comprenant deux matrices :

- Une matrice en 4 dimensions, représentant la variation acoustique au fil du temps d'un voxel. (paramètre désactivé)
- Une matrice en 3 dimensions qui contient la pression maximale enregistrée pour chaque point de la grille tridimensionnelle simulée. (fig.6)

2. Resultat fig. 9 Annexes, http ://www.k-wave.org/

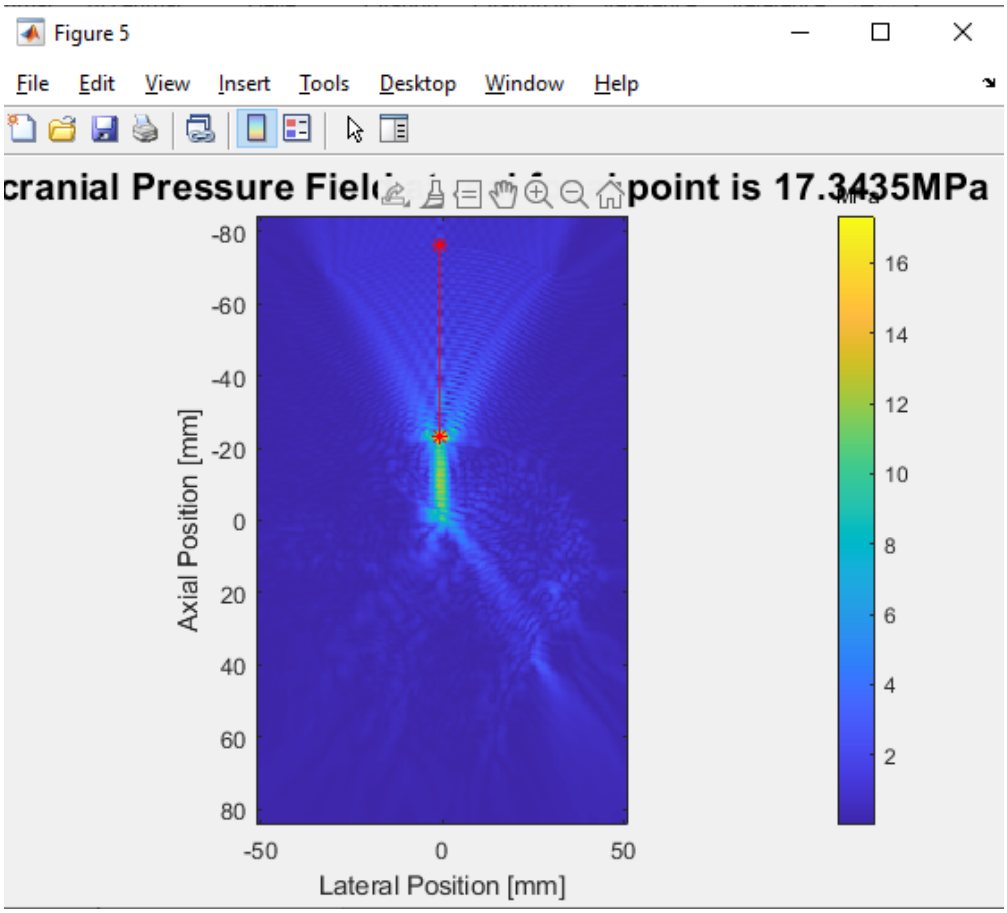

FIGURE 8 – Champ de pression transcrânien

Cette étape de simulation joue un rôle crucial dans la collecte de données et la création d'images utiles pour l'analyse ultérieure. Après troncage et avec une fréquence d'ondes de 0.5 MHz la simulation prend 50 minutes. Il est à noter que plus la fréquence est élevée ou plus le nombre de points par période est important, plus la durée de la simulation s'allongera.

3.4 Limitations du script initial

Le script conçu par L. Dapeng est fonctionnel, cependant, son utilisation présente plusieurs inconvénients :

- Bloc de code monolithique de 700 lignes, il est particulièrement ardu de s'immerger dans un code aussi étendu, ce qui le rend assez rébarbatif à utiliser.
- Redondance de définitions de variables et manque d'abstractions. Les variables ne sont pas isolées dans le code, entraînant des redéfinitions multiples, ce qui crée un inconvénient supplémentaire en raison du manque de généralisation. Voir ci dessous.

```
%Ligne 40
%% 获得 CT 和 MRI 图像的分辨率
MRI_space=[];%MRI 的三维分辨率（mm）
MRI_space(1:2)=info2.PixelSpacing;
MRI_space(3)=info2.SliceThickness;

CT_space=MRI_space;%CT 的三维分辨率（mm）,CT 和 MRI 已配准，具有相同的空间分辨率

%% 设置颅骨 CT 数据的上下限及分层（水，软组织和猴子颅骨），不同数据阈值不同
CT_th1=0;% 设置水的 CT 阈值为 0 HU，低于 0 的全部置为水
CT_th2=1800;% 设置颅骨的 CT 阈值下限为 1800HU，0-1800HU 的认为是软组织
CT_th3=7000;% 设置颅骨的 CT 阈值上限为 7000HU,1800-7000HU 的认为是颅骨，大于 7000 的置为 2400HU
CT(CT<CT_th1)=CT_th1;
CT(CT<CT_th2&CT>CT_th1)=1800;
CT(CT>CT_th3)=CT_th3;

%% 选择同一位置（水平面的中心）MRI 和 CT 图像查看配准效果（可选的）
axial_plane_center1 = round(size(CT,3)/2);...
...
% Ligne 99
%%% 注意：Vq = interp3(V,Xq,Yq,Zq) 假定一个默认的样本点网格。默认网格点覆盖区域 X=1:n、Y=1:m 和
%Z=1:p，其中 [m,n,p] = size(V)
[xq,yq,zq] = meshgrid(1:kwave_step/CT_space(1):m,1:kwave_step/CT_space(2):
n,1:kwave_step/CT_space(3):k);
CT_inter = interp3(x,y,z,CT,xq,yq,zq);% 插值后的 CT 数据
MRI_inter = interp3(x,y,z,MRI,xq,yq,zq);% 插值后的 MRI 数据

%% 设置颅骨 CT 数据的上下限及分层（水，软组织和猴子颅骨），插值后需要再次分层
CT_th1=0;% 设置水的 CT 阈值为 0 HU，低于 0 的全部置为水
CT_th2=1800;% 设置颅骨的 CT 阈值下限为 1800HU，0-1800HU 的认为是软组织
CT_th3=7000;% 设置颅骨的 CT 阈值上限为 7000HU,1800-7000HU 的认为是颅骨，大于 7000 的置为 2400HU
CT_inter(CT_inter<CT_th1)=CT_th1;
CT_inter(CT_inter<CT_th2&CT_inter>CT_th1)=1800;
CT_inter(CT_inter>CT_th3)=CT_th3;
...
% Ligne 456


%% 设置颅骨 CT 数据的上下限及分层（水，软组织和颅骨 ），插值后需要再次分层
CT_th1=0;% 设置水的 CT 阈值为 0 HU，低于 0 的全部置为水
CT_th2=1800;% 设置颅骨的 CT 阈值下限为 150HU，0-150HU 的认为是软组织
CT_th3=7000;% 设置颅骨的 CT 阈值上限为 2400HU,150-2400HU 的认为是颅骨，大于 2400 的置为 2400HU
CT_final(CT_final<CT_th1)=CT_th1;
CT_final(CT_final<CT_th2&CT_final>CT_th1)=1800;
CT_final(CT_final>CT_th3)=CT_th3;

```

```
%% 定义声学常数
HU_max=max(CT_final(:));
```

- Manque de modularité et de généralisation. Dans le code, certaines variables sont définies de manière statique, alors qu'elles devraient dépendre des volumes entrés. Cela rend le script inadaptable à l'utilisation de volumes différents, car les paramètres ne s'ajustent pas automatiquement en fonction des changements de volume. Par exemple, la cible est actuellement fixée dans le code, mais en utilisant des volumes différents, cette valeur devrait être dynamiquement ajustée en conséquence. Exemple ci dessous.

```
% 找到靶点所在的平面，确定是矢状面，横截面还是冠状面
target_plane_position=98;% 找到 ACC 所在的平面 %%ici

%% 画出靶点所在的平面
target_plane=squeeze(MRI_inter(:,:,target_plane_position));
figure
imagesc(target_plane)
axis equal
axis tight
colormap gray;

%% 确定靶点的其余坐标
ACC_target_xy=[120,120];% 靶点 %% ici
figure
imagesc(target_plane)
axis equal
axis tight
colormap gray;
hold on
plot((ACC_target_xy(1)),(ACC_target_xy(2)),'r+')


%% 计算颅骨的轮廓作为其切线方向
target_plane_CT=squeeze(CT_inter(:,:,target_plane_position));
```

- L'utilisation de ce programme peut être complexe, voire impossible, pour des utilisateurs qui ne sont pas familiers avec le langage MATLAB. De plus, la présence de commentaires initialement rédigés en mandarin peut rendre la traduction par Deepl parfois peu fiable, ce qui peut entraîner des difficultés de compréhension supplémentaires.

4 Explication technique du projet

4.1 Modèle-Vue-Contrôleur (MVC)

L'application a été conçue en suivant le modèle de conception Modèle-Vue-Contrôleur (MVC), qui permet de séparer les préoccupations en termes de gestion des données, d'interface utilisateur et de contrôle. Ce modèle favorise la maintenabilité, la réutilisabilité et la scalabilité du code. Voici un aperçu du fonctionnement de l'application avec MVC.

4.1.1 Modèle

Le Modèle représente la structure de données sous-jacente et les opérations associées. Dans notre application, le Modèle est principalement géré par la classe `Simulation`. Cette classe contient les données de la simulation, telles que les propriétés relatives aux fichiers CT et IRM, les paramètres de simulation, ainsi que les résultats. Elle est également responsable de l'ouverture de fichiers Dicom, de la configuration de la simulation et de l'exécution de cette dernière.

4.1.2 Vue

La Vue est responsable de l'interface utilisateur (UI) et de l'affichage des données. Dans notre application, la classe `View` gère les interactions avec l'utilisateur. Elle fournit des fonctions pour afficher les informations, désactiver ou activer des boutons, et récupérer des données de configuration depuis l'interface utilisateur. La classe `View` est associée à l'interface utilisateur de l'application.

4.1.3 Contrôleur

Le Contrôleur agit comme un intermédiaire entre le Modèle et la Vue. Dans notre application, la classe `Controller` joue ce rôle. Le Contrôleur gère la logique métier, relie les actions de l'utilisateur aux opérations du Modèle et met à jour l'interface utilisateur en conséquence. Il coordonne les actions, telles que l'ouverture de fichiers, la configuration de la simulation et le lancement de la simulation elle-même.
L'utilisation du modèle MVC permet une séparation claire des responsabilités dans l'application, facilitant ainsi la maintenance et l'extension du code.

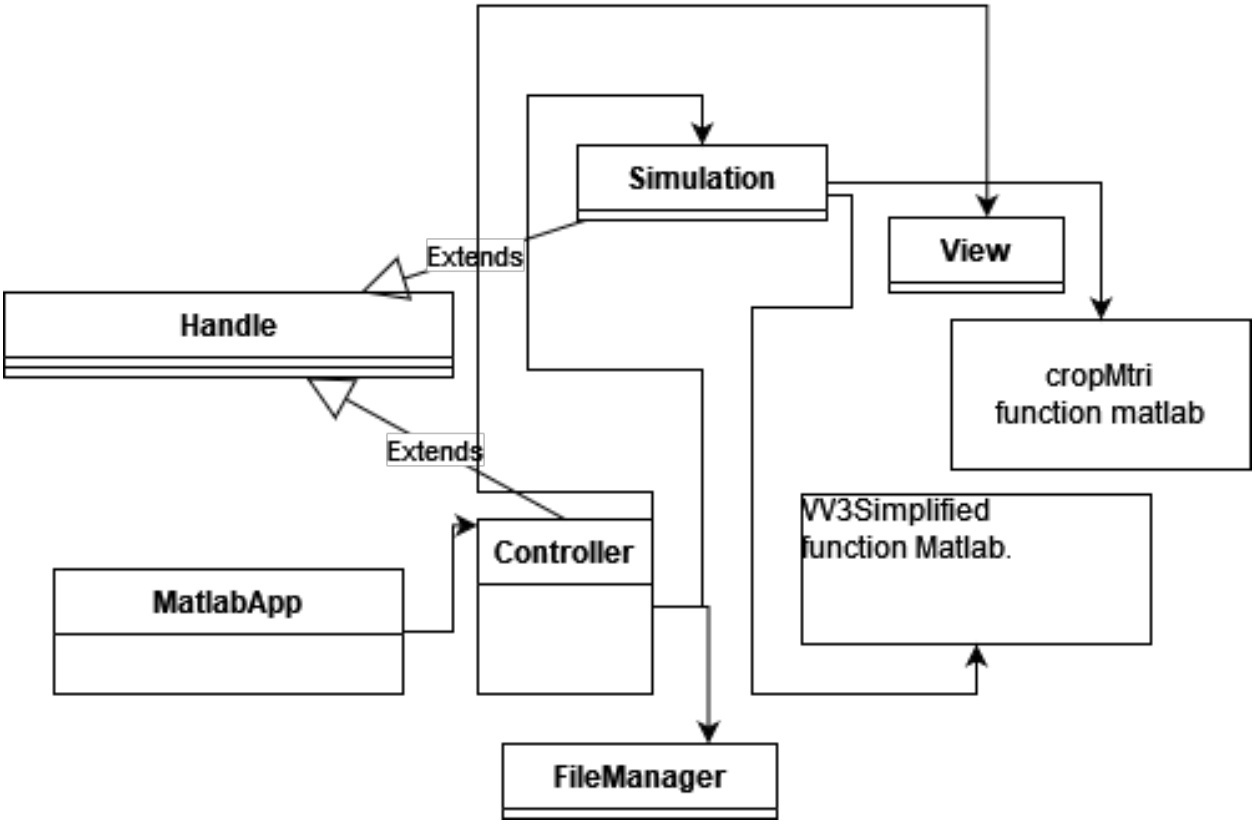

FIGURE 9 – Diagramme de la représentation MVC de mon application fait sur Draw.io

4.2 Modèle

En premier lieu, la classe Simulation englobe le script initial modifié. Pour préserver la stabilité des données et du script, aucune modification profonde n'a été apportée. Compte tenu de la complexité du code source, particulièrement pour quelqu'un sans expertise dans le domaine des ondes acoustiques, sa compréhension intégrale reste ardue. Malgré cela, j'ai investi du temps pour m'y intéresser et adapter le code afin qu'il soit compatible avec mon application, tout en le rendant plus cohérent et modulaire.

Il convient de noter que L. Dapeng a rédigé un code exécutable pour une configuration spécifique. Tous les paramètres sont définis de manière à obtenir un résultat connu. La classe Simulation prend en charge la manipulation complète des données et hérite d'une classe de gestion d'objet (Handle). Cela implique que l'instance manipulée par la classe sera modifiée en profondeur. La classe Simulation fournit au Controller les données nécessaires.

4.3 Simulation

La classe `Simulation` est responsable de la gestion des données de simulation, de l'ouverture de fichiers CT et IRM, ainsi que de la configuration et de l'exécution de la simulation. Elle hérite également de la classe `handle`. La classe `Simulation` contient les propriétés et les méthodes suivantes :

- `properties`
  - `animalName` : Le nom de l'animal utilisé pour la simulation (par défaut, "Macaque").
  - `filenameCT` et `filepath1CT` : Le nom et le chemin du fichier CT sélectionné.
  - `info1` : Les informations DICOM du fichier CT.
  - `V1`, `spatial1`, `dim1`, `D1`, et `CT` : Les données CT et les informations associées.
  - `filenameIMR` et `filepathIMR` : Le nom et le chemin du fichier IRM sélectionné.
  - `info2` : Les informations DICOM du fichier IRM.
  - `V2`, `spatial2`, `dim2`, `D2`, et `MRI` : Les données IRM et les informations associées.
  - `wave_speed` : La vitesse des ondes dans l'eau (par défaut, 1500 m/s).
  - `CT_th1`, `CT_th2`, et `CT_th3` : Les Seuil HU CT pour l'eau, les tissus mous et le crâne (par défaut, 0, 1800, 7000 HU).
  - `wave_frequency` : La fréquence des ondes ultrasonores (par défaut, 0.5 MHz).
  - `ppw` : Le nombre de longueurs d'onde par point de grille (par défaut, 5).
  - `angel` : L'angle de la simulation (non initialisé).
  - `dwater`, `stissue`, `dtissue`, et `sbone` : Les paramètres acoustiques de l'eau, des tissus mous, des tissus mous denses et du crâne (par défaut, 1000 kg/mş, 1560 m/s, 1030 kg/mş, 3100 m/s).
  - `source_roc` : Le rayon de courbure du transducteur (par défaut, 66 mm).
  - `source_diameter` : Le diamètre efficace du transducteur (par défaut, 64 mm).
  - `source_amp` : La pression acoustique de surface du transducteur (par défaut, 1e6 Pa).
  - `PPP` : Le nombre de points par période (par défaut, 25).
  - `matriceT` : La matrice de données de simulation (non initialisée).
- `methods`
  - `Simulation()` : Le constructeur de la classe, qui initialise les propriétés de la simulation.
  - `openCTFile()` : Cette méthode permet d'ouvrir un fichier CT, de lire ses informations DICOM et de stocker les données associées.

- `openIRMFile()` : Cette méthode permet d'ouvrir un fichier IRM, de lire ses informations DICOM et de stocker les données associées.
- `makeSimulation()` : Cette méthode démarre la simulation en appelant `simulationWorker()`.
- `simulationWorker()` : Cette méthode contient le code de la simulation, y compris la configuration avancée.
- `exportData()` : Cette méthode exporte les données de simulation, en excluant les propriétés spécifiées dans `attributNotSave`.
- `loadConfig()` : Cette méthode permet de charger une configuration depuis un fichier JSON et de l'appliquer à la simulation.
- `loadData(data)` : Cette méthode charge les données de configuration dans la simulation à partir de la structure `data`.

4.4 Controlleur

Le `Controller` s'occupe du transfert des données du modèle vers la vue. Il hérite également de `Handle`.

- `properties`
  - `simulation` : Cette propriété représente un objet de la classe "Simulation" qui gère les calculs et la simulation de l'application.
  - `view` : Cette propriété représente un objet de la classe "View" qui gère l'interface utilisateur et l'affichage des résultats.
  - `fileManager` : Cette propriété représente un gestionnaire de fichiers qui est utilisé pour gérer la sauvegarde des données de simulation.
- `methods (Access = public)`
  - `setUpDataConfig()` : Cette méthode configure les paramètres par défaut de la vue en utilisant les paramètres de simulation.
  - `Controller(appParam)` : Le constructeur de la classe prend en paramètre un objet `appParam` et crée une instance de la classe "View" et de la classe "Simulation" pour gérer l'interface utilisateur et les simulations.
  - `makeSimulation(model)` : Cette méthode démarre la simulation, désactive les boutons pendant la simulation, affiche un message de début, effectue la simulation, réactive les boutons après la simulation, affiche un message de fin, puis exporte les données.
  - `openCTFile()` : Cette méthode ouvre un fichier CT, vérifie les fichiers requis, affiche le chemin du fichier CT et gère les exceptions.
  - `openIRMFile()` : Cette méthode ouvre un fichier IRM, vérifie les fichiers requis, affiche le chemin du fichier IRM et gère les exceptions.

- `saveData(obj, app)` : Cette méthode valide et enregistre les données saisies dans l'interface utilisateur, en vérifiant que les entrées sont valides.
- `updateStatus(obj, boolean)` : Cette méthode met à jour l'état de la vue en fonction d'un booléen.
- `exportData()` : Cette méthode exporte les données de simulation dans un fichier en utilisant un gestionnaire de fichiers.
- `loadConfig()` : Cette méthode charge une configuration précédemment enregistrée et configure les paramètres de la vue en conséquence.

4.5 Vue

La classe `View` est responsable de la gestion de l'interface utilisateur et de l'affichage des résultats de la simulation. Elle contient les propriétés et les méthodes suivantes :

- `properties`
  - `app` : Cette propriété représente un objet de l'application, qui est généralement passé en tant que paramètre au constructeur de la classe. Elle permet d'accéder aux composants de l'interface utilisateur de l'application.
- `methods`
  - `View(appParam)` : Le constructeur de la classe prend en paramètre un objet `appParam` et initialise la propriété `app`.
  - `defaultParam(obj, simulation)` : Cette méthode configure les paramètres par défaut de l'interface utilisateur en utilisant les paramètres de simulation passés en argument.
  - `showMessage(obj, exception)` : Cette méthode affiche un message d'erreur en utilisant les informations de l'exception passée en argument.
  - `disableButtonWhilesimulation(obj)` : Cette méthode désactive les boutons de l'interface utilisateur pendant la simulation.
  - `enableButtonWhilesimulation(obj)` : Cette méthode réactive les boutons de l'interface utilisateur après la simulation.
  - `displaypathOfFileCT(obj, simulation)` : Cette méthode affiche le chemin du fichier CT sélectionné dans l'interface utilisateur.
  - `displayPathOfFileIrm(obj, simulation)` : Cette méthode affiche le chemin du fichier IRM sélectionné dans l'interface utilisateur.
  - `checkRequirementFiles(obj, filepathCt, filepathIrm)` : Cette méthode vérifie si les fichiers CT et IRM requis ont été sélectionnés, et active le bouton de simulation si c'est le cas.

- `showMessageStartSimulation(obj)` : Cette méthode affiche un message de début de simulation.
- `showMessageEndSimulation(obj)` : Cette méthode affiche un message de fin de simulation.
- `showMessageUI(obj, string)` : Cette méthode affiche un message personnalisé dans l'interface utilisateur.
- `saveData(obj)` : Cette méthode affiche un message indiquant que les données ont été sauvegardées avec succès.
- `unsaveData(obj)` : Cette méthode affiche un message indiquant que les données n'ont pas été sauvegardées.
- `exportData(obj)` : Cette méthode affiche un message indiquant que les données de simulation ont été exportées avec succès.
- `showPathConfig(obj, string)` : Cette méthode affiche le chemin du fichier de configuration JSON chargé.

4.6 Fonctions supplémentaires

L'application offre également la possibilité d'exporter et d'importer des données à partir d'un fichier au format JSON. À cette fin, un singleton a été mis en place qui agit comme un unique point d'accès pour la gestion de l'écriture du fichier. Le contrôleur prend en charge cette instance du singleton pour effectuer ces opérations.

4.7 Fonctions utiles

Trois Fonctions ont été rajouté au code principal, VV3Simplified, cropMtri,foundFransductor.

- VV3simplified est une fonction qui a pour but de sélectionner un point dans un volume en 3 dimensions. Il est possible de naviguer à travers la matrice en faisant défiler la souris. Vous pouvez choisir un point une fois le plan trouvé en cliquant sur "Choice Target". Une fois la cible choisie, il est possible de l'enregistrer en appuyant sur la croix ou en appuyant sur "Save Target". La fonction prend un seul paramètre, la matrice.

- cropMtri est une fonction qui affiche les trois plans d'un volume médical (Sagittal, frontal, transversal). Des curseurs permettent de découper la matrice afin de la rendre plus petite dans les trois plans, réduisant ainsi la propagation des ultrasons et, par conséquent, le temps de calcul. Cette fonction prend en paramètre la matrice et la cible exposée. Les plans affichés dépendent directement de la cible.

- La fonction foundTransductor affiche divers éléments, dont une image CT, un polygone rouge représentant l'enveloppe de convolution du crâne, un demi-cercle bleu, un curseur, et une ligne bleue. Le demi-cercle bleu indique toutes les positions possibles du transducteur, et le curseur permet de régler la position en fonction de l'angle.

L'objectif principal de cette fonction est de déterminer l'angle entre la ligne bleue et l'enveloppe de convolution du crâne (polygone rouge). Les limites de cet angle sont fixées entre 60 et 120 degrés. Il est essentiel de noter que plus l'angle s'éloigne de la perpendiculaire (angle droit), moins la fiabilité de la mesure est élevée. Ceci s'explique par les propriétés de diffusion des ondes dans les tissus osseux denses, qui peuvent entraîner des phénomènes tels que la diffraction et influencer la précision des mesures ultrasonores. En ajustant la position du transducteur en fonction de l'angle optimal, la fonction cherche à obtenir des mesures fiables tout en tenant compte de ces caractéristiques.

5 Evolution du code initial

Voici différentes évolution du code initiale.

- Certaines variables sont définies dans l'instance de la classe Simulation.
- Utilisation de variables appropriés et non pas des immédiats + supression de code redondant.
- Utilisation de la fonction VV3Simplified(mri) pour extraire les coordonnées de la cible. En conséquence, modification des coordonnées initialement fixées en dur.
- Suppression de multiples figures affichées.
- Modification du code qui permet de trouver l'angle d'exposition entre le crane et le faisceau du tir.
- Ajout d'un code pour trouver l'angle.
- Ajout d'une fonction pour rogner la matrice cropMtri(MRI), renvoie les coordonées minimum et maximum de chaque plan afin d'en rogner plus tard la matrice.
- Ajout d'un graphique de la variation acoustique dans le temps.
- Création de l'environnement et somme de matrice pour obtenir l'IRM du crane avec le transducteur.

6 Principaux problèmes de développement

6.1 Threads

Le principal problème non résolu de cette application réside dans le bloquage de l'interface lors du lancement de la simulation. En effet, étant donné que les calculs nécessitent un temps considérable (environ 1 heure et 10 minutes), il aurait été judicieux d'exécuter la simulation dans un thread secondaire. Cependant, un obstacle se présente : l'affichage de la fenêtre ne s'effectue que sur le thread principal, tandis que l'application elle-même est exécutée sur ce même thread. Par conséquent, un dilemme se pose : soit exécuter la fonction de simulation sur un thread secondaire au risque de ne pas pouvoir visualiser la simulation ni vérifier son bon déroulement, soit exécuter la fonction dans le thread principal, entraînant ainsi le bloquage de l'application principale. Le choix a été de garder le bloquage de l'application.

6.2 Tests Unitaires

Pour cette application, aucun test n'a été developpé. Idéalement, il aurait été intéressant d'effectuer des tests unitaires sur le modèle, notamment sur les divers calculs de la configuration. Cependant, en raison de la complexité du code, j'ai rencontré des difficultés à élaborer des tests permettant de démontrer le bon fonctionnement du code. De plus, le code de simulation est dense, ce qui le rend trop ardu à tester. En ce qui concerne les tests de l'interface graphique, il n'existe pas de solutions concrètes étant donné que Matlab App Designer est une technologie relativement récente.

6.3 Architecture MVC

Lorsqu'on adopte l'architecture MVC, il est essentiel de maintenir une structure de code rigoureuse. De manière générale, j'ai veillé à respecter cette organisation. Cependant, en ce qui concerne la section dédiée à la simulation, j'ai été contraint de conserver le bloc de code tel qu'il était. En effet, les données et variables sont fortement entremêlées, et certaines subissent plusieurs modifications tout au long du code. Mon niveau de compréhension de la logique de la simulation ne me permet pas d'effectuer des modifications substantielles.

7 Compléments

Ayant fini le projet de simulation plutot, j'ai débuté un nouveau projet initié par Damien Fouan. Il convient de noter que les images CT et IRM chargées dans la simulation sont préalablement orientées. Les données brutes consistent en volumes de têtes de singes dans divers plans. Toutefois, il est impératif, pour les besoins de la simulation (notamment le calcul de l'angle du transducteur), que les images CT et IRM soient dans le même plan. Le défi réside dans la nécessité d'effectuer cette manipulation de manière interactive à l'aide d'un script. Cette opération s'avère chronophage, car à chaque fois que Damien souhaite

observer le mouvement de la tête du singe, il doit relancer le script. Dans le cadre de la résolution de cette problématique, je développe une application permettant de visualiser en temps réel le déplacement ou la rotation de la tête du singe. Cette application offre également la possibilité de sauvegarder l'état de l'ensemble des volumes dans des fichiers DICOM.

7.1 Améliorations possibles

Des améliorations peuvent être envisagées, notamment en ce qui concerne le blocage de l'application. Pour résoudre ce problème, il serait nécessaire de se rendre directement dans la fonction de la boîte à outils k-Wave appelée kspaceFirstOrder3D. De plus, il est envisageable d'apporter une mise à jour de l'interface utilisateur en utilisant du HTML et du CSS, que l'on peut intégrer dans une application MATLAB.

8 Conclusion

À mon arrivée et lors des premières explications, j'ai ressenti un certain niveau de confusion. J'étais entouré de doctorants et de chercheurs, chacun ayant des spécialisations distinctes mais partageant tous un intérêt commun pour les ondes ultrasonores. Tous semblaient posséder une connaissance de base et une compréhension des concepts sous-jacents, alors que j'étais dans la situation inverse, en train d'acquérir encore ces fondamentaux.

Il était évident que mon projet se concentrerait sur la manipulation d'images médicales et la simulation d'ondes. Au fil de l'exposition du projet, j'ai progressivement compris que je n'avais pas nécessairement besoin de maîtriser chaque ligne de code en détail. Mon rôle principal serait de concevoir et de mettre en oeuvre une application MATLAB fonctionnelle, adaptée aux besoins de la recherche.

En conclusion, mon expérience de stage au sein du laboratoire de science a été une période enrichissante au cours de laquelle je me suis plongé dans le domaine captivant du traitement d'images avec MATLAB. La découverte approfondie du fonctionnement du langage MATLAB lui-même a constitué une part significative de mon apprentissage.

L'autonomie dont j'ai bénéficié tout au long du stage a été à la fois stimulante et formatrice. Cependant, cette expérience m'a également fait prendre conscience de certaines limites inhérentes à l'autodidaxie. Bien que j'aie apprécié la liberté d'explorer et d'apprendre par moi-même, je constate qu'il est particulièrement difficile d'acquérir suffisamment de connaissances pour comprendre le code et les algorithmes écrits par L. Dapeng, ce qui m'a quelque peu frustré.

L'ambiance sereine et non stressante du laboratoire a grandement contribué à rendre cette période de stage agréable. Finalement, j'ai réalisé le cahier des charges qui m'a été confié. L'application est fonctionnelle, mais il reste plusieurs aspects à améliorer. J'espère que cette tâche sera assurée par un autre stagiaire prochainement.

9 Bibliographie

1. Bradley Treeby, Ben Cox, and Jiri Jaros, *k-Wave*, Manual Version 1.1 (August 27, 2016), Toolbox Release 1.1,http://www.k-wave.org/manual/k-wave_user_manual_1.1.pdf

2. *MATLAB*, Object-Oriented Programming, Mathworks

3. *MATLAB*, *Graphics*, Mathworks, https://nl.mathworks.com/help/pdf_doc/matlab/creating_plots.pdf

4. *Matlab app designer*, https://nl.mathworks.com/help/matlab/app-designer.html

Glossaire

bloquage de l'interface Un bloquage ou "freeze de l'interface" se produit lorsqu'une interface utilisateur informatique devient temporairement non réactive, figée ou gelée, empêchant l'utilisateur d'effectuer des actions ou de interagir normalement avec le système. Cela peut résulter de divers problèmes, tels que des erreurs logicielles, des conflits de programmes, des limitations matérielles ou d'autres facteurs, entraînant un état momentané d'inactivité de l'interface.. 25

Dicom Les images DICOM, ou Digital Imaging and Communications in Medicine, sont un format standard utilisé dans le domaine médical pour stocker, partager et transmettre des images médicales. Ces images comprennent souvent des données provenant de scanners CT (tomodensitométrie), IRM (imagerie par résonance magnétique), radiographies et autres modalités d'imagerie médicale. Les fichiers DICOM contiennent non seulement l'image elle-même, mais aussi des informations associées telles que les données du patient, les paramètres d'acquisition de l'image, la date et l'heure, etc. Ces informations sont essentielles pour garantir l'intégrité et la traçabilité des données médicales.. 18

MHz Mégahertz. Unité de mesure de la fréquence, valant un million de hertz. Grandeur liée à un phénomène périodique, qui mesure le nombre de fois où ce phénomène se reproduit dans un intervalle donné. . 15

période Dans le contexte des ondes, une période se réfère à la durée nécessaire pour qu'une onde complète (comme une crête et un creux successifs) se reproduise. C'est la mesure du temps entre deux points identiques sur l'onde.. 20

Seuil HU Le seuil HU (Unités Hounsfield) est une mesure utilisée en imagerie médicale, en particulier dans les scanners CT (tomodensitométrie). Les unités Hounsfield représentent l'atténuation des rayons X par les tissus et sont calibrées de manière à attribuer des valeurs spécifiques à différents matériaux anatomiques. Le zéro HU est souvent attribué à l'eau, et les valeurs positives et négatives représentent respectivement une densité tissulaire plus élevée ou plus faible que celle de l'eau. Par exemple, les tissus denses comme l'os auront des valeurs HU positives, tandis que les tissus plus mous comme les organes auront des valeurs HU généralement positives mais plus proches de zéro.. 20

transducteur Un transducteur est un dispositif qui convertit un type d'énergie en un autre. Dans le contexte médical ou ultrasonore, un transducteur est souvent utilisé pour convertir l'énergie électrique en ondes ultrasonores et vice versa, permettant ainsi la création d'images échographiques en utilisant les échos des ultrasons.. 10, 11

voxel Le voxel, mot-valise créé en contractant ń volume ż et ń pixel ż (ce dernier est lui-même une contraction de ń picture ż et ń element ż), est à la 3D ce que le pixel est à la 2D. Il stocke une information physique (couleur, densité, intensité, etc.) d'un point d'un volume sur un maillage régulier. . 14

10 Annexes

10.1 Script Matlab

Script du code de L. Dapeng modifié par mes soins pour qu'il tourne en tant que script.

```
1
2  %% Le programme principal de simulation de champ sonore ultrasonore
3  %transcrânien comprend principalement : la sélection du point cible et du point incident,
4  %l'acquisition des paramètres acoustiques du crâne, la simulation K_WAVE
5  %%%Version : 2022.9.21, modifiée par : Li Dapeng, utilisant
6  %le programme : AB data, macaque E (enregistré)
7
8  %% initialisation
9  clc;
10 clear;
11 % close all;
12
13 %% Ouvrir l'image CT (CT a été enregistré)
14 [filename,filepath1]=uigetfile('E:\data\配准后数据\E\CT\.dcm');
15 %Sélectionnez simplement n'importe quel fichier DIOCM pour obtenir les informations de l'image CT
16 filepath2=[filepath1 filename];
17 info1 = dicominfo(filepath2); % Lire les informations des données DICOM
18 [V1,spatial1,dim1] = dicomreadVolume(filepath1);
19 %Lire les données, V1 est une matrice tridimensionnelle, représentant les données brutes CT
20 (par rapport à l'enregistrement IRM)
21 %spatial:Résolution spatiale et coordonnées, dim : dimension:dimension
22 V1 = squeeze(V1);%Éliminer les cotes de longueur 1
23 D1=single (V1);%convertir en simple précision
24 clear V1;
25 %%Échelle de gris à la valeur CT
26 CT = D1.* info1.RescaleSlope + info1.RescaleIntercept;
27
28 %% Ouvrir l'image IRM
29 % filename2 est le nom du fichier DICOM que vous souhaitez utiliser
30 [filename,filepath1]=uigetfile('E:\data\配准后数据\E\MRI\.dcm');
31 filepath2=[filepath1 filename];
32 info2 = dicominfo(filepath2);
33 [V2,spatial2,dim2] = dicomreadVolume(filepath1);
34 %V:Données brutes (4D),spatial:Résolution spatiale et coordonnées, dim : dimension:dimension
35 V2 = squeeze(V2);%Éliminer les cotes de longueur 1
36 D2=single (V2);%convertir en simple précision
37 clear V2;
38 MRI=D2;
39
40
41 %% Acquérir la résolution des images CT et IRM
42 MRI_space=[];%Résolution tridimensionnelle de l'IRM (mm)
43 MRI_space(1:2)=info2.PixelSpacing;
44 MRI_space(3)=info2.SliceThickness;
45
46 CT_space=MRI_space;%La résolution tridimensionnelle du CT (mm), du CT et de
47 %l'IRM a été enregistrée et a la même résolution spatiale
48
49 %definitions des variables
50 %% Définissez les limites supérieures et inférieures et les couches
51 %de données CT du crâne (eau, tissus mous et crâne de singe),
52 %différents seuils de données sont différents
53 CT_th1=0;%Réglez le seuil CT de l'eau sur 0 HU,
54 %et tous ceux en dessous de 0 sont définis comme de l'eau
55 CT_th2=1800;%Réglez la limite inférieure du seuil CT du crâne à 1800HU,
56 %et 0-1800HU est considéré comme un tissu mou
57 CT_th3=7000;%Définissez la limite supérieure du seuil CT du crâne sur 7000HU
```

```
%, 1800-7000HU est considéré comme le crâne et le seuil CT supérieur à 7000 est défini sur 2400HU
wave_speed=1500;%vitesse des vagues dans l'eau m/s
wave_frequency=0.5; %la fréquence MHz
ppw=5;% Combien de longueurs d'onde par point de grille (k-wave)

CT(CT<CT_th1)=CT_th1;
CT(CT<CT_th2&CT>CT_th1)=CT_th2;
CT(CT>CT_th3)=CT_th3;

%% Sélectionnez le même emplacement (centre du plan horizontal)
%images IRM et CT pour voir l'effet d'enregistrement (facultatif)
axial_plane_center1 = round(size(CT,3)/2);
axial_plane_center2 = round(size(MRI,3)/2);
CT_im=CT(:,:,axial_plane_center1);
MRI_im=MRI(:,:,axial_plane_center2);
CTandMRI=CT_im*0.4+MRI_im;%image de superposition
figure()
subplot(131)
imagesc(CT_im)
axis equal
axis tight
colormap gray;
colorbar;
title('CT image')
subplot(132)
imagesc(MRI_im)
axis equal
axis tight
colormap gray;
colorbar;
title('MRI image')
subplot(133)
imagesc(CTandMRI)
axis equal
axis tight
colormap gray;
colorbar;
title('CT and MRI iamge')


%% Calculer le pas spatial de l'onde k de la longueur d'onde ultrasonore
%(la simulation tridimensionnelle contient au moins 3 longueurs d'onde par pas)
wave_length=wave_speed/(wave_frequency*1e6)*1000;%longueur d'onde mm
kwave_step=wave_length/ppw;% La taille de pas de chaque point de la grille dans k-wave


%% Interpolation des données brutes CT et IRM
[m,n,k]=size(CT);
[x,y,z] = meshgrid(1:n,1:m,1:k);
%%%Remarquer: Vq = interp3(V,Xq,Yq,Zq) Supposons une grille par défaut de points d'échantillonnage.
%Zone de couverture des points de grille par défaut X=1:n、Y=1:m 和 Z=1:p, 其中 [m,n,p] = size(V)
[xq,yq,zq] = meshgrid(1:kwave_step/CT_space(1):m,
1:kwave_step/CT_space(2):n,1:kwave_step/CT_space(3):k);
CT_inter = interp3(x,y,z,CT,xq,yq,zq);%Données CT interpolées
MRI_inter = interp3(x,y,z,MRI,xq,yq,zq);%Données IRM interpolées

%% Définissez les limites supérieure et inférieure et la stratification des données CT du crâne
%(eau, tissus mous et crâne de singe), et devez stratifier à nouveau après l'interpolation
CT_inter(CT_inter<CT_th1)=CT_th1;
CT_inter(CT_inter<CT_th2&CT_inter>CT_th1)=CT_th2;
CT_inter(CT_inter>CT_th3)=CT_th3;

%% Recadrez ou amplifiez les données brutes pour réserver une place au transducteur
%%Nécessité d'introduire le transducteur pour connaître la taille d'amplification
% %augmentation des données
```

```
[m,n,k]=size(MRI_inter);%taille actuelle des données
add_number=30;%nombre de calques à ajouter
%Augmentation des données IRM
MRI_add=zeros(m+add_number,n,k);
MRI_add(add_number+1:end,:,:)=MRI_inter;
%Augmentation des données CT
CT_add=zeros(m+add_number,n,k);
CT_add(add_number+1:end,:,:)=CT_inter;
%Observer la taille de l'image après amplification
% VolumeViewer3D(MRI_add)%boîte à outils
MRI_inter=[];
MRI_inter=MRI_add;
CT_inter=[];
CT_inter=CT_add;

%% Trouvez le plan où se trouve le point cible et trouvez le reste des coordonnées
%en fonction de l'image du plan
%%value = VV3DModified(MRI_inter);
%La boîte à outils est requise et Coordinate correspond respectivement aux lignes,
%aux colonnes et aux pages de la matrice
%Trouver le plan où se trouve la cible, qu'elle soit sagittale, transversale ou coronale
val = vv3Simplified(MRI_inter);
target_plane_position=val(3);
%Trouver le plan où se trouve l'ACC macaque k: 99, 140, 110, macaque E: 98,120, 120


%% Dessinez le plan où se trouve la cible
target_plane=squeeze(MRI_inter(:,:,target_plane_position));
% figure
% imagesc(target_plane)
% axis equal
% axis tight
% colormap gray;

%% Déterminer les coordonnées restantes de la cible
ACC_target_xy=[val(1),val(2)];%cible


%% Calculer le contour du crâne comme sa direction tangente
target_plane_CT=squeeze(CT_inter(:,:,target_plane_position));



[m n]=find(target_plane_CT>CT_th2);
[k1,av] = convhull(n,m);
x_convhull=n(k1);
y_convhull=m(k1);

%% Déterminer la position du transducteur et calculer l'angle d'incidence et la trajectoire
focol_distance=63.6;%La distance focale réelle du transducteur focalisé
%(calculée à partir du sommet du transducteur)
%%Transformez le transducteur dans le plan sagittal pour maintenir la mise au point
%%Générer un cercle avec le point cible comme centre et la distance focale comme rayon
theta = 1:0.5:360;
x0 = (ACC_target_xy(1));
y0 = (ACC_target_xy(2));
r = focol_distance/kwave_step;%combien de grilles le rayon occupe
x_circle= x0 + r*cosd(theta);
% x_circle=round(x_circle);
y_circle = y0 + r*sind(theta);
% y_circle=round(y_circle);
%%point réservé à l'extérieur du crâne
in=[];%La caractéristique de la position enregistrée, 0 signifie à l'extérieur du polygone,
%1 signifie à l'intérieur et à l'extérieur du polygone
for i=1:1:length(x_circle)
    in(i)=inpolygon(x_circle(i),y_circle(i),x_convhull,y_convhull);
```

```
end
x_circle1=x_circle(in==0);%obtenir le point intérieur
y_circle1=y_circle(in==0);%obtenir le point intérieur

%% Limiter les points à sélectionner et exclure les points déraisonnables
x_circle1_cut=[];
y_circle1_cut=[];
i = find (y_circle1<ACC_target_xy(2)-(focol_distance/2));
%%i=find((x_circle1>50 & x_circle1<190 & y_circle1<ACC_target_xy(2)));
%i=find((x_circle1>50 & x_circle1<190));
x_circle1_cut= x_circle1(i);%obtenir le point intérieur
y_circle1_cut=y_circle1(i);%obtenir le point intérieur



%% Obtenir les informations sur le crâne, les coordonnées x, y et la valeur HU
%du faisceau sonore à travers le crâne (modifié)
pos_final_all={};%L'ensemble des coordonnées de tous les points passant par le crâne
skull_profile_all={};%Une collection de pixels avec tous les points passant par le crâne
angle_inside_all=[];%Ensemble des angles de tous les points passant par le crâne
x2=x0;y2=y0;%coordonnées cibles
for i1=1:length(x_circle1_cut)
    % x1=199;y1=121;
    x1=x_circle1_cut(i1);%Coordonnées du sommet du transducteur
    y1=y_circle1_cut(i1);

    %%Calculer les coordonnées en pixels de l'image traversées par la ligne
    pos1=[];
    pos2=[];
    pos3=[];
    pos_final=[];
    pos1 = [x1,y1;x2,y2];%Obtenir les coordonnées du point final de la ligne
    dx=pos1(2,1)-pos1(1,1);%La première colonne est l'abscisse
    dz=pos1(2,2)-pos1(1,2);%La deuxième colonne est la coordonnée y
    if dz==0
        pos2(:,1)=min(pos1(1,1),pos1(2,1)):max(pos1(1,1),pos1(2,1));
        pos2(:,2)=pos1(2,2);
        pos_final=round(pos2);
    elseif dx==0
        pos2(:,2)=min(pos1(1,2),pos1(2,2)):max(pos1(1,2),pos1(2,2));
        pos2(:,1)=pos1(1,1);
        pos_final=round(pos2);
    else
        k=dz/dx;
        b=pos1(1,2)-pos1(1,1)*k;
        %générer une ligne: (1)z=kx+b;(2)x=(z-b)/k;
        %Calculer avec l'abscisse
        x11=min(pos1(2,1),pos1(1,1));x22=max(pos1(2,1),pos1(1,1));
        pos2(:,1)=x11:0.25:x22;
        pos2(:,2)=pos2(:,1).*k+b;
        pos2=round(pos2);
        pos2=unique(pos2,'rows','stable');
        %Calculé avec les coordonnées z
        z11=min(pos1(1,2),pos1(2,2));z22=max(pos1(1,2),pos1(2,2));
        pos3(:,2)=z11:0.25:z22;
        pos3(:,1)=(pos3(:,2)-b)./k;
        pos3=round(pos3);
        pos3=unique(pos3,'rows','stable');
        if size(pos2,1)>=size(pos3,1)
            pos_final=pos2;
        else
            pos_final=pos3;
        end
    end
    clear pos1 pos2 pos3
```

```
    if pos_final(end,:)~=[x2 y2]
        pos_final=flip(pos_final);
    end
    pos_final_all{i1}=pos_final;
    %%Obtenez le contour du crâne en fonction des pixels passés par la ligne droite et
    %trouvez le point d'entrée du crâne
    skull_profile=[];
    for i2=1:1:length(pos_final)
        skull_profile(i2)=target_plane_CT(pos_final(i2,2),pos_final(i2,1));
    end
    skull_profile_all{i1}=skull_profile;
    %%Calculer quelques informations crâniennes (épaisseur, porosité, variation de vitesse, etc.)
    a=find(skull_profile~=0);%Trouver tous les points du crâne qui ne sont pas 0
    x_inside=pos_final(a(1),1);
    y_inside=pos_final(a(1),2);%Trouver les coordonnées du point incident

    %%Calculer l'angle entre deux droites
    %%Trouver les coordonnées des extrémités tangentes correspondantes
    %des deux côtés du point incident
    for i3=1:1:length(x_convhull)-1
        u = ((x1-x_convhull(i3))*(y1-y2) - (y1-y_convhull(i3))*(x1-x2)) /
        ((x1-x2)*(y_convhull(i3)-y_convhull(i3+1))-(y1-y2)*(x_convhull(i3)-x_convhull(i3+1)));
        t = ((x1-x_convhull(i3))*(y_convhull(i3)-y_convhull(i3+1)) -
        (y1-y_convhull(i3))*(x_convhull(i3)-x_convhull(i3+1))) /
        ((x1-x2)*(y_convhull(i3)-y_convhull(i3+1))-(y1-y2)*(x_convhull(i3)-x_convhull(i3+1)));
        if (u >= 0 && u <= 1.0) && (t >= 0 && t <= 1.0)

            % Crossing
            %fprintf('They cross or intersect.\n');
            x3=x_convhull(i3);
            y3=y_convhull(i3);
            x4=x_convhull(i3+1);
            y4=y_convhull(i3+1);
        else
            % ^^INo crossing
            %fprintf('They do not cross or they completely overlap.\n');
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%
    %%Calculez l'angle en utilisant la formule du produit scalaire des vecteurs a.b=|a|*|b|*con(l);
    v1 = [x1,y1] - [x2,y2];
    v2 = [x4,y4] - [x3,y3];
    angle_inside=acos(dot(v1,v2)/(norm(v1)*norm(v2)))*(180/pi);
    %Angle de rotation radian* (180/pi)
    angle_inside_all(i1)=angle_inside;
end

%% Trouvez la position du sommet et l'angle de l'angle d'incidence le plus proche
%de l'ange ř (x est l'angle requis, tel que 90 degrés)
angel=foundTransductor(x_circle1_cut,y_circle1_cut,angle_inside_all,
target_plane_CT,ACC_target_xy,x_convhull,y_convhull,x0,y0);
x_angel=[];
y_angel=[];
angle_angel=[];
j=1;
diff_angel=abs(angle_inside_all-angel);
k=find(diff_angel==min(diff_angel));
x_angel=x_circle1_cut(k);
y_angel=y_circle1_cut(k);
angle_angel=angle_inside_all(k);


% %% Trouver le profil transcrânien pour un angle d'incidence angelř
%(l'incidence oblique n'est pas précise)
skull_profile_1=skull_profile_all{k};
```

```
skull_profile_x=0:kwave_step:(length(skull_profile_1)-1)*kwave_step;%mm
skull_profile_2=skull_profile_1(skull_profile_1>1800);
Skull_thickness=(length(skull_profile_2)-1)*kwave_step;


%% Tracez une ligne droite reliant le point cible et le point incident,
%et déterminez l'angle de rotation (en rendant le faisceau sonore perpendiculaire
%à l'image pour une observation facile)
transducer_xy=[x_angel(1),y_angel(1)];%cibler
v1=ACC_target_xy;%cibler
v2=transducer_xy;%sommet du transducteur
v3=[];%pointe verticale
v3(1)=ACC_target_xy(1);v3(2)=transducer_xy(2);
distance_tran_tar=round(norm(v2-v1));
%Distance entre le sommet du transducteur et le point cible (maille)
%Calculer l'angle entre la direction du faisceau sonore ultrasonique et la normale
rotate_angle=acos((norm(v3-v1)/norm(v2-v1)))*(180/pi);

%% Déterminer les coordonnées du point cible et les coordonnées du
%point incident ultrasonore à partir de l'image planaire contenant le point cible, voici un exemple
transducer_position=[v2(2),v2(1),target_plane_position];
target_position=[v1(2),v1(1),target_plane_position];
transform_matrx=MRI_inter*0;%Utilisé pour enregistrer
%les sommets et les cibles du transducteur
transform_matrx(target_position(1)-2:target_position(1)+2,
target_position(2)-2:target_position(2)+2,target_position(3)-2:target_position(3)+2)=200;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Faites pivoter les données en fonction de l'angle calculé
%(ici, faites pivoter autour de l'axe z, volumeViewer affiche la zone cible dans la tranche xy)

MRI_rotate=imrotate3(MRI_inter,-rotate_angle,[0 0 1],'linear','loose','FillValues',0);
CT_rotate=imrotate3(CT_inter,-rotate_angle,[0 0 1],'linear','loose','FillValues',0);
transform_matrx_rotate=imrotate3(transform_matrx,-rotate_angle,[0 0 1],
'linear','loose','FillValues',0);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
source_roc      = 66e-3;
source_diameter = 64e-3;
Lax1=find(transform_matrx_rotate==200);%
s=size(MRI_rotate);%Calculer la taille d'un tableau tridimensionnel
[m,n,k]=ind2sub(s,Lax1(round(length(Lax1)/2)));
%Convertir un indice unique maximum en indice multidimensionnel tridimensionnel
targrt_position_new1=[m,n,k];
transducer_position_new1=[m-distance_tran_tar,n,k];
diameter1=0;
if(mod(round(source_diameter/kwave_step*1000),2)==0)
    diameter1 = round(source_diameter/kwave_step*1000)+1;
else
    diameter1 = round(source_diameter/kwave_step*1000);
end
bowlVolshow1 = makeBowl(size(MRI_rotate),transducer_position_new1,
round(source_roc/kwave_step*1000),diameter1,[m ,n, k]);
bowlVolshow1 = bowlVolshow1*900;
bowlVolshow1 = MRI_rotate+bowlVolshow1;


dimensioncut =cropMtri(bowlVolshow1,targrt_position_new1);
clear bowlVolshow1 CT_inter MRI_inter MRI_im MRI_add CT_add CT_im xq yq zq

%% Observez les données tournées
% VolumeViewer3D(MRI_rotate);%La boîte à outils est requise et Coordinate
%correspond respectivement aux lignes, aux colonnes et aux pages de la matrice

%% Découpez les données (y compris CT, IRM et matrice de position)
%pour réduire la quantité de calculs
MRI_final=[];
```

```
CT_final=[];
transform_matrx_final=[];
row_set=[dimensioncut(3),dimensioncut(4)];
column_set=[dimensioncut(1),dimensioncut(2)];
page_set=[dimensioncut(5),dimensioncut(6)];
MRI_final=MRI_rotate(row_set(1):row_set(2),column_set(1):column_set(2),page_set(1):page_set(2));
CT_final=CT_rotate(row_set(1):row_set(2),column_set(1):column_set(2),page_set(1):page_set(2));
transform_matrx_final=transform_matrx_rotate(row_set(1):
row_set(2),column_set(1):column_set(2),page_set(1):page_set(2));

% VolumeViewer3D(CT_final);
% VolumeViewer3D(transform_matrx_final);

%% Trouver les sommets et les points cibles des transducteurs transformés
Lax=find(transform_matrx_final==200);%
s=size(transform_matrx_final);%Calculer la taille d'un tableau tridimensionnel
[m,n,k]=ind2sub(s,Lax(round(length(Lax)/2)));%Convertir un indice unique
%maximum en indice multidimensionnel tridimensionnel
targrt_position_new=[m,n,k];
transducer_position_new=[m-distance_tran_tar,n,k];


%% Observez la position relative de la cible de stimulus transformée
%et du point incident du transducteur dans le nouveau plan
v11=[targrt_position_new(2),targrt_position_new(1)];%cible
v22=[transducer_position_new(2),transducer_position_new(1)];%sommet du transducteur

CT_final_Sagittla=squeeze(CT_final(:,:,targrt_position_new(3)));

%% Calculer l'épaisseur du crâne et la distribution des valeurs CT à travers
%lesquelles passe le faisceau sonore
Transcranial_HU=CT_final_Sagittla(v22(2):v11(2),v11(1));
position_skll=(0:abs(v11(2)-v22(2))).*kwave_step;

skull_index=find(Transcranial_HU>1800);%Trouver le point de coupure du crâne
skull_length=abs((skull_index(1)-skull_index(end)))*kwave_step;

figure()
plot(position_skll,Transcranial_HU)
xlabel('Ditance (mm)','FontSize',16);
ylabel('HU','FontSize',16);
title(['The thickness of the skull crossed by the ultrasound is ',
num2str(skull_length),'mm'],'FontSize',16)


%% Définissez les limites supérieure et inférieure et la stratification
%(eau, tissus mous et crâne) des données CT du crâne, et devez stratifier
%à nouveau après l'interpolation
CT_final(CT_final<CT_th1)=CT_th1;
CT_final(CT_final<CT_th2&CT_final>CT_th1)=CT_th2;
CT_final(CT_final>CT_th3)=CT_th3;

%% définir les constantes acoustiques
HU_max=max(CT_final(:));
HU_water=0;
swater=wave_speed;%Règle la vitesse du son dans l'eau, correspondant au minimum HU
dwater=1000;%densité de l'eau
stissue=1560;%vitesse du son des tissus mous
dtissue=1030;%densité des tissus mous
sbone=3100;%Supposée être la vitesse du son dans l'os cortical, correspondant au maximum HU
dbone=1900;%Régler sur la densité dans l'os cortical, correspondant au plus grand HU
f=wave_frequency;%Tester la fréquence ultrasonique pour ce coefficient d'atténuation (MHz)
b=1.1;%Coefficient d'absorption a=a0*f^y ; a0 doit être transformé en [dB/(MHz^y cm)]
amin=0.2;%Attenuation  [dB/(MHz^y cm)]
amax=8;%%Attenuation  [dB/(MHz^y cm)]
atissue=0.6;%Attenuation  [dB/(MHz^y cm)]
%alpha_dB = 20*log10  (exp (alpha_Neper))
```

```
%alpha_dB= alpha_Neper * 20*log10 (exp)  = 8.686 * alpha_Neper

%% Calcul des constantes de champ sonore, y compris la vitesse, la densité et l'atténuation du son
axial_plane_center = round(size(CT_final,3)/2);


%Utilisez HU pour former une relation linéaire avec la densité et la vitesse pour résoudre
density=(CT_final-HU_water).*((dbone-dwater)/(HU_max-HU_water))+dwater;%Calculer la densité
speed=(CT_final-HU_water).*((sbone-swater)/(HU_max-HU_water))+swater;%Calculer la vitesse
k=(dbone-density)./(dbone-dwater);%Calculer la porosité
atta=amin+(amax-amin).*(k.^0.5);%Calculer le facteur d'atténuation
density(CT_final==1800)=dtissue;%densité des tissus mous
speed(CT_final==1800)=stissue;%vitesse des tissus mous
atta(CT_final==1800)=atissue;%coefficient d'atténuation des tissus mous
atta(CT_final==0)=amin;%Régler l'atténuation de l'eau sur amin

%% Afficher les résultats de calcul des paramètres acoustiques (facultatif)
axial_plane_center = round(size(CT_final,3)/2);
HU_im=CT_final(:,:,axial_plane_center);
density_im=density(:,:,axial_plane_center);
speed_im=speed(:,:,axial_plane_center);
atta_im=atta(:,:,axial_plane_center);


%% Définir les conditions de simulation de l'onde k %%

% Définir la grille de calcul
[Nx,Ny,Nz]=size(CT_final);
dx = kwave_step/1e3;    ^^I% La taille du pas dans la direction x
dy = dx ;            % La taille du pas dans la direction y
dz=dx ; % La taille du pas dans la direction z
kgrid = kWaveGrid(Nx, dx, Ny, dy, Nz, dz);

%Définir les propriétés du support
medium.sound_speed = swater * ones(Nx, Ny, Nz);^^I% [m/s]
medium.density = dwater * ones(Nx, Ny, Nz);       % [kg/m^3]
medium.alpha_coeff = 0 * ones(Nx, Ny, Nz);  % [dB/(MHz^y cm)]
medium.alpha_power=b;

%définir la couche d'atténuation
medium.sound_speed = speed;^^I% [m/s]
medium.density = density;       % [kg/m^3]
medium.alpha_coeff=atta; % [dB/(MHz^y cm)]

source_f0       = wave_frequency*1e6;      % source frequency [Hz]
%Définir la grille horaire
% cfl             = 0.5;      % CFL Numéro
cfl             = 0.2;      % CFL Numéro
PPP = round(ppw / cfl);%Combien de cycles par point
dt = 1 / (PPP * source_f0);
%(1)Définir automatiquement l'heure
% kgrid.makeTime(medium.sound_speed);
%(2)Définir manuellement l'heure
% cfl             = 0.5;      % CFL number
max_distance=sqrt(Nx^2+Ny^2+Nz^2)*dx;%La distance maximale de propagation des ultrasons
t_end           = max_distance/1500;
% Temps de calcul total (doit être supérieur au temps de stabilisation du système)
% PPP = round(ppw / cfl);%Calculer combien de points par période
Nt = round(t_end / dt);
kgrid.setTime(Nt, dt);

%définir la source
% Définir les propriétés du transducteur
source_f0       = wave_frequency*1e6;      % source frequency [Hz]
source_roc      = 66e-3;    % Rayon de courbure du transducteur [m]
source_diameter = 64e-3;    % Diamètre effectif du transducteur [m]
```

```
source_amp      = 1e6;      % La pression acoustique de surface du transducteur [Pa]

%Définit le type de transducteur, ici un bol de focalisation
% Définir la position spatiale et la rotation du bol
bowl_pos = [kgrid.x_vec(transducer_position_new(1)),
kgrid.y_vec(transducer_position_new(2)), kgrid.z_vec(transducer_position_new(3))];
focus_pos = [kgrid.x_vec(targrt_position_new(1)),
kgrid.y_vec(targrt_position_new(2)), kgrid.z_vec(targrt_position_new(3))];

% définir un vide kWaveArray
karray = kWaveArray('BLITolerance', 0.01, 'UpsamplingRate', 10);
% Augmenter le transducteur en forme de bol
karray.addBowlElement(bowl_pos, source_roc, source_diameter, focus_pos);

%% définition binary mask
source.p_mask = karray.getArrayBinaryMask(kgrid);
% Définir la série temporelle du signal
source_sig = createCWSignals(kgrid.t_array, source_f0, source_amp, 0);
% Définir la source du signal, la redistribution du signal,
%cette étape demande beaucoup de temps et une grande mémoire
source.p = karray.getDistributedSourceSignal(kgrid, source_sig);
% VolumeViewer3D(int8(source.p_mask));
%
% VolumeViewer3D(medium.sound_speed);
% VolumeViewer3D(medium.density);
% VolumeViewer3D(medium.alpha_coeff);
% définir le capteur
sensor.mask = [1,1,1, Nx, Ny,Nz].';%La grille à enregistrer est la grille entière
sensor.record = {'p_max'};
%Le paramètre à enregistrer est la valeur maximale de la pression acoustique
% sensor.record = {'p_min'};%Le paramètre à enregistrer est
%la valeur minimale de la pression acoustique
% N'enregistrer que des valeurs stables
record_periods  = 3;        % nombre de cycles à enregistrer
sensor.record_start_index =1 ;%%kgrid.Nt - record_periods * PPP + 1

%% Définir les conditions de sortie de la simulation
% display_mask = source.p_mask;%Afficher la grille des transducteurs
display_mask = CT_final*0;%montrer le crâne
display_mask(CT_final>1800)=1;
% VolumeViewer3D(display_mask);
input_args = {'DisplayMask', display_mask, 'PlotLayout',
true, 'PMLInside', false, 'PlotPML', false,...
    'DataCast', 'single', 'DataRecast', true,'PlotScale', ...
    [-1, 1] * source_amp};%cpu opération
% input_args = {'DisplayMask', display_mask, 'RecordMovie',true,'MovieName'
,'1','PlotLayout', true,  'MovieProfile', 'MPEG-4', 'MovieArgs', {'FrameRate', 10},...
%     'PMLInside', false, 'PlotPML', false,...
%     'DataCast', 'gpuArray-single', 'DataRecast', true,'PlotScale', ...
%     [-1, 1] * source_amp};%gpu opération,et enregistrer une vidéo
sensor_data = kspaceFirstOrder3D(kgrid, medium, source, sensor, input_args{:});
%Démarrer la simulation d'onde k 3D

%% Évaluation des résultats de simulation %%
%% Extraire l'amplitude du signal du capteur
% amp = extractAmpPhase(sensor_data.p_max, 1/kgrid.dt, source_f0, ...
%     'Dim', 2, 'Window', 'Rectangular', 'FFTPadding', 1);
%La deuxième dimension représente le temps
% % Matrice de reconstruction
% amp = reshape(amp, Nx, Ny,Nz);
amp=sensor_data.p_max;

%% Obtenir les coordonnées 3 axes
x_vec = kgrid.x_vec;
y_vec = kgrid.y_vec;
```

```
z_vec = kgrid.z_vec;

%% Trouver la mise au point, calculer la profondeur de mise au point
% %Exclure les points de valeur maximale sur le support
Estimated_focal_length=10/1000;%distance de mise au point visuelle
start_n=round(Estimated_focal_length/dx);
amp_exculude=amp;
amp_exculude(1:start_n,:,:)=0;

amp_max=max(amp_exculude(:));
%Calculer la valeur maximale d'un tableau tridimensionnel,Pa
s=size(amp);%Calculer la taille d'un tableau tridimensionnel
Lax=find(amp==amp_max);
%Calculer l'indice unique de la position maximale
[m,n,k]=ind2sub(s,Lax);
%Convertir un indice unique maximum en indice multidimensionnel tridimensionnel
Loc_focus=[m,n,k];%indice de la valeur maximale
focus_coordinate=[x_vec(m),y_vec(n),z_vec(k)];
focal_length=norm(focus_coordinate-bowl_pos)*1000;
%Calculer la distance du point focal au sommet du transducteur,mm


%% Dessiner des images comparatives de CT, IRM et champ sonore
amp_focus_coordinate_oral=field_im1(targrt_position_new(1),targrt_position_new(2));
%Trouver la pression acoustique du point cible d'origine (Pa)
focus_coordinate_oral=[x_vec(targrt_position_new(1)),
y_vec(targrt_position_new(2)),z_vec(targrt_position_new(3))];
figure
subplot(1,3,1)
imagesc(squeeze(MRI_final(:,:,targrt_position_new(3))))
axis equal
axis tight
hold on
plot([v11(1),v22(1)],[v11(2),v22(2)],'r');
title(['MRI image'])
subplot(1,3,2)
imagesc(CT_final_Sagittla)
axis equal
axis tight
hold on
plot([v11(1),v22(1)],[v11(2),v22(2)],'r');
title(['CT image in Sagittla Plan'])
subplot(1,3,3)
imagesc(1e3 * x_vec, 1e3 * y_vec, field_im1'/1e6);
xlabel('Axial Position [mm]');
ylabel('Lateral Position [mm]');
axis image;
title(['Transcranial Pressure Field at targrt (ACC) is ',
num2str(amp_focus_coordinate_oral/1e6),'MPa'],'FontSize',16)
h=colorbar;
set(get(h,'title'),'string','MPa');
hold on
plot(bowl_pos(1)*1000,bowl_pos(2)*1000,'r*')
hold on
plot(focus_coordinate_oral(1)*1000,focus_coordinate_oral(2)*1000,'r*')
hold on
plot([bowl_pos(1)*1000,focus_coordinate_oral(1)*1000],
[bowl_pos(2)*1000,focus_coordinate_oral(2)*1000],'r')
view(-90,-90)

%% Trouvez la mise au point d'origine et calculez la profondeur de mise au point
% %Exclure les points de valeur maximale sur le support

focus_coordinate_oral=[x_vec(targrt_position_new(1)),
y_vec(targrt_position_new(2)),z_vec(targrt_position_new(3))];

```

```
% Contient la distribution du champ sonore ainsi que les points cibles et incidents et les médias
amp_focus_coordinate_oral=field_im1(targrt_position_new(1),targrt_position_new(2))
;%Trouver la pression acoustique du point cible d'origine (Pa)
figure;
imagesc(1e3 * x_vec, 1e3 * y_vec, field_im1'/1e6);
xlabel('Axial Position [mm]');
ylabel('Lateral Position [mm]');
axis image;
title(['Transcranial Pressure Field at targrt (ACC) is ',
num2str(amp_focus_coordinate_oral/1e6),'MPa'],'FontSize',16)
h=colorbar;
set(get(h,'title'),'string','MPa');
hold on
plot(bowl_pos(1)*1000,bowl_pos(2)*1000,'r*')
hold on
plot(focus_coordinate_oral(1)*1000,focus_coordinate_oral(2)*1000,'r*')
hold on
plot([bowl_pos(1)*1000,focus_coordinate_oral(1)*1000],
[bowl_pos(2)*1000,focus_coordinate_oral(2)*1000],'r')
view(-90,-90)


%% résultat de sortie
clc
fprintf('The wavelength is %g mm\n', wave_length);
fprintf('The number of points per wave is %g \n', ppw);
fprintf('The k-wave step is %g mm\n', kwave_step);
fprintf('The surface pressure  is %6.2f MPa\n',source_amp/1e6);
fprintf('The  pressure at focal point is %6.2f MPa\n',amp_max/1e6);
fprintf('The   pressure at targrt is %6.2f MPa\n',amp_focus_coordinate_oral/1e6);
fprintf('The focal length  is %6.2f mm\n',focal_length);
% fprintf(' Largeur axe long à mi-hauteur=%g mm\n',b*dx*1000);
%Considérez l'unité de longueur de la grille, multipliée par la taille du pas
% fprintf(' Largeur du petit axe à mi-hauteur=%g mm\n',a*dx*1000);
```

10.2 Images provenant de la simulation

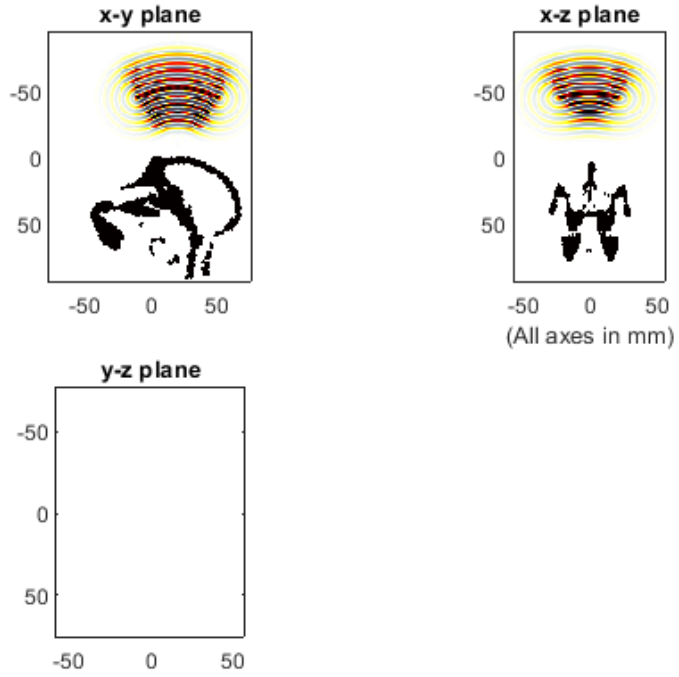

FIGURE 10 – Images provenant de la simulation

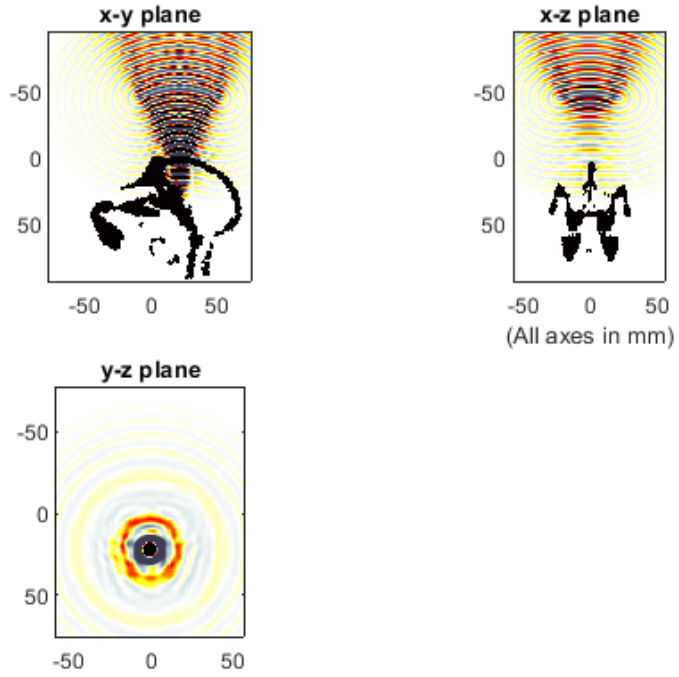

FIGURE 11 – Images provenant de la simulation

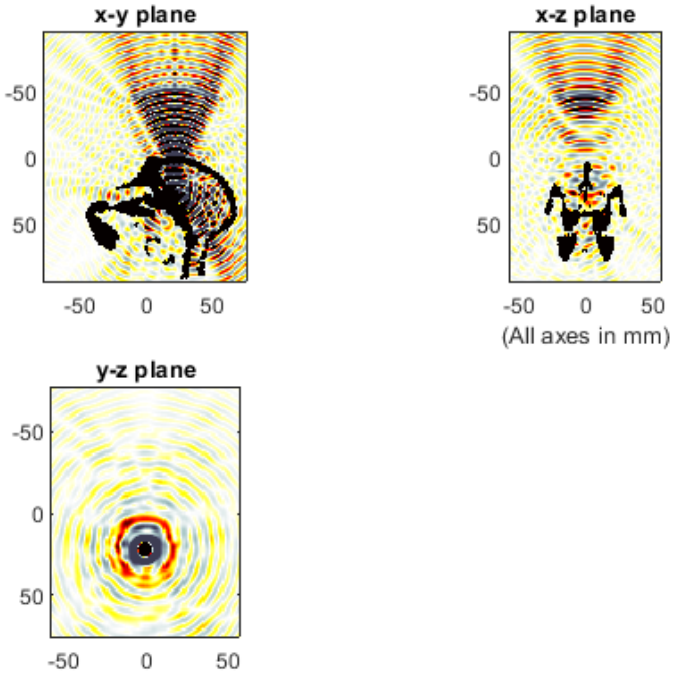

FIGURE 12 – Images provenant de la simulation