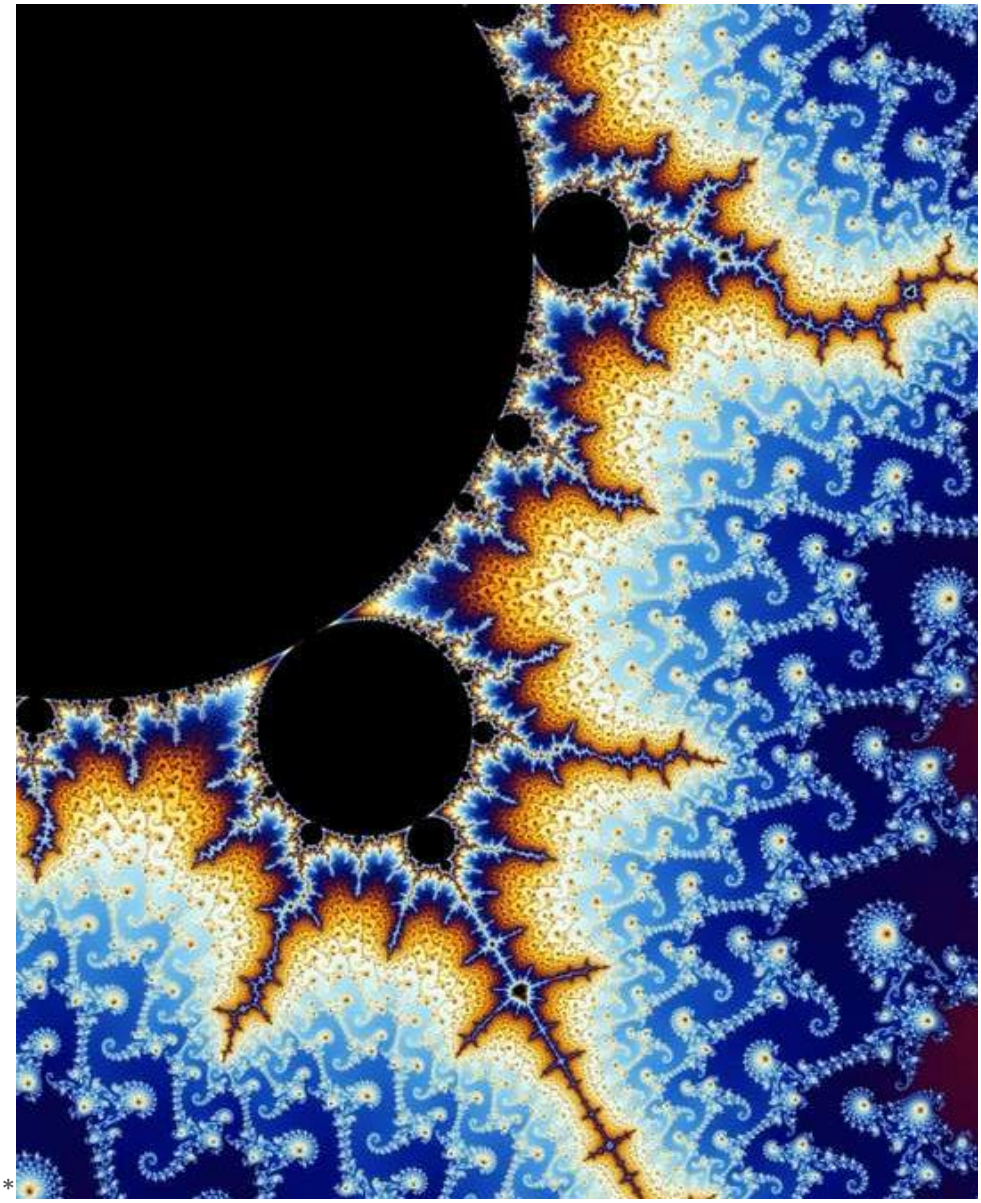


*

Projet final

CONTROLE DES ONDES ACOUSTIQUES

Quentin Delacroix | Pollution acoustique et EM | 2021

Introduction

Le module contexte et enjeux de la ST Pollution acoustique et électromagnétique a mis en évidence le besoin de contrôler les ondes acoustiques. Nous avons alors vu que des géométries particulières sur les frontières des domaines où les ondes acoustiques peuvent évoluer peuvent permettre de localiser les ondes, et ainsi contrôler la pollution acoustique en épargnant certaines zones d'intérêts.

Le but de ce projet est de constater cet effet en simulant l'impact d'une bordure en forme de fractale sur la propagation d'une onde plane. Le travail principal réside donc dans la génération d'une maille en forme de fractale, qui sera le sujet de la partie I. Dans la partie II, on s'intéresse à l'approximation du paramètre alpha, permettant de simplifier la modélisation d'une surface poreuse à la frontière du domaine. Enfin, dans la partie III, l'équation aux dérivées partielles est résolue sur le maillage fractal, puis la solution est tracée.

Part I - Control of the numerical solution accuracy of acoustic wave by changing the quality of the mesh

Fichier de code associé : part1.py

Le but des questions ici est de construire petit à petit tous les outils nécessaires à la génération de la maille fractale. Toute la difficulté réside dans la manipulation de la structure de donnée utilisée : la maille est ici stockée dans une matrice creuse, plus adaptée à l'usage que nous souhaitons en faire. Cependant, ne connaissant initialement pas cette structure, sa manipulation a été relativement maladroite.

J'ai ainsi commencé mon travail sur une maille de quadrangles, carré, que j'ai par la suite modifié à l'aide des diverses fonctions que les questions demandaient de coder.

Dans le code, chaque fonction est détaillée par des commentaires. J'invite ainsi le lecteur à consulter le fichier pour comprendre plus en détail comment ont été réalisées les fonctions.

- Question 1 et 2 :

Fonctions associées : aspect_ratio_tri ; edge_length_factor_tri

Les questions 1 et 2 demandaient de calculer l'aspect ratio et le edge length factor pour un élément. Ce n'était pas bien difficile : il suffisait d'appliquer les formules telles que vues dans le cours. A noter cela dit que j'ai réalisé des fonctions permettant de les calculer sur des éléments triangles ; puisque le sujet amènera de toute manière à changer tous les quadrangles en triangles.

- Question 3 :

Fonction associée : shift_internal_nodes_coords

Le but de cette fonction est de modifier aléatoirement (légèrement) l'emplacement des nœuds à l'intérieur du domaine (en excluant ceux sur la frontière), pour obtenir une maille plus chaotique. Je les ai simplement translatés entre -1/3 et 1/3 de la longueur d'un élément quadrangle, dans les deux dimensions.

Figure 1 : shift_internal_nodes_coords appliquée à une maille carrée 32x32

À noter que cette fonction ne marche actuellement que sur une maille carrée : sur la maille fractale, il faudrait pouvoir trouver les nœuds qui constituent la frontière, pour ne pas les translater. Ceci est le but de la Question 8, optionnelle, que je n'ai malheureusement pas eu le temps de traiter.

- Question 4

Fonctions associées : add_node_to_mesh ; remove_node_to_mesh ; add_elem_to_mesh ; remove_elem_to_mesh

Cette question, qui demande un peu de travail, permet de se familiariser avec la structure de donnée.

Ce n'est vraiment pas compliqué d'ajouter un élément, et surtout un nœud. En revanche, retirer un élément demande un peu plus de réflexion, puisqu'il faut localiser dans elem2nodes les nœuds à retirer, puis décrémenter p_elem2nodes pour garder la structure intacte. Pour ce qui est de retirer un nœud, il faut en même temps trouver tous les éléments qui contiennent ce nœud, pour les supprimer aussi. Il y a une manière simple de le faire, en une ligne, mais elle m'était inconnue lorsque j'ai travaillé sur cette question. Cela m'a donc demandé plus d'efforts que nécessaire…

- Question 5

Fonction associée : compute_barycenter_of_element

Cette fonction est relativement simple. Il suffit de trouver les nœuds qui composent l'élément, puis d'utiliser leurs coordonnées pour calculer la position du barycentre.

- Fonctions supplémentaires

Fonctions associées : remove_all_inside_rectangle ; remove_orphan_nodes

Ces deux fonctions n'étaient pas demandées explicitement, mais j'ai tout de même eu à les coder pour pouvoir aborder plus confortablement la suite.

La première fonction, comme son nom l'indique, permet de supprimer tous les éléments à l'intérieur d'un rectangle. En entrée, on indique les coordonnées du coin inférieur gauche et supérieur droit, et cette fonction supprime tous les éléments dont le barycentre se trouve dans le rectangle. Ainsi, elle fait appel à compute_barycenter_of_element et remove_elem_to_mesh.

La seconde fonction permet de supprimer les nœuds orphelins, n'appartenant à aucun élément. J'ai eu à coder cette fonction vers la fin du projet, les nœuds orphelins posant un problème d'inversibilité de matrice. En réalité, elle est très peu optimisée, et sert à détruire les nœuds non supprimés par la fonction précédente. Avec plus de temps, j'aurais fait les choses plus proprement.

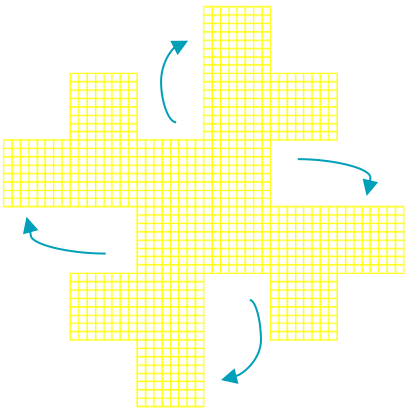
- Question 6

Fonction associée : generate_fractal_mesh

La question principale. C'est de loin ce qui m'a pris le plus de temps à réaliser de tout le projet. C'est celle qui utilise toutes les fonctions définies précédemment pour générer la fractale, à partir de la maille régulière initiale.

La fractale que j'ai choisi de former est un peu particulière. Elle repose sur 4 « patterns » de déplacement d'un carré sur le bord : déplacement du bas vers la droite, de la droite vers le haut, du haut vers la gauche, et de la gauche vers le bas (voir flèches sur l'image ci-dessous). A chaque opération réalisée sur une face, l'ordre de la fractale sur cette face augmente :

Ordre 0



Ordre 1

Ordre 2

Ordre 3

Ordre 4

Il faut, entre chaque opération, mettre à jour l'état des fractales sur chaque face : quelle va être la prochaine opération à appliquer, et quelles sont les coordonnées du rectangle qu'il va falloir retirer. Pour cela, j'ai choisi d'enregistrer l'état de chaque face dans des dictionnaires. Il ne reste plus qu'à itérer sur chaque face, une nombre de fois égal à l'ordre.

Il faut également veiller à ce qu'il ait suffisemment d'éléments sur la maille d'origine, pour atteindre un ordre plus élevé. En pratique, dans des temps raisonnables, le plus grand ordre que j'ai réussi à atteindre est le $6^{ème}$.

- Question 7 :

Fonction associée : shift_quad_to_tri_all (et shift_quad_to_tri)

Cette fonction permet de convertir tous les quadrangles en triangles. Pour cela, j'ai du coder une fonction shift_quad_to_tri qui divise un élément en deux triangles, et les rajoute à la structure de donnée. Si il y a N quadrangles dans la maille, on itère sur chaque quadrangle pour former 2*N triangles. Il ne reste plus qu'à supprimer les N premiers éléments de la structure, pour ne garder que les triangles.

Part II - Control of acoustic wave by changing the damping on a surface

Fichier de code associé : part2.py

Le but de cette partie est d'approximer le coefficient alpha tel qu'expliqué auparavant, en minimisant la fonction d'erreur epsilon(alpha) à l'aide de scipy.optimize.minimize. La principale difficulté de cette partie était ainsi de recopier les (longues) formules du cours sans se tromper…

Pour réaliser cela, j'ai défini une grosse fonction epsilon, puis une nouvelle fonction epsilon_k à l'intérieur, puis plein de petites fonctions encore à l'intérieur pour calculer les différents agrégats de la formule d'epsilon_k(alpha) tel que donné dans le cours. Ce n'était pas dur, mais laborieux ; et trouver les erreurs dans le code n'a pas toujours été évident.

J'ai également défini tous les paramètres utiles à l'intérieur d'epsilon, à partir de la porosité, la résistivité et la tortuosité du milieu. Les valeurs prises en pratique dans le code correspondent aux caractéristiques du Melamine foam, tel que donné dans l'énoncé.

Enfin, pour la fonction g au bord, j'ai choisi un simple Dirac. Sa transformée de Fourier g_k est donc 1.

Voici les courbes obtenues pour le Melamine foam :

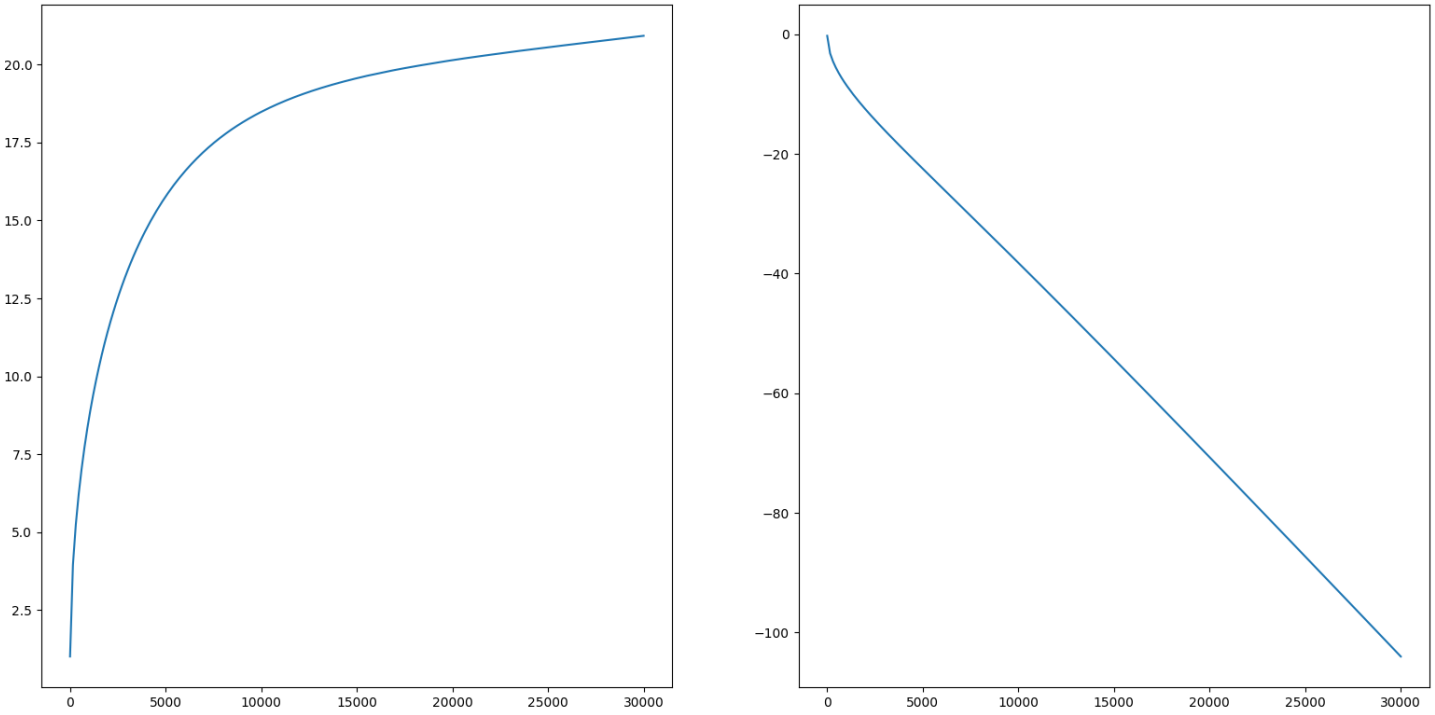


Figure 3 : partie réelle (à gauche) et partie imaginaire (à droite) du coefficient alpha en fonction de la pulsation omega

En effet, cette courbe est très similaire à celle donnée dans le cours pour le Melamine foam.

Malheureusement, je n'ai pas eu le temps d'essayer ce programme pour d'autres matériaux. Il suffirait pour cela de trouver la porosité, la tortuosité et la résistivité de différents matériaux. Cela permettrait de comparer l'efficacité de différents matériaux à absorber les ondes acoustiques.

Part III - Control of acoustic wave by changing the geometry on a surface

Fichier de code associé : part3.py

Dans cette partie, on résout l'EDP régissant la propagation du notre onde sur la maille fractale. On s'attend donc que, pour certaines fréquences, l'amplitude de l'onde est localisée à des endroits précis.

Voici ci-dessous la maille sur laquelle nous allons résoudre l'EDP. Elle a été générée par le fichier part1.py, dans lequel p_elem2nodes, elem2nodes et node_coords ont été sauvegardés dans un fichier .dat, pour ne pas avoir à relancer la génération à chaque fois. Ces matrices sont importées dans part3.py au début du code.

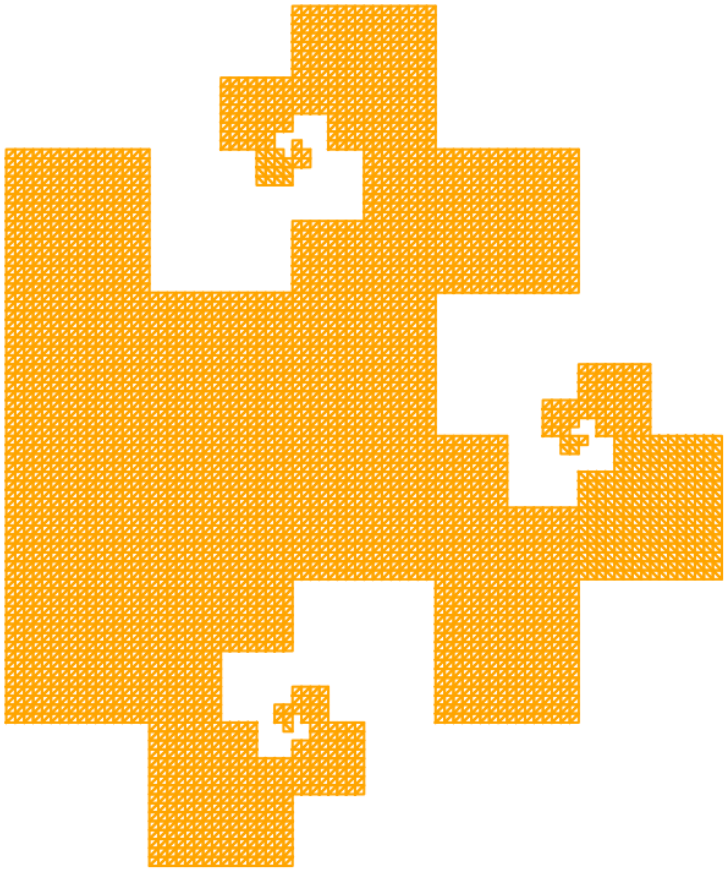


Figure 4 : maille fractale de calcul. La condition de dirichlet sera appliquée sur le bord gauche.

Nous pouvons alors appliquer une condition de dirichlet sur les nœuds situés sur le bord gauche, sans fractale. J'aurais souhaité pouvoir appliqué la condition sur une bordure fractale, mais par manque de temps, je n'ai pas codé la fonction localisant les nœuds du bord fractal.

En appliquant la condition g=1 à tous les nœuds sur le bord gauche, on peut enfin tracer la solution du problème aux EDP. Ci-dessous, la solution a été tracée pour k=pi*1.7 :

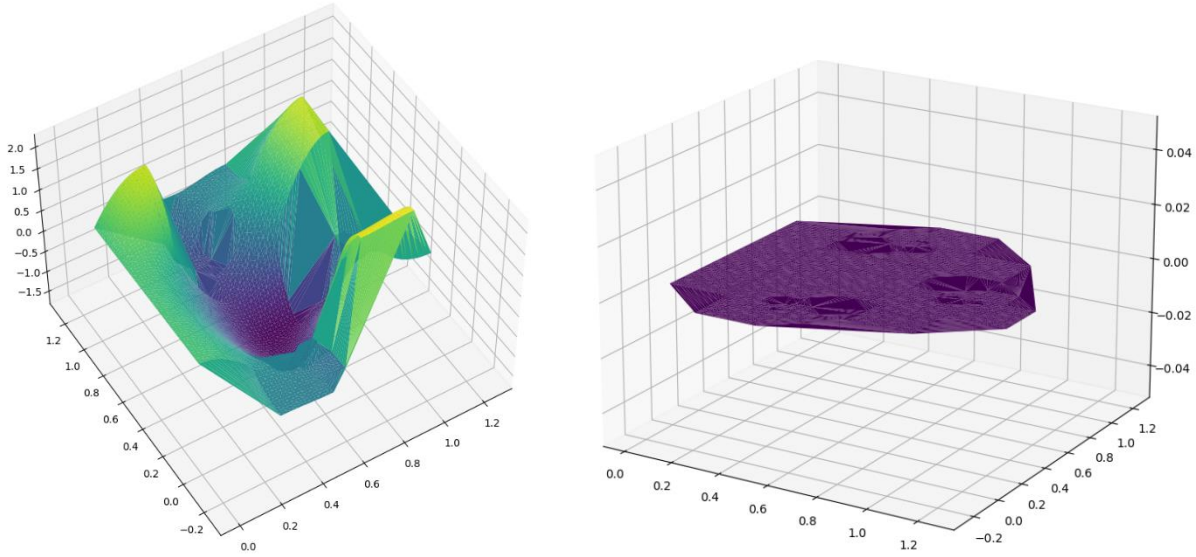


*Figure 5 : solution de l'équation aux EDP sur la maille fractale, pour k=pi*1.7. Partie réelle à gauche, imaginaire à droite*

On remarque d'abord que la partie imaginaire est nulle, ce qui est normal, puisque la partie imaginaire de la condition de dirichlet est nulle (principe de superposition). Pour ce qui est de la partie réelle, pour cette fréquence, on peut voir trois maximums d'amplitude localisés dans chaque coin de la fractale. On peut supposer qu'il s'agit là de l'effet des fractales sur chacun des trois bords : dans ce cas, la géométrie de la bordure permet bel et bien de localiser l'onde.

Cependant, lorsque k change légèrement, la solution peut changer radicalement, et parfois donner des résultats qui semblent aberrants. Je ne suis donc pas certain que mon implémentation se passe comme prévu. Cela dit, il est normal que l'efficacité du contrôle des ondes dépende de la longueur d'onde, et que pour certaines fréquences, la géométrie s'avère inefficace.

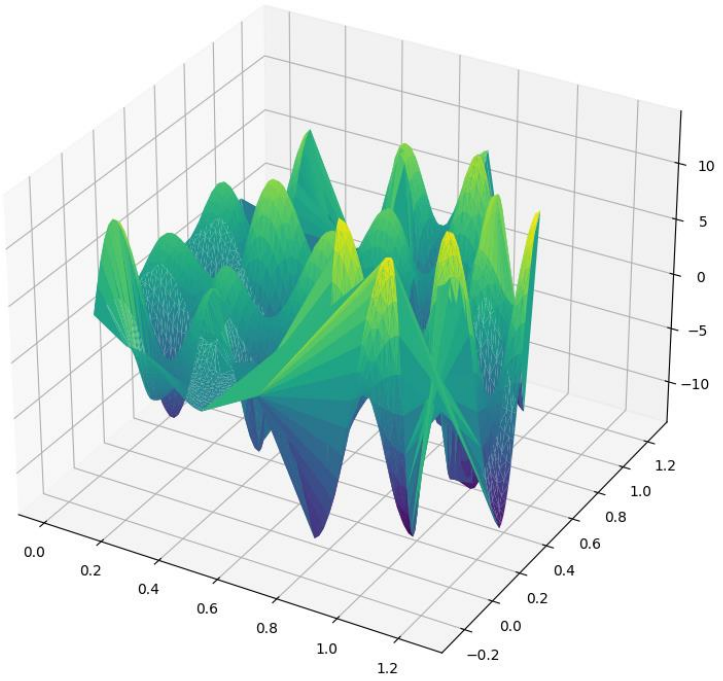


*Figure 6 : solution de l'équation aux EDP sur la maille fractale, pour k=pi*7. C'est moins concluant…*