



UNIVERSITÉ PARIS-SACLAY

CENTRALESUPÉLEC

# RAPPORT ENSEIGNEMENT D'INTÉGRATION

AMAR James - james.amar@student-cs.fr

AUDUSSE Elio - elio.audusse@student-cs.fr

BOURNEZ Pierrick - pierrick.bournez@student-cs.fr

BOUINIÈRE Thomas - thomas.bouiniere@student-cs.fr

DURAND DELBECQUE Lilou - lilou.durand@student-cs.fr

BADAOUI Abdennacer - abdennacer.badaoui@student-cs.fr

---

## POLLUTION ACOUSTIQUE ET ÉLECTROMAGNÉTIQUE

EI - ONERA

---

Professeur : Frédéric Magoules

Novembre, 2022

November 25, 2022

## Contents

<b>1</b>	<b>Introduction et objectifs</b>	<b>2</b>
1.1	Introduction . . . . .	2
1.2	Objectifs . . . . .	3
1.3	Étapes de résolution du problème . . . . .	3
<b>2</b>	<b>Étude préliminaire</b>	<b>4</b>
2.1	Définition de l'énergie . . . . .	4
2.2	Choix des matériau et coefficients d'absorption $\alpha(\omega)$ . . . . .	4
<b>3</b>	<b>Descente de gradient</b>	<b>7</b>
3.1	Principe général . . . . .	7
3.2	Choix du $\chi$ initial . . . . .	10
3.3	Résultats à fréquence fixée . . . . .	12
<b>4</b>	<b>Minimisation multi-fréquentielle</b>	<b>14</b>
4.1	Nouvelle fonction coût . . . . .	14
4.2	Influence de la quantité de matériaux . . . . .	16
<b>5</b>	<b>Conclusions et réponses au problème</b>	<b>18</b>
<b>6</b>	<b>Annexe</b>	<b>19</b>

# 1 Introduction et objectifs

## 1.1 Introduction

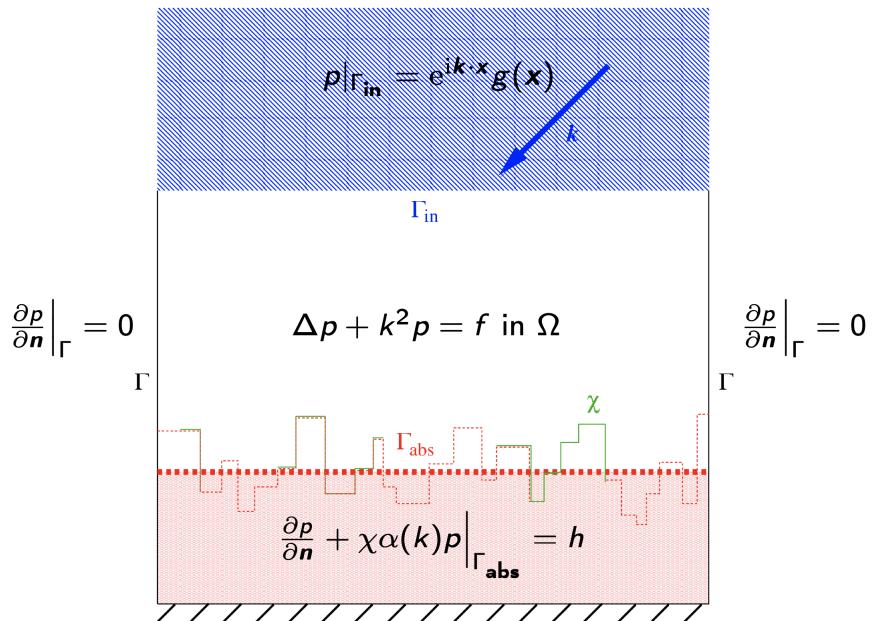
La pollution acoustique et électromagnétique est un secteur clé des sociétés modernes. En effet, les enjeux modernes tels que l'impact des portables sur la santé ou le bruit dans les grandes villes, sont au cœur des sujets de recherche. Une grande force d'innovation est nécessaire dans ce domaine pour apporter des réponses aux victimes des préjudices de quelque pollution que ce soit pour leur confort et leur santé. Afin de limiter cette pollution acoustique et électromagnétique, les acteurs de ce secteur cherchent des solutions pour réduire la transmission des ondes par l'insertion de matériaux absorbants.

L'étude que nous allons menée dans ce rapport est assez générale. Néanmoins, nous allons nous intéresser plus particulièrement à son application dans le domaine de l'aéronautique afin de minimiser le bruit d'un réacteur d'avion avec une structure de liner. Par là, nous voulons diminuer l'énergie sonore émise dans le moteur. Pour y parvenir, nous avons modélisé la situation de la manière suivante : nous nous plaçons dans un domaine rectangulaire avec des conditions de Neumann sur les bords latéraux et des conditions de Dirichlet et de Robin sur les derniers bords. L'objectif principal tout au long de ce rapport sera de trouver une fonction caractéristique  $\chi$  à valeurs dans  $\{0, 1\}$  modélisant la présence ou non du liner et de sorte que l'on minimise l'énergie  $J(\chi) = \int_{\Omega} |p_{\chi}|^2 dx$ , avec  $p_{\chi}$  la solution du système suivant :

$$\begin{cases} \Delta p + k^2 p = 0 & \text{sur } \Omega \\ p = g & \text{sur } \Gamma_{in} \\ \frac{\partial p}{\partial n} = 0 & \text{sur } \Gamma \\ \frac{\partial p}{\partial n} + \alpha \chi p = 0 & \text{sur } \Gamma_{abs} \\ \int_{\Gamma} \chi dS = \beta & 0 < \beta < \mu(\Gamma) \end{cases}$$

Nous noterons que  $\Gamma_{abs}$  dépend de  $\chi$  et donc que la solution dépend bien du placement du matériaux absorbant. Dans ce problème, la première équation correspond à un problème d'onde classique sur un domaine  $\Omega$ , les trois équations suivantes représentent les conditions aux bords de notre problème et enfin la dernière équation représente la contrainte économique de notre problème, c'est à dire le fait que l'on ne dispose que d'une quantité limitée de matériau poreux. Cette quantité de matière est non nulle et doit évidemment être strictement inférieure à la mesure du bord sur lequel elle est disposée.

On peut illustrer ce problème avec le schéma suivant avec  $h = 0$  :



## 1.2 Objectifs

Outre le fait de trouver la répartition optimale du matériau, nous avons aussi pour objectif de trouver quel matériau est le plus à même de remplir le rôle d'absorbant acoustique tout en respectant certaines contraintes comme le budget et la masse. En effet, d'un point de vue industriel, la masse de l'avion est une contrainte très importante car celle-ci va augmenter la force de traînée de l'avion, et donc augmenter la consommation de carburant pour le vol. Enfin, nous avons aussi pour objectif de trouver la géométrie du liner la plus adaptée à notre problème. En effet une géométrie fractale tend à être plus efficace en matière de localisation d'onde, mais toutefois plus compliquée à mettre en place et industrialiser. Finalement, nous proposerons plusieurs solutions en effectuant des compromis entre toutes les différentes contraintes. Nous commencerons donc par réaliser ces trois études en parallèle et, pour finir, nous mettrons en commun les différents résultats pour proposer une solution de géométrie avec une répartition du matériau optimal permettant de minimiser au mieux la pollution acoustique.

## 1.3 Étapes de résolution du problème

1. Étude préliminaires des matériaux et enjeux du problèmes
2. Minimisation de l'énergie sonore par descente de gradient
3. Comparaison des résultats expérimentaux
4. Conclusions et propositions de solutions

## 2 Étude préliminaire

### 2.1 Définition de l'énergie

Pour le moment, nous travaillons à fréquence fixée  $\omega \in I_\omega$ . L'énergie est définie de la façon suivante :

$$J(\chi) = \int_{\Omega} |p_\chi|^2 dx$$

Pour l'implémenter dans notre code, nous avons multiplié chaque valeur prise par la fonction  $p$  sur le domaine et multiplié par la surface de chaque cases du maillage :

```

1 def your_compute_objective_function(domain_omega, u, spacestep, mu1, V_0):
2     """
3         This function compute the objective function:
4         J(u, domain_omega)= \int_{domain_omega} ||u||^2
5         + mu1*(Vol(domain_omega)-V_0)
6         """
7
8     energy = 0.0
9     (M, N) = numpy.shape(domain_omega)
10    for i in range(1, M-1):
11        for j in range(1, N-1):
12            energy += u[i, j]*numpy.conjugate(u[i, j])
13    energy = energy*(spacestep**2) + mu1*(M*N*(spacestep**2)-V_0)
14    # print(numpy.real(energy))
15    return numpy.real(energy)

```

### 2.2 Choix des matériau et coefficients d'absorption $\alpha(\omega)$

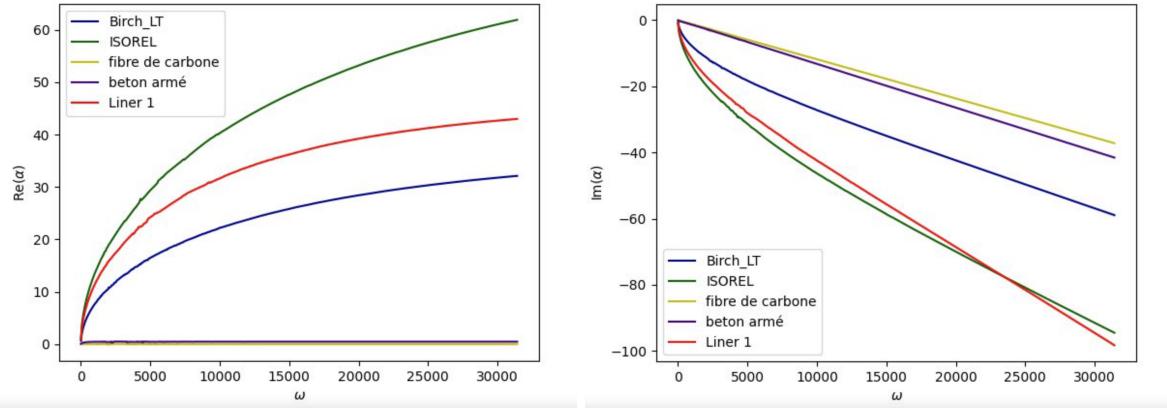
Tout d'abord, nous choisissons l'intervalle de fréquences auquel nous nous intéressons dans le cadre de notre étude. Comme nous nous plaçons dans le cadre de l'étude de la réduction sonore d'un moteur d'avion, il est nécessaire de diminuer la nuisance des fréquences audibles par l'être humains. Comme l'humain entend entre 20 Hz et 20 kHz, notre intervalle d'intérêt sera inclus dans celui-ci. De plus, nous savons que ce sont les basses fréquences qui causent les plus importants désagréments. Nous avons donc choisi l'intervalle

$$I_\omega = [50\text{Hz}, 1\text{kHz}]$$

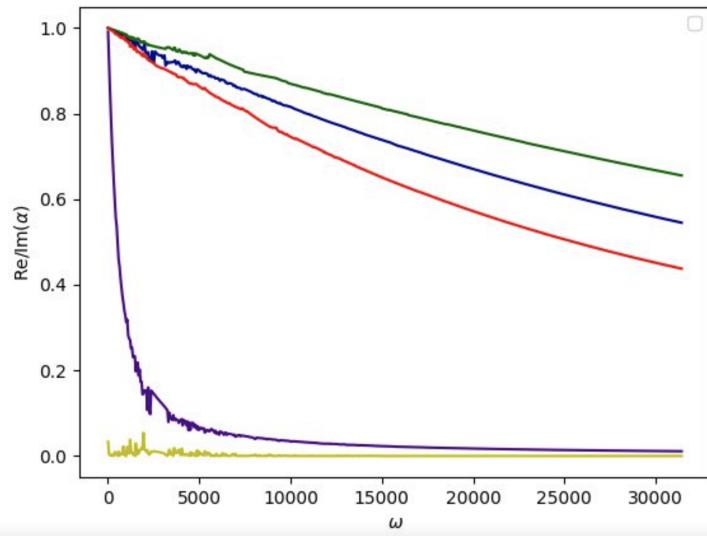
Nous pouvons remarquer que dans notre problème, la condition au bord sud de notre domaine est une condition de robin qui dépend d'un paramètre  $\alpha(\omega)$ . Ce paramètre représente l'absorption du matériau d'étude. Cette absorption va évidemment varier selon le matériau utilisé, mais également selon la fréquence d'étude  $\omega$ . Nous allons donc calculer d'abord le coefficient d'absorption  $\alpha(\omega)$  pour chaque fréquence dans l'intervalle d'étude  $I_\omega$ . Comme le matériau a vocation à être installé dans un avion, le cahier des charges nous impose un matériau léger, résistant et poreux. Nous avons décidé de comparer quatre différents matériaux: l'Isorel, les nanofibres de carbones, le bouleau et un liner utilisé dans certains types d'avions. Ces matériaux ont différentes caractéristiques intrinsèques qui vont avoir un impact sur leur capacité d'absorbance. Les grandeurs caractéristiques pour ce problèmes sont la porosité  $\Phi$ , la résistivité  $\sigma$  ainsi que la tortuosité  $\alpha_h$ .

Caractéristique	Fibre de carbone	liner d'avion	Bouleau	Isorel
$\Phi$	0.508	0.8	0.529	0.70
$\sigma$	3700	50000	151429	142300
$\alpha_h$	2.23	0.98	1.37	1.15

Nous avons donc calculé le coefficient  $\alpha(\omega)$  de ces différents matériaux sur la plage de fréquence  $I_\omega$  afin d'étudier leur absorbance. Nous avons obtenu les résultats suivants par simulations :



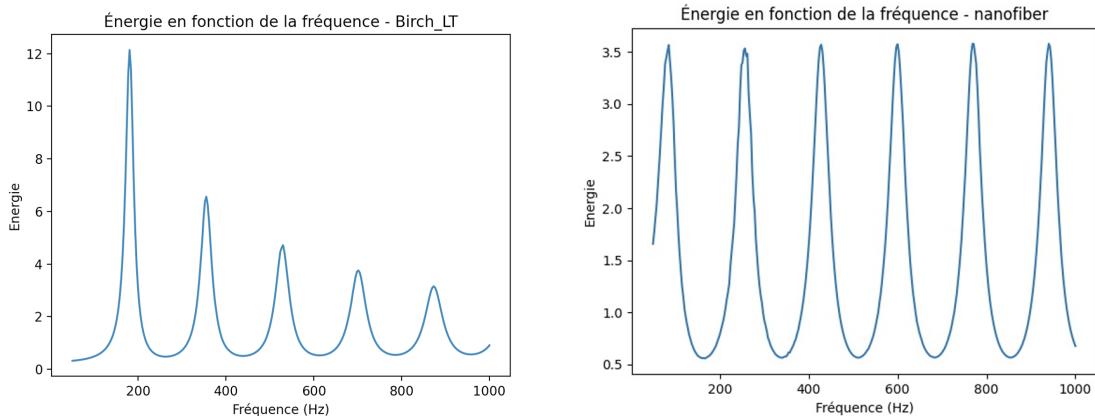
Évolution de  $\alpha$  en fonction de  $\omega$

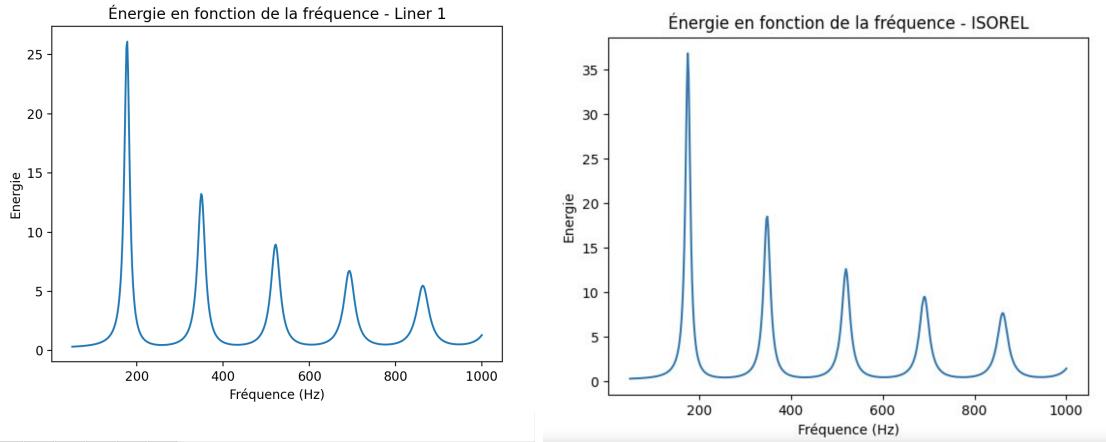


Évolution du rapport  $\frac{\text{Re}(\alpha)}{\text{Im}(\alpha)}$  des différents matériaux par rapport à la fréquence

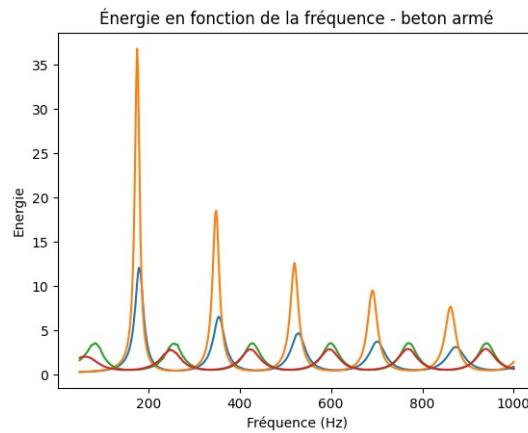
De prime abord, nous pourrions être interpellé par l'allure des différentes courbes. La compréhension des courbes passe par l'évolution de la partie réelle et imaginaire de  $\alpha$ . En effet, plus le rapport de  $\frac{\text{Re}(\alpha)}{\text{Im}(\alpha)}$  est proche de 1 plus le matériau est absorbant. C'est pour ça que la décroissance du bouleau et de l'Isorel est aussi forte. Toutefois les pics sont plus élevés pour le bouleau que pour les autres matériaux.

Afin d'aller plus loin dans l'étude de l'absorbance de ces différents matériaux, nous avons décidé de recouvrir le bord choisi rectiligne  $\Gamma$  avec chacun des matériaux et de relever l'énergie sonore en fonction de  $\omega \in I_\omega$





Énergie des différents matériaux en fonction de la fréquence



Energie des différents matériaux en fonction de la fréquence.

l'orange correspond à l'isorel, bleu est le liner d'avion, le vert est le nanofibre et le rouge est le bouleau

Nous voulons choisir le matériaux avant de faire des analyses plus poussées sur le système de l'avion. Nous savons que la réduction des ondes par le choix de la frontière est très efficace pour une fréquence donnée ou pour une intervalle de fréquence fixe. Le rêve est donc d'avoir une onde très localisé sur une bande de fréquence et qui s'atténue très vite quand la fréquence augmente. Comme la nanofibre n'absorbe pas l'énergie lorsque la fréquence augmente, nous pouvons donc écarter le choix de la nanofibre.

Un autre critère est le poids, le bouleau a une très bonne localisation autour de 200 hertz mais est beaucoup trop lourd pour rentrer dans un avion, nous avons donc écarter cette possibilité. Il ne reste plus que deux matériaux: le liner 1 et isorel. Les deux ont un pic très fort autour de 200 hertz et une décroissance rapide pour les fréquences supérieurs, nous choisissons arbitrairement le liner 1 qui a un pic fondamentale plus faible et étant implémenter en pratique sur de vrais avions, son poids doit être relativement acceptable. Nous allons alors faire une étude plus poussée de l'absorption en fonction du niveau de fractale pour le matériaux de liner d'avion.

### 3 Descente de gradient

#### 3.1 Principe général

A matériau, fréquence et quantité de matériau fixée, nous pouvons nous questionner sur la répartition optimale de celui-ci, cela revient à déterminer  $\chi$  tel que l'énergie  $J(\chi)$  de la solution du problème d'Helmotz soit minimale. Afin de minimiser cette fonction, nous allons utiliser une méthode classique : la descente de gradient projetée. Cette méthode est assez simple, elle consiste en un choix d'une valeur initiale de  $\chi$  (nous discuterons du choix de cette valeur par la suite), puis à chaque itération, on retranche à la valeur de  $\chi$  la dérivée de Fréchet de notre énergie multipliée par un coefficient  $\mu$  qui représente le pas de notre descente de gradient projetée qu'on projette dans l'ensemble des solutions recherché.

Nous avons donc besoin de l'expression de la dérivée de  $J(\chi)$  qui a pour expression dans notre cas

$$J'(\chi) = -Re(\alpha p_\chi q_\chi) \quad (1)$$

où  $q_\chi$  est la solution du problème adjoint ( $P^*\chi$ ) suivant :

$$\begin{cases} \Delta q + k^2 q = -2\bar{p}_\chi & \text{sur } \Omega \\ q = 0 & \text{sur } \Gamma_{in} \\ \frac{\partial q}{\partial n} = 0 & \text{sur } \Gamma \\ \frac{\partial q}{\partial n} + \alpha \chi q = 0 & \text{sur } \Gamma_{abs} \end{cases}$$

Cette descente de gradient est codée selon le programme suivant :

```

 $\chi^{(0)} \in U_{ad}(\beta); \mu = \mu^{(0)};$ 
for  $k = 0$  :  $K$  do
    compute  $p^{(k)}$ ;  $q^{(k)}$ ;  $J(\chi^{(k)})$ ;  $J'(\chi^{(k)})$ ;
     $E = J(\chi^{(k)})$ ;
    while  $E \geq J(\chi^{(k)})$  &  $\mu > \epsilon_0$  do
         $\ell = 0$ ;
         $\chi^{(k+1)} = \mathcal{P}_\ell [\chi^{(k)} - \mu J'(\chi^{(k)})]$ ;
        while  $|\int_{\Gamma_{abs}} \chi^{(k+1)} dS - \beta| \geq \epsilon_1$  do
            if  $\int_{\Gamma_{abs}} \chi^{(k+1)} dS \geq \beta$  then
                 $\ell \leftarrow \ell - \epsilon_2$ ;
            else
                 $\ell \leftarrow \ell + \epsilon_2$ ;
            end if
             $\chi^{(k+1)} = \mathcal{P}_\ell [\chi^{(k)} - \mu J'(\chi^{(k)})]$ ;
        end while
        compute  $p^{(k+1)}$ ;  $J(\chi^{(k+1)})$ ;
         $E = J(\chi^{(k+1)})$ ;
        if  $E < J(\chi^{(k)})$  then
             $\mu \leftarrow \mu + \epsilon_3$ ;
        else
             $\mu \leftarrow \mu / 2$ ;
        end if
    end while
end for

```

Figure 2 : Pseudo-code de la descente de gradient

Il est important de noter que notre descente de gradient s'effectue pour des fonctions  $\chi$  appartenant non pas à l'ensemble des fonctions caractéristiques  $U_{ad}(\beta)$  mais sur sa fermeture algébrique pour la convergence faible étoile  $U_{ad}^*(\beta)$ . Avec,

$$\begin{aligned} U_{ad}(\beta) &= \{\chi \in L^\infty \mid \forall x \in \Gamma, \forall \chi(x) \in \{0, 1\}, 0 < \int_\Gamma \chi dS = \beta < \mu(\Gamma)\} \\ U_{ad}^*(\beta) &= \{\chi \in L^\infty \mid \forall x \in \Gamma, \forall \chi(x) \in [0, 1], 0 < \int_\Gamma \chi dS = \beta < \mu(\Gamma)\} \end{aligned} \quad (2)$$

En effet, il est cohérent de chercher un minimum de  $J(\chi)$  pour  $\chi \in U_{ad}^*(\beta)$  car  $J(\chi)$  est une fonction continue au sens la convergence faible étoile sur  $U_{ad}^*(\beta)$ . Et de plus,  $U_{ad}^*(\beta)$  est un compact au sens de la convergence faible étoile. C'est pourquoi  $J$  atteint ses bornes sur cet ensemble et atteint en particulier son minimum.

Bien que la méthode théorique de la descente de gradient semble aisée, il y a tout de même plusieurs subtilités auxquelles il nous faut prêter attention : La première est que nous cherchons des solutions sur l'ensemble particulier  $U_{ad}^*$  et en faisant la descente de gradient naïvement, les valeurs de la fonction  $\chi$  peuvent sortir de l'intervalle  $[0, 1]$ . Comme l'ensemble des valeurs est un intervalle, l'algorithme le plus simple de minimisation sur un ensemble convexe est celui du gradient projeté car son projecteur  $P_l$  afin de forcer les valeurs prises par  $\chi$  à rester dans l'intervalle  $[0, 1]$  est :

$$P_l(\chi) = \max(0, \min(\chi + l, 1)) \quad (3)$$

```

1 def P_l(l, chi):
2     M, N = numpy.shape(chi)
3     mat_min = numpy.zeros((M, N), dtype=numpy.float64)
4     for i in range(M):
5         for j in range(N):
6             element = (chi+l*numpy.ones((M, N)))[i, j]
7             mat_min[i, j] = min(element, 1)
8     mat_max = numpy.zeros((M, N), dtype=numpy.float64)
9     for i in range(M):
10        for j in range(N):
11            mat_max[i, j] = max(0, mat_min[i, j])
12
13 return mat_max

```

La seconde subtilité est que le nouveau  $\chi$  obtenu à chaque itération peut ne plus respecter la contrainte de budget imposée dans notre problème, et ainsi il faut trouver une nouvelle valeur proche mais dont l'intégrale sur la frontière est bien égale à  $\beta$  à epsilon près. Pour se faire, nous avons utilisé une méthode par approche dichotomique afin de trouver le meilleur  $\chi$  permettant de répondre aux contraintes de matériau. Voici la manière dont cette approche a été codée :

```

1 def compute_projected(chi, domain, V_obj):
2     """This function performs the projection of $\chi^n - mu*grad
3
4     (M, N) = numpy.shape(domain)
5     S = 0
6     for i in range(M):
7         for j in range(N):
8             if domain[i, j] == _env.NODE_ROBIN:
9                 S = S + 1
10
11    B = chi.copy()
12    l = 0
13    chi = processing.set2zero(chi, domain)
14
15    V = numpy.sum(numpy.sum(chi)) / S
16    debut = -numpy.max(chi)
17    fin = numpy.max(chi)
18    ecart = fin - debut
19    # We use dichotomy to find a constant
20    # such that
21    # chi^{n+1} = max(0, min(chi^{n}+l, 1)) is an element
22    # of the admissible space
23    while ecart > 10 ** -4:
24        # calcul du milieu
25        l = (debut + fin) / 2
26        for i in range(M):
27            for j in range(N):
28                chi[i, j] = numpy.maximum(0, numpy.minimum(B[i, j] + l, 1))
29        chi = processing.set2zero(chi, domain)
30        V = sum(sum(chi)) / S
31        if V > V_obj:

```

```

32         fin = l
33     else:
34         debut = l
35     ecart = fin - debut
36     # print('le volume est', V, 'le volume objectif est', V_obj)
37
38 return chi

```

Enfin, une fois la descente de gradient effectuée, nous obtenons un  $\chi$  à valeurs dans  $[0,1]$  car  $\chi \in U_{ad}^*(\beta)$ . Il est donc nécessaire de projeter cette fonction dans  $U_{ad}(\beta)$ . Pour cela, il est nécessaire de modifier les plus grandes valeurs pour qu'elles valent 1 et que les autres valent 0. Il faut toutefois faire attention à la quantité de matériau qui doit correspondre à la quantité attendue. Pour cela nous avons codé la fonction suivante :

```

1 def projected_0_1(chi, N):
2     '''N est le nombre de points valant 1, permet d'obtenir un
3     nouveau chi d'integrale sur Gamma égale beta'''
4     m,n = numpy.shape(chi)
5     liste_values = []
6     for i in range(m):
7         for j in range(n):
8             liste_values.append(((i,j),chi[i,j]))
9     liste_sorted = sorted(liste_values, key=lambda x:x[1])
10    indices=liste_sorted[-N:]
11    indices_sorted = []
12    for x,_ in indices:
13        indices_sorted.append(x)
14    new_chi = []
15    for i in range(m):
16        for j in range(n):
17            if (i,j) in indices_sorted:
18                new_chi.append(1)
19            else:
20                new_chi.append(0)
21    return numpy.array(new_chi).reshape(m,n)

```

### 3.2 Choix du $\chi$ initial

Un des problèmes de la descente de gradient est qu'il est possible pour un mauvais choix de valeur initiale de rester bloqué sur un minimum local de la fonction que l'on souhaite minimiser (voir figure ci-dessous). Il existe plusieurs façons de résoudre ce genre de problème, et la solution la plus simple est de tester le programme avec plusieurs valeurs initiales différentes.

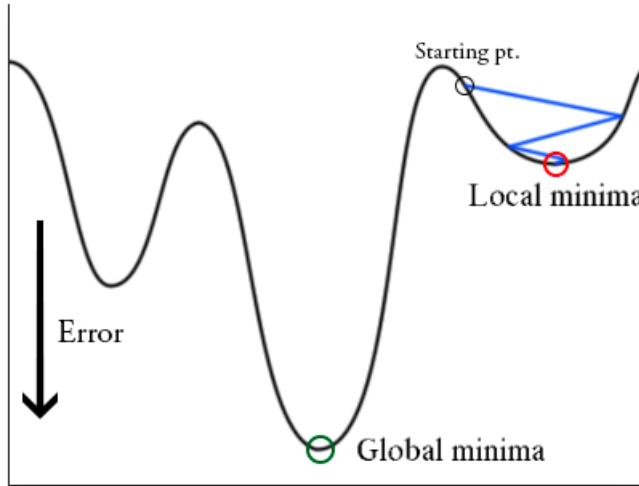


Figure 3 : Minimum local et minimum global

Dans le code qui nous a été fourni,  $\chi_0$  est initialisé de façon à remplir le tronçon de 1/5 à 3/5 de notre frontière. Cela peut donc être intéressant car ce tronçon est plutôt centré mais ce choix pose néanmoins plusieurs problèmes.

Tout d'abord, le choix d'un  $\chi$  initial en un seul morceau rend plus difficile l'obtention d'un  $\chi$  final morcelé. Cela peut être intéressant pour des contraintes industrielles mais peut nous empêcher de découvrir des minimums globaux. De plus, ce choix implique une valeur de  $\beta$  (notre contrainte budgétaire) plus grande en fonction du niveau de notre fractale. Cela ne semble pas très réaliste car il n'y a pas de raison d'avoir un budget plus grand pour un niveau de fractale différent, au contraire, il serait plus intéressant de regarder si certains niveaux de fractale trop élevés sont obsolètes à cause d'un budget trop faible.

Pour ce faire, nous avons donc écrit deux nouvelles fonctions (voir en annexe). La première, `_set_chi_alea`, prend en argument deux nouveaux paramètres : un entier  $n$  qui correspond au nombre de cases où l'on veut avoir une valeur de  $\chi$  égale à 1 et un booléen "alea" qui indique si l'on veut ou non placer le  $\chi$  initial aléatoirement. Cette fonction permet donc d'avoir une contrainte budgétaire constante en fonction du niveau de la fractale, néanmoins elle nécessite quelques modifications dans le code et il nous faut également connaître la taille de la frontière pour en déduire le  $n$  maximal.

La seconde, `_set_chi_n`, fonction prends également en argument un entier  $n$  mais cette fois-ci représente le nombre de divisions que l'on souhaite sur notre frontière. La fonction initialise alors  $\chi$  de sorte à ce qu'un segment sur deux possède de l'isolant.

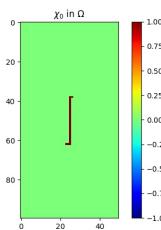


Figure 1: `_set_alpha_alea 1`

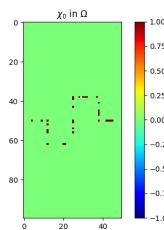


Figure 2: `_set_alpha_alea 2`

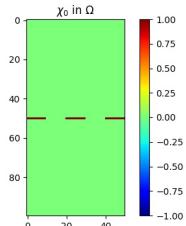


Figure 3: `_set_alpha_n` avec comme paramètre 3

Voici donc comment la distribution finale de  $\chi$  change en fonction de sa distribution initiale :

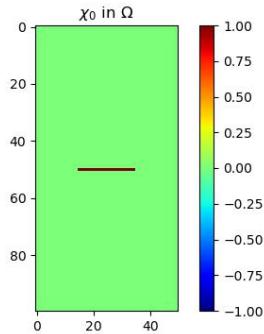


Figure 4:  $\chi_0$  (\_set\_chi\_alea n=20 )

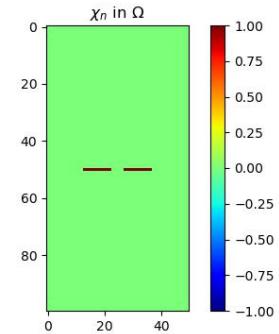


Figure 5:  $\chi_n$  (\_set\_chi\_alea n=20 )

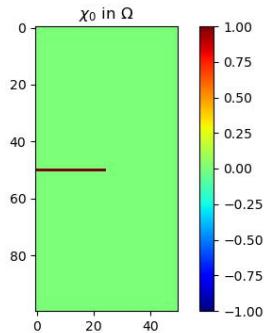


Figure 6:  $\chi_0$  (\_set\_chi\_n n=2 )

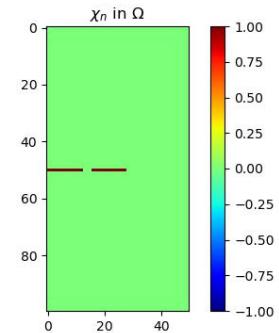


Figure 7:  $\chi_n$  (\_set\_chi\_n n=2 )

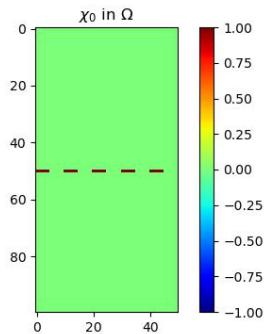


Figure 8:  $\chi_0$  (\_set\_chi\_n n=10 )

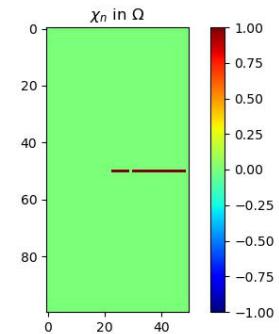


Figure 9:  $\chi_n$  (\_set\_chi\_n n=10 )

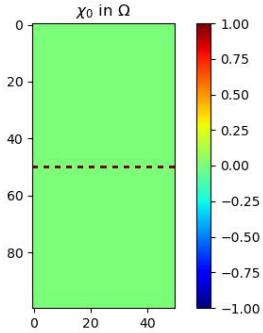


Figure 10:  $\chi_0$  (\_set\_chi\_n n=25 )

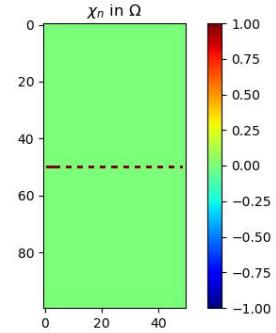


Figure 11:  $\chi_n$  (\_set\_chi\_n n=25 )

Nous pouvons donc observer qu'en fonction des conditions initiales de la descente de gradient, nous obtenons des résultats de  $\chi$  différents, et donc des niveaux d'énergies différents. Afin de s'assurer de trouver un minimum global de l'énergie, il est donc judicieux d'effectuer la même descente de gradient avec différentes conditions initiales.

### 3.3 Résultats à fréquence fixée

Dans cette section, nous nous intéressons à la minimisation de l'énergie pour une fréquence donnée. Pour savoir à quelle fréquence il est important de se placer, nous avons tracé l'énergie de la solution à l'équation d'Helmholtz pour les fréquences  $\omega \in I_\omega$ . Nous avons vu à l'aide de la figure précédente que les énergies en fonction de la fréquence présentent certains pics pour des fréquences précises. Théoriquement, puisque nous considérons un rectangle de taille  $ab$ , les fréquences de résonances sont de la forme

$$f_{m,n} = \frac{c_0}{2} \sqrt{\frac{m^2}{a^2} + \frac{n^2}{b^2}}, \quad (m, n) \in \mathbb{N}^2 \quad (4)$$

Nous avons alors vérifié la coïncidence de ces fréquences théoriques avec les fréquences expérimentalement trouvées avec nos simulations. Comme les pics fondamentaux sont déjà très atténus, nous nous concentrerons à la réduction de l'énergie du mode fondamental. Par la suite nous avons décidé de vous présenter uniquement les résultats obtenus pour le liner 1 qui est le matériau qui s'avère être le plus intéressant. La fréquence utilisée ici est de 100 Hz. Voici les résultats que nous avons obtenus :

- Niveau 0 :

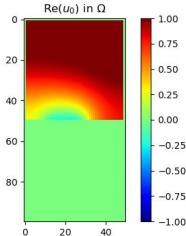


Figure 12:  $\text{Re}(u_0)$

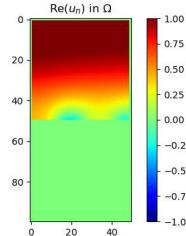


Figure 13:  $\text{Re}(u_n)$



Figure 14: Energie

- Niveau 1 :

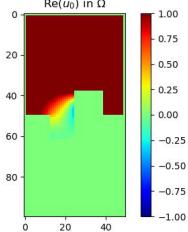


Figure 15:  $\text{Re}(u_0)$

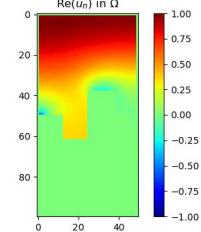


Figure 16:  $\text{Re}(u_n)$

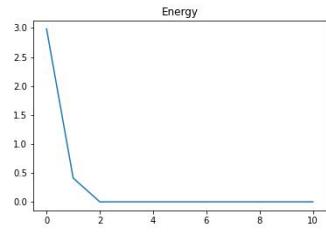


Figure 17: Energie

- Niveau 2 :

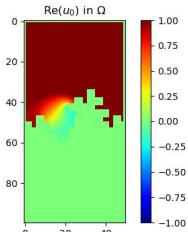


Figure 18:  $\text{Re}(u_0)$

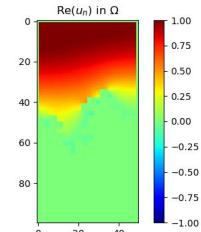


Figure 19:  $\text{Re}(u_n)$

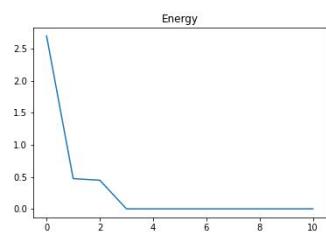
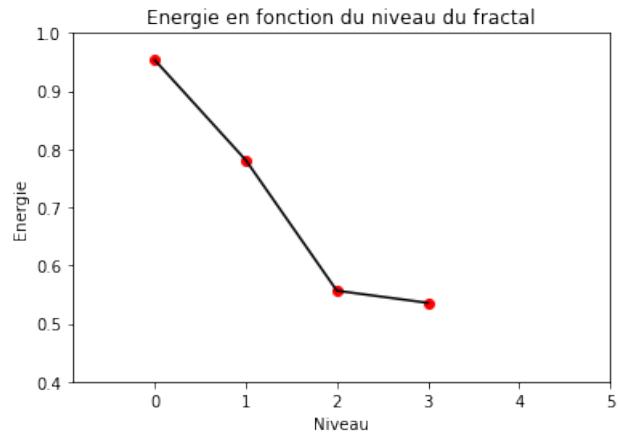


Figure 20: Energie



Nous pouvons d'abord observer que l'énergie finale diminue avec l'augmentation du niveau de fractale. Ceci est un résultat auquel nous nous attendions car comme étudié en cours, les bords préfactales avec un niveau plus grand permet de mieux localiser les ondes et de diminuer l'énergie. Cependant, il est intéressant de chercher un compromis entre réduction de l'énergie et simplicité du bord. En effet, dans une application industrielle, il est nécessaire de limiter les coups financiers et la complexité de fabrication tout en maximisant la réduction acoustique. Ici, nous observons donc que la réduction de l'énergie est relativement faible entre le niveau de fractale 2 et le niveau 3, alors que le motif de niveau 2 est bien plus simple que le niveau 3. Il semble donc convenable dans cette situation d'opter pour un bord fractale de niveau 2. N'ayant pas de capacité de calcul suffisant, nous n'avons pas pu pousser le calcul de l'énergie à des niveaux plus grands que 4.

## 4 Minimisation multi-fréquentielle

### 4.1 Nouvelle fonction coût

Dans cette partie nous nous intéressons à la réduction sonore d'une plage de fréquences. En effet, nous avons observé dans les parties précédentes que l'énergie était maximale pour différentes fréquences  $\omega \in O_{max}^{(n)}$ . Où  $O_{max}^{(n)}$  est l'ensemble des n premières fréquences pour lesquelles l'énergie est maximale. Nous définissons alors une nouvelle fonction coût  $J_{tot}(\chi)$

$$J_{tot}(\chi) = \sum_{\omega \in O_{max}^{(n)}} J(\chi, \omega) = \sum_{\omega \in O_{max}^{(n)}} \int_{\Omega} |p_{\chi}(\omega)|^2 dx \quad (5)$$

Cette fonction a été codée de la façon suivante :

```

1 def compute_objective_function_multirange(domain_omega, u_range, spacestep,
2                                         mu1, V_0):
3     return sum([your_compute_objective_function(
4         domain_omega, u, spacestep, mu1, V_0) for u in u_range])

```

Nous avons adapté la méthode de descente de gradient présentée et utilisée précédemment afin de minimiser cette nouvelle fonction de coût. Nous avons effectué cette nouvelle descente de gradient et avons obtenu les résultats suivants avec  $n = 5$  le nombre de pics minimisés:

**Pour le Liner 1:** • Niveau de fractale 0 :

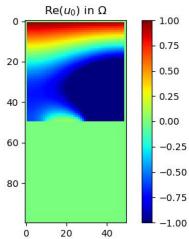


Figure 21:  $Re(u_0)$  avec  $f = 175$  Hz

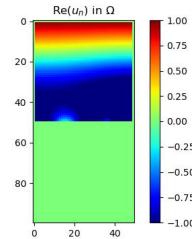


Figure 22:  $Re(u_n)$  avec  $f = 175$  Hz

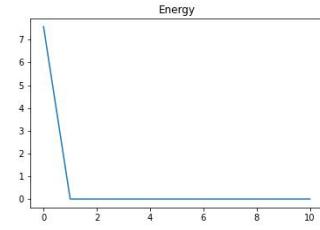


Figure 23: Energie

• Niveau de fractale 1 :

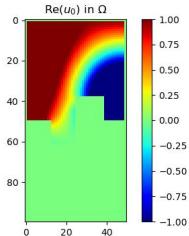


Figure 24:  $Re(u_0)$  avec  $f = 175$  Hz

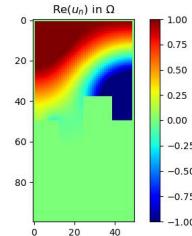


Figure 25:  $Re(u_n)$  avec  $f = 175$  Hz

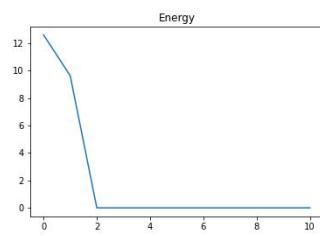


Figure 26: Energie

• Niveau de fractale 2 :

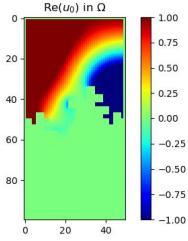


Figure 27:  $\text{Re}(u_0)$  avec  $f = 175$  Hz      Figure 28:  $\text{Re}(u_n)$  avec  $f = 175$  Hz

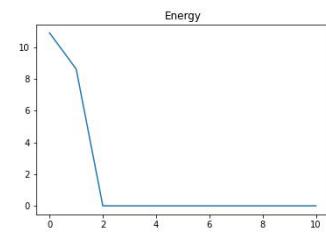
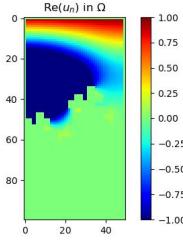


Figure 29: Energie

**Pour l'Isorel:** • Niveau de fractale 0 :

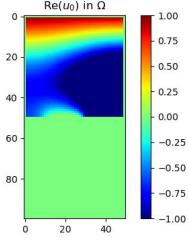


Figure 30:  $\text{Re}(u_0)$  avec  $f = 175$  Hz      Figure 31:  $\text{Re}(u_n)$  avec  $f = 175$  Hz

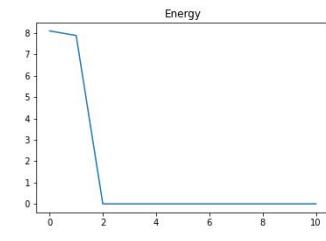
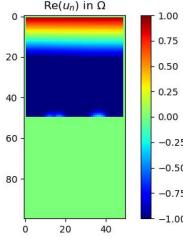


Figure 32: Energie

• Niveau de fractale 1 :

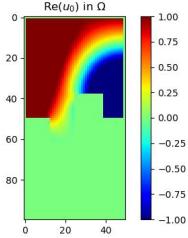


Figure 33:  $\text{Re}(u_0)$  avec  $f = 175$  Hz      Figure 34:  $\text{Re}(u_n)$  avec  $f = 175$  Hz

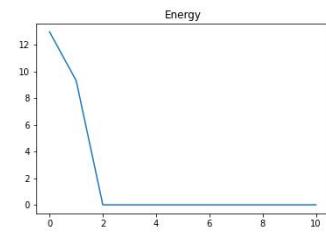
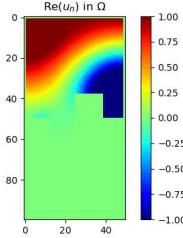


Figure 35: Energie

• Niveau de fractale 2 :

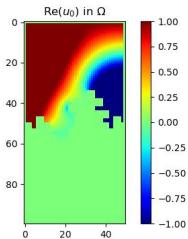


Figure 36:  $\text{Re}(u_0)$  avec  $f = 175$  Hz      Figure 37:  $\text{Re}(u_n)$  avec  $f = 175$  Hz

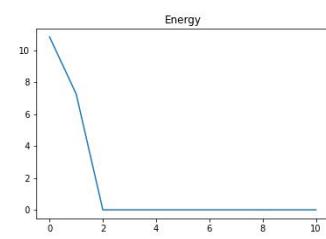
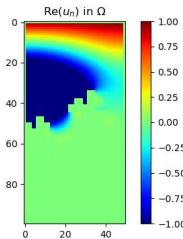


Figure 38: Energie

**Interprétations** Nous pouvons donc observer que notre descente de gradient fonctionne avec plusieurs fréquences puisque l'énergie diminue bien. Nous pouvons aussi observé que l'énergie diminue lorsque l'on augmente le niveau de fractale ce qui correspond aux attentes théoriques.

## 4.2 Influence de la quantité de matériaux

Maintenant que nous avons étudié les différentes méthodes de minimisation de l'énergie, que ça soit à fréquence unique ou avec plusieurs fréquences, nous allons nous intéresser à l'influence de la quantité de matériau  $\beta$ .

- Pour  $\beta = 0.2$  :

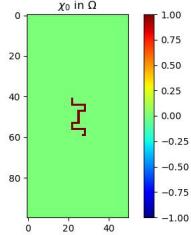


Figure 39:  $\chi_0$  avec  $\beta = 0.2$

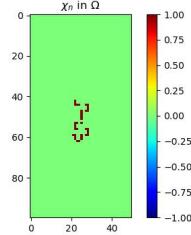


Figure 40:  $\chi_n$  avec  $\beta = 0.2$

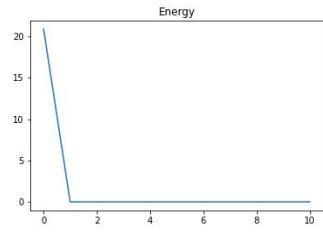


Figure 41: Energie

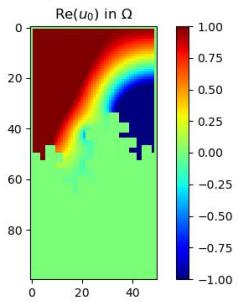


Figure 42:  $Re(u_0)$  avec  $f = 175$  Hz

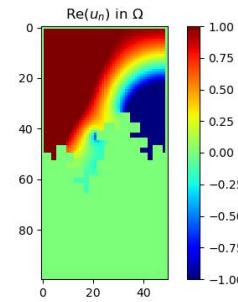


Figure 43:  $Re(u_n)$  avec  $f = 175$  Hz

- Pour  $\beta = 0.4$  :

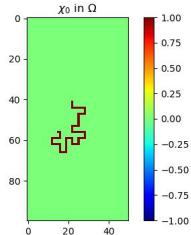


Figure 44:  $\chi_0$  avec  $\beta = 0.4$

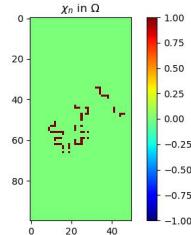


Figure 45:  $\chi_n$  avec  $\beta = 0.4$

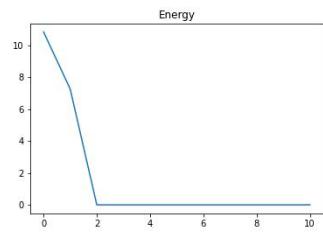


Figure 46: Energie

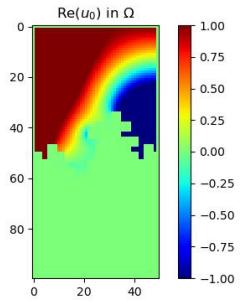


Figure 47:  $Re(u_0)$  avec  $f = 175$  Hz

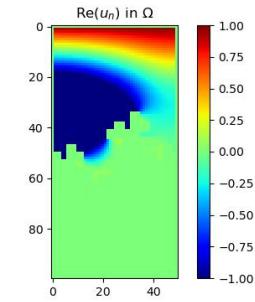


Figure 48:  $Re(u_n)$  avec  $f = 175$  Hz

- Pour  $\beta = 0.6$  :

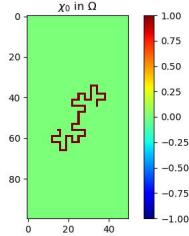


Figure 49:  $\chi_0$  avec  $\beta = 0.6$

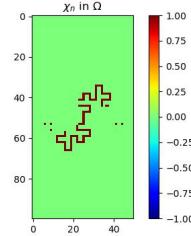


Figure 50:  $\chi_n$  avec  $\beta = 0.6$

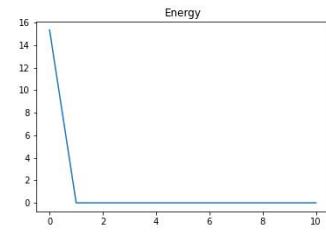


Figure 51: Energie

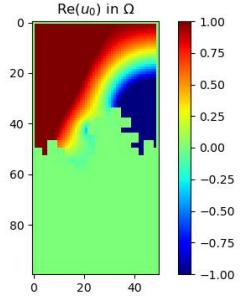


Figure 52:  $Re(u_0)$  avec  $f = 175$  Hz

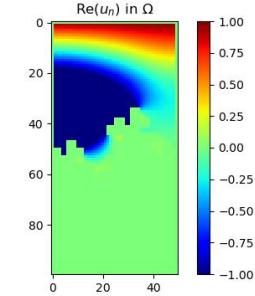


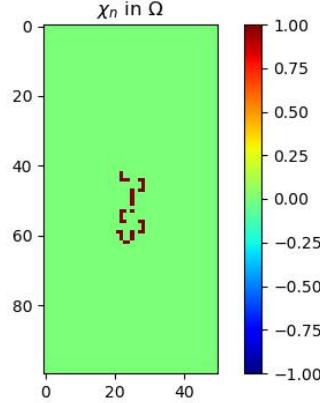
Figure 53:  $Re(u_n)$  avec  $f = 175$  Hz

**Interprétations** Nous pouvons observer que bien évidemment, l'énergie diminue lorsque la quantité de matériaux augmente. Pour parvenir à un choix de cette quantité, il faudra donc évaluer le coût de la présence du matériau. En effet, il y a le coût de la matière première qui entre en compte mais aussi la masse de celui-ci.

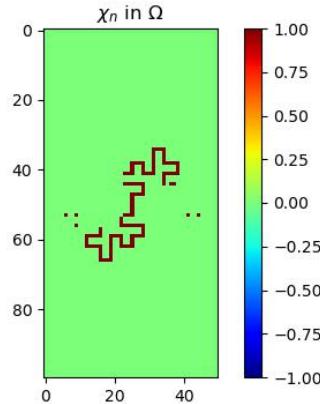
## 5 Conclusions et réponses au problème

Au regard de toute notre étude, nous sommes maintenant en capacité de vous proposer deux offres pour le revêtement de votre avion.

**Proposition 1** Cette proposition est le choix relativement économique, nous optons pour l'Isorel avec une quantité de matériau  $\beta = 0,2$ . L'Isorel a divers avantages: il est peu couteux 6,38 euros le kilogramme, léger 230kg/m<sup>3</sup>. Nous avons opté pour la répartition fractale de niveau 2 car facilement industrialisable. En termes de performances, l'énergie est diminuée de 28%.



**Proposition 2** Cette seconde proposition a un coût plus élevé mais ce prix est justifié par des performances bien meilleures. Nous optons ici pour un revêtement déjà utilisé dans plusieurs moteurs de la NASA qui a donc des capacités d'atténuations de bruit très avancées. Cependant, ce matériau est plus lourd 1600kg/m<sup>3</sup> et plus cher 500 euros le kilogramme. Nous proposons la répartition suivante avec un bord fractale de niveau 2 et une quantité de matériau importante  $\beta = 0,6$ . L'énergie est ici 2 fois plus basse que dans la proposition précédente.



## 6 Annexe

```
1 def _set_chi_alea(M, N, x, y, n, alea = True):
2     """
3         Initialise le chi_0,
4         n : nombre de case o    l'on veut mettre de l'isolant
5         """
6         chi = numpy.zeros((M, N), dtype=numpy.float64)
7         val = 1.0
8         if alea :
9             list_choice = numpy.zeros(len(x), dtype = int)
10            for i in range(len(x)) :
11                list_choice[i] = i
12            print("list_choice:", list_choice)
13            for i in range(n) :
14                rand = numpy.random.choice(list_choice)
15                list_choice = numpy.setdiff1d(list_choice, rand)
16                chi[int(y[rand]), int(x[rand])] = val
17        else :
18            for k in range(len(x)//2 - n//2, len(x)//2 + n//2):
19                chi[int(y[k]), int(x[k])] = val
20
21        return chi
22
23 def _set_chi_n(M, N, x, y, n):
24     """
25         Initialise chi avec des segments de taille 1/n
26         n : Division demand e (max 50 pour niveau 1)
27         """
28         val =1.0
29         chi = numpy.zeros((M, N), dtype=numpy.float64)
30         print(len(x)//n)
31         for i in range((n+1)//2):
32             for k in range(2*i*len(x)//n, (2*i+1)*(len(x)//n)):
33                 print("k:", k)
34                 chi[int(y[k]), int(x[k])] = val
35
36         return chi
```