Rapport de projet de physique numérique

Département de Physique de l'ENS

Modélisation bidimensionnelle des instruments à vent

Elèves :
Elie Gouzien
Sophie Marbach

Supervision :
Emmanuel Dormy
Avec l'aimable participation de
Ludivine Oruba

Novembre 2011

Table des matières

Introduction

Qui ne s'est pas demandé comment cela était possible qu'un instrument aussi simple que la flûte - littéralement un bâton de bois creusé, à peine taillé, avec quelques trous - pouvait faire des sons aussi doux et aussi timbrés ? Finalement, n'est-ce pas surprenant qu'en mettant ses doigts n'importe où, on crée des sons aussi différents, juste en soufflant ?

Notre projet a le but clairement ambitieux mais extrêmement enrichissant de comprendre pourquoi en soufflant dans une flûte, on peut créer des sons et notamment des sons aussi riches.

Nous expliquerons d'abord notre démarche de numérisation. Nous survoleront ensuite les premières tentatives de modélisations d'une flûte simple, puis nous évoluerons assez rapidement vers le modèle final de numérisation, qui nous a servi pour les exploitations futures. Ce programme est capable de prendre des instruments très divers et aux formes relativement abouties (s'approchant notamment d'une flûte traversière et d'une clarinette) et de regarder ce qui s'y passe quand on envoie le son que l'on veut dedans. On finera par essayer de tracer le spectre des deux instruments que nous auront créés.

I Modèle de modélisation 2D de propagation du son

I.1 Grille de modélisation

On décide d'effectuer la modélisation sur les variables de pression et de vitesse à la fois. On propose donc de définir dans l'espace une grille principale sur laquelle chaque noeud représente une valeur de pression, et intercalée, deux grilles secondaires correspondant aux vitesses verticales et horizontales, comme illustré sur la figure 1.

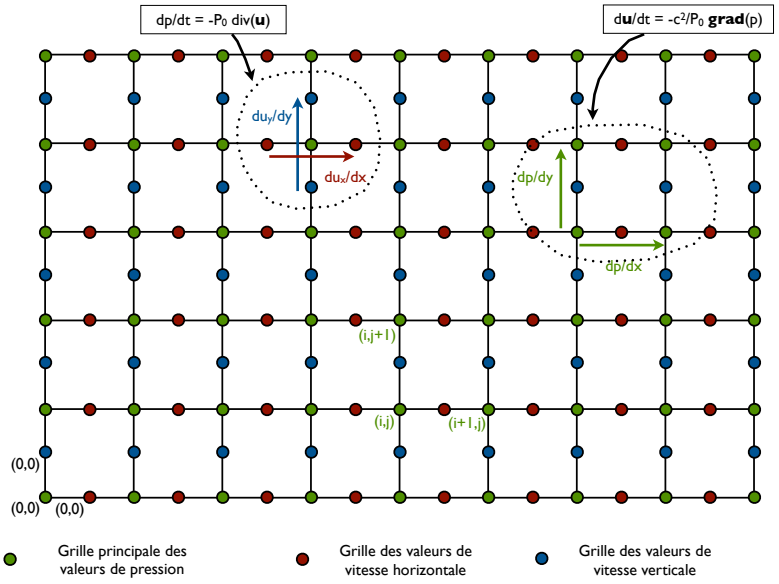

FIGURE 1 – Grille de modélisation 2D

L'étude linéaire des équations de propagation des ondes dans l'air donne les deux formules de propagation suivantes :

$$\begin{cases} \dfrac{\partial \vec{u}}{\partial t} = -\dfrac{c^2}{P_0}\overrightarrow{grad}(p) \\ \dfrac{\partial p}{\partial t} = -P_0 div(\vec{u}) \end{cases}$$

Nous choisissons alors le schéma explicite suivant, qui est résumé sur la figure, avec n désignant les indices temporels :

$$\begin{cases} \dfrac{p_{(i,j)}^{(n+1)} - p_{(i,j)}^{(n)}}{\Delta t} = -P_0 \left(\dfrac{u_{x(i,j)}^{(n)} - u_{x(i-1,j)}^{(n)}}{\Delta x} + \dfrac{u_{y(i,j)}^{(n)} - u_{y(i,j-1)}^{(n)}}{\Delta y} \right) \\ \dfrac{u_{x(i,j)}^{(n+1)} - u_{x(i,j)}^{(n)}}{\Delta t} = -\dfrac{c^2}{P_0} \left(\dfrac{p_{(i+1,j)}^{(n+1)} - p_{(i,j)}^{(n+1)}}{\Delta x} \right) \\ \dfrac{u_{y(i,j)}^{(n+1)} - u_{y(i,j)}^{(n)}}{\Delta t} = -\dfrac{c^2}{P_0} \left(\dfrac{p_{(i,j+1)}^{(n+1)} - p_{(i,j)}^{(n+1)}}{\Delta y} \right) \end{cases}$$

I.2 Ordre et stabilité du schéma

Ordre du schéma

Soit $\mathcal{L}$ l'opérateur de d'Alembert et $\hat{\mathcal{L}}$ l'opérateur de discrétisation de l'opérateur de d'Alembert.

En combinant astucieusement les 3 équations ci-dessus, on obtient le schéma suivant pour P :

$$\frac{p_{(i,j)}^{(n+1)} - 2p_{(i,j)}^{(n)} + p_{(i,j)}^{(n-1)}}{\Delta t^2} = c^2 \left(\frac{p_{(i+1,j)}^{(n)} - 2p_{(i,j)}^{(n)} + p_{(i-1,j)}^{(n)}}{\Delta x^2} + \frac{p_{(i,j+1)}^{(n)} - 2p_{(i,j)}^{(n)} + p_{(i,j-1)}^{(n)}}{\Delta y^2} \right)$$

D'où :

$$\mathcal{L}(p_{(i,j)}^{(n)}) - \hat{\mathcal{L}}(p_{(i,j)}^{(n)}) = -\frac{(\Delta t^2)}{12}\left(\frac{\partial^4 p}{\partial t^4}\right) + c^2\left[\frac{(\Delta x^2)}{12}\left(\frac{\partial^4 p}{\partial x^4}\right) + \frac{(\Delta y^2)}{12}\left(\frac{\partial^4 p}{\partial y^4}\right)\right] + O((\Delta t)^2, (\Delta x)^2, (\Delta y)^2)$$

où on a laissé tomber les indices pour plus de lisibilité.

Soit encore, en utilisant le fait que p vérifie l'équation de d'Alembert :

$$\mathcal{L}(p_{(i,j)}^{(n)}) - \hat{\mathcal{L}}(p_{(i,j)}^{(n)}) = c^2\left[\frac{\Delta x^2}{12} - c^2\frac{\Delta t^2}{12}\right]\frac{\partial^4 p}{\partial x^4} + c^2\left[\frac{\Delta y^2}{12} - c^2\frac{\Delta t^2}{12}\right]\frac{\partial^4 p}{\partial y^4} - \frac{c^2\Delta t^2}{6}\left(\frac{\partial^4 p}{\partial y^2 \partial x^2}\right) + O(...)$$

Ce qui prouve que le schéma est consistant. En effet, lorsque les pas temporels et spatials tendent vers 0, la solution discretisée coïncide avec la solution exacte.

On remarque que le schéma est alors nécessairement **d'ordre** 2. En effet, sous réserve de prendre les mêmes pas horizontaux et verticaux, on serait tenté de poser $c = \frac{\Delta x}{\Delta t} = \frac{\Delta y}{\Delta t}$ mais ceci ne permet que d'enlever les premiers termes, et il restera toujours le terme croisé d'ordre 2. On ne pourra donc pas améliorer la convergence du schéma.

Remarque : on peut voir en revanche qu'en dimension 1 le problème ne se pose pas, puisqu'il n'y a pas de termes croisés, et on peut alors avoir une précision infinie sur la solution exacte en posant $c = \frac{\Delta x}{\Delta t}$

Dispersion liée au schéma

On cherche les solutions du schéma sous forme d'ondes planes : $p_{(i,j)}^{(n)} = e^{\mathbf{i}(k\cos(\theta)i\Delta x + k\sin(\theta)j\Delta y - n\omega\Delta t)}$ où $\vec{k} = (k\cos(\theta), k\sin(\theta))$ est le vecteur d'onde et ω est éventuellement complexe.

Pour simplifier les expressions nous allons prendre, comme ce sera le cas dans la majorité de nos programmes : $\Delta x = \Delta y = h$.

On obtient en injectant dans le schéma de discrétisation :

$$\sin\left(\frac{\omega\Delta t}{2}\right) = \alpha^2\left[\sin\left(\frac{k\cos(\theta)h}{2}\right) + \sin\left(\frac{k\sin(\theta)h}{2}\right)\right]$$

où on a posé $\alpha = \frac{c\Delta t}{h}$ souvent appelé courant de Friedrichs-Lewy.

On peut prendre pour y voir plus clair la racine puis l'arcsinus de cette expression et effectuer un développement limité dans la limite où h et Δt tendent vers 0.

$$\frac{\omega}{k} = c\left(1 + \frac{h^2k^2}{24}(\alpha^2 - \cos^4(\theta) - \sin^4(\theta)) + o(h^3)\right)$$

On voit ici qu'il est impossible de choisir α tel que le terme de dispersion disparaisse. On aura donc forcément de la dispersion. En revanche, si $hk \ll 1$, les effets de la dispersion seront faibles. Quand on observe des ondes de longueur d'onde de l'ordre de 1 pixel, il sera donc impossible d'en tirer une quelconque conclusion.

Stabilité du schéma

Pour rechercher un critère de stabilité, on peut aussi utiliser les ondes planes et vérifier que pour tout k, $\omega(k)$ est réel. Par simple lecture de la relation de dispersion, non développée, ci dessus, on en déduit que, comme le sinus doit être inférieur un 1 si ω est réel, alors $\alpha^2 \leq \frac{1}{2}$, ce qui donne :

$$\frac{h}{\Delta t} \geq \sqrt{2}c$$

On peut aussi partir de la relation de dispersion développée, et dire que pour qu'il n'y ait pas d'ondes de choc, il est nécessaire que $\frac{d\omega}{dk} \leq c$; c'est-à-dire $\alpha^2 \leq \cos^4(\theta) + \sin^4(\theta)$. Or $\cos^4 + \sin^4$ a des valeurs comprises entre 1/2 et 1 donc il suffit que : $\alpha^2 \leq \frac{1}{2}$.

Lorsqu'on reprend l'étude pour deux valeurs différentes de Δx et Δy, mettons $\Delta y = \kappa \Delta x$ où κ est un réel plus grand que 1, on obtient la condition :

$$\frac{\Delta y}{\Delta t} \geq c\left(\frac{\kappa^2+1}{\kappa^2}\right)^{1/2}$$

Comme la fonction $\frac{\kappa^2+1}{\kappa^2} \leq 2$ pour $\kappa \geq 1$, il suffit alors d'avoir :

$$\frac{\Delta y}{\Delta t} \geq \sqrt{2}c$$

Finalement on retiendra :

$$min\left(\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t}\right) \geq \sqrt{2}c \qquad \text{Condition de stabilite}$$

I.3 Conditions au bord

I.3.1 Conditions d'absorption sur les bords de la pièce

Condition de non réflexion

Pour nos différentes modélisations, nous allons considérer les instruments de musique dans des pièces. Nous avons donc besoin, pour éviter de cumuler l'influence des bords de la pièce sur le son créé par le souffle dans l'instrument, que les bords de la pièce absorbent le son.

On considère donc une onde plane arrivant sur un mur. $\vec{e_x}$ qui définit l'orientation de l'axe des x est la normale entrant dans le mur.

L'onde plane incidente vérifie : $\frac{\partial p}{\partial t} + c\frac{\partial p}{\partial x} = 0$.

S'il existe une onde plane réfléchie, de la forme : $p_0 e^{\mathbf{i}(\omega t + kx)}$, elle vérifie : $\frac{\partial p}{\partial t} - c\frac{\partial p}{\partial x} = 0$.

On peut donc éviter la formation d'ondes réfléchies en posant la condition au bord suivante :

$$\frac{\partial p}{\partial t} + c\frac{\partial p}{\partial x} = 0$$

Adaptation numérique

Par exemple, pour la bordure verticale à gauche, on a pris :

$$p_{(0,j)}^{(n+1)} = p_{(0,j)}^{(n)} + \frac{c\Delta t}{\Delta x}\left(p_{(1,j)}^{(n)} - p_{(0,j)}^{(n)}\right)$$

Une telle condition a pour effet de diminuer l'ordre du schéma, mais on ne s'attardera pas davantage sur cette question.

Limitations

La condition choisie n'est légitime que pour une onde plane arrivant avec un front d'onde parallèle au mur, ce qui n'est pas le cas de tout le front d'onde en deux dimensions, clairement (et pour des longueurs d'onde devenant de l'ordre du pixel, notre estimation du gradient notamment sera très mauvaise).

Il existe deux solutions pour éviter ce problème :

- on peut prendre une pièce très grande devant les dimensions caractéristiques de la flûte, pour que les effets de la réflexion aient lieu loin.
- on peut effectuer une modélisation de "perfectly matched layers" en diminuant l'intensité du front d'onde sur une distance de quelques pixels aux bords (par exemple en jouant sur le terme associé à la dérivée spatiale...).

En constatant sur nos différents programmes que les effets de la réflexion sur les bords intervenaient peu, visuellement, nous n'avons pas éprouvé le besoin d'aller plus loin dans cette direction.

I.3.2 Condition de réflexion sur les bords de l'instrument

On prendra pour commencer une flûte d'une simplicité poussée à l'extrême : il s'agit d'une flûte constituée d'un unique cylindre / tube droit, dans lequel on souffle par une extrêmité. On discutera de modèles de flûtes améliorés ultérieurement.

Sur les bords intérieurs de l'instrument on prend des conditions de réflexion simples : la vitesse normale aux bords doit donc s'annuler. Par exemple pour un bord en $i = D$ et l'autre en $D + L$ où L est la taille en pixel de la flûte, on prendra : $u^{(n)}_{x(D,j)} = u^{(n)}_{x(D+1,j)}$ et $u^{(n)}_{x(D+L,j)} = u^{(n)}_{x(D+L-1,j)}$.

Sur les bords extérieurs de la flûte, le débat s'installe. En fait, en pratique, les bords extérieurs de la flûte réfléchissent le son aussi, et on adoptera donc la même condition. Toutefois, pour certaines études numériques (les premières notamment où on ne fait que "tatonner", on adoptera des conditions d'absorption sur les bords extérieurs afin de "voir mieux" ce qui est projeté par la flûte.

I.4 Souffle dans l'instrument

Modéliser le souffle produit par l'homme dans l'instrument de musique est de loin la chose la plus délicate à envisager de notre projet. Dans le fond, lorsque les joues d'un flûtiste se gonflent et expirent de l'air, en fait elles dégagent une surpression dans le corps de l'instrument. Mais en fait, c'est tout le corps même de l'instrumentiste qui va servir aussi d'instrument : on a constaté en effet que les lèvres-mêmes vibrent lorsque l'instrumentiste souffle, sinon comment pourrions nous siffler ?

Pour les clarinettes ou autres instruments à hanche, c'est un mécanisme plus compliqué : une petite lamelle (le plus souvent en roseau pour les clarinettes) sert d'opérateur pour l'admission et la rétention de l'air en supression dans la bouche de l'instrumentiste. De façon extrêmement simplifiée, c'est elle qui permet de "battre" l'air et de créer une perturbation périodique.

Finalement, si on modélise l'admission d'air dans l'instrument par une sinusoïde pure, on n'est pas très proche de la réalité, mais le comportement obtenu de l'onde initialement admise sera relativement justifié. Ensuite, si on reste dans le cadre d'une physique linéaire, il n'y a plus vraiment à se poser de questions, puisque les modes n'interragissent pas entre eux.

II Différentes optimisations de la modélisation à 2D

II.1 Flûte simple dans une pièce

Pour débuter, nous n'avons pas voulu nous compliquer trop la vie. Après un essai fructueux de propagation dans une "flûte" toute droite sans pièce, de laquelle part à une extrêmité une onde sinusoïdale, nous avons juste élargi l'espace de travail en mettant une pièce autour de la flûte. On utilise pour l'instant une formulation récursive la plus simple qu'il soit pour l'évolution de chaque point de pression. Bien sûr, le programme est alors très très lent dès qu'on augmente le nombre de points dans la pièce.

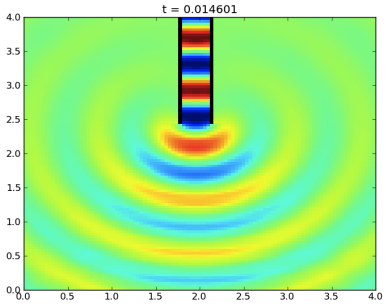

FIGURE 2 – Visualisation de la pression à un temps $t = 14,6ms$ dans une flûte énorme après le départ initial de la sinusoïde et propagation dans une pièce de 4m par 4m

Le programme correspondant se nomme **flute_piece1.py**.
Il faut maintenant trouver une méthode pour améliorer la vitesse de ce code!

II.2 Vectorisation

La vectorisation consiste simplement à appliquer la fonction de propagation a une liste de points plutôt que de faire deux boucles récursives imbriquées. On procède de la sorte :
- on transforme $p_{(i,j)}$ en $\tilde{p}_{(NY*i+j)}$ par exemple.
- on crée les listes de points sur lesquels doivent s'appliquer les fonctions de propagation du son.
- on utilise la fonction map() sous python pour les appliquer.

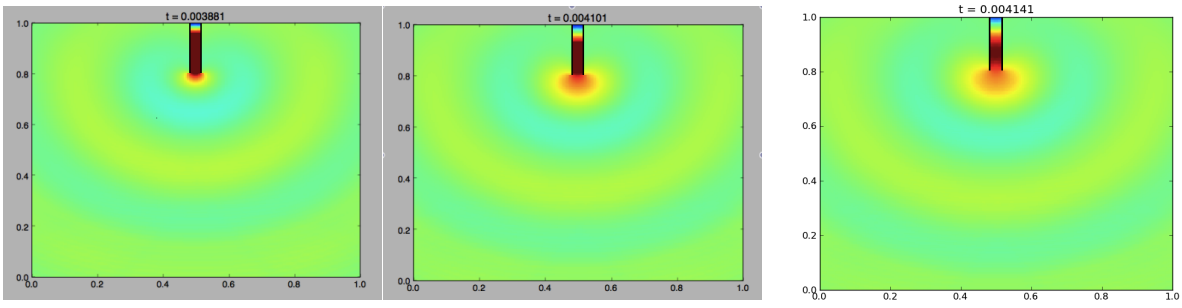

FIGURE 3 – Arrivée du front d'onde sur une flûte de taille comparable à une flûte réelle dans une pièce de 1m par 1m

La différence de vitesse est assez remarquable! De plus on constate bien la différence d'intensité entre les ondes à l'extérieur et à l'intérieur de la flûte. En effet, la puissance à l'intérieur de la flûte est beaucoup plus importante qu'à l'extérieur.
Le programme correspondant se nomme **flute_piece2.py**.

II.3 Matrices à trous

Une des idées que nous avions était aussi de créer une énorme matrice dont la multiplication seule suffirait à transformer p,u_x et u_y.

Construction de la super matrice

La construction de la matrice n'est pas complètement évidente. On peut construire facilement l'intérieur brut de la matrice concernant la propagation du son. En revanche, dès qu'on touche les conditions aux bords sur les instruments, cela devient très vite compliqué, et ingérable quand il s'agit de changer d'instrument.

Comme nous avions déjà des programmes qui tournaient mais sans erreur manifeste, nous avons eu l'idée de reprendre la fonction timeStep et la modifier pour construire directement la matrice. Pour cela, on remplace l'affectation d'une valeur par celle d'une ligne de la matrice. On a ainsi une méthode qui permet de construire la matrice même dans les cas les plus tordus où on se serait inévitablement trompé en essayant de le faire à la main.

Plusieurs tentatives de construction ont d'abord avorté (pour cause de matrices trop vite trop grandes) avant que nous trouvions le formalisme des matrices *sparse* sous python. En effet, nos matrices de transfert étant essentiellement "pleines de zéros" considérant que seuls les carreaux attenants les uns aux autres interragissaient, il ne servait à rien de retenir dans la mémoire autant de choses pour si peu d'information...

Armés donc du formalisme des matrices sparse, nous avons créé un programme matriciel sous python. La vitesse d'exécution est alors bien supérieure encore à celle du programme vectorisé, en revanche, nos ordinateurs ne supportaient pas la création de matrices sparse trop grandes (au bout d'une centaine de points, la création de la matrice était impossible - ci dessous on représente tout de même du 100 par 100).

Le programme correspondant regroupe :

- **constantes.py** qui définit les constantes du programme
- **courbes.py** qui définit le type des courbes qu'on utilisera ensuite pour construire l'instrument.
- **indices.py** qui définit la convention choisie pour les indices
- **paroi.py** qui définit les conditions aux bords pour un instrument donné (nous consacrons une partie ultérieure de notre rapport à ce programme en lui-même).
- **matrice.py** qui crée la matrice d'interaction pour un instrument donné (en utilisant Paroi en particulier).
- **flute_piece_top.py** qui est le programme principal.

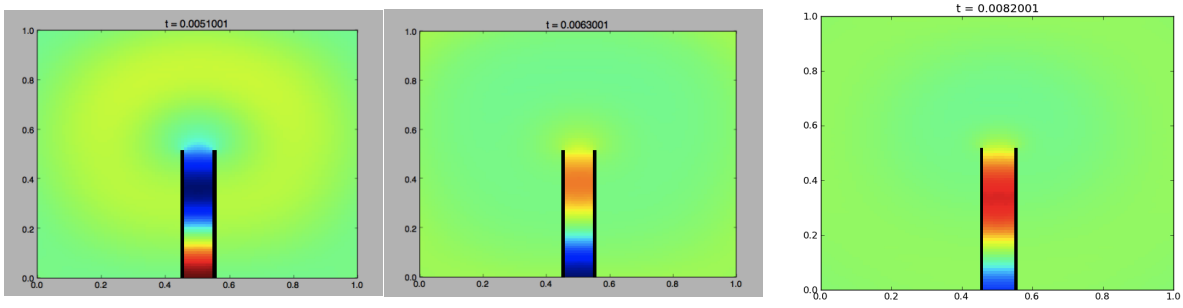

FIGURE 4 – Arrivée du front d'onde sur une flûte de taille peu comparable à une flûte réelle dans une pièce de 1m par 1m

Diagonalisation de la super matrice Si nous notons $\vec{P}$ l'énorme vecteur formé des vecteurs concernant p, u_x et u_y, on a alors : $\vec{P}^{(n+1)} = M\vec{P}^{(n)}$, autrement dit, $\frac{\partial \vec{P}}{\partial t} \simeq \frac{M-Id}{dt}\vec{P}$

Donc, les valeurs propres de la matrice $\frac{M-Id}{dt}$ seront potentiellement des imaginaires purs dont la norme correspondra aux fréquences propres de l'instrument (et de la pièce).

Suite à plusieurs échecs, nous avons tout de même essayé de diagonaliser la matrice. Le paquet scipy.sparce propose une fonction de diagonalisation. Mais celle-ci donne des résultat aberrants : une seule valeur propre réelle.

La diagonalisation de matrice pleines fonctionne un peu mieux : elle donne des valeurs réalistes bien que la reconstruction de la matrice d'origine à partir de la forme diagonalisée soit plus qu'approximative :

- un premier résultat encourageant est que l'on retrouve la stabilité du schéma puisque toutes les valeurs propres sont de partie réelle négative.

- parmi les vecteurs propres, la plupart correspondent à des artefacts numérique ou des résonances de la pièce (extérieur de la flûte), comme on le voit ci-dessous.

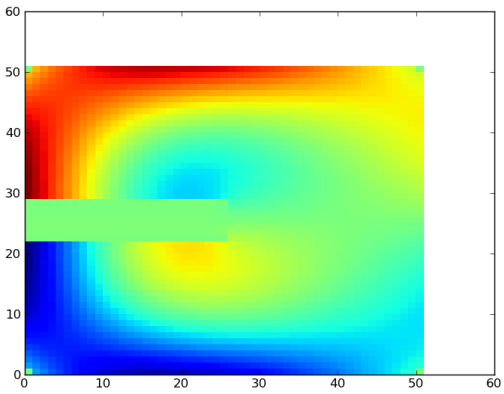

FIGURE 5 – Mode à l'extérieur de la flûte.

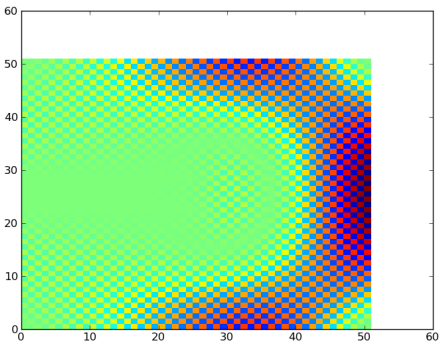

FIGURE 6 – mode artefact numérique : en moyennant sur quelques pixels, la pression est uniformément nulle

- En repérant les valeurs propres telles que la fréquence soit raisonnable (pas trop grande ni trop faible) et l'amortissement faible, nous arrivons à trouver quelques modes propres de la flûte. Une autre difficulté est que les modes du système total sont dégénérés par rapport à ceux de la flûte. Ainsi pour chaque mode de la flûte on va obtenir un ensemble de modes, chacun correspondant à un mode extérieur.

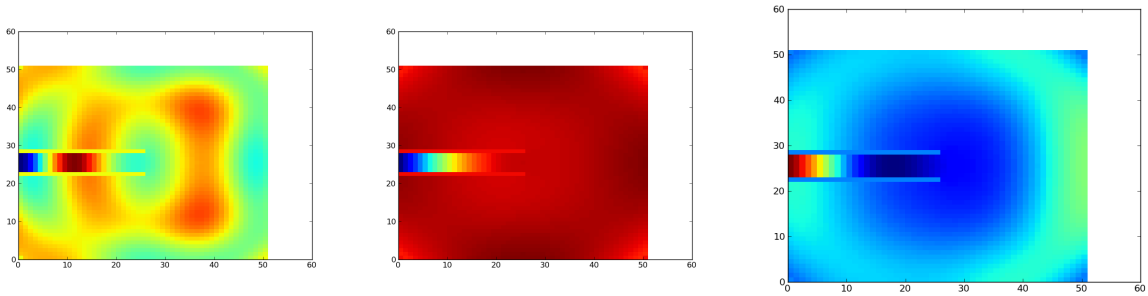

FIGURE 7 – Différents modes de résonance de la flute.

Pour que la diagonalisation soit réellement exploitable, il faudrait être capable de sélectionner automatiquement les modes physiques. Il aurait également fallut apprendre à se servir finement des outils de gestion des matrices creuses, voire de passer à un langage permettant de mieux gérer la mémoire.

Cela aurait nécessité du temps que nous n'avons malheureusement pas eu. De plus la simple construction de la matrice demeure plus longue que l'exécution sur un temps raisonnable du programme en C++.

Conclusion

Au terme de ces nombreux essais, nous avons (car cela avait été construit en parallèle), préféré poursuivre notre projet en C++, pour des simples raisons de commodité et rapidité. En effet, le programme tournait beaucoup plus rapidement en C++, avec une simple flûte, et la gestion des nouveaux instruments nous y était préférée - la tentative de construction d'une flûte à bec notamment avait été une épreuve sans pareil et qui n'a pas su aboutir sur des résultats très concluants.

III Conditions aux limites des instruments particuliers

Initialement, nous avions pris en compte la présence de l'instrument directement dans la fonction qui faisait évoluer dans le temps les variables. Très vite il a fallu commencer à penser à changer d'instrument et il fallait donc passer à des méthodes telles que le description de la flûte soit plus facile à donner à la machine. C'est pour cela que nous avons décidé de décrire l'instrument comme un ensemble de paroi.

III.1 Physique d'une paroi

Dans le cadre de l'acoustique où l'on étudie des ondes en régime linéaire non visqueux, la condition aux bords d'une paroi est la nullité de la vitesse normale. La nullité de la vitesse tangentielle viendrait si l'on était en régime visqueux. Mais qu'est-ce que la normale à une courbe composée de pixels ?

Tout d'abord, précisons que nous considérons que les points de la grille de pression décrivent la courbe et que les points de vitesse adjacents forment le bord de la paroi. Nous avons décidé de répartir les pixels d'une paroi en différentes catégorie selon la direction de la normale que nous avons attribué. Les normales retenues sont : les axes des pixels et les diagonales, comme on le voit sur le schéma ci dessous.

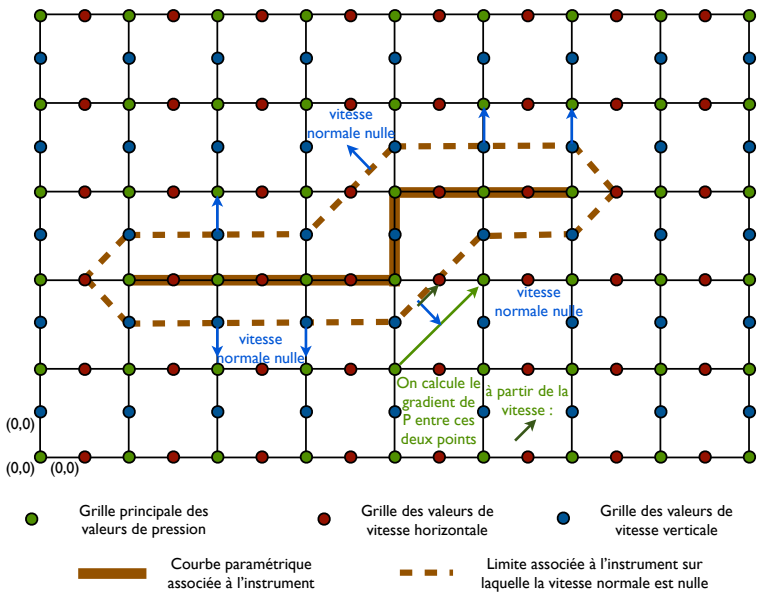

FIGURE 8 – illustration du mécanisme de numérisation des bords des instruments

Maintenant que nous connaissons l'effet d'une paroi, il faut encore être capable d'en créer simplement une.

III.2 Construction d'une paroi

Une solution basique serait de tout simplement donner une liste de pixels pour la paroi. Cette solution présente l'inconvénient d'être très longue à réaliser et en plus cela rend impossible de changer les paramètres numériques (nombre de points...).

On va donc créer nos parois sous forme de courbes paramétrées. Mais passer d'une fonction paramétrée de manière quelconque à un ensemble de pixel n'est pas trivial. La solution retenue consiste, à partir d'un pixel donné correspondant à un point de la courbe, à rechercher le plus petit incrément dans le paramètre de la courbe tel que l'on ne bouge que d'un pixel par rapport à la position précédente.

Il y a une certaine tolérance et ce que l'on veut est en fait trouver un intervalle sachant qu'on peut tester la position de tout nombre par rapport cet intervalle. La méthode retenue est finalement d'encadrer (au sens large) l'intervalle cible et de regarder au milieu du segment joignant les bornes. Si on est dans l'intervalle, on a gagné et sinon on définit ce milieu comme une nouvelle borne.

Une fois la liste des points récupérée, il faut déterminer l'orientation de la normale. Pour cela on détecte les voisins de chaque point et selon leur nombre et position on décide de la direction de la normale (voir notamment la figure ci dessus).

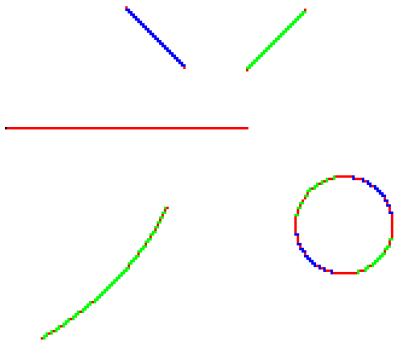

FIGURE 9 – Parois variées : la couleur représente les différents types de physique

```
vector<Paroi*> instrument;

//construction à l'ancienne
vector<int> x,y;
for(int i=50;i<75;i++)
{
    x.push_back(i);
    y.push_back(i);
    x.push_back(i);
    y.push_back(i+1);
}
instrument.push_back(new Paroi(x,y)); //on ajoute une paroi vertical
x.clear();
y.clear();

for(int i=50;i<75;i++)
{
    x.push_back(i+50);
    y.push_back(NY-i-75);
    x.push_back(i+50);
    y.push_back(NY-i-1-75);
}
instrument.push_back(new Paroi(x,y));
x.clear();
y.clear();

//Initialisation à partir d'une fonction quelconque
Segment segment1(0,DY/2,DX/2,DY/2);
instrument.push_back(new Paroi(segment1,0,1));

Cercle cercle1(DX/2+DX/5,DY/2+0.2,DX/10);
instrument.push_back(new Paroi(cercle1,0,10));

ExponentielleVerticale expo(DX/3,2*DY/3,DX/7,8*DY/9);
instrument.push_back(new Paroi(expo,0,1.2));
```

FIGURE 10 – Code permettant de créer les parois

En pratique, nous avons créé une classe Paroi (aussi bien en c++ qu'en python) pour gérer tout cela et rendre le code principal plus lisible.

III.3 Construction d'un instrument

Un instrument étant simplement un ensemble de parois, il suffit de le construire en le décrivant comme un ensemble de courbes paramétriques.

Compte tenu du travail effectué sur les parois et comme on peut construire une courbe à partir d'éléments simples (typiquement rayon et centre pour un cercle, ou points pour un segment), l'instrument se construit bouts à bouts. Par exemple une flute peut être décrite par des segments donc par des points. La seule lourdeur restante provient de la complexité de la géométrie d'un instrument : le décrire avec des courbes usuelles nécessite de le découper en beaucoup de morceaux.

III.4 Bilan

On a pu ainsi construire un modèle plus raffiné de flûte, s'apparentant à la flûte traversière, dans laquelle on envoie du son par le petit orifice haut, et le "flûtiste" joue avec sa "bouche" qui est énorme et dont les bords absorbent les "fuites" du son qu'il produit. Quel ne fût pas notre émerveillement quand nous avons vu encore de très belles ondes stationnaires s'établir dans la flûte!

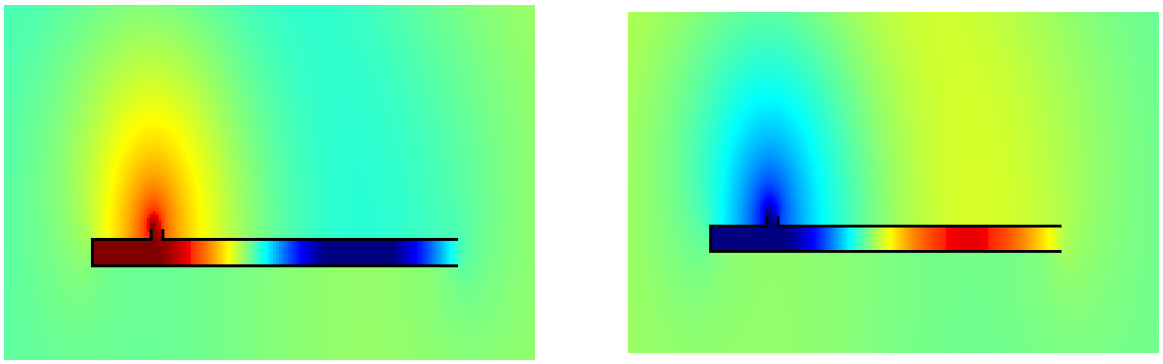

FIGURE 11 – Etablissement des ondes stationnaires dans la flûte traversière

IV Etude des différents instruments

IV.1 La flûte

Le son grave

Après quelques recherches, nous avons fini par adopter un modèle de flûte traversière avec un léger rebord pour y "poser" le son, comme le feraient les flûtistes en question, qui soufflent à raz quasiment.

On a alors décidé d'en faire "l'étude fréquentielle", c'est-à-dire qu'on a déposé une sonde de pression à un point bien placé de notre pièce, et on a regardé la valeur efficace de la pression en cette sonde, en fonction de la fréquence. Un balayage pour des fréquences allant de 10 à 5000Hz par pas de 5Hz a alors permis l'établissement du graphe suivant :

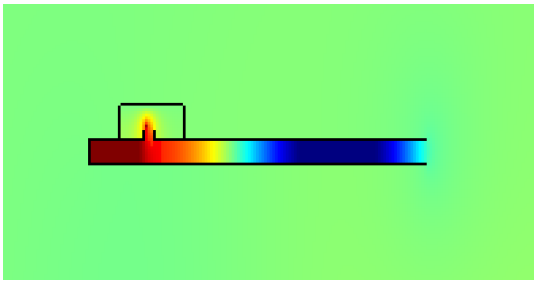

FIGURE 12 – Image de la flûte traversière

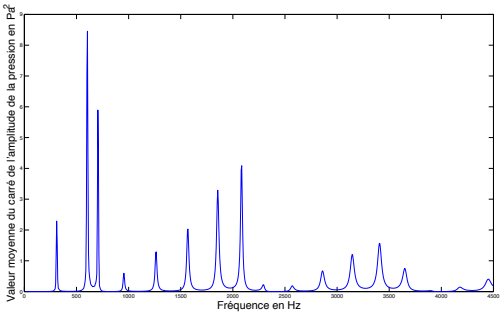

FIGURE 13 – Spectre en fréquence de la flûte tous trous bouchés

Quel ne fût pas notre émerveillement et notre satisfaction de voir comme la flûte avait pu aussi précisément "sélectionner" des modes ! On voit donc que si l'on joue le son relativement grave associé à cette note ($\simeq 600Hz$), le son produit par la flûte sera possiblement doté de toutes les harmoniques présentes ci dessus (des harmoniques à l'octave donc le double de la fréquence, mais on voit aussi des fréquences plus écartées les unes des autres qui peuvent correspondre à d'autres modes de résonance, à la quinte par exemple). Le son de la flûte est donc très riche - très timbré.

Les autres sons Forts de ce succès, nous nous sommes empressés de voir ce que cela donnait avec des trous !

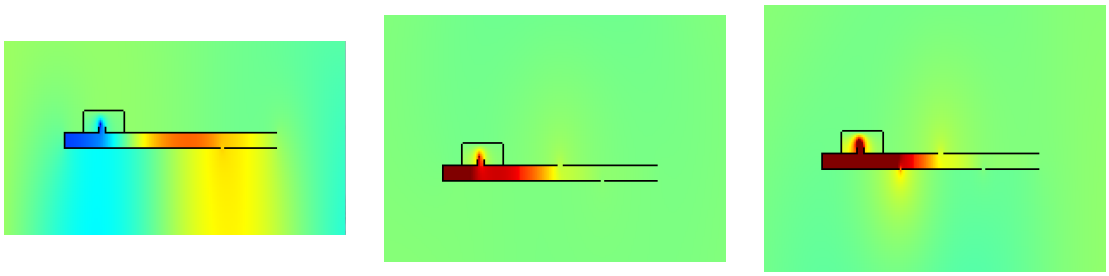

FIGURE 14 – Images de la flûte traversière avec 1, 2 ou 3 trous

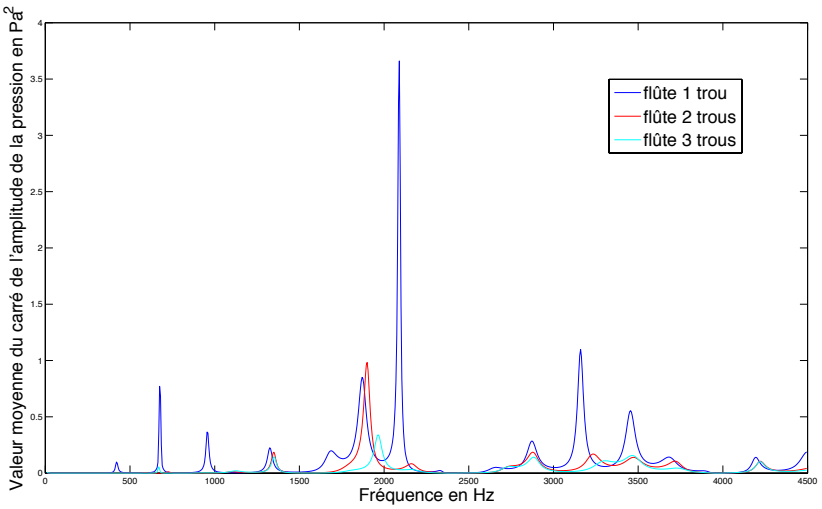

FIGURE 15 – Spectre en fréquence de la flûte pour plusieurs configurations des trous

Sur ce spectre on constate d'abord que l'intensité des pics est globalement moins grande que celle des pics la flûte étant complètement bouchée. Ensuite, on voit que pour certains trous, certains pics sont confondus, d'autres sont disjoints : ce qui illustre bien qu'on peut jouer des notes différentes avec différentes configurations de trous ; mais on verra ça avec plus de précision sur la clarinette.

IV.2 La clarinette

Le son grave

Notre clarinette, elle, possède une embouchure, qui est sensée modéliser un bec, et possède une extrêmité très évasée en exponentielle. Que va donner l'étude fréquentielle ?

On réalise le même balayage que précédemment et on obtient le graphe suivant :

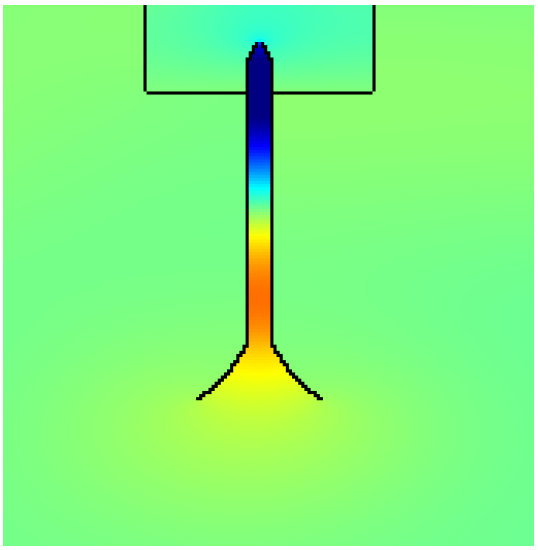

FIGURE 16 – Image de la clarinette

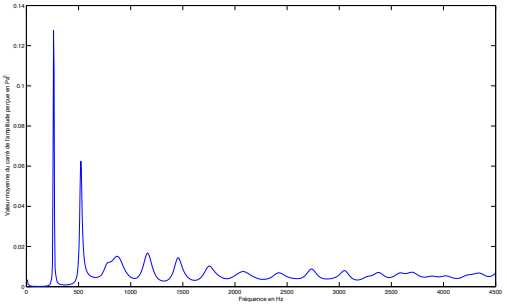

FIGURE 17 – Spectre en fréquence de la clarinette tous trous bouchés

Tout se passe aussi bien : la clarinette sélectionne elle aussi des modes. En revanche, elle les sélectionne davantage que la flûte. Son son est donc beaucoup plus pur que celui de la flûte (c'est-à-dire moins timbré).

Ici on a décidé d'affiner le balayage en fréquence sur les premiers modes pour voir si les fréquences étaient multiples les unes des autres :

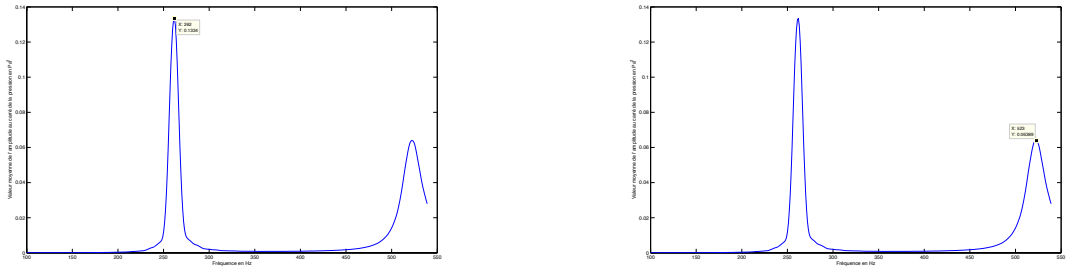

FIGURE 18 – Spectre en fréquence fin de la clarinette tous trous bouchés

Le premier pic étant à $262Hz$ et le second à $523Hz$ nous pensons pouvoir dire avec une très bonne approximation que le second est le double du premier : on a des harmoniques à l'octave traditionnel. Mais, ce n'est pas le cas de tous les pics secondaires qui dénotent l'apparition d'harmoniques à la quinte, etc et qui font également la richesse du son.

Les autres sons Et pour la clarinette, avec des trous ?

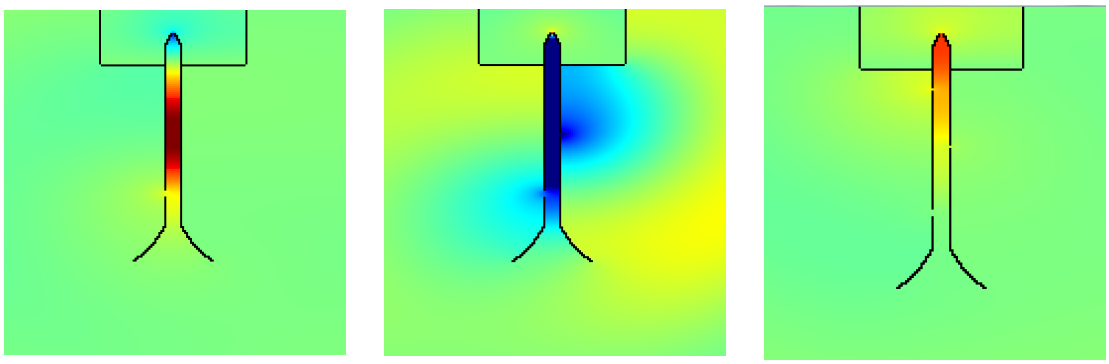

FIGURE 19 – Images de la clarinette avec 1, 2 ou 3 trous

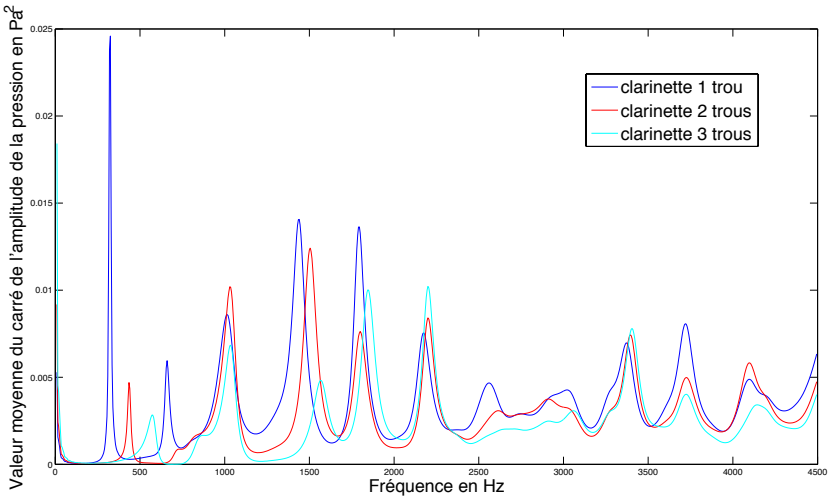

FIGURE 20 – Spectre en fréquence de la clarinette pour plusieurs configurations des trous

Ici, on voit de même que pour la flûte, que l'intensité a grandement diminué. En revanche on voit beaucoup mieux le phénomène suivants :

- certaines harmoniques sont communes à plusieurs trous
- mais le spectre de chaque trou est très différent parce que certaines harmoniques se décalent bien ce qui correspond au filtrage différent : donc on joue nécessairement des notes différentes avec différentes configurations de trous (le pic principal se décale...). Par exemple, on remarque que plus on augmente le nombre de trous, c'est-à-dire, plus on permet au son de "s'échapper" plus tôt, c'est-à-dire, plus près de l'embouchure, plus il est aigü ! Et en effet, si on diminue la longueur sur laquelle les ondes stationnaires s'établissent, on diminue λ et on augmente f ! Tout se tient.

Conclusion

Finalement, nous avons réussi à construire deux instruments en deux dimensions, et à illustrer la variété et la richesse de leurs timbres respectifs. Nous pouvons donc, à la lumière de cette analyse numérique, dire que les flûtes sont des instruments toujours aussi merveilleux !

Références

[1] *La physique des Cuivres*
http ://la.trompette.free.fr/theorie.htm

[2] *Notes on perfectly matched layers*
par Steven G. Johnson, 2007

[3] *The logical clarinet : numerical optimization of the geometry of woodwind instruments*
par D. Noreland, J.Kergomard, F. Laloë, C. Vergez, P. Guillemain, et A. Guilloteau

[4] *La Clarinette*
par Suzanne et Franck Laloë, Pour la Science, 1985

[5] *Simulation numérique et retournement temporel*
par F. Marbach, 2010