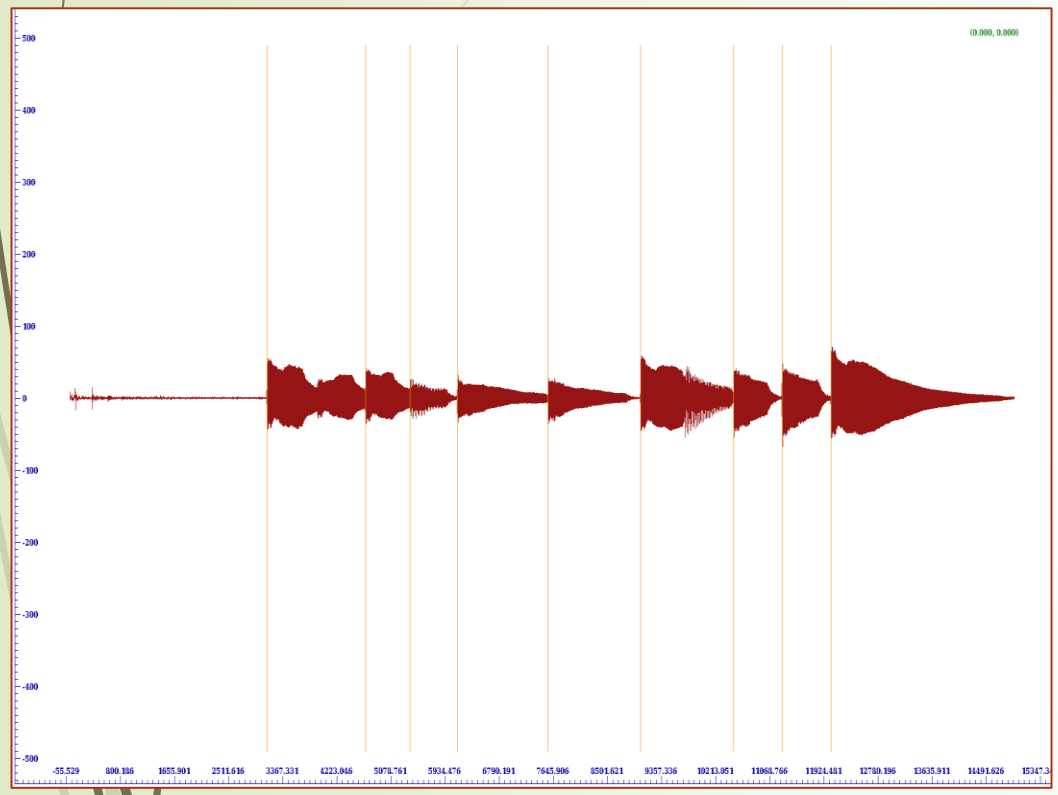
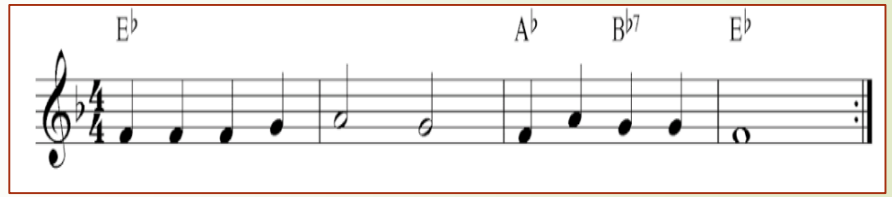
Projet : Retranscrire une partition musicale depuis un signal sonore.

Itération 3

Groupe 2 : Carreteros Laetitia , Duraj Bastien , Grolleau Tao , Kempenaers Francis , Jouvet Lucas

Objectif

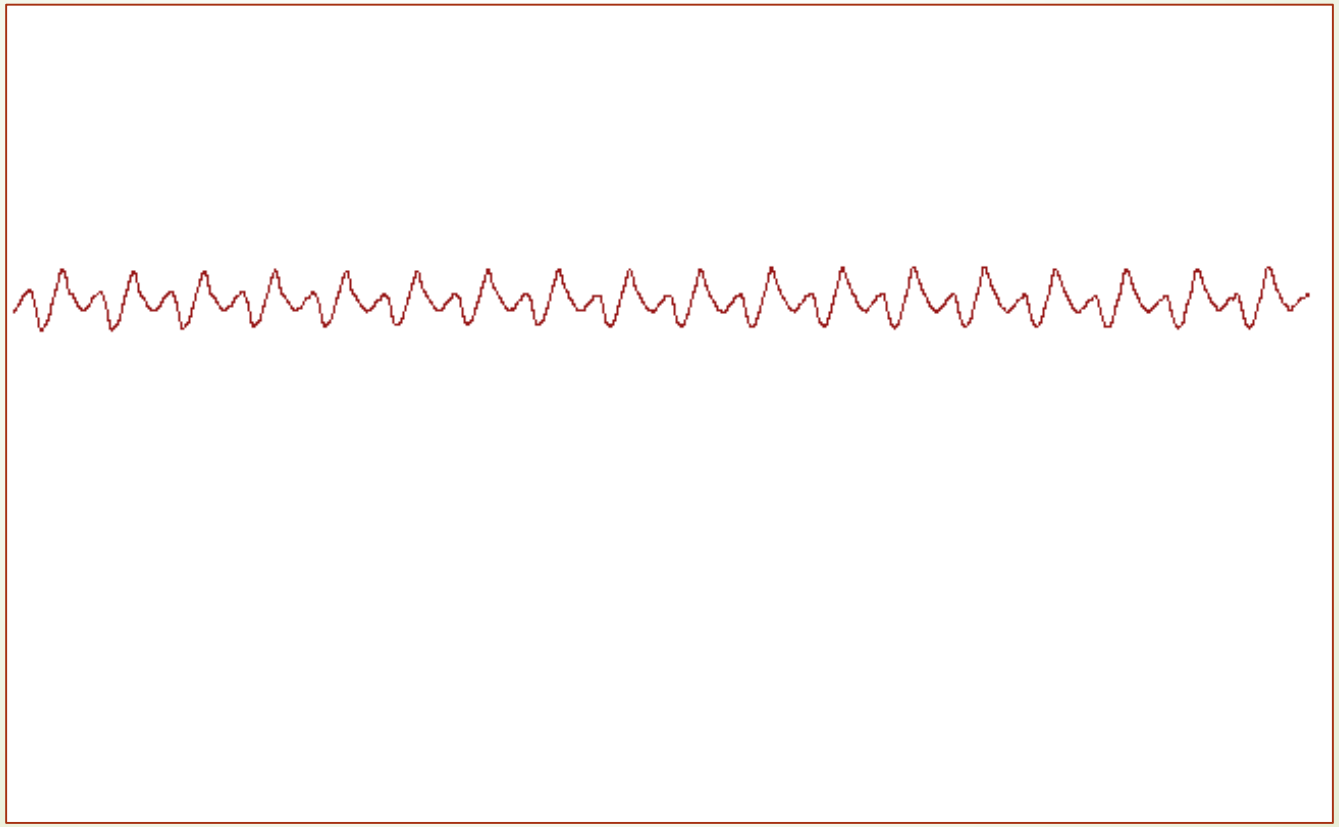




Répartition de notre travail

- Gestion fichier, Analyse des signaux
  - Bastien DURAJ
  - Lucas JOUVET
- Création de la partition
  - Laetitia CARRETEROS
  - Tao GROLLEAU
- Parser Midi , interface graphique
  - Francis KEMPENAERS

Analyse note



Analyse note

```
Frequence 2 : 173.619671
note : fa 2
Frequence 3 : 269.443625
note : do# 3
Frequence 4 : 979.977570
note : si 4Note 1: si 4
Frequence 1 : 157.595947
note : re# 2
```

Analyse note : Méthode 1

- Moyenne de toutes les fréquences
- Suppression des fréquences 2 fois < moyenne et 2 fois > moyenne
- Moyenne de toutes les fréquences

Analyse note : Méthode 2

- Tri des fréquences
- Récuperation de la fréquence médianne

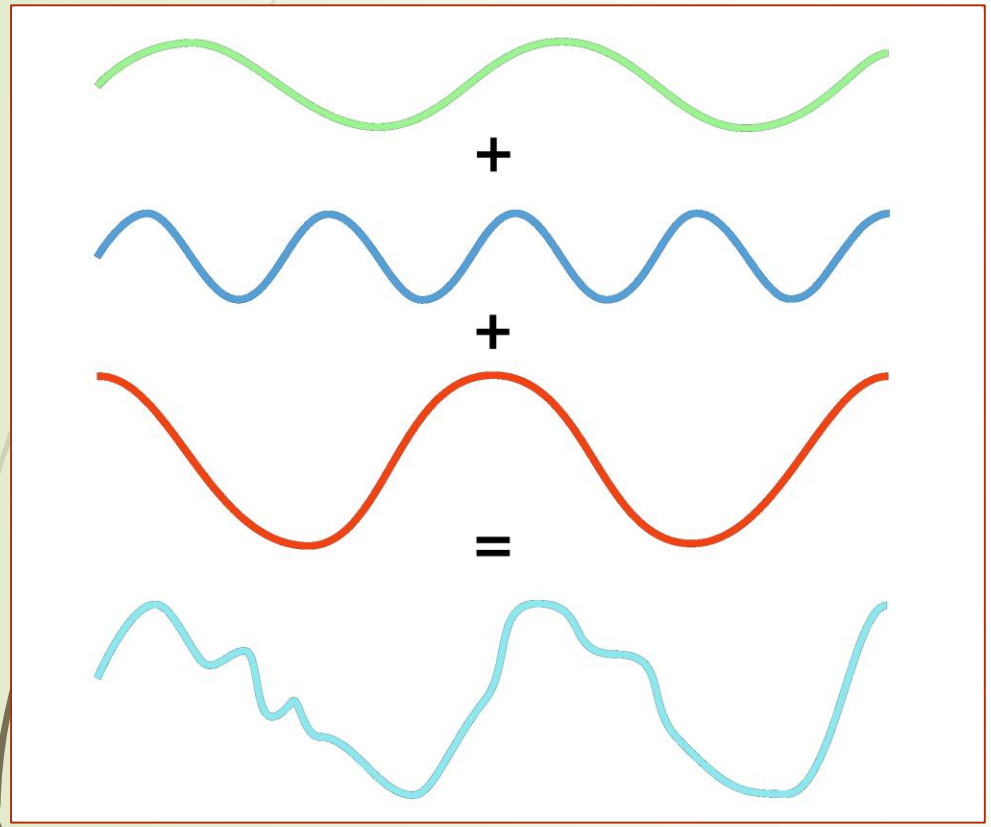
Analyse note : Méthode 3

- Tri des fréquences
- Suppression 1% des fréquences maximales
- Fréquence moyenne

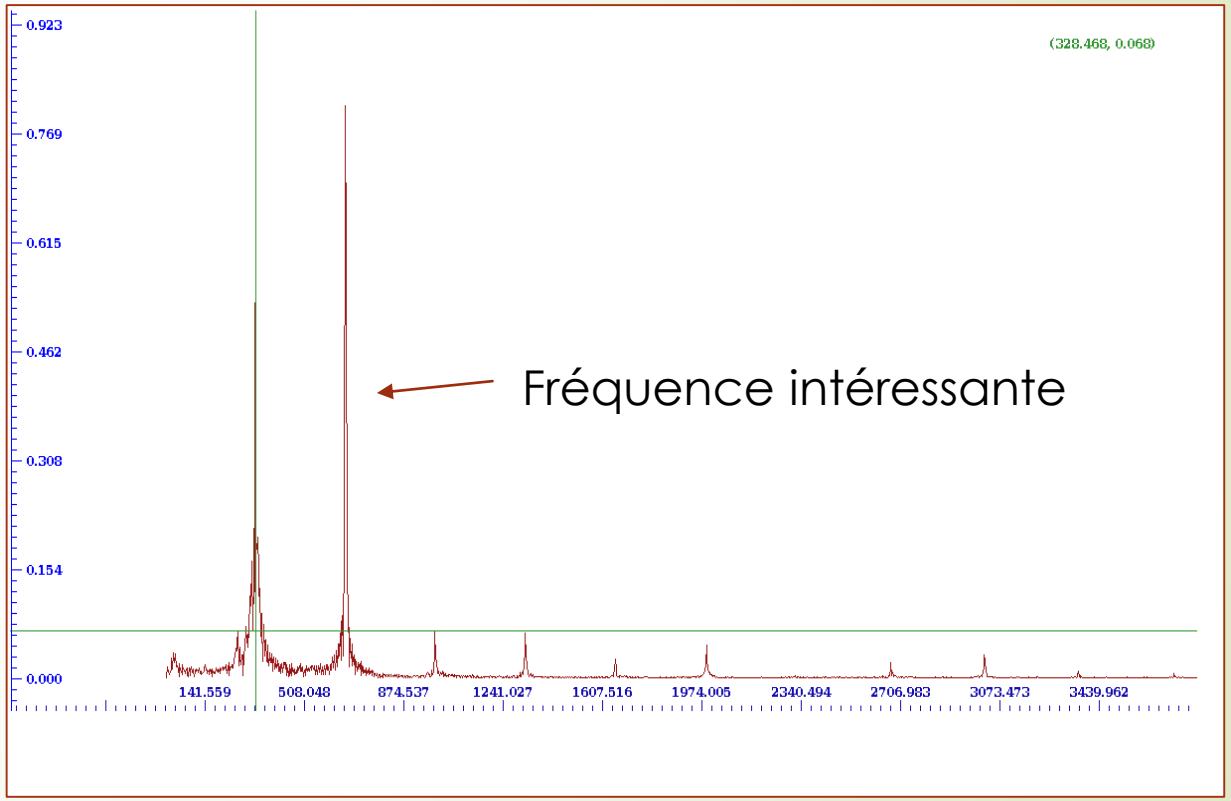
Analyse note : Méthode 4

- Tri des notes
- Récuperation de la frequence la plus redondante

L’importance de la Fast Fourier Transform



Composition d’une onde périodique

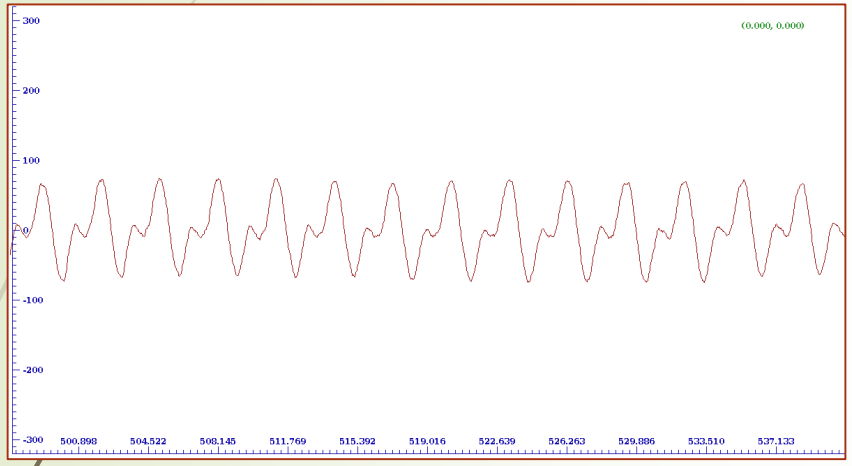


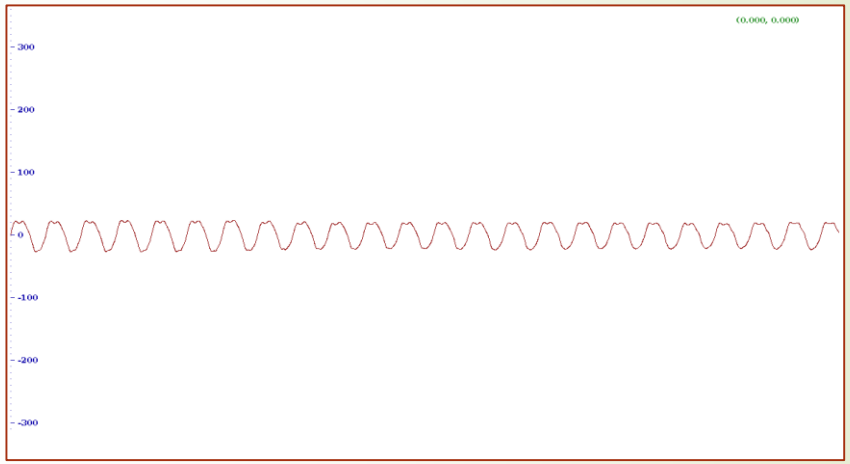
- Spectre

Découpage d'un signal en notes

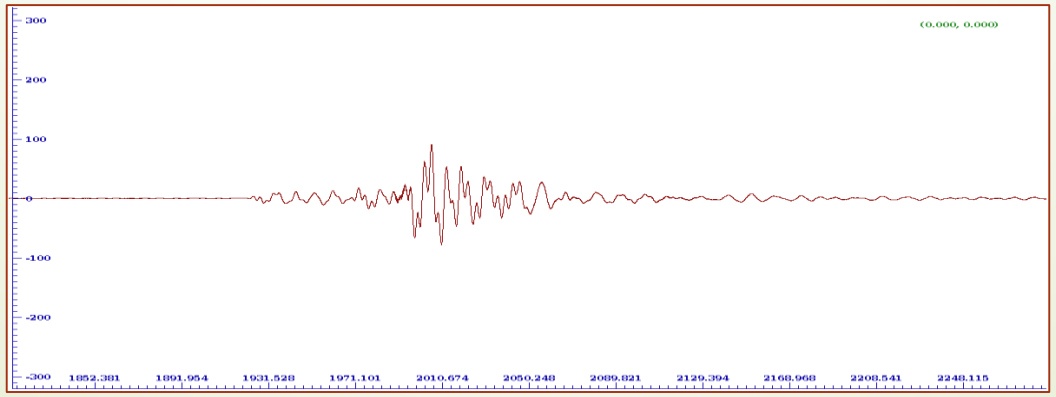
Problèmes rencontrés

- Un signal n'a pas toujours la même forme





- L'amplitude du signal varie énormément



Découpage d'un signal en notes

Algorithmes créés

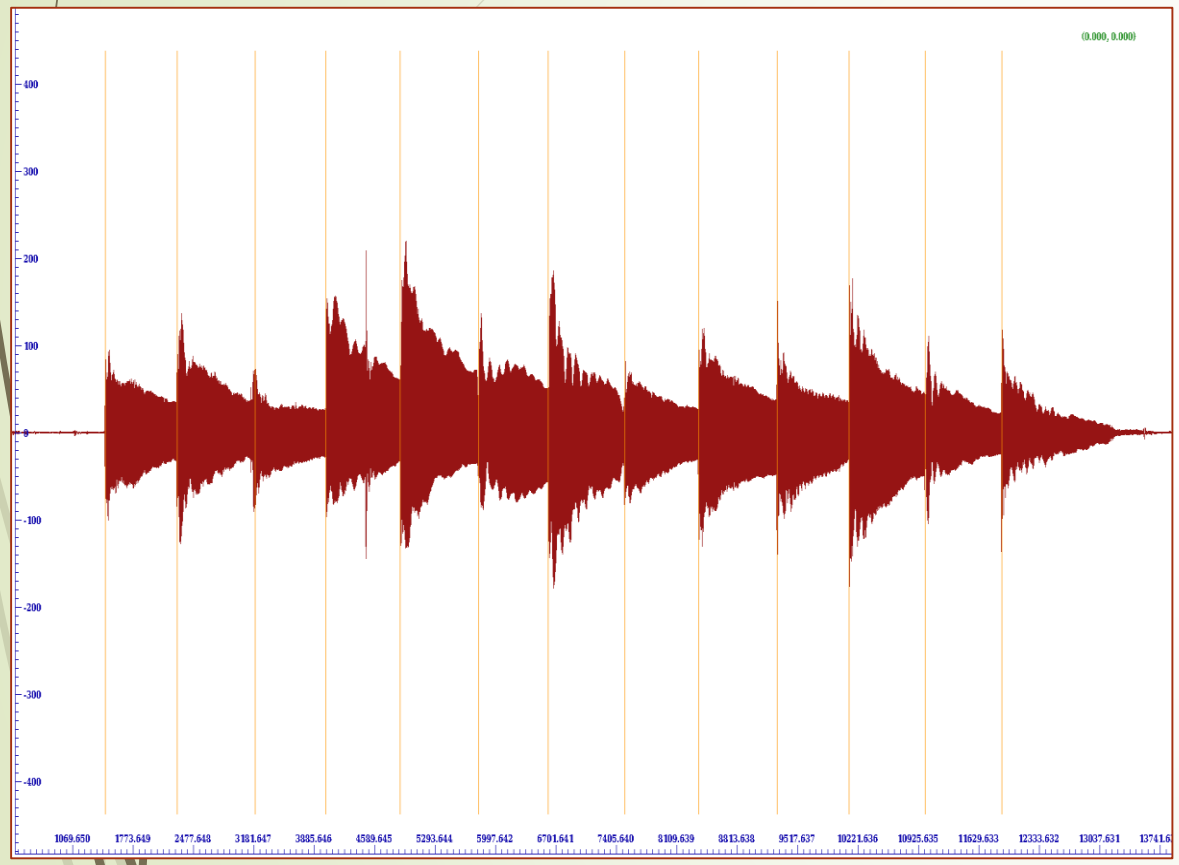
- Premier algorithme
  - Utilisation de la longueur d'une période
  - Problème si la même note est jouée 2 fois de suite
- Deuxième algorithme
  - Utilisation des baisses d'amplitudes entre 2 notes
  - Problème pour fixer le seuil

Problème commun: Aucun algorithme permettant de découper le signal en période

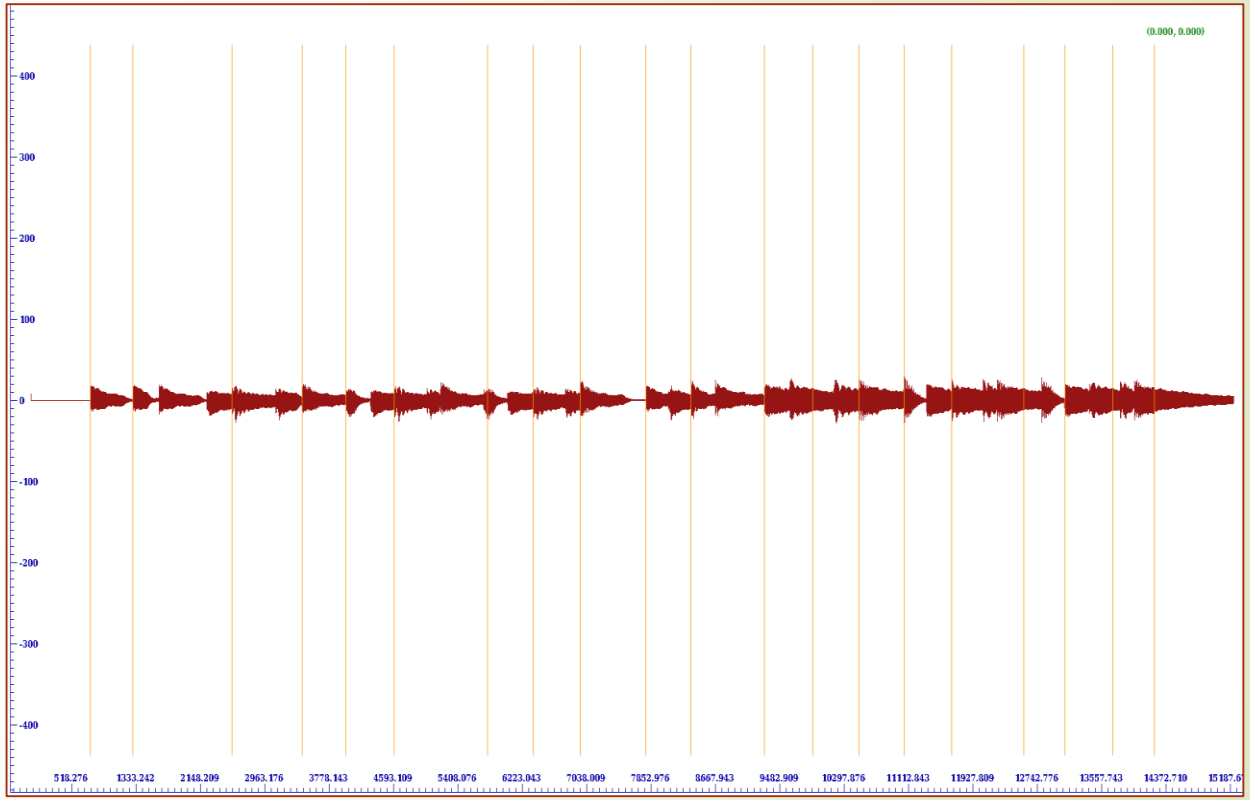
Présentation du deuxième algorithme

```
Entrée: un signal  # sous forme d'un tableau d'amplitudes
Sortie: un tableau de notes # avec l'indice de debut et de fin de chaque note

separation_signal(signal):
    debut
        seuil <- 2 * ecart_type(signal)  # calcule la limite au-dessus de laquelle une période est valide
        periodes <- recupere_periode_haute(signal, seuil)  # recupere les periodes dans un tableau avec l'indice de chacune d'elles dans les amplitudes
        notes <- recupere_note(periodes)     # recupere les notes en créant des séparations la ou il y a des trous entre les periodes

        notes <- enleve_note_courte(notes)    # si debut et fin de note trop court alors on enleve la note
        notes <- analyse_note_longue(notes)   # si debut et fin trop eloigné alors on analyse a nouveau cette partie et ajoute les nouvelles notes trouvées

        return notes   # renvoie le résultat
    fin
```

Résultats



Signal correctement segmenté



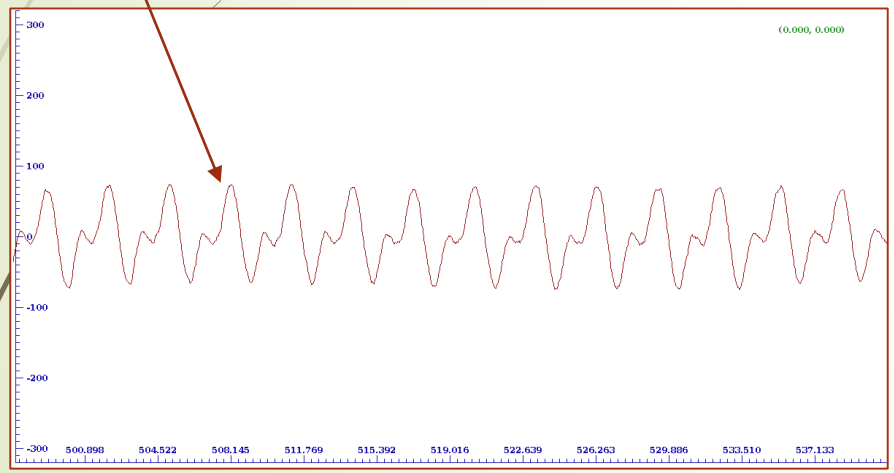
Signal mal segmenté

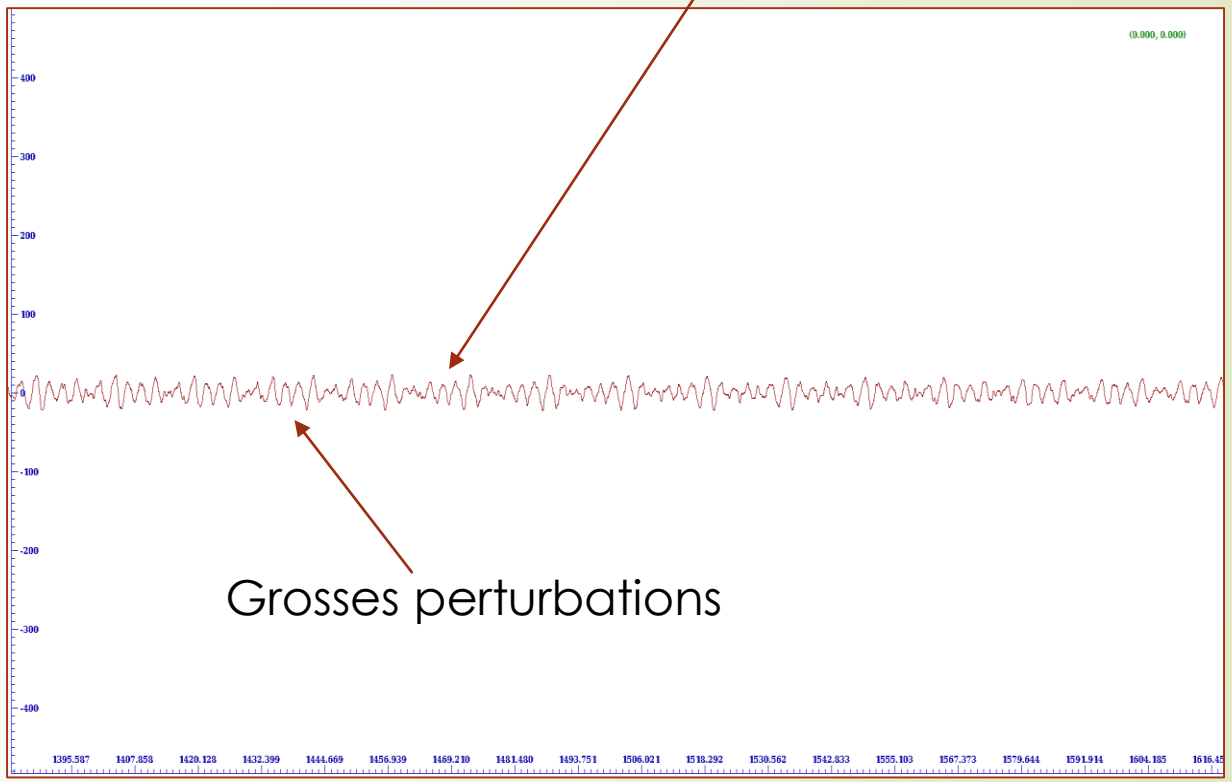
Un petit mot sur les accords

- Problèmes encore sans solution

Périodes très longues

Périodes nettes et précises





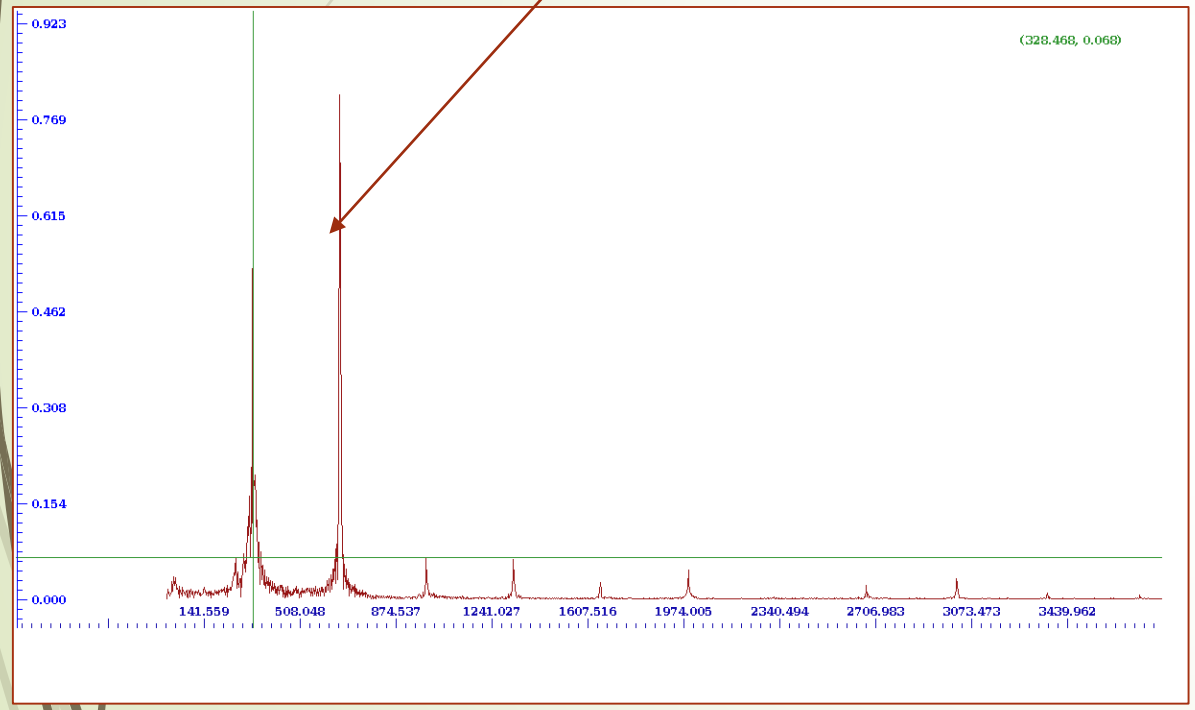
Grosses perturbations

Signal d'une note

Signal d'un accord

Un petit mot sur les accords

Présence d'une forte harmonique 2



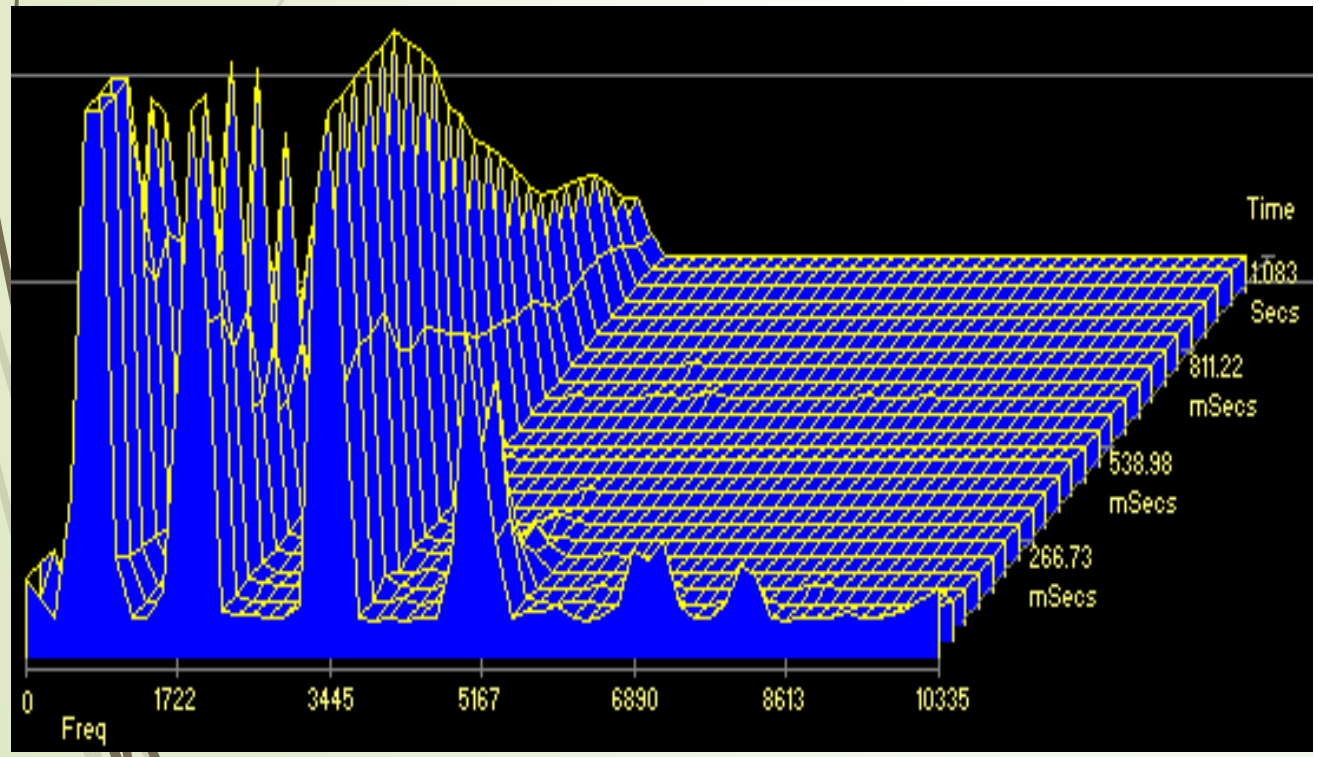
Spectre d'une note

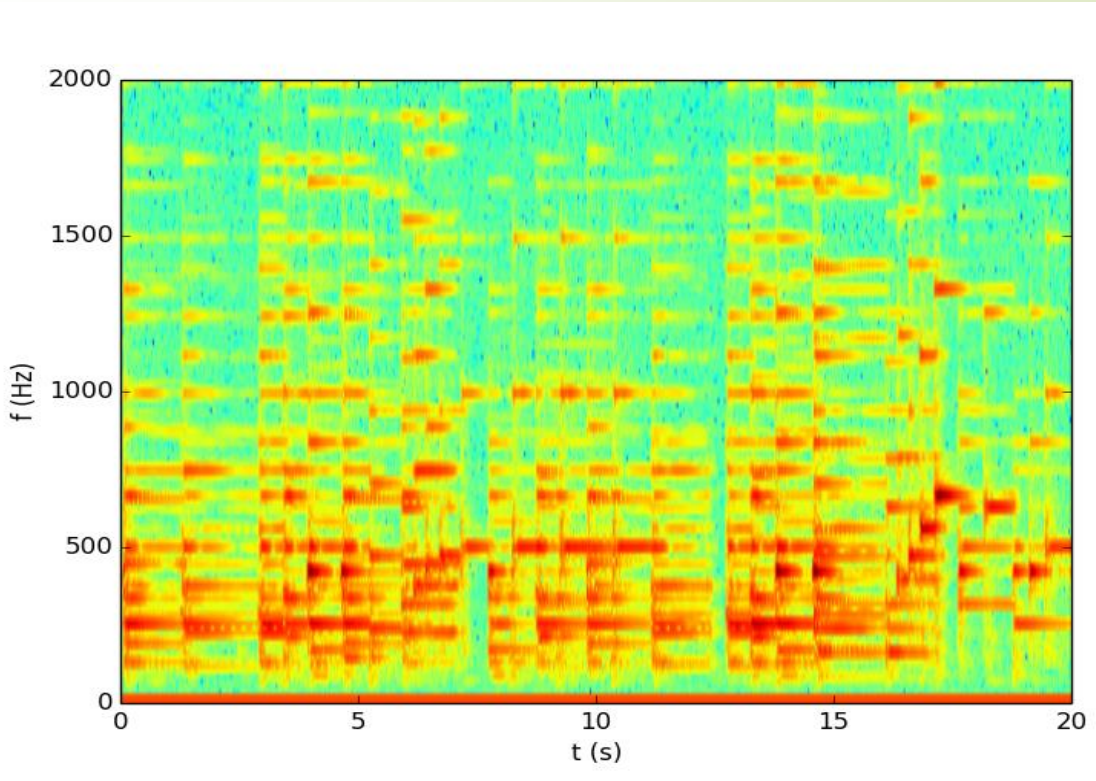
Ne correspond pas à des harmoniques

Spectre d'un accord

Analyse du signal: voie à explorer

- Utilisation de l'analyse temps-fréquence





Parser Midi
midi vers fichier intermédiaire

- Structure d'un fichier midi :
  - Métadonnées (version, nombre de pistes)
  - Suite d'événements (note on, note off mais aussi changement de tempo…)
  - Deux données par événement, dans le cas d'un note on/off la touche concernée et la vélocité.
  - Ces événements surviennent à un tick donné (compté depuis le début de la lecture)

Traitement du fichier midi

- 128 touches
- 12 notes (C, C#, D, D#, E, F, F#, G, G#, A, A#, B)
- Retrouver la note et son octave à partir de la touche
- Utiliser les ticks pour calculer les durées, silences et accords
  - Notation : 1 pour une ronde, 4 pour une noire, 8 pour une croche…
- Dans le fichier intermédiaire : nombre de notes, découpage de la mesure

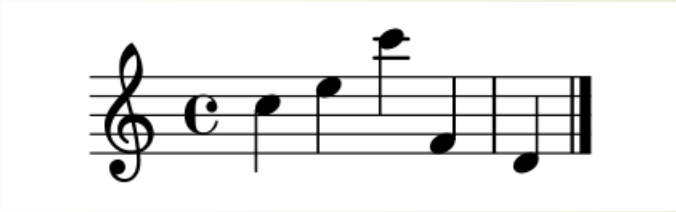
  Une ligne par note : note diese(0 ou 1) octave durée silence accord

Traitement du fichier intermédiaire en partition Lilypond

La gestion de la hauteur des notes:

Exemple :

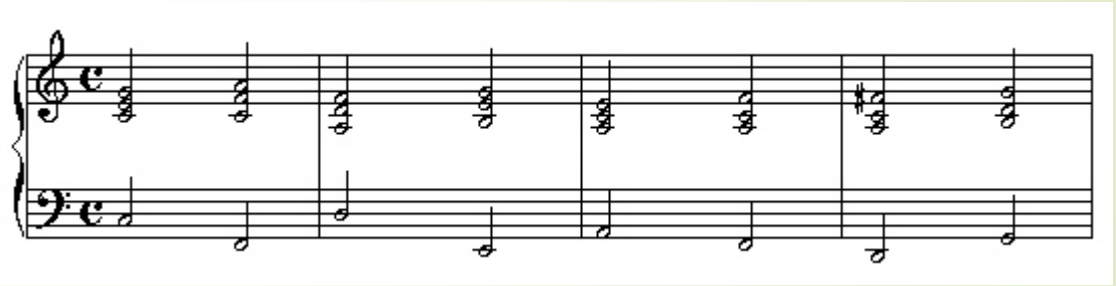
c' e' c'' f, d,



Un accord dans une partition :

Exemple :

< c e g >3



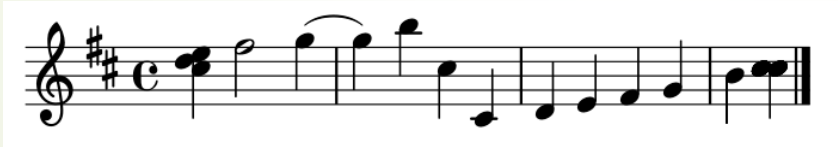
Traitement du fichier intermédiaire en partition Lilypond

Gestion de l 'armure de la partition par rapport à la gamme utilisée.

Exemple:

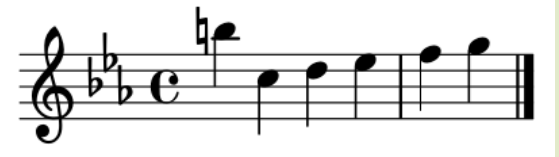
```
c 1 4 4 0 1
d 0 4 4 0 1
e 0 4 4 0 1
f 1 4 2 0 0
g 0 4 2 0 0
b 0 4 4 0 0
c 1 4 4 0 0
c 1 3 4 0 0
d 0 3 4 0 0
e 0 3 4 0 0
f 1 3 4 0 0
g 0 3 4 0 0
b 0 3 4 0 0
c 1 4 4 0 1
```

- En Dmajeur :



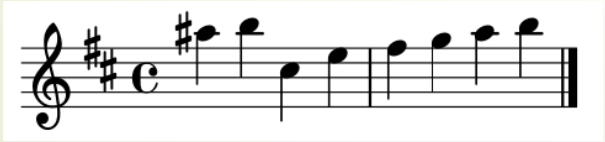
- En Cmineur :

```
b 0 4 4 0 0
c 0 4 4 0 0
d 0 4 4 0 0
d 1 4 4 0 0
f 0 4 4 0 0
g 0 4 4 0 0
```



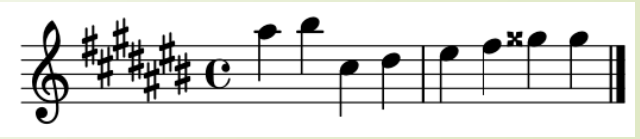
- En Bmineur

```
a 1 4 4 0 0
b 0 4 4 0 0
c 1 4 4 0 0
e 0 4 4 0 0
f 1 4 4 0 0
g 0 4 4 0 0
a 0 4 4 0 0
b 0 4 4 0 0
```



- En A#mineur:

```
a 1 4 4 0 0
b 1 4 4 0 0
c 1 4 4 0 0
d 1 4 4 0 0
f 0 4 4 0 0
f 1 4 4 0 0
a 0 4 4 0 0
```



Algorithme de reconnaissance de gamme

Reconnaissance_gamme(Tableau_notes){

Tableau_cherche_gamme<-remplir(tableau_notes)
Tableaux_gammes<-Initialisation_tableau_gammes();
Tableau_gamme_minimaliste<-Trie_notes(tableau_chercher_gamme);
Indice_gamme<-
chercher_gamme(Tableaux_gammes,Tableau_gamme_minimaliste;

Tableau_notes<-Modification_tableau_note(Tableau_notes,Indice_gamme);

On écrit dans le fichier Lilypond le nom de la gamme;
}

Algo Initialisation_tableau_gamme

```
Initialisation_tableau_gamme(){
        noms_tonalites[30]; // contient le noms des différentes tonalités
        tableau_diese[14]; //Contient les indices permettant de construire les gammes
                                        contenant des dièses par rapport à la précédente
        tableau_bemol[14]; //Contient les indices permettant de construire les gammes
                                        contenant des bémols par rapport à la précédente
        mineur_diese[7];//Permet d'ajouter la sensible des gammes mineur harmonique
        mineur_bemol[7];
        j=0,k=0;

        Pour i=2 à i<=14 de pas i+=2{
                Construction gammes avec dièses
                        Majeur:
                                tableaux_gammes[ i ][diese[ j ]] = 0;
                                tableaux_gammes[ i ][diese[ j +1]] = 1;
                        mineur
                                tableaux_gammes[ i +1][mineur_diese[ k ]] = 1;
                                tableaux_gammes[ i  +1][mineur_diese[ k] -1] = 0;
                Construction gammes avec bémol
                                Comme pour les dièses , indices différents
                j++,k++;
        }
}
```

Algorithme de reconnaissance de gamme

Reconnaissance_gamme(Tableau_notes){

Tableau_cherche_gamme<-remplir(tableau_notes)
Tableaux_gammes<-Initialisation_tableau_gammes();
Tableau_gamme_minimaliste<-Trie_notes(tableau_chercher_gamme);
Indice_gamme<-
chercher_gamme(Tableaux_gammes,Tableau_gamme_minimaliste;

Tableau_notes<-Modification_tableau_note(Tableau_notes,Indice_gamme);

On écrit dans le fichier Lilypond le nom de la gamme;
}

Algorithme découpage de notes

Une note en Lilypond est équivalente à une puissance de 2. Plus elle est élevée, plus la note est longue.

Découpage de notes (
Soit 2^i avec i = 0.
Si on trouve 2^i égal au temps restant, on affiche la note.
Si on trouve 2^i plus grand, on incrémente la puissance de 2.
Si on trouve 2^i plus petit, on calcule le temps de la note pointée.
En fonction de celui-ci, on redécoupe le temps restant en notes

On s'arrête si on a trouvé une combinaison de notes équivalente au temps ou si on a atteint 2^6 (équivalent à une quadruple croche).

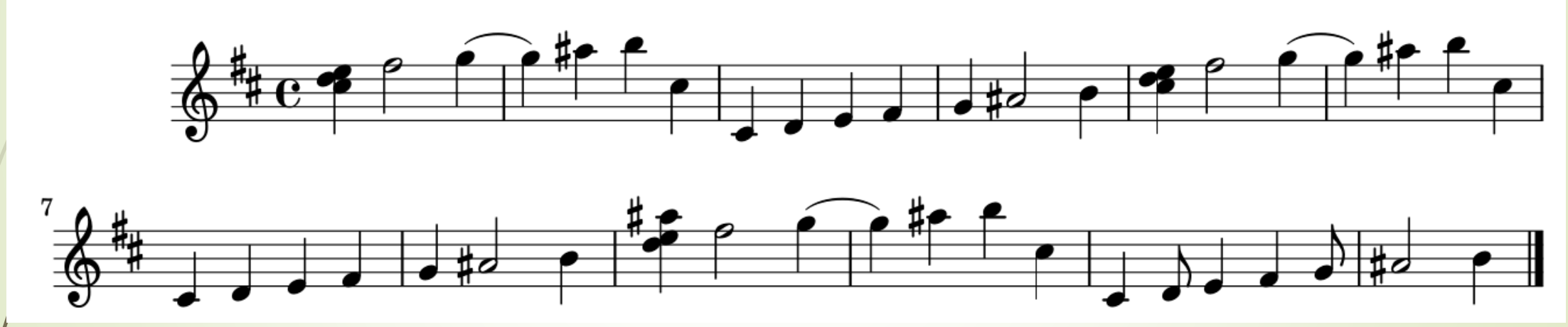
Le fonctionnement est similaire pour le temps après la barre.

Algorithme écrire

```
Temps_restant = temps_mesure
Pour chaque notes
        temps_restant -= temps_note
        Si temps_restant < 0
                Si on peux découper la note en deux de manière sysmétrique
                        On affiche la note ou l'accord
                On découpe la note
        Si temps_restant = 0
                On affiche la note ou l'accord
                temps_restant = temps_mesure
        Si temps_restant > 0
                On affiche la note ou l'accord
```

Traitement du fichier intermédiaire en partition Lilypond

Un exemple :



Démonstration