MASTER 2 ATIAM 2012 - 2013

Segmentation des séries temporelles pour l'orchestration automatique

Mémoire de stage

présenté par :
Maria Caterina Mannone

Directeur de stage :
Carlos Agon

IRCAM, Equipe de Représentations Musicales

1 mars - 31 juillet 2013

Table des matières

Abstract

In the field of increasing interest concerning automatic orchestration, we propose here a new method. This proposal is oriented towards the symbolic domain, differentiating from existing approaches which are related rather to the signal domain.

The aim of this project is to represent the typical approach followed by a composition student, a composer, or an orchestral conductor.

Our idea is closely linked to the visual approach of composers and conductors to the score : to imagine the overall effect of a score, without listening nor reading the notes, but by only glancing at the organization of lines, figures and blocks. The idea is to assign the solutions, learned by reading a score, to orchestrate a piano piece with similar drawings.

Here we take as a model *The rite of Sping* by Igor Stravinsky. The idea is first to automatically analyze the score, looking for musical figures per line and blocks of figures, and then applying the information obtained (e.g. the choice of instruments that play one or more gestures, the mix of instruments, other lines of a more decorative character) to similar figures identified in a piano piece for orchestration. In the first instance, we will concentrate on paradigms of segmenting time series, used here for the detection of musical figures.

Keywords : orchestration, informatics, segmentation.

Résumé

Dans le scénario d'intérêt croissant des tentatives d'orchestration automatique, on propose ici une nouvelle méthode. Cette proposition est orientée vers le domaine du symbolique, à la différence des autres approches existants, liées plutôt au signal.

On cherche ici à représenter l'approche suivie par un étudiant du cours de Composition, un compositeur ou un chef d'orchestre.

Notre idée est strictement liée à l'approche visuelle des compositeurs et chefs d'orchestre à la partition : imaginer l'effet général sans l'écouter ni lire les notes, en jetant seulement un coup d'œil à la disposition des lignes, des figures, des blocs. L'idée est alors d'attribuer les solutions, apprises en lisant une partition, au morceau pianistique à orchestrer ayant des dessins similaires.

Nous prenons, ici, comme modèle *Le Sacre du Printemps* par Igor Stravinsky. L'idée est d'abord de bien analyser automatiquement la partition, en cherchant les figures musicales pour chaque ligne et les blocs des figures, et d'appliquer les informations obtenues (choix des instruments qui jouent quelques figures, ou combinaisons, mélange des instruments, autres lignes au caractère plus décoratif) aux figures similaires identifiées dans un morceau pianistique à orchestrer. Dans un premier temps, nous nous concentrerons aux paradigmes de segmentation des séries temporelles, utilisés ici pour la recherche des figures musicales.

Mot clés : orchestration, informatique, segmentation.

Introduction. Comment peut-on enseigner l'orchestration ?

Lady Ada Lovelace l'avait prophétisé au milieu du XIXème siècle : nous pourrons un jour utiliser des machines à calcul pour la musique. En fait il y a dans la musique un double côté, de spontanéité dans l'idéation et d'obéissance à certaines règles ; deux aspects qui ne sont pas vraiment en contradiction.

L'amateur de musique n'ayant pas une grande confiance dans les nombres pourrait crier au scandale ; où ira l'art, si tout devient un produit créé par la machine ? Même l'étudiant des nombres qui aime écouter de la musique pour le plaisir pourrait pleurer en imaginant l'art mourir. En fait, l'automatisation appliquée à des œuvres d'art fait tout de suite revenir aux utopies négatives comme celles citées dans l'orwellien *1984*, où on parlait des machines pour écrire des textes et des accords pour des chansons destinées au peuple, ou bien aussi aux écrans pour la reconnaissance gestuelle (et le contrôle), des technologies qui ont déjà vu la lumière il y a longtemps (hélas !, aurait dit Orwell).

Même dans un cadre un peu plus académique, pas de machine, pas d'automatisation, seulement crayon, papier et bougies pour composer la nuit, nous pouvons toujours nous poser une question très délicate : qu'est-ce qu'on peut vraiment formaliser de la composition, ou, plus particulièrement, de l'orchestration ?

Quelques fois, il arrive que des compositions naissent de changements générés par l'émotion ou la maladie. Toutefois, dans le cadre du cours au Conservatoire, nous n'imaginons pas les professeurs faire pleurer leurs élèves ou leur donner des médicaments pour stimuler leur imagination.

Nous avons cependant des techniques qui peuvent bien être enseignées, un peu comme dans l'enseignement d'une langue : nous apprenons la syntaxe, l'orthographe et la grammaire.

Mais, dans l'écriture d'un roman comme dans les études de composition, nous ne pouvons pas enseigner la fantaisie ou la génialité ; en revanche, il est possible de définir des techniques pour bien utiliser et encadrer l'originalité. Le compositeur génial, même en suivant des règles, sait obtenir de merveilleux résultats. Les règles peuvent formaliser des techniques d'écriture, par exemple le contrepoint. En regardant un texte de contrepoint, on trouve des règles basées sur les lois de la consonance et la possibilité de distinguer différentes voix. Nous pouvons alors écrire des chefs d'oeuvres en suivant ces règles, et arriver à de bons résultats grâce à l'ordinateur - la machine par définition. Il est donc

possible et légitime de chercher à automatiser des processus utilisés dans le cadre de l'orchestration.

Dans l'orchestration nous retrouvons deux approches différentes : d'un côté, le monde symbolique lié à la partition, et de l'autre, le monde du son, lié au timbre. Une étude qui soit vraiment complète doit comprendre les deux côtés.

À l'Ircam, ce problème a été traité dans deux thèses doctorales ; il s'agit de deux travaux liés à l'approche signal.

Dans la thèse de G. Carpentier [8], on traite les combinaisons instrumentales pour trouver un certain timbre souhaité. Il est vrai que le problème des combinaisons instrumentales qui donnent un certain timbre est abordé dans tous les traités d'orchestration. Le logiciel *Orchidée*, dont les bases sont contenues dans cette thèse, permet aux compositeurs de trouver les combinaisons instrumentales qui donnent une approximation d'une *cible*, donnée *a priori*. Ce résultat est obtenu en considérant simultanément plusieurs paramètres, avec une optimisation combinatoire. Entre les descripteurs considérés il y a la largeur spectrale, l'enveloppe spectrale, la modulation d'amplitudes, les principaux partiels et les temps d'attaque. En utilisant vingt-six instruments, avec plusieurs modes de jeu, nous pouvons immédiatement imaginer l'impressionnant nombre de combinaisons possibles. Le problème est complexe et il sera résolu à l'aide des algorithmes génétiques (le logiciel propose différentes solutions, l'utilisateur humain choisit entre eux, et les prochaines solutions proposées par le logiciel changeront selon les choix).

Dans la thèse de doctorat de P. Esling [13], le travail est différencié de la thèse de Carpentier, car les descripteurs sont combinés et étudiés en tenant compte de leur évolution temporelle.

Dans ce travail de stage nous abordons le problème sous un autre angle, d'une façon différente, mais complémentaire aux approches citées. La nouveauté consiste dans la formulation d'une méthode qui soit exclusivement liée au domaine symbolique. Nous cherchons à imiter le processus d'apprentissage suivi par les étudiants lors du cours d'Orchestration. Tout d'abord, ils lisent dans des manuels [5, 23, 16, 1, 9] tout ce qui concerne les caractéristiques techniques des instruments orchestraux ; puis ils lisent des exemples des orchestrations bien ou mal réussies, pour créer, en autonomie, un vocabulaire initial pour une propre écriture personnelle.

Dans son étude, l'étudiant suit cette démarche : en lisant beaucoup de partitions (et en les écoutant), il apprend des exemples d'orchestrations plus ou moins bien réussies. Le bon élève, le compositeur qui maitrise son travail, et le chef d'orchestre, sont capables d'écouter avec leur oreille interne le son général d'une partition, mais aussi en regardant le dessin caché dans la partition même : avant de lire chaque note, nous pouvons visualiser les mouvements des lignes, les superpositions, les variations de densité des points, les groupements, les déplacements vers l'aigu ou vers le grave. Notre idée est alors de s'inspirer de la méthode de travail précédente pour réaliser une orchestration automatique.

Nous voulons ici faire abstraction des concepts d'harmonie, d'analyse musicale, de toutes les connaissances préalables dans le domaine théorique, pour chercher à concevoir une méthode pour ordinateur idéalement basée seulement sur l'analyse et la comparaison

des lignes.

Il faut, tout d'abord, analyser la partition choisie comme modèle. Le cœur du travail de stage sera la segmentation d'une pièce orchestrale en séquences avec des caractéristiques communes. Il s'agit d'une étape fondamentale pour en extraire des éléments primitifs qui nous permettront de réaliser d'autres mélanges orchestraux. Donc, au niveau informatique, le centre du travail sera l'étude des techniques de segmentation des séries temporelles [14, 12, 17, 28, 26, 15]. Nous segmenterons des pièces pour piano et des pièces orchestrales, choisies comme modèle pour l'écriture symphonique. Si nous trouvons des correspondances entre des séquences de la pièce pour piano et le modèle orchestral, nous appliquerons le choix des instruments et la distribution des figures musicales du modèle à la pièce à orchestrer, pour réaliser une nouvelle partition. L'idée de regarder l'orchestration comme ensemble des figures, des lignes, des blocs au niveau visuel a été proposée par Salvatore Sciarrino [24]. Elle a été récemment reprise dans le mémoire d'un Master à la Sorbonne [11], mais toujours au niveau de l'analyse. Son application à l'orchestration est à notre connaissance inédite.

Pour segmenter des partitions, nous testerons deux algorithmes inédits : l'inconstance [2], et la platitude de la courbe des coefficients de Fourier [3]. Les résultats trouvés seront concrètement appliqués à deux cas d'orchestration.

Le modèle choisi pour le stage est la partition du *Sacre du Printemps* de Igor Stravinsky dû à la variété des figures musicales contenues (fig. 1).

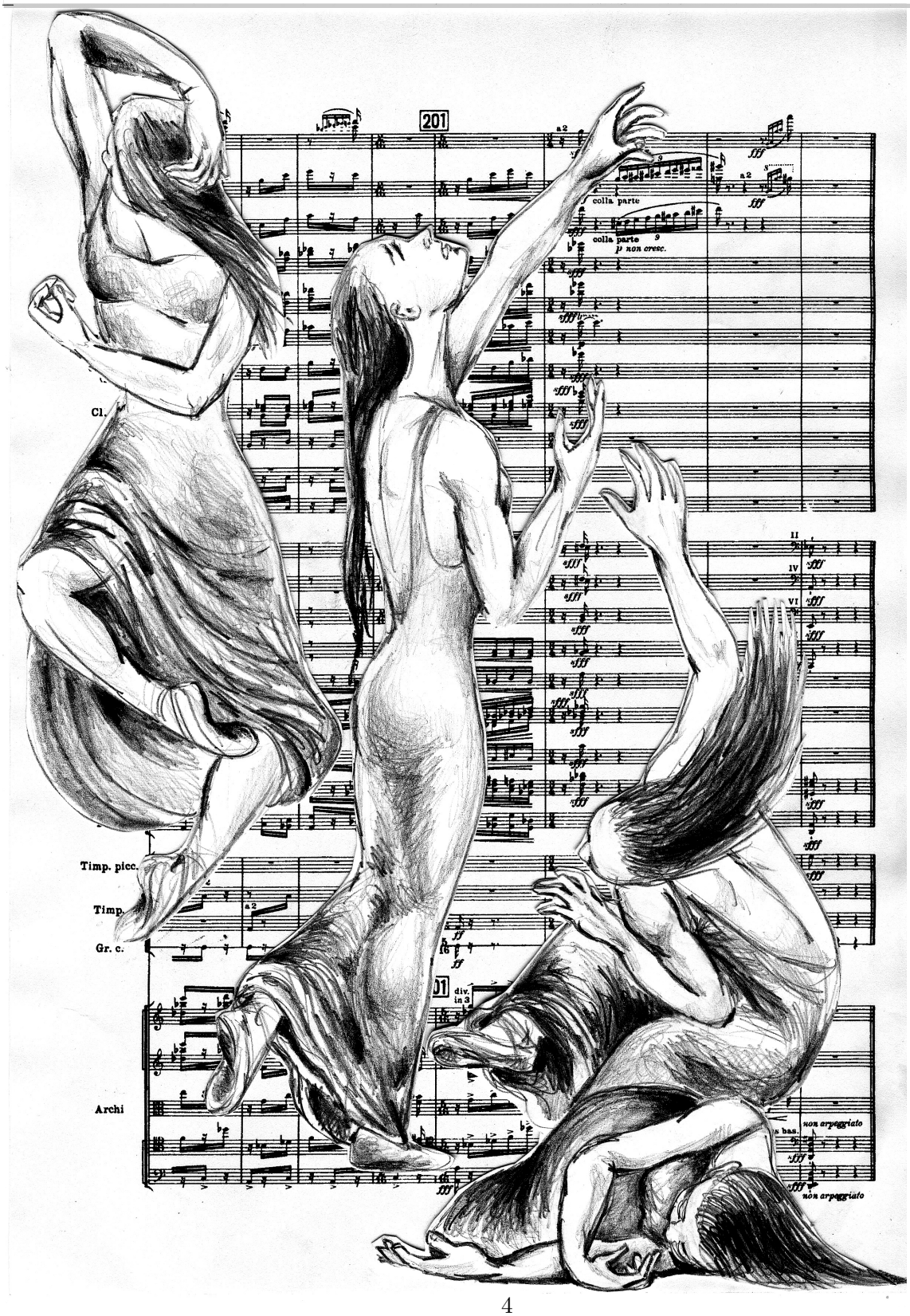


FIGURE 1 – Interprétation originale de la conclusion du *Sacre du Printemps*. Dessin de Maria Mannone.

Chapitre 1

Proposition d'une méthode d'orchestration automatique

Nous présentons ici un modèle d'orchestration général ; il s'agit d'un grand plan réalisable dans une échelle temporelle qui dépasse le cadre de ce stage. Toutefois, après une description de l'idée, nous attirerons l'attention sur une partie spécifique et très importante de ce plan, le problème de la segmentation des séries temporelles.

L'orchestrateur humain apprend des règles liées aux caractéristiques de chaque instrument : l'ambitus, les différences entre les registres, les nuances possibles et celles qui ne le sont pas. Ensuite, il regardera les règles sur des mélanges des instruments (souvent tirées des partitions), et, enfin, il étudiera des partitions écrites par les grands compositeurs, en cherchant à extraire des informations, qui, dans le cas humain, sont liées aux effets *qui plaisent et qui ne plaisent pas.*

Il s'agit, il faut bien le remarquer, seulement d'une des possibilités de travail ; toutefois elle est très simple, et possible à schématiser et ainsi à bien utiliser.

Cette méthode est fondée sur l'écoute des partitions, ainsi que sur le côté graphique. En prenant de la distance pour la lecture de la partition, il est évident qu'il n'est plus possible de reconnaitre la position de chaque note et donc de penser à sa hauteur absolue ; cependant, nous avons une idée de la distribution générale des *lignes*, des *figures* et des *blocs* des figures. Nous pouvons considérer une figure comme une ligne ou une combinaison de lignes ayant une importance musicale et des traits reconnaissables. Nous pouvons penser aux figures qui acquièrent une importance en étant répétées en blocs et, bien sûr, aux figures qui pourraient appartenir à quelque vocabulaire des primitives [7].

L'idée principale de ce stage est d'extraire les figures contenues dans une partition orchestrale, les comparer avec les figures dans une pièce à orchestrer, et, dans le cas de correspondance, d'effectuer l'assignation des choix instrumentaux pour les figures trouvées (mais aussi la création d'autres lignes 'de contour' et 'd'embellissement' s'il en existe dans le modèle) au morceau à orchestrer, en prêtant bien attention à garder les hauteurs originales dans ce morceau. En attribuant des parties d'embellissement, il faut détecter les rapports intervallaires avec les notes de la figure trouvée, et reproduire les mêmes intervalles dans le morceau. Il faut que le modèle soit le plus varié possible et

riche de figures. L'idée d'abord d'une analyse graphique est empruntée a la méthode proposée par Sciarrino dans son livre [24], où il voit toute la composition comme un jeu de variation, reproduction, multiplication, enrichissement, simplification des modules représentables à l'aide des graphiques. Cette idée graphique a été récemment utilisée pour la composition à partir des images [19] en généralisant l'idée xenakienne [27], et pour l'affichage graphique des caractéristiques du développement temporel des compositions musicales, comme les répétitions des séquences. En particulier, les représentations graphiques donnent une idée de la quantité de ces répétitions. Une proposition de mesure du degré de mémoire qui a été faite dans [20], inspirée par la Physique [21], est facilement représentable de manière graphique.

1.1 La méthode

Le schéma de l'image 1.1 donne une vision générale de ce processus d'apprentissage à l'orchestration. Il faut le regarder de gauche à droite. L'élève apprend les caractéristiques des instruments orchestraux (a), des mélanges des instruments (a/b), aussi en lisant des partitions (b). Dans la partition, nous cherchons aussi des figures et blocs, en regardant leur réalisation (choix des instruments, lignes complémentaires...). En parallèle, l'élève analyse la pièce à orchestrer, par exemple un morceau pianistique ; il identifie les figures, et, dans le cas de correspondance avec le modèle, l'élève utilise les choix instrumentaux effectués dans le modèle, en respectant les caractéristiques des instruments, et, dans des applications plus avancés, en gardant les caractéristiques harmoniques de la pièce à orchestrer. Chaque figure musicale sera représentée par des séries temporelles, et ainsi que chaque morceau pianistique à orchestrer. Si leur distance est minimisée (par rapport à quelque valeur de seuil à obtenir en comparant une pièce pianistique avec la même pièce orchestrée), les combinaisons correspondantes des instruments dans le modèle seront appliquées à la partie du morceau à orchestrer. La fig. 1.2 montre l'idée (les lignes isolées sont reportées dans la fig. 1.3). Il faut distinguer, à l'intérieur d'un segment, entre lignes maitres et lignes secondaires. Les lignes maitres dans le modèle sont celles qui correspondent au lignes dans la pièce à orchestrer ; les lignes secondaires sont celles qui n'ont pas de correspondances dans la pièce à orchestrer, mais qui peuvent lui être appliquées, pour ré-créer le même effet que le modèle. Le résultat de l'orchestration automatique souhaitée est montrée dans la fig. 1.4 (les lignes isolées sont reportées dans la fig. 1.5). Les groupes des lignes ayant la même couleur sont les *blocs* (figures répétées).

Les paramètres à extraire de la partition, qui vont générer nos séries temporelles, sont : hauteurs, temps d'attaques et durées, intensités. Nous focalisons ici l'attention seulement sur les paramètres hauteurs, d'abord, puis hauteurs et temps d'attaque. Il y a déjà dans la littérature plusieurs exemples de segmentation des séquences musicales [4, 7, 22]. Ce type de recherche a de grands points en commun avec la reconnaissance des motifs, en anglais *pattern recognition* [22]. Afin de rejoindre la plus grande généralité possible, nous travaillerons avec des algorithmes développés pour des séries temporelles génériques [14, 12, 28, 26, 15, 25, 2, 3].

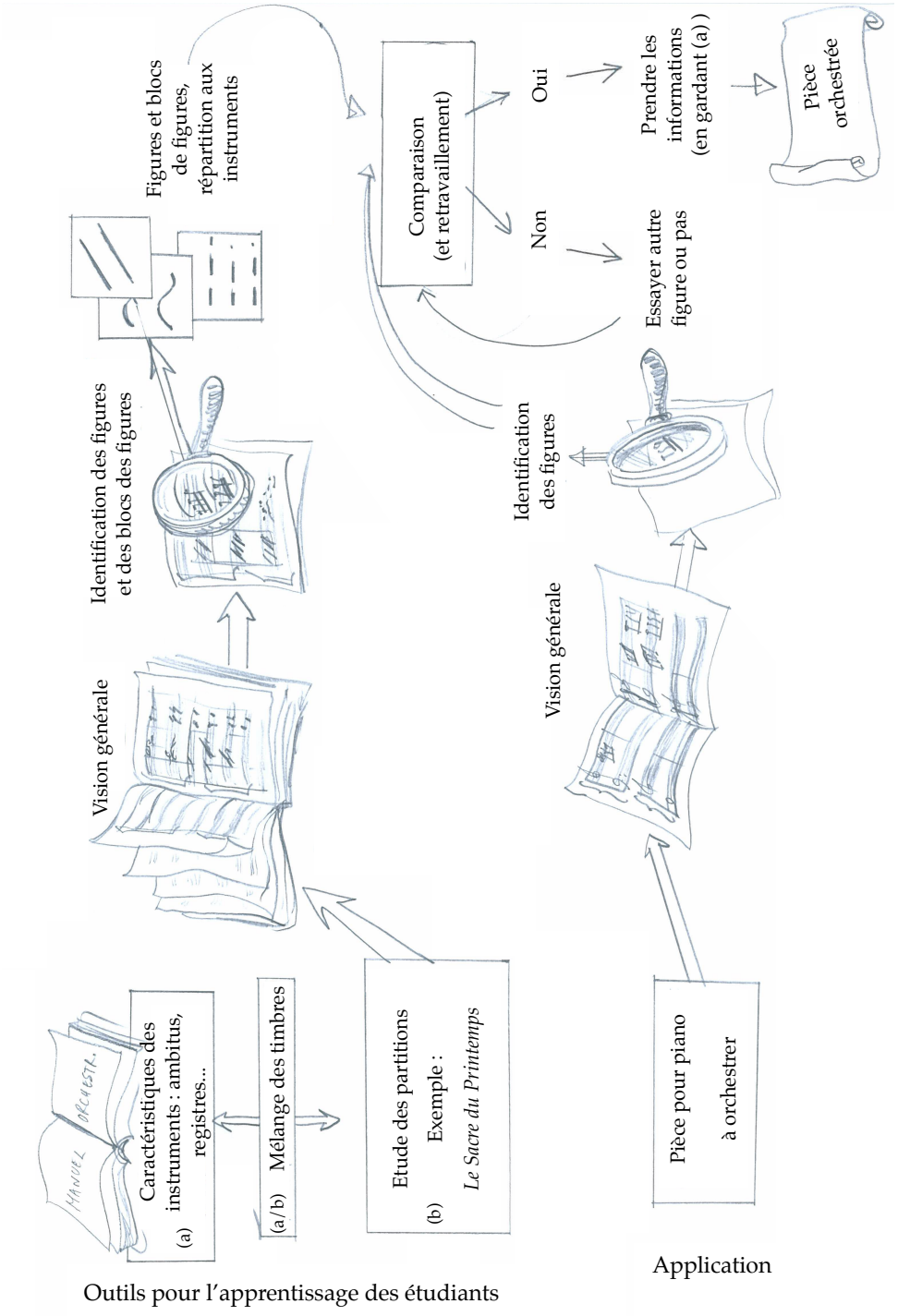


FIGURE 1.1 – Schématisation d'une méthode pour l'apprentissage humain à l'orchestration. (Dessins de Maria Mannone)

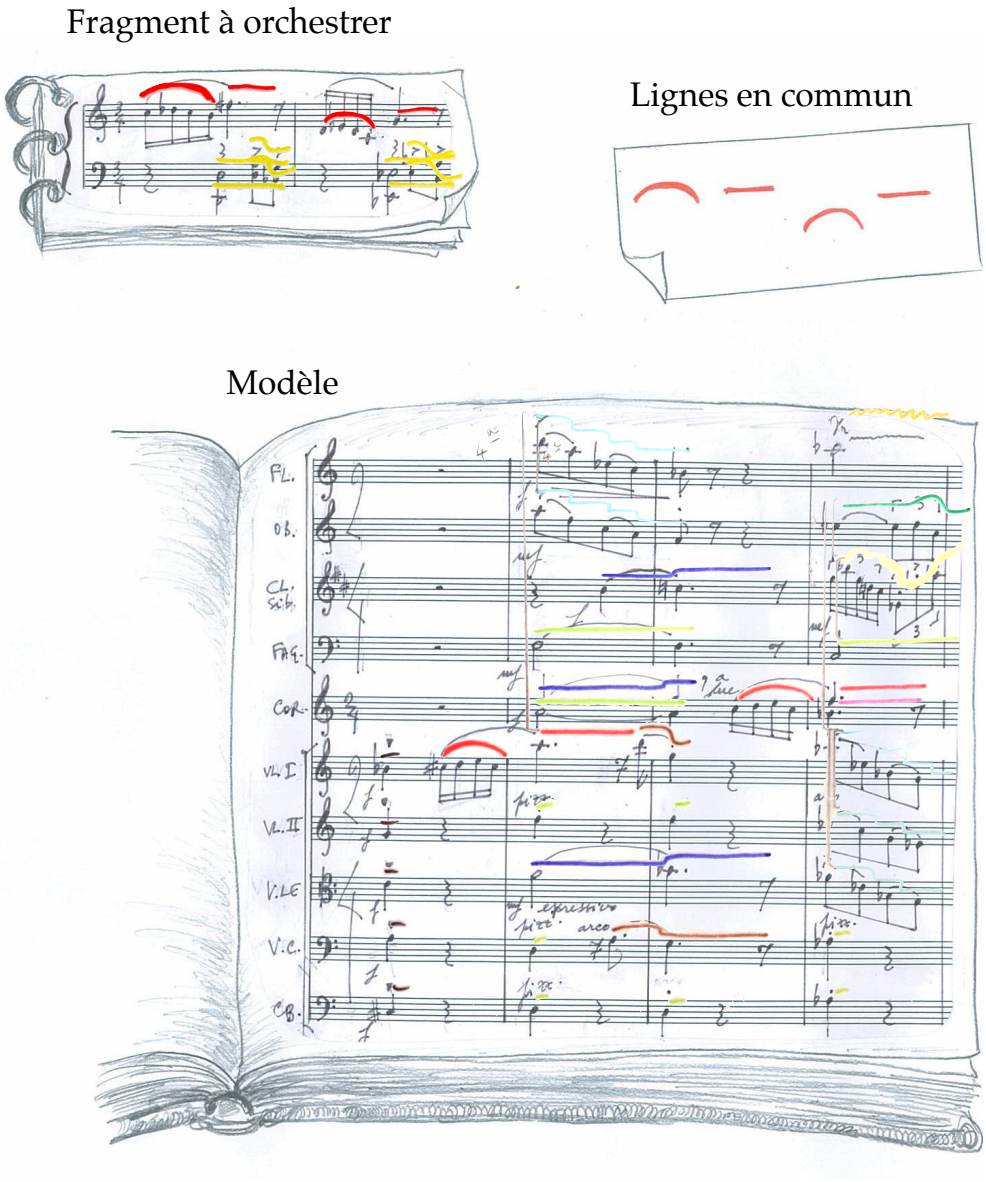
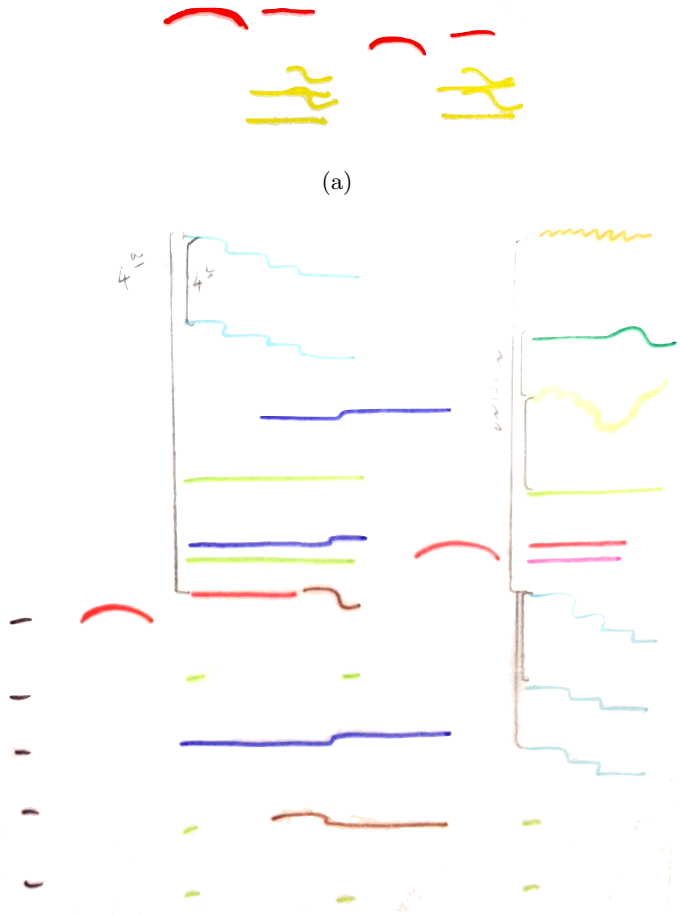


FIGURE 1.2 – Exemple de la méthode. Un morceaux pour piano (en haute à gauche) sera schématisé en différents lignes, une pour chaque figure musicale. Il faut chercher ces lignes dans un morceaux orchestral choisi comme modèle ; dans ce cas, les lignes en commun sont indiquées en haute à droite. Nous appellerons ces lignes *maitres*. Dans la partition orchestrale nous avons des lignes (indiquées à l'aide d'autres couleurs) qui ne sont pas présentes dans le morceau pour piano ; nous appellerons ces lignes *secondaires*. Il faudra les ajouter, en gardant les intervalles entre ces lignes et celles rouges. Dans le modèle orchestral, nous avons des groupes de lignes marquées par la même couleur : il s'agit des *blocs* de figures musicales identiques ou similaires. (Dessin de Maria Mannone)



(a)

(b)

FIGURE 1.3 – Détail des lignes dans le fragment à orchestrer (a) et dans le modèle (b), tirées da la fig. 1.2.

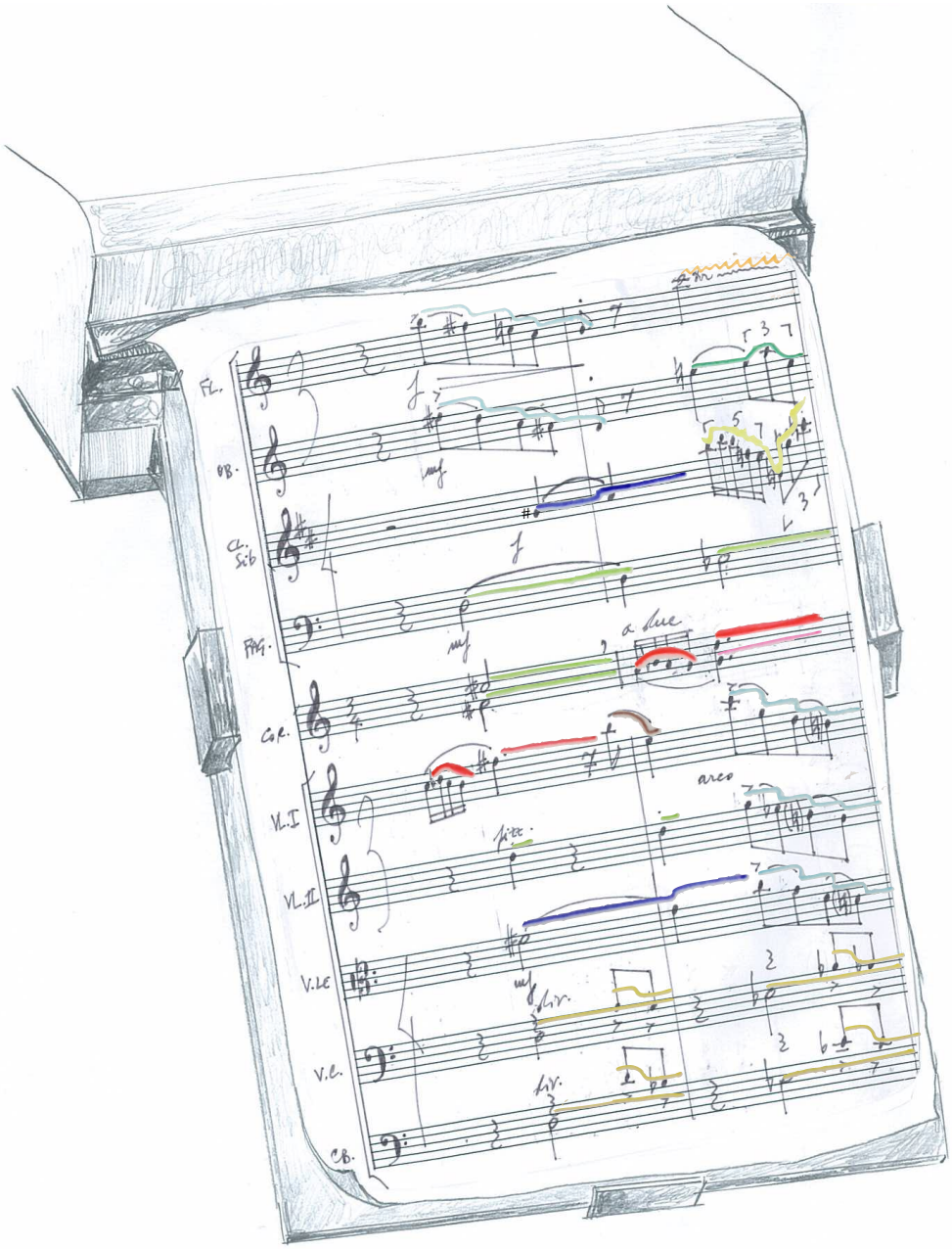


FIGURE 1.4 – Fragment orchestré sortant d'une imprimante : il s'agit de l'orchestration que nous souhaitons obtenir par l'ordinateur. Nous pouvons ici retrouver les lignes rouges assignées aux hauteurs du morceau pour piano, jouées par les instruments du modèle. Les autres lignes du piano (jaune foncé) sont assignées aux cordes graves ; les lignes secondaires (autres couleurs) sont attribuées selon les blocs et les intervalles du modèle, mais avec d'autres sons. Il ne s'agit cependant que d'un matériel de départ pour le travail du compositeur. Dans un prototype plus avancé, il faudra ajouter une autre phase, la ré-élaboration du modèle, pour introduire des variations. (Dessin de Maria Mannone)

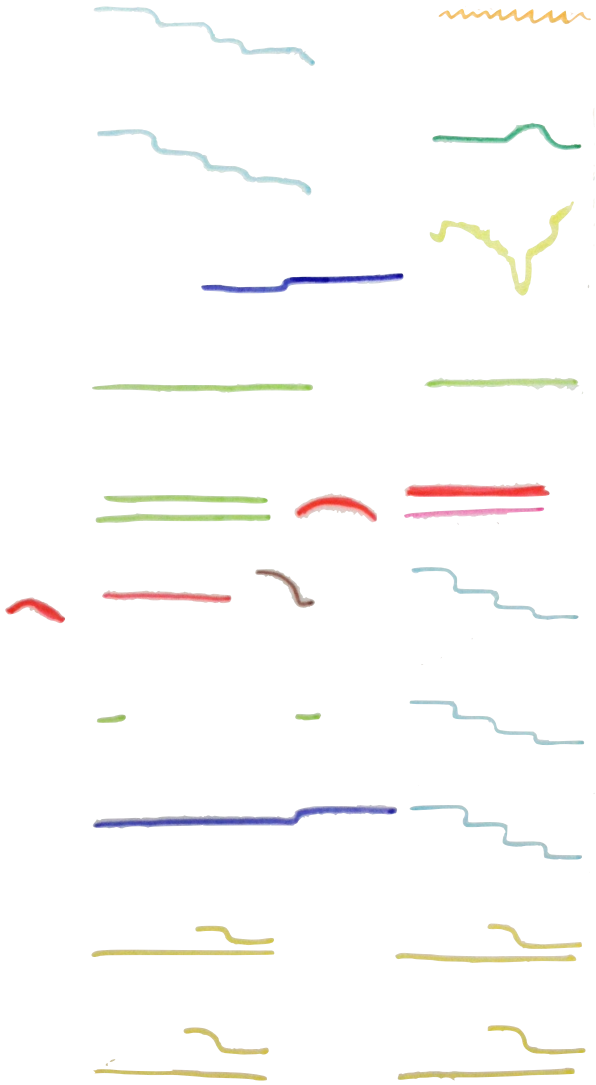


FIGURE 1.5 – Détail des lignes dans le fragment orchestré de fig. 1.4.

Chapitre 2

Segmentation

2.1 Séries temporelles

Les informations symboliques contenues dans une partition sont des suites de données (hauteurs, intensités...) dans le temps. L'outil mathématique fondamental pour les étudier est donc le concept de série temporelle.

Une série temporelle est une collection d'observations réalisées consécutivement au cours du temps. Plus formellement, nous pouvons définir une série temporelle T comme une séquence, temporellement ordonnée, de n variables à valeurs réelles [14]

$$T = (t_1, ..., t_n) \;\; t_i \in \mathbb{R}. \tag{2.1}$$

Une série temporelle est reliée à l'observation d'un processus, donc à l'observations des événements dans le temps. Nous pouvons avoir des séries temporelles multidimensionnelles : $T = ((t_{1,1}, \ldots t_{n,1}), \ldots (t_{1,d}, \ldots t_{n,d}))$, $t_{i,j} \in \mathbb{R}$. Il est possible d'obtenir des sous-ensemble des séries, donc des sous-séquences, de les comparer, d'en définir des distances réciproques, d'en évaluer la similarité, et d'en trouver des répétitions (à la recherche des *motifs*). La segmentation des séries temporelles s'impose logiquement comme l'outil pour extraire des motifs et, plus en général, des informations sur la quantité décrite par la série elle même.

2.2 Analyse des partitions à l'aide des séries temporelles

Dans le cadre de la méthode d'orchestration automatique proposée, un rôle décisif est joué par l'analyse préliminaire des partitions. Afin d'identifier ses différentes figures musicales et donc de constituer des blocs d'orchestration, il faut utiliser quelques algorithmes de segmentation. Nous ne voulons pas ici introduire des concepts de théorie et analyse musicale dans les programmes ; nous voulons schématiser la suite numérique de paramètres (hauteurs, temps d'attaque, durées, intensités) pour chaque instrument de l'orchestre (et pour l'instrument solo dont la partie doit être orchestrée) comme des séries temporelles. Le premier problème à résoudre est donc la segmentation des ces

séries. Nous avons, dans la littérature, des algorithmes de segmentation, différenciés selon les nécessités et le but de la recherche. Ici nous examinerons deux exemples d'algorithmes tirés de la littérature. Des résultats plus fiables pour le contexte musical seront obtenus en utilisant l'inconstance (chapitre 3) et la platitude des coefficients de Fourier appliqués au rythme (chapitre 4).

Une étude idéale pourrait considérer les quatre paramètres fondamentaux : hauteurs, temps d'attaque, durées et intensités. Toutefois, le problème de la segmentation au niveau technique présente déjà des difficultés sur une ou deux dimensions. Nous avons donc établi de commencer à implémenter, tester et comparer des algorithmes pour chaque paramètre isolé. Nous avons donc considéré d'abord le paramètre hauteur, en utilisant, comme indication du temps, simplement des indexes d'ordre croissant pour chaque note. Dans le cas d'un accord joué par la même voix, nous prenons la hauteur la plus aiguë.

Dans cette approche numérique, le but est de trouver une technique de segmentation qui soit la plus correspondante possible au résultat trouvé par l'humain, qui regarde une partition en cherchant des figures et une subdivision en séquences élémentaires.

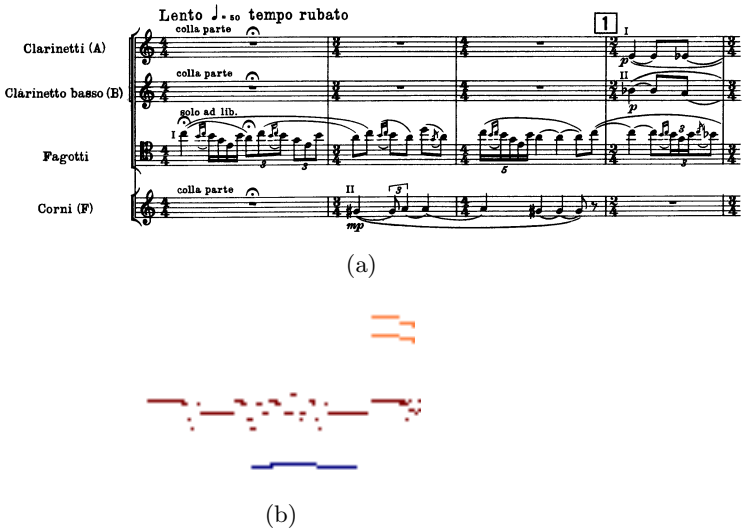
Il existe plusieurs travaux sur la segmentation et la reconnaissance des motifs [14, 12, 28, 26, 15, 25]. Il se trouve que le problème de la segmentation est beaucoup plus général ; ainsi la variété des algorithmes déjà développés dépend des différents buts pour lesquels nous cherchons à segmenter. Il y a, cependant, des algorithmes classiques, et aussi des modèles basilaires adaptables aux circonstances.

La reconnaissance des patterns pourrait être menée dans deux façons différentes : ou en cherchant des dessins répétées, ou bien en cherchant, dans la pièce à étudier, des séquences faisant partie d'un vocabulaire précédemment défini [22]. La question de la répétition est très importante, car la répétition en soi peut donner de l'importance et de la visibilité aux groupes des figures. Il y a, dans la littérature, pour le domaine du signal, des travaux qui ont pour objectif de trouver les séquences répétées, en définissant des distances et en calculant leurs minimisations [10].

Bref, l'idée suivie ici est d'implémenter des algorithmes de segmentation, en l'appliquant au *Sacre*, et en comparant les résultats avec des segmentations nouvelles données *a priori*. Dans la fig. 2.1 se présent la partition du début du Sacre *Sacre* et le correspondant extrait du fichier midi. Nous voulons segmenter les séries temporelles des paramètres dans le midi, qui nous donnent des indications pour la segmentation de la partition.

2.3 Algorithmes

Pour me familiariser avec l'environnement informatique choisi, j'ai commencé à appliquer deux algorithmes classiques de la littérature à la segmentation des pièces musicales. Nous présenterons, brièvement, cette expérience (les codes sont reportés dans l'annexe B).



(a)

(b)

FIGURE 2.1 – (a) Partition du début du *Sacre* ; (b) correspondante représentation temps-hauteurs du contenu du fichier midi, dans différents canaux.

2.3.1 Douglas-Peucker

L'idée est de tracer une ligne qui soit la régression linéaire pour tout l'ensemble des points ; puis nous calculons la position du point le plus loin de la ligne droite de régression, et nous calculons, à gauche et à droite, les différentes nouvelles régressions. Nous répétons le calcul, jusqu'à obtenir une erreur qui soit plus petite qu'une certaine limite fixée. La fig. 2.2 montre un exemple d'application. Cet algorithme, dans le cas des séries très longues et très variées comme celles du *Sacre*, ne donne pas de bons résultats. Ceci, du fait que la segmentation s'arrête là où la distance, entre les points de la courbe et la ligne de régression, est plus petite qu'un certain niveau établi par l'utilisateur. Son application aux exemples musicaux ne se révèle donc pas très pertinent.

2.3.2 Bottom-Up

Nous partons de la subdivision la plus subtile : un segment pour chaque couple des valeurs consécutives. Ensuite nous collons les segments, jusqu'à rejoindre une certaine valeur de désaccord entre les valeurs de la série et les points sur les segments de régression linéaire.

Si nous considérons, par exemple, la liste numérique (1 1 1 3 3 3 5 8 2 2 2 2), avec une erreur égale à 1, elle est segmentée comme ((1 1 1 3 3 3) (5 8 2 2 2 2)) ; avec un choix d'erreur beaucoup plus petite, 0.001, la segmentation est ((1 1) (1 3) (3 3) (5 8 2 2 2 2)). Dans le cas d'erreur égale à zéro, nous retrouvons la liste ((1 1) (1 3) (3 3) (5 8) (2 2) (2 2)), correspondante au début de la segmentation, avec un segment pour chaque couple de valeurs. Pour représenter en graphique les résultats de la segmentation, il faut obtenir des valeurs d'abscisse pour comprendre où couper les morceaux. Dans le cas, par exemple, de

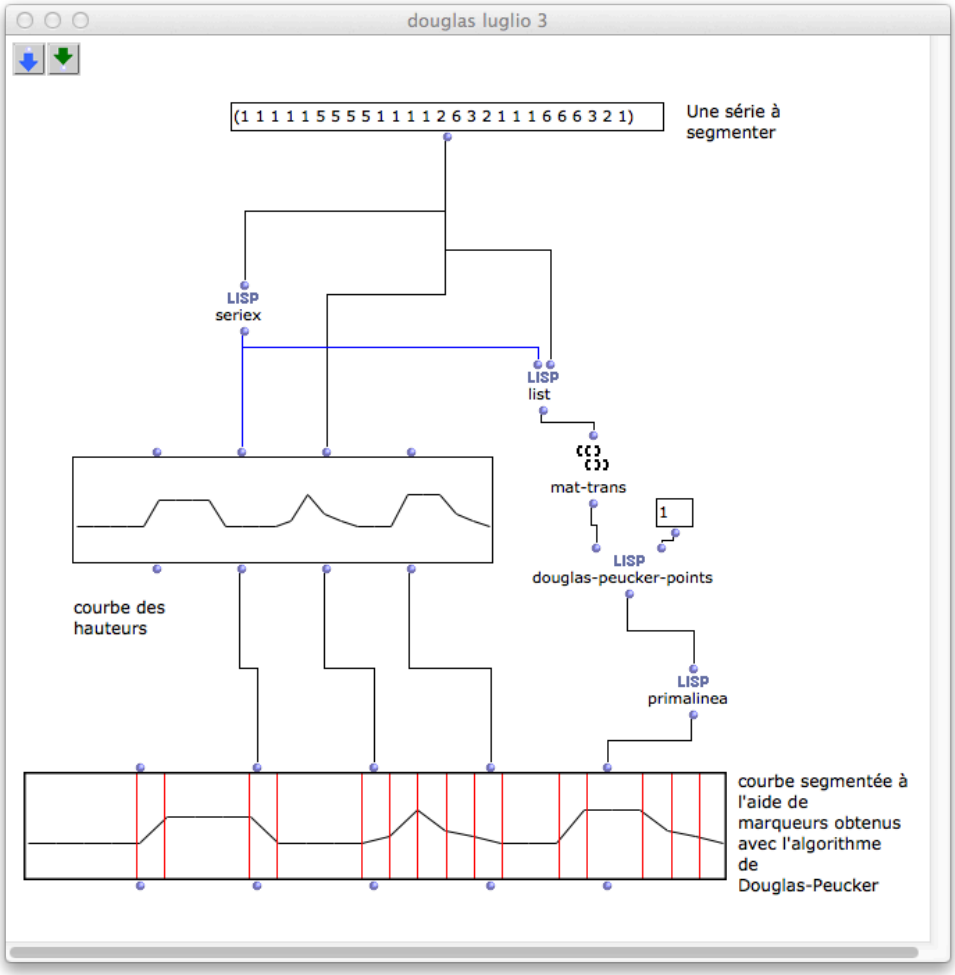


FIGURE 2.2 – Application de l'algorithme de Douglas-Peucker (paragraphe 2.3.1) dans un patch OpenMusic pour segmenter une série.

((1 2) (1 1 1)), nous avons deux branches, et chaque branche a une longueur différente. Nous utilisons alors une indication comme (0 2 5), où 0 est le départ de la première séquence, 2 de la deuxième et 5 d'une hypothétique troisième. La fig. 2.3 montre un exemple d'application à la même courbe de la fig. 2.2.

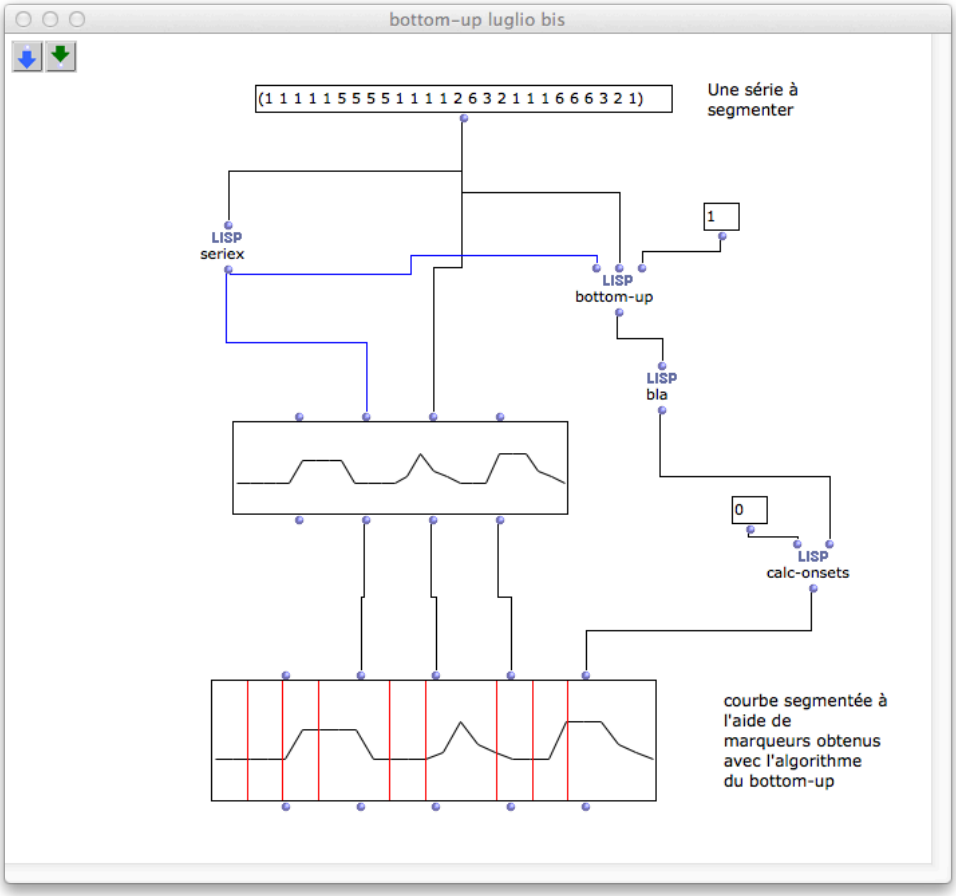


FIGURE 2.3 – Application de l'algorithme du bottom-up (paragraphe 2.3.2) dans un patch OpenMusic pour segmenter une série. En sortie, l'algorithme ne donne pas directement des marqueurs, mais la série segmentée : par exemple, (1 1 2 2) devient ((1 1) (2 2)). La fonction *bla* calcule la longueur des branches, donc 2 et 2. La fonction *calc-onsets* donne les positions des marqueurs ; dans le cas de cet exemple nous avons seulement un marqueur, posé sur 1 (en évaluant à partir de 0).

2.4 Paramètres et problèmes

Les algorithmes utilisés marchent bien dans le cas des séries temporelles très simples, avec un petit écart entre ses éléments. Dans le cas de notre intérêt, où nous nous proposons de segmenter des séries de l'ordre de milliers d'éléments, ayant une grande variabilité, ces algorithmes classiques ne se révèlent pas très satisfaisantes, même en réglant le paramètre de l'erreur. Dans l'algorithme de Douglas-Peucker l'erreur ϵ, choisi par l'utilisateur, est comparée avec la distance entre l'ordonnée de l'i-ème point et le segment de droite ayant la même valeur d'abscisse. La distance est définie comme

```
(defun distancePointSegmentK (i seriex seriey)
(abs (/ (- (retta i seriex seriey) (nth i seriey))
(sqrt (+ 1 (expt (b seriex seriey) 2))))))
```

La distance d-max est initialement égalisée à zéro, et si la quantité

```
(d (distancePointSegmentK i seriex seriey))
```

est plus grand que $d_{\max}$, on pose $d = d_{\max}$; donc, l'algorithme de segmentation est itéré d'une façon récursive si $d_{\max} \geq \epsilon$.

Nous allons étudier dans les chapitres suivants deux propositions de segmentation ainsi appliquées au domaine musical.

Chapitre 3

Inconstance

3.1 Définition mathématique

Bien que la constance dans l'esprit humain soit une qualité positive, dans le cas de la géométrie d'une courbe il est plus intéressant de regarder les propriétés d'inconstance. L'exemple de constance maximale d'une courbe est donné par une ligne droite, qui ne change jamais son inclinaison par respect d'elle-même. Cependant, les cas intéressants sont là où existent des variations. Comment les quantifier d'une façon simple et presque intuitive ?

Le problème d'une technique simple pour quantifier le degré du "désordre" d'une courbe a été récemment traité dans un article de J.-P. Allouche [2]. Ce travail utilise le théorème de Cauchy-Crofton, qui définit l'inconstance selon le nombre de croisements de la courbe étudiée avec une ligne droite d'inclinaison variable. En laissant tomber un crayon sur une courbe dessinée sur un papier, la probabilité de croisement avec la courbe augmente si la ligne dessinée est très éloignée d'une droite, avec donc des variations comme l'image d'un fleuve vue de dessus.

En des termes plus mathématiques, le nombre moyen d'intersections est donné de la façon suivante. Soient Γ une courbe plaine, $l(\Gamma)$ sa longueur, et $\delta(\Gamma)$ le périmètre de la courbe fermée qui donne l'enveloppe de Γ. Soit Ω un ensemble des lignes droites qui croisent Γ. Le théorème de Cauchy-Crofton récite que le nombre moyen de points d'intersection entre Γ et Ω est égal à [2] :

$$\mathcal{I}(\Gamma) = \frac{2l(\Gamma)}{\delta(\Gamma)}, \tag{3.1}$$

où $\mathcal{I}$ indique la quantité nommée comme *inconstance*, égal à ce nombre. Nous pouvons aussi quantifier la variabilité d'une courbe en utilisant d'autres critères, comme la variance résiduelle de la régression linéaire. Toutefois, nous avons des cas dans lesquels nous ne faisons pas de distinction entre deux courbes en utilisant la variance, mais elles sont bien distinguées à l'aide de l'inconstance.

FIGURE 3.1 – Les points 1 et 4 sont des autres côté du 2 et 3 (a) ; en suppriment le point 3, nous obtenons un segment de contour (b). En suivant la même méthode pour tous les points, nous obtenons l'enveloppe convexe d'un ensemble des points.

3.2 Implémentation

Nous avons utilisé un algorithme pour le calcul numérique de l'inconstance, en langage Common Lisp pour pouvoir directement l'utiliser dans l'environnement OpenMusic. L'inconstance $\mathcal{I}$ définie dans l'equ. 3.1 peut être écrite comme

```
(defun cauchy-crofton (points)
  (/ (* 2 (length-curbe points))
   (length-curbe (append (enveloppe-convexe points) ))))
```

où la fonction *length-courbe* mesure la distance entre chaque couple des points, en ajoutant les valeurs pour obtenir la longueur totale ; les lignes de code sont les suivantes :

```
(defun length-curbe (points)
  (let ((rep 0))
    (loop for p1 in points
          for p2 in (cdr points) do
          (setf rep (+ rep (point-distance p1 p2))))
          rep))
```

La fonction *enveloppe-convexe* est reportée dans l'annexe A.

L'idée est la suivante. Etant donné un ensemble de points dont nous voulons calculer l'enveloppe convexe, il faut tout d'abord prendre le point le plus bas, le lier avec tous les autres, et les numéroter dans le sens croissant des angles. Ensuite, en levant les segments mais en gardant l'énumération, nous partons toujours du point le plus bas, numéroté comme le premier, et nous relierons ce point avec le deuxième, puis le deuxième avec le troisième, puis le troisième avec le quatrième. Si les coordonnées du troisième sont plus 'internes' par respect au quatrième, nous relierons directement le deuxième avec le quatrième, en obtenant un segment de contour (fig. 3.1). Nous répéterons le même processus pour tous les points. En terme plus mathématiques, un segment est dans l'enveloppe convexe si tous les autres points sont du même côté [6].

Cet algorithme a été utilisé à l'aide d'une fenêtre glissante sur les éléments de la série temporelle. Le code pour la fenêtre est le suivant :

```
(defun finestra-mobile-sovrapp (seriex seriey taglia passo)
(let ((i 0))
  (loop while (< i (- (- (length seriex) 1) taglia))
        collect
          (let* ((sub1 (subseq seriex i (+ i taglia)))
                 (sub2 (subseq seriey i (+ i taglia))))
            (setf i (+ i passo))
            (cauchy-crofton (list2points sub1 sub2))))))
```

Il est possible de modifier la taille de la fenêtre (*taglia* en italien), et la vitesse d'avancement, l'étape (*passo*). Nous considérons deux séries en entrée, car nous pouvons, en principe, utiliser une série, par exemple y, pour les valeurs des hauteurs, et une autre, x, pour les correspondants temps d'attaque. Dans le cas de cette première application, nous avons choisi d'analyser un seul paramètre à la fois ; donc, si nous considérons les hauteurs, les x seront données seulement par des indexes d'ordre.

3.3 Application au *Sacre*

En général, si nous avons un brusque changement dans la courbe, nous pouvons prévoir une croissance de l'inconstance, et à l'inverse. Ainsi, nous pouvons utiliser ce paramètre pour la segmentation des séries temporelles, extraits d'une analyse musicale. L'idée est donc de chercher à obtenir quelques indications sur la segmentation des séries temporelles dans lesquelles avait été décomposée la partition du *Sacre du printemps*.

La méthodologie suivie a été de comparer les segmentations automatiques des courbes des hauteurs avec celles "modèle", réalisées personnellement. De fiables segmentations automatiques apparaissent selon le degré de leur similarité avec la segmentation manuelle.

La courbe d'inconstance obtenue après le glissement de la fenêtre a été segmentée en utilisant des variations de moyenne. Les marqueurs (qui délimitent les régions dans les courbes d'inconstance) ont été appliqués aux séries temporelles originelles, pour voir si la segmentation obtenue était bonne ou non, par rapport à l'exemple de *bonne segmentation* réalisé avant.

Le fichier utilisé pour cette expérience est celui du *Sacre*, du début jusqu'à la section 52 dans la partition orchestrale. Il contient un très grand nombre de figures musicales, et donc l'analyse acquiert beaucoup d'intérêt.

3.3.1 Segmentation des courbes des hauteurs

La technique d'analyse choisie est la suivante.

1. Nous prenons un canal du fichier midi, par exemple le cinquième, dont nous examinons la première ligne (correspondant au premier basson). Nous représentons les hauteurs en terme d'indices d'ordre. Nous segmentons la courbe obtenue d'abord en utilisant la segmentation manuelle, comme montre la fig. 3.2.

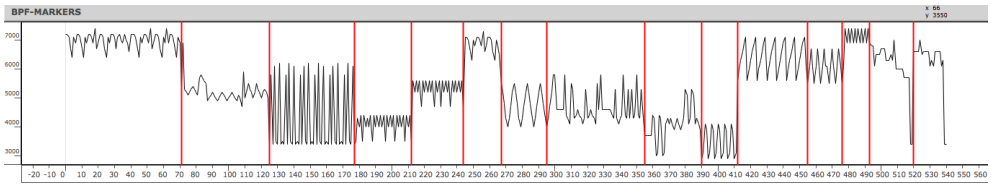


FIGURE 3.2 – L'image représente la segmentation souhaitée des hauteurs relatives à la première ligne des basson (cinquième canal du fichier midi), du début jusqu'à la section 52. Dans l'axe des ordonnées on a les valeurs d'hauteurs en midicents, dans l'axe des abscisses on a des indexes d'ordre pour chaque valeur. Nous souhaitons réussir à obtenir la même segmentation par la machine.

2. Nous trouvons alors la courbe d'inconstance (3.3), à l'aide de l'algorithme cité dans le paragraphe 3.2, en choisissant une taille de fenêtre égale à dix points (nous discuterons par la suite de ce qui concerne le choix de la taille) et des étapes d'avancement égales à un point.
   Cette courbe est segmentée à l'aide d'une simple fonction avec une fenêtre qui glisse (sans aucune superposition) et calcule les valeurs des moyennes, puis en compare chacun avec le précédent, et pose un marqueur si l'écart de la moyenne précédente est supérieur à un certain niveau établi, comme montre le code suivante :

```
(defun finestra_medie (seriey taglia passo errore)
(let ((i 0))
  (loop while (< i (- (- (length seriey) 1) taglia))
        collect
          (let* ((sub1 (subseq seriey i (+ i taglia)))
                 (sub2 (subseq seriey (+ i taglia) (+ i (+ taglia 2)))))
            (setf i (+ i passo))
            (if (> (variazione (calcolo_medie sub1)
            (calcolo_medie sub2)) errore)
            (progn (finestra_medie sub2 taglia passo errore)
              (setf mark i))
           (finestra_medie sub2 taglia passo errore))))))

(defun calcolo_medie (serieNum)
(/ (apply '+ serieNum) (length serieNum)))

(defun variazione (num1 num2) (expt (- num1 num2) 2))
```

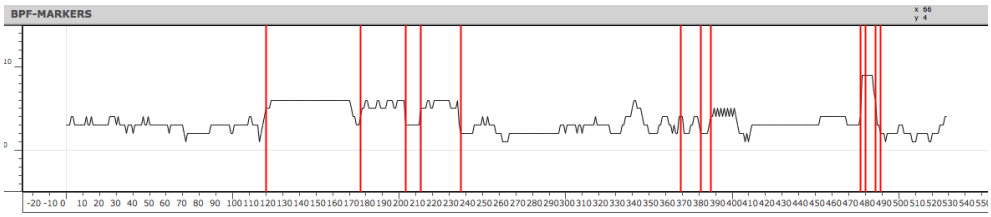


FIGURE 3.3 – Canal 5 du midi, fenêtre de taille 10, courbe de l'inconstance.

3. Enfin, les marqueurs obtenus ont été appliqués, toujours automatiquement, à la courbe originale des valeurs des hauteurs (fig. 3.4). Plus proche est la segmentation trouvée à celle obtenue manuellement, et meilleur est le résultat.

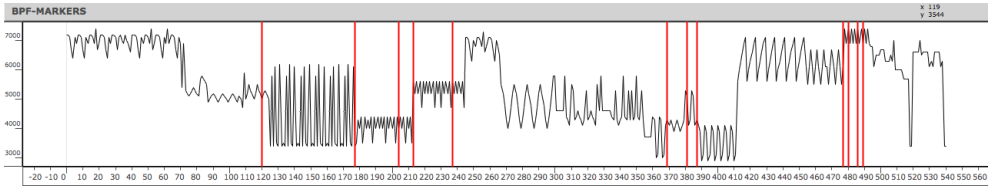


FIGURE 3.4 – Canal 5 du midi, courbe des hauteurs, segmenté à l'aide des marqueurs obtenus dans la figure 3.3.

Même si la comparaison entre les courbes peut être facilement réalisée de façon visuelle, nous avons besoin de quelques critères pour l'effectuer automatiquement.

3.3.2 La segmentation est-elle bonne ou non ?

Il existe toutefois encore des problèmes ouverts, comme, par exemple, un critère universel pour choisir la taille de la fenêtre. Deux critères ont été utilisés pour établir le résultat de la comparaison entre la segmentation automatique et la *bonne* segmentation manuelle ; cependant, ces critères dans certains cas ne donnent pas toujours de réponses correspondants aux expectatives.

Le premier critère utilisé consiste dans une comparaison entre le nombre de séquences trouvé à la main, et le nombre trouvé à l'aide de la segmentation automatique, divisé par le nombre de notes dans la série considérée. Plus petit est le nombre obtenu, et plus grande est la concordance entre segmentation automatique et segmentation souhaitée.

Le deuxième critère consiste dans l'évaluation de la largeur, en termes de nombres des points (des notes), de chaque séquence trouvée en utilisant la segmentation. Par exemple, étant donné la suite des marqueurs (a, b, c, d), le vecteur des largeurs est : $(b-a,\, c-b,\, d-c) = (\alpha, \beta, \gamma)$. Une autre segmentation du même morceaux pourrait donner (a', b', c', d'), et donc $(b'-a',\, c'-b', d'-c') = (\alpha', \beta', \gamma')$. Les régions sont-elles presque les mêmes, et donc peut-on isoler les mêmes figures musicales ? Il faut les comparer ; pour cela le module de la distance entre les composantes des vecteurs (α, β, γ) et $(\alpha', \beta', \gamma')$ a été évalué. Dans le cas où le nombre des séquences des deux segmentations à comparer est différent, dû à un différent nombre de marqueurs, nous considérons le nombre minimum de coordonnés entre les deux vecteurs. Comme dans le cas du premier critère, la fiabilité de la segmentation est témoignée par valeurs plus petites obtenues.

Ce deuxième critère se révèle être plus fiable par respect au premier.

Le code relatif au deuxième critère est le suivant :

```
(defun modulo-distanza (serie1 serie2)
  (let ((sum 0)) (do ((i 0 (1+ i)))
                     ((>= i (length (confronto serie1 serie2))))
                     (setq sum
```

```
                    + sum (expt (nth i (confronto serie1 serie2)) 2))))
    (sqrt sum)))
```

où :

```
(defun vettore-distanze (seriey)
  (let (valori)
    (loop for x in seriey
          for x1 in (cdr seriey) collect (- x1 x))))

          (defun confronto (vet1 vet2) (mapcar #'- vet1 vet2))
```

Clairement, il ne s'agit que d'une première méthode de comparaison. Le problème le plus relevant dans ce contexte, est que, dans le cas où il n'y aurait pas le même nombre de marqueurs, la comparaison sera faite sur un même nombre commun, le plus petit. Par conséquent, une partie des informations données par la segmentation issue d'un très grand nombre de séparations, à partir d'un point donnée, sera perdue. Dans la poursuite du travail, il faudra définir un critère qui puisse tenir compte de toutes les informations.

3.3.3 Résultats

Nous avons effectué plusieurs tentatives en variant différents paramètres des fenêtres pour le calcul de l'inconstance et de ces moyennes. Nous pouvons résumer ces résultats dans le schéma du tableau 3.1. Plus petites étaient les valeurs indiquées par les critères, et meilleur était l'accord. Nous utilisons toujours une étape d'avancement égal à 1.

3.3.4 Interprétation

Afin de trouver la segmentation automatique la plus proche de celle obtenue manuellement des courbes des hauteurs, des résultats ont été acquis en variant les paramètres relatifs à la taille de la fenêtre pour le calcul de l'inconstance et son étape d'avancement, et la taille de la fenêtre pour le calcul des moyennes.

Une dépendance spécifique des caractéristiques d'une séquence en particulier n'a pas été révélée. Nous avons, en revanche, trouvé de meilleurs résultats en utilisant des fenêtres de taille autour de 10 ou 20 points. Au-dessus et au-dessous de ces valeurs, en fait, nous avons une perte d'information utile. Il s'agit d'un résultat exclusivement expérimental.

Pour le canal 5, nous avos l'indication d'une fenêtre de taille autour de 10 points. Même si des informations qui ne dépendent pas strictement des caractéristiques de la courbe traitée ont été obtenues, il faut noter que nous n'avons pas encore trouvée une méthode qui soit complètement générale. Un autre obstacle à la définitive généralisation de la méthode est le passage intermédiaire du positionnement des marqueurs.

En particulier, pour le cinquième canal nous trouvons que dans la plupart des cas nous n'avons aucune indication des marqueurs entre la 70-ème et 80-ème note environ (pour le canal 5). En regardant la partition, cela en effet correspond à une transposition du même dessin vers le grave, avec une restriction de l'ambitus utilisé. Toutefois il s'agit d'un dessin très proche pour être révélé comme une variation d'inconstance significative. Dans ce cas, une suggestion importante a été obtenue pour l'analyse musicale d'une façon complètement automatique, fondée seulement sur l'analyse des séries temporelles. Il s'agit, en effet, du but de ce travail : chercher des algorithmes, ou bien des pistes à suivre pour des développements futures, pour une analyse complètement automatisée et basée sur le côté figuratif de la musique, à utiliser dans le cadre de l'orchestration.

canal midi	instruments	taille fenêtre inconstance	fenêtre moyenne : taille, erreur	critère 1 ($*10^{-4}$)	critère 2
5	basson	3	5, 0.2	55 (74)	229.09 (341.61)
5		5	3, 1	37 (18)	326.58 (435.88)
5		10	2, 1.5	80 (18)	195.01 (114.35)
5		15	5, 2	92 (110)	374.00 (452.59)
5		20	4, 2	92 (74)	123.00 (137.05)
5		50	4, 1.5	18 (0)	360.18 (477.77)
5		100	4, 1.5	40 (55)	627.38 (701.59)
5		200	4, 1	11 (110)	210.25 (319.70)
0	cordes sans *pizzicato*	5	5, 1	53	78.55
0		20	4, 1	80	341.48
0		50	10, 2	110	235.49
2	flûtes	5	5, 1	77	251.85
2		50	5, 2	52	341.49
8	cordes avec *pizzicato*	5	12, 0.5	101	617.19
8		50	4, 1.5	43	365.26

TABLE 3.1 – Tableau comparatif des résultats de l'application de l'inconstance à l'aide des fenêtres de taille variable pour la segmentation des séries des hauteurs. Les critères utilisés pour la comparaison sont décrits dans la section 3.3.2. Les valeurs entre parenthèse sont référées aux résultats de la comparaison avec des segmentations souhaitées ayant un marqueur de moins, comme expliquée dans la section 3.3.4.

Quand nous avons utilisé une taille de la fenêtre pour l'évaluation de l'inconstance bien plus grande que 10 ou 20 points (ou plus petite que 5), des résultats assez mauvais ont été obtenus. Dans le cas d'une fenêtre de 200 points nous perdons beaucoup d'informations, donc la méthode ne se révèle pas très satisfaisante, comme montrent les fig. 3.5 et 3.6.

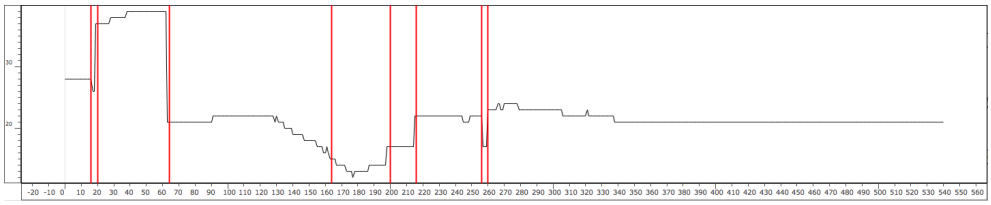


FIGURE 3.5 – Canal 5 du midi, fenêtre de taille 200, graphique de l'inconstance.

Pour le canal 8, avec un choix de 8 points de taille pour la fenêtre d'inconstance, nous obtenons un résultat assez bon pour les côtés, mais avec une segmentation trop subtile entre les indexes 190 et 260 (fig. 3.9) par respect à la segmentation manuelle (fig. 3.7). Ce détail pèse sur le résultat des critères (table 3.1), en faisant baisser le niveau de correspondance avec le

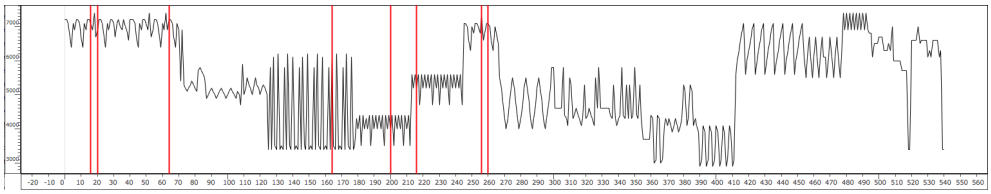


FIGURE 3.6 – Canal 5 du midi, graphique des hauteurs, segmenté à l'aide des marqueurs obtenus dans la figure 3.5. Nous pouvons voir que, en augmentant la taille de la fenêtre, on obtient des résultats de plus en plus mauvais. Nous obtenons des résultats pas très bons avec des tailles plus petites que 5.

modèle. Cependant, il ne s'agit pas d'un problème de l'application de l'inconstance, car dans la courbe de fig. 3.8 la séquence entre 190 et 260 est très régulière, même si le programme pour le calcul des moyenne en fenêtres glissantes relève, pour l'étape choisi, des variations qui justifient le positionnement d'un marqueur.

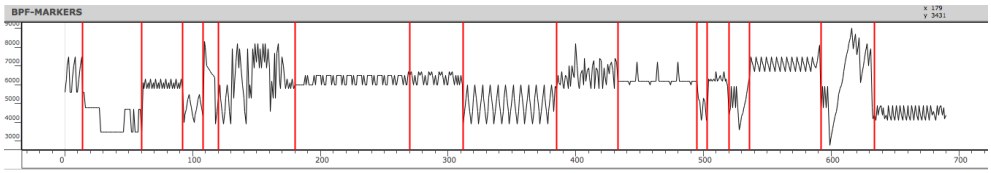


FIGURE 3.7 – Canal 8 du midi, exemple de la segmentation souhaitée.

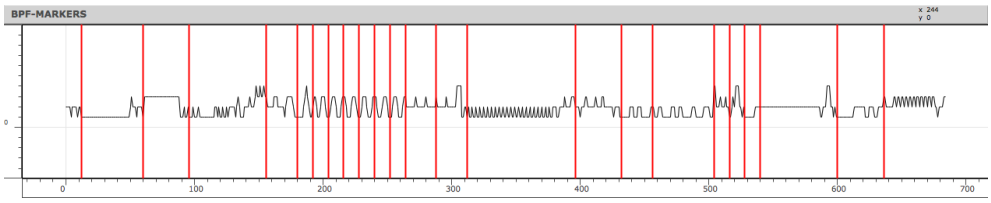


FIGURE 3.8 – Canal 8 du midi, fenêtre de taille 5, graphique de l'inconstance.

Une version plus avancée de cette partie pourrait avoir un réglage automatisé capable de choisir les paramètres utiles pour minimiser le désaccord avec la segmentation souhaitée. De plus, dans le cas d'un grand ensemble des œuvres utilisées comme modèle orchestral, il dévient presque impossible, en terme de temps à consacrer, d'obtenir les segmentation souhaitées (et il ne serait pas très utile). Nous pourrions essayer avec des petits extraits de la pièce-modèle, en trouvant les bons paramètres pour ce type de modèle, afin de commencer le travail automatique. Dans le cas des hauteurs, la méthode de l'inconstance a été utile parce-que la variabilité est exprimée par des valeurs de plus en plus différentes, et la constance par des valeurs très proches les unes des autres, ou à la limite toutes égales, des *pitches*. Toutefois, dans le cas des temps d'attaque nous avons des valeurs toujours différentes (sauf pour des notes en même temps, comme dans un accord). L'extrême régularité des temps d'attaque n'est pas indiquée, en fait, par des valeurs égales, mais par des nombres ayant toujours la même distance entre eux, comme des

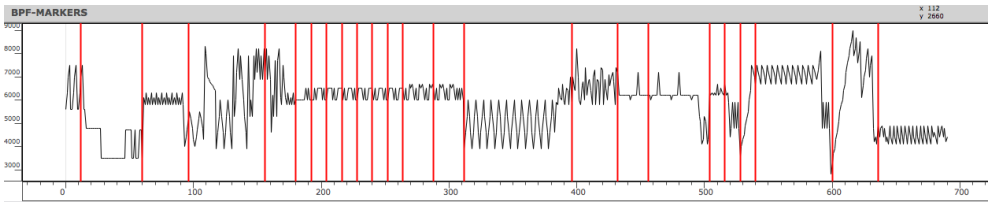


FIGURE 3.9 – Canal 8 du midi, graphique des hauteurs, segmenté à l'aide des marqueurs obtenus dans la figure 3.8. Il s'agit d'un des meilleurs résultats obtenus avec cette technique. Toutefois, les critères utilisés pour quantifier l'accord avec la segmentation manuelle (cf. fig. 3.7) ne donnent pas une information très fiable.

points équidistants sur une ligne droite ou bien sur un cercle (qui est l'équivalent topologique d'un segment de droite). Donc, pour l'analyse des temps d'attaque, nous utiliserons une autre technique.

Chapitre 4

Transformée de Fourier

4.1 Transformée de Fourier discrète pour l'analyse du rythme

La transformée de Fourier discrète a été utilisée pour distinguer les intervalles et la variabilité des intervalles dans les gammes, par D. Lewin [18, 3].

Considérons la transformée de Fourier de $f : \mathbb{Z}_c \to \mathbb{C}$:

$$\mathcal{F}(f) : t \to \sum_{k \in \mathbb{Z}_c} f(k) e^{-2ik\pi t/c} \tag{4.1}$$

La transformée de Fourier d'un sous-ensemble A dans $\mathbb{Z}_c$ sera la transformée de la fonction caractéristique 1_A de A [3] :

$$\mathcal{F}_A : t \to \sum_{k \in A} f(k) e^{-2ik\pi t/c} \tag{4.2}$$

Le k-ème coefficient de Fourier $\mathcal{F}(f)(k)$ donne la périodicité de la fonction f (périodicité égale à c/k).

Nous pouvons, également, définir le *contenu intervallique* du sous-ensemble A de $\mathbb{Z}_c$ par le nombre d'occurrence de chaque intervalle entre deux éléments de A :

$$IC_A(k) = Card\left\{(x, y) \in A \times A,\, t.q.\, x + k = y\right\}. \tag{4.3}$$

Un théorème [3] montre que la transformée de Fourier de cette quantité est égale au module carré de $\mathcal{F}_A$.

Encore une dernière définition : le sous-ensemble A, avec cardinalité d, est *bien réparti* si le module du d-ème coefficient de Fourier est maximale, donc si

$$\sqrt{\mathcal{F}(IC_A)(d)} = |\mathcal{F}_A(d)| \geq |\mathcal{F}_{A'}(d)|,\ \forall A' \subset \mathbb{Z}_C,\ Card\, A' = d. \tag{4.4}$$

L'extension de cette idée aux rythmes utilise l'idée de régularité d'une suite : un rythme régulier est une succession d'accents et de pulsations régulier ; chaque divergence de ce modèle peut être quantifiée, en donnant une mesure de la variabilité rythmique.

Le travail à suivre présente la première application de cet outil mathématique pour la segmentation, et pour l'analyse d'un morceaux de grandes proportions. Ainsi, nous avons décidé, pour quantifier la variabilité et donc caractériser les séquences des rythmes dans le *Sacre*, d'utiliser la transformée de Fourier.

4.2 La méthode

Nous pouvons visualiser la régularité d'un rythme à l'aide d'un cercle et de l'opération de *modulo*. Nous considérons par exemple une mesure de quatre quarts avec... tout simplement quatre quarts. Si nous utilisons un cercle discret avec huit points, nous aurons une disposition régulière des quatre noirs. Si nous avons des noirs et des croches, la disposition pourrait être moins symétrique, et ainsi de suite. Si nous voulons analyser deux mesures, nous pouvons prendre un cercle ayant un double nombre des points ; cela serait toutefois une solution triviale. En utilisant toujours un cercle avec le même nombre de points, soit N, et en calculant le *modulo* N des temps d'attaque des notes dans la deuxième mesure, nous pouvons disposer les points dans le même cercle, en visualisant s'il y a des régularités, des superpositions ou non (fig. 4.1 b).

Pour quantifier la variabilité d'un rythme, nous avons évalué la platitude des coefficients de Fourier. Comme montre l'image 4.2, les coefficients de Fourier sont différents pour rythmes différents ; mieux le rythme est réparti, plus élevée est sa platitude. La platitude est calculée selon l'écart de la moyenne des coefficients.

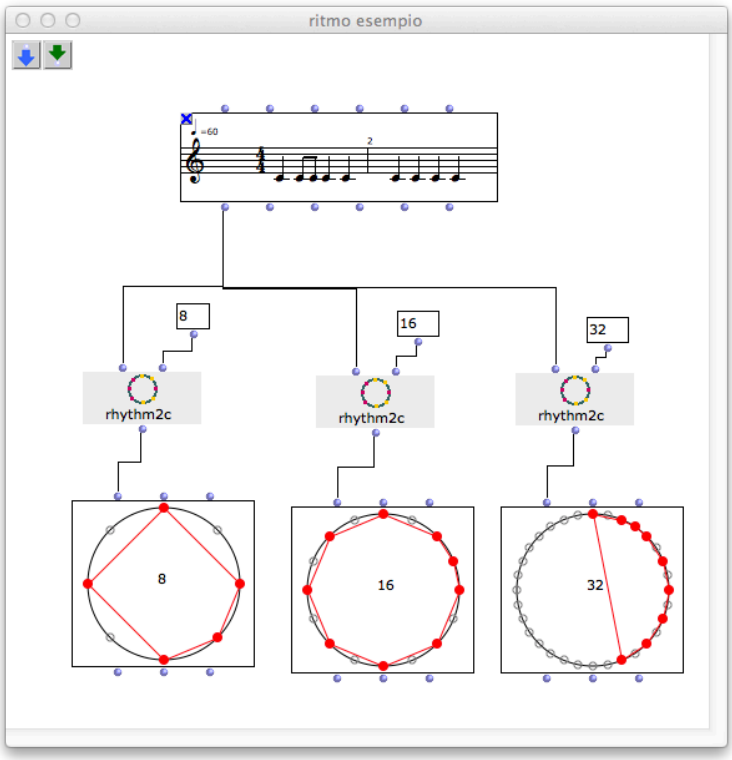
Si les mesures analysées ont des durées différentes, il est toujours possible de réaliser une analyse rythmique, en considérant toutefois une certaine valeur de taille, qui soit égale pour toute la pièce. Par exemple, dans le cas du *Sacre*, ont été comparés les résultats de l'analyse pour des fenêtres de durées en millisecondes. De plus, dans les figures 4.5 et 4.6, nous obtenons des courbes ayant dans l'axe des ordonnées les valeurs de platitude des coefficients de Fourier, et dans l'axe des abscisses, non plus des indices (comme dans le cas des courbes d'inconstance), mais des temps en millisecondes. Cette caractéristique permet de comparer tout de suite les marqueurs obtenus aux régions dans le graphique du fichier midi, et dans les fenêtres *chord-seq* de OpenMusic.

Vu la variabilité des rythmes et du *tempo*, pour analyser la distribution des temps d'attaques, une fenêtre temporelle générique de durée donnée a été utilisée, entre 1000 et 8000 millisecondes. Le nombre de notes avec un temps d'attaque à l'intérieure de la fenêtre a été calculé, en évaluant le *modulo* de ces valeurs de temps. Un intervalle de valeurs autour 4000 ms est justifié de la façon suivante. L'indication du métronome noir = 60 indique que, dans une minute, nous avons soixante noirs. Donc, chaque noir vaut une seconde. Dans une mesure de quatre quarts, nous avons alors quatre seconds, donc quatre mille millisecondes. Le début du *Sacre* contient une mesure de quatre quarts avec un métronome noir = 50, puis une mesure de trois quarts et ainsi de suite, donc une taille de fenêtre autour 3000 - 4000 ms se révèle appropriée.

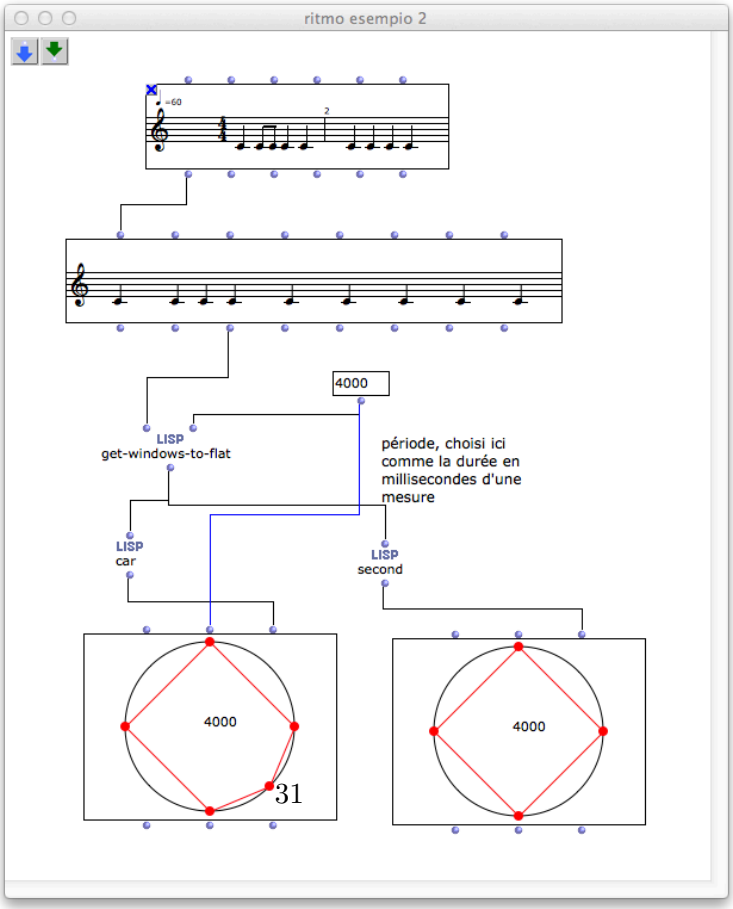
4.3 Application au *Sacre*

Nous considérons les temps d'attaque. Une fenêtre glissante calcule le nombre des attaques par unité de temps (par exemple 2000 ms), et leur position par rapport au modulo 2000. Le code est le suivant :

```
(defun get-windows-to-flat (list step)
 (let ((limit step) win rep)
   (loop while list do
         (if (< (car list) limit)
             (push (mod (pop list) step) win)
           (progn
             (push (reverse win) rep)
             (setf win nil)
             (setf limit (+ limit step))  ))) (reverse rep)))
```



(a)



(b)

FIGURE 4.1 – (a) Exemple d'utilisation d'un *n-cercle* pour la visualisation des temps d'attaque. (b) Autre exemple : nous utilisons ici une période en millisecondes.

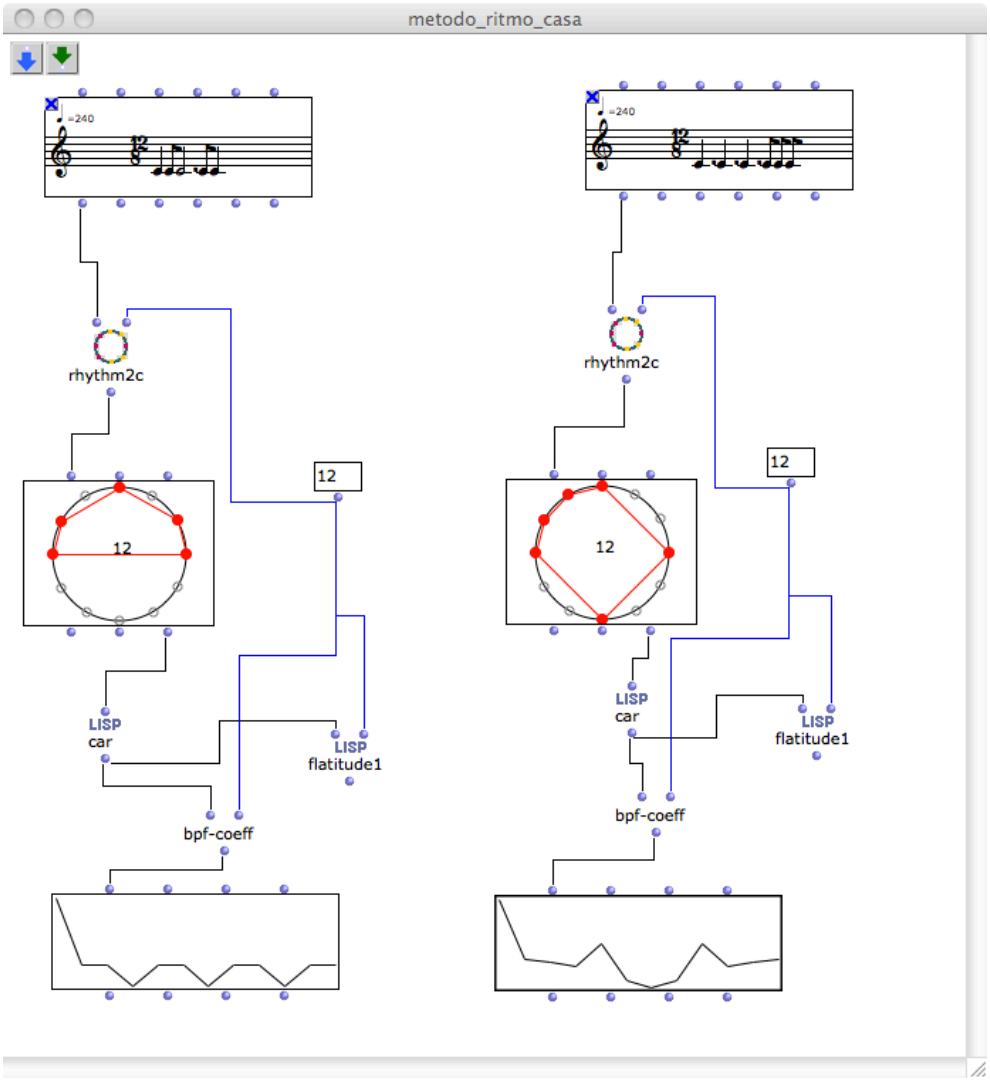


FIGURE 4.2 – Comparaison entre rythme différents à l'aide de la platitude des coefficients de Fourier. Dans le premier cas, plus irrégulier, nous avons une valeur de platitude égale à 0.44, dans le deuxième cas 0.91.

Les valeurs trouvées par fenêtre ont été visualisés dans un *n-cercle*. Les valeurs obtenues avec *get-windows-to-flat* sont à l'intérieur des listes ; pour chaque liste, de longueur variable, il faut calculer une valeur de platitude, à l'aide des coefficients de Fourier. Le code à utiliser est :

```
(defun finestra-flatitude-2 (seriey period)
  (loop for item in seriey collect
      (if item (if (= (length item) 1) 0.0001 (flatitude1 item period)) 0)))
(defun ecartype (list)
  (let* ((n (length list))
        (moyenne (/ (reduce '+ list) n)))
    (sqrt (/ (loop for item in  list sum
                   (sqr (- item moyenne))) n))))

(defun flatitude1 (list period)
  (let ((coeff (norm-coeff-list list period))) (ecartype (cdr coeff))))
```

Dans le cas où nous voulons utiliser une fenêtre glissante avec des superpositions, nous pouvons remplacer les lignes suivantes dans *get-windows-to-flat* :

```
  (defun get-windows-to-flat-Gliss (list step size)
```

et

```
 (push (mod (pop list) size) win)
```

La segmentation automatique, déjà décrite dans le paragraphe 3.3.1, a donc permis de segmenter automatiquement les courbes de la platitude. Afin de bien poser les marqueurs sur le graphique de la platitude, en respectant les valeurs d'abscisse en millisecondes, nous avons utilisé le code :

```
(defun aggiungi-decimali (seriey valore)
  (append (loop for i from 0 to (- (length seriey) 1)
  collect (* (nth i seriey) valore))))
```

La comparaison entre segmentation obtenue et segmentation souhaitée a été effectuée à l'aide des critères décrits dans la section 3.3.2. Une comparaison des résultats trouvés est donné dans le tableau 4.1, et, à l'aide d'une fenêtre glissante avec des superpositions, dans le tableau 4.2. Le patch de OpenMusic utilisé pour l'analyse est reporté dans la fig. 4.3.

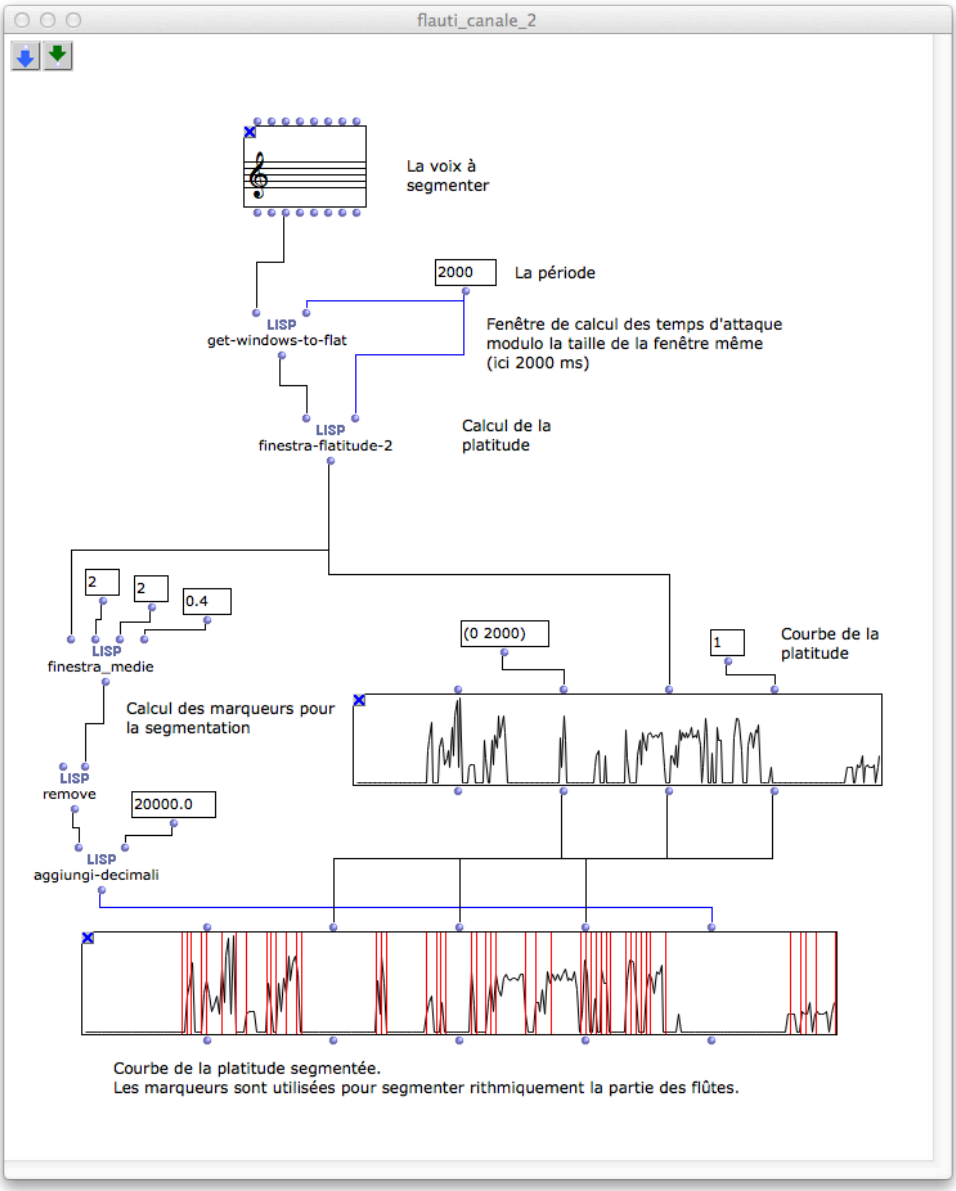


FIGURE 4.3 – Patch de OpenMusic utilisé pour l'analyse de la platitude, pour le canal 2.

canal midi	instruments	taille fenêtre	critère 1 ($*10^{-3}$)	critère 2 ($*10^{7}$)
0	cordes	1000	32	1.21
0	sans	2000	34	1.16
0	*pizz.*	4000	23	1.30
0		8000	38	1.14
5		1000	0	1.65
5	basson	2000	0.74	1.66
5		4000	19	1.94
5		8000	29	1.58

TABLE 4.1 – Tableau comparatif des résultats de l'application de la platitude aux canaux du fichier midi du *Sacre*, pour l'identification des séquences rythmiques. Les résultats donnent des informations pour la comparaison entre segmentation manuelle (réalisée directement sur la partition midi et sur les objets *chord-seq* de OpenMusic), et segmentation obtenue à l'aide de la platitude. Les critères utilisés pour la comparaison ont été définis dans la section 3.3.2. Pour le canal 0, un choix des paramètres pour la segmentation (taille de la fenêtre de comparaison, étape, écart) de 2, 2, 0.5 a été utilisée ; pour le canal 5, de 2, 2, 0.4.

canal midi	instrument	dimension fenêtre	étape 500 critère 1 ($*10^{-3}$)	étape 500 critère 2 ($*10^{7}$)	étape 800 critère 1 ($*10^{-3}$)	étape 800 critère 2 ($*10^{7}$)
0	cordes	1000	85	9.04	120	8.55
0	sans	2000	85	9.04	120	8.55
0	*pizz.*	4000	85	9.04	120	8.55
0		8000	85	9.04	120	8.55
5		1000	40	1.37	10	1.80
5	basson	2000	40	1.37	10	1.80
5		4000	40	1.37	10	1.80
5		8000	40	1.37	12	1.10

TABLE 4.2 – Tableau comparatif dans le cas de l'utilisation d'une fenêtre avec des superpositions correspondantes aux étapes de 500 ou 800 ms, et comparaison avec la segmentation souhaitée. La technique d'analyse est la même utilisée dans le table 4.1. Pour le canal 0, un choix des paramètres pour la segmentation (taille fenêtre de comparaison, étape, écart) de 2, 2, 0.5 a été utilisée ; pour le canal 5, de 2, 2, 0.4.

4.4 Interprétation

Les résultats des tableaux 4.1 et 4.2 montrent que nous n'avons pas, dans le cas de la platitude, une grande différence dans le choix de la taille de la fenêtre, face à la majeure variabilité des résultats pour le calcul de l'inconstance (tableau 3.1).

Dans les deux analyses, le facteur multiplicatif devant chaque terme du deuxième critère ne doit pas faire peur, car ces résultats tiennent compte des différences des distances entre les marqueurs des différentes segmentations, distances qui, dans le cas considéré, impliquent des grands nombres (étant chaque pas dans les graphiques de 1000 ms et pas de 10 points, comme dans le cas de l'inconstance, où les valeurs d'abscisse étaient seulement des indices d'ordre).

Pour chaque critère, plus petit est le nombre obtenu, et meilleure est la concordance entre segmentation automatique et segmentation souhaitée. Conceptuellement, le deuxième critère est plus fiable que le premier, mais aussi le deuxième doit être perfectionné (comme il a déjà été expliqué dans la section 3.3.2).

Nous obtenons une bonne coïncidence avec la segmentation manuelle dans le cas, par exemple, du canal 0 à l'aide d'une fenêtre d'analyse de 2000 ms. Dans la section 13 de la partition du *Sacre*, *Les Augures Printaniers*, les figures en obstiné rythmique de croches aux cordes, qui commencent au peu près de $20.82 * 10^4$ ms du midi, instant qui correspond au point $20.81 * 10^4$ ms dans la courbe de la platitude (qu'il vaut environ 1.1). Un dessin différent, avec des triolets piquées qui se trouvent autour de $28.5 * 10^4$ ms, est bien détecté par un marqueur posé automatiquement à $28.4 * 10^4$ ms. Le graphique de la platitude est reporté dans la fig. 4.4. Une comparaison avec la séquence dans le *chord-seq* de OpenMusic segmenté manuellement est reportée dans la fig. 5.4, car cette analyse a été reprise dans le chapitre 5, pour l'application du modèle stravinskien à une partition à orchestrer, contenant des figues plus proches.

Dans le cas d'une fenêtre d'analyse qui glisse avec des superpositions, nous obtenons un recouvrement plus fin, mais qui donne parfois des résultats globaux plus moyennés, et donc moins outils pour distinguer les figures rythmiques, spécialement dans le cas des étapes d'avancement égale à 500 plutôt qu'à 800 (fig. 4.4 et 4.6). En effet, le tableau 4.2 donne des valeurs inférieures, donc meilleures, pour une étape de 800 que pour 500 dans le cas du canal 0. En présence de superpositions, les résultats ne dépendent pas de la taille de la fenêtre, mais de l'étape d'avancement.

Les informations qui proviennent d'une analyse de la platitude, étant déjà référées aux temps, peuvent donner des indications très fiables pour l'identification des blocs, avec une importance cruciale pour l'étude de l'orchestration.

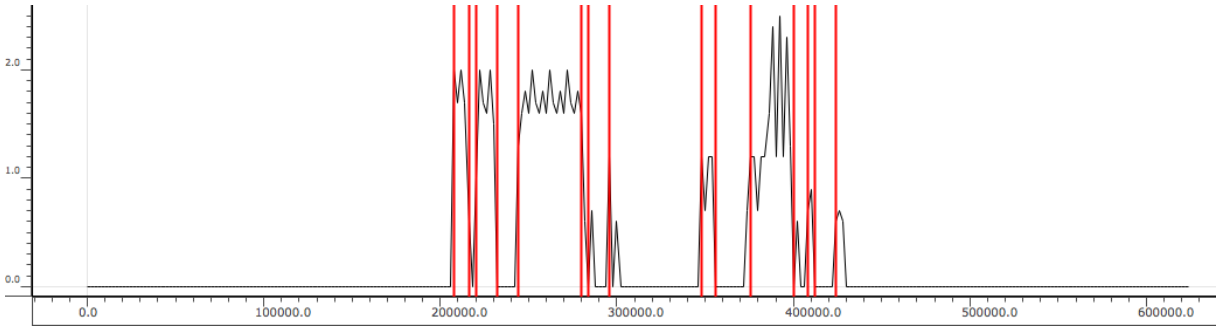


FIGURE 4.4 – Graphique de la platitude du canal 0 du fichier midi du *Sacre*, fenêtre de 2000 ms.

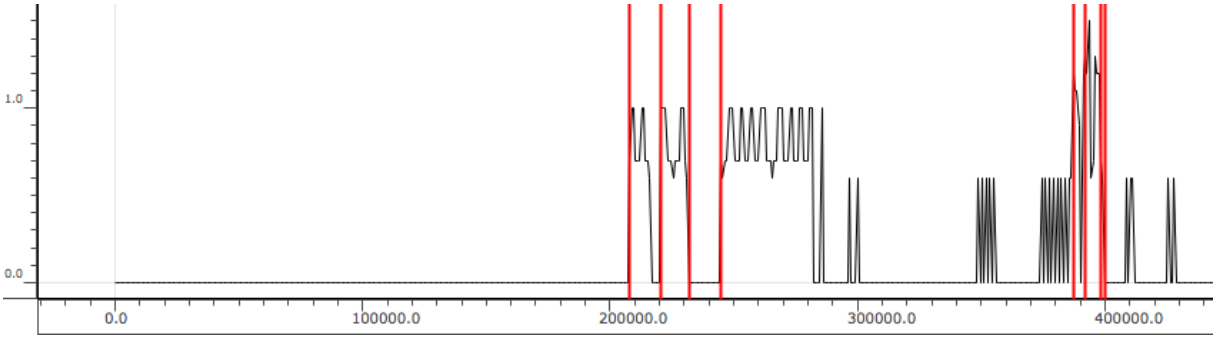


FIGURE 4.5 – Graphique de la platitude du canal 0 du fichier midi du *Sacre*, fenêtre de 2000 ms, étape d'avancement égal à 800.

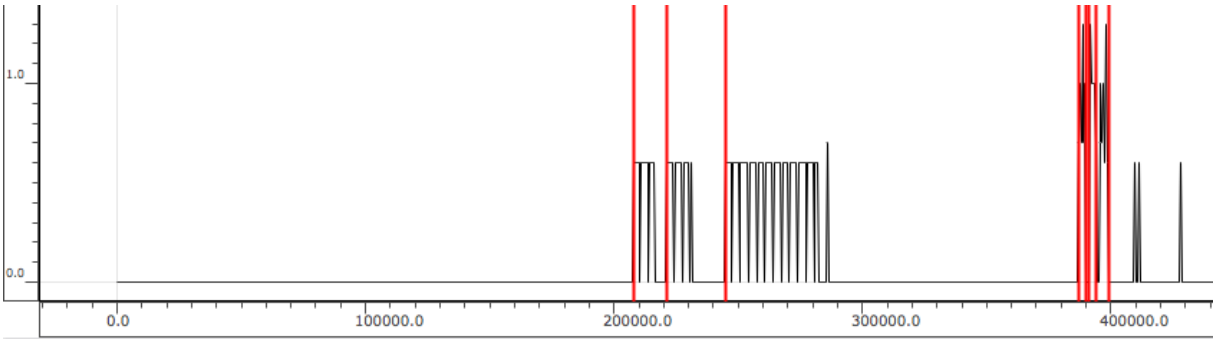


FIGURE 4.6 – Graphique de la platitude du canal 0 du fichier midi du *Sacre*, fenêtre de 2000 ms, étape d'avancement égal à 500.

4.5 Identification des figures et des blocs

Après avoir défini des outils pour l'étude des lignes isolées, nous pouvons aussi les utiliser pour comparer les lignes entre elles-mêmes, afin d'identifier des répétitions, des figures ayant une importance musicale, et des blocs. Il est possible donc d'étudier la vrai orchestration. Nous donnerons ici juste quelques exemples de la méthode de travail à suivre. La fig. 4.7 montre l'utilisation des courbes de la platitude pour identifier le motif rythmique des triolets répétés. Aux rythmes similaires correspondent des valeurs proches de platitude. À titre d'exemple, dans l'extrait de partition de la fig. 4.8 nous visualisons des blocs de figures proches ou bien identiques. La fig. 4.9 montre que les valeurs de platitude des parties pour le même bloc sont très proches, identiques pour les groupes irréguliers de neuf + 1 demi-croches détachés (platitude égale à 0.4) et assez proches pour des figures similaires, le groupe de six + 1 demi-croches détachés (platitude égale à à 0.7). La fig. 4.7 montre que le motif des triolets de croches répétées donne des valeurs de platitude autour de 1 - 1.5. Dans la phase d'analyse, l'identification des blocs permet d'apprendre comment le compositeur a distribué un motif entre plusieurs instruments ; dans la phase d'écriture pour orchestre, cette information permet d'utiliser la même technique avec des motifs similaires. Il s'agit d'un type de comparaison *interne* à la pièce orchestrale considérée ; un autre type d'analyse pour l'orchestration est ce que nous pouvons peut définir comparaison

externe, dont nous parlerons dans le chapitre 5. Il s'agit de trouver les figures les plus proches entre une pièce à orchestrer et une pièce-modèle orchestral, afin d'orchestrer la première selon le modèle de la deuxième.

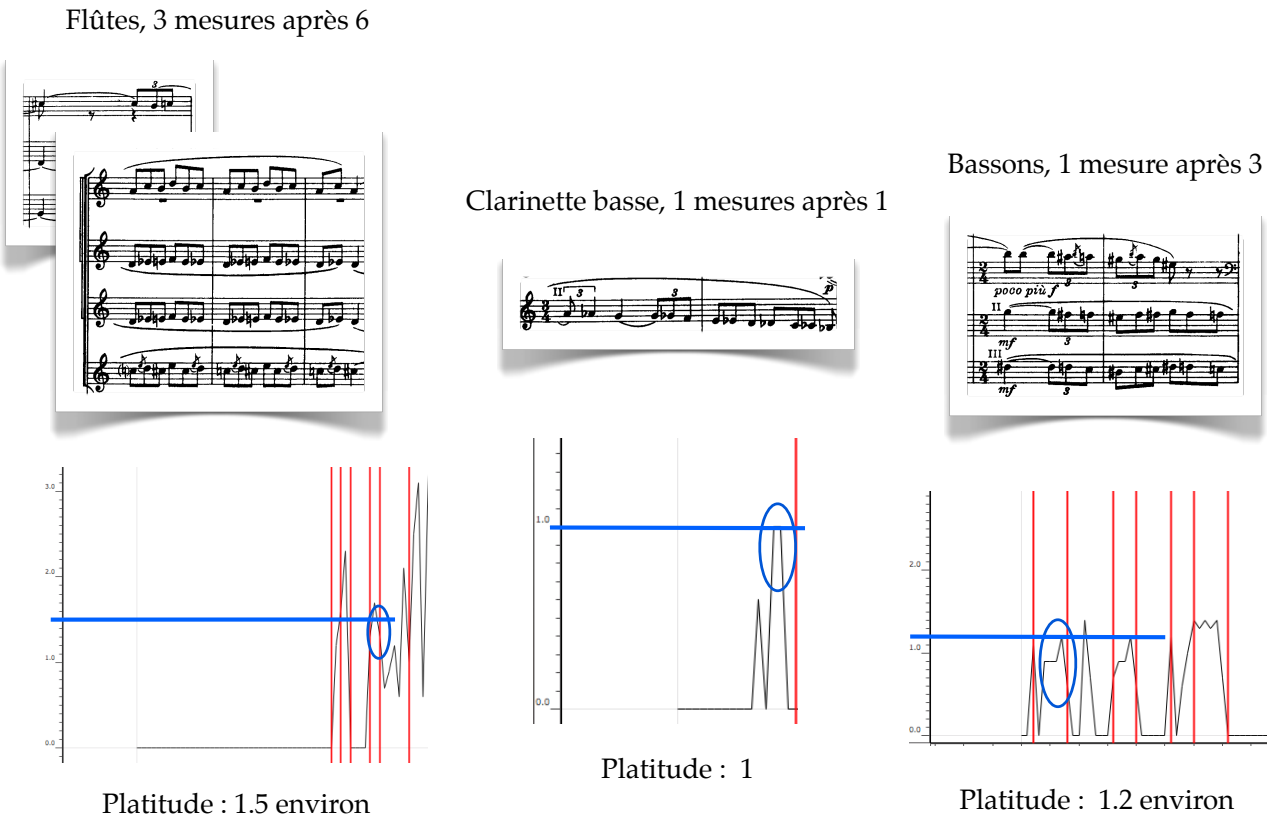


FIGURE 4.7 – Identification d'une figure musicale entre différents instruments, à l'aide des valeurs de platitude.

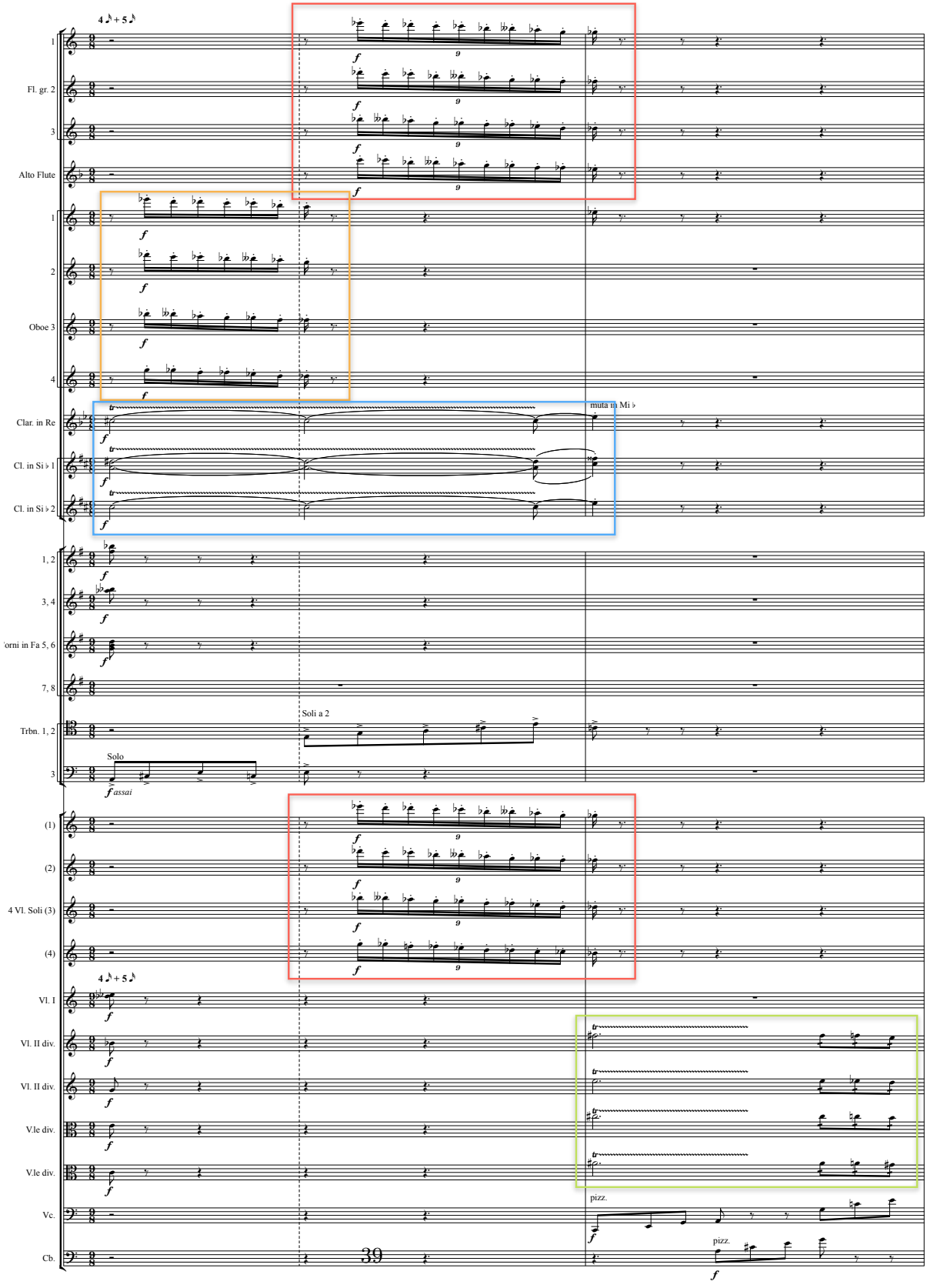


FIGURE 4.8 – Section 39, début. Les carrés indiquent des blocs de la même figure mélodique et rythmique. Les blocs marqués en rouge et en orange sont différents mais assez proches ; ils sont analysés dans la figure 4.9.

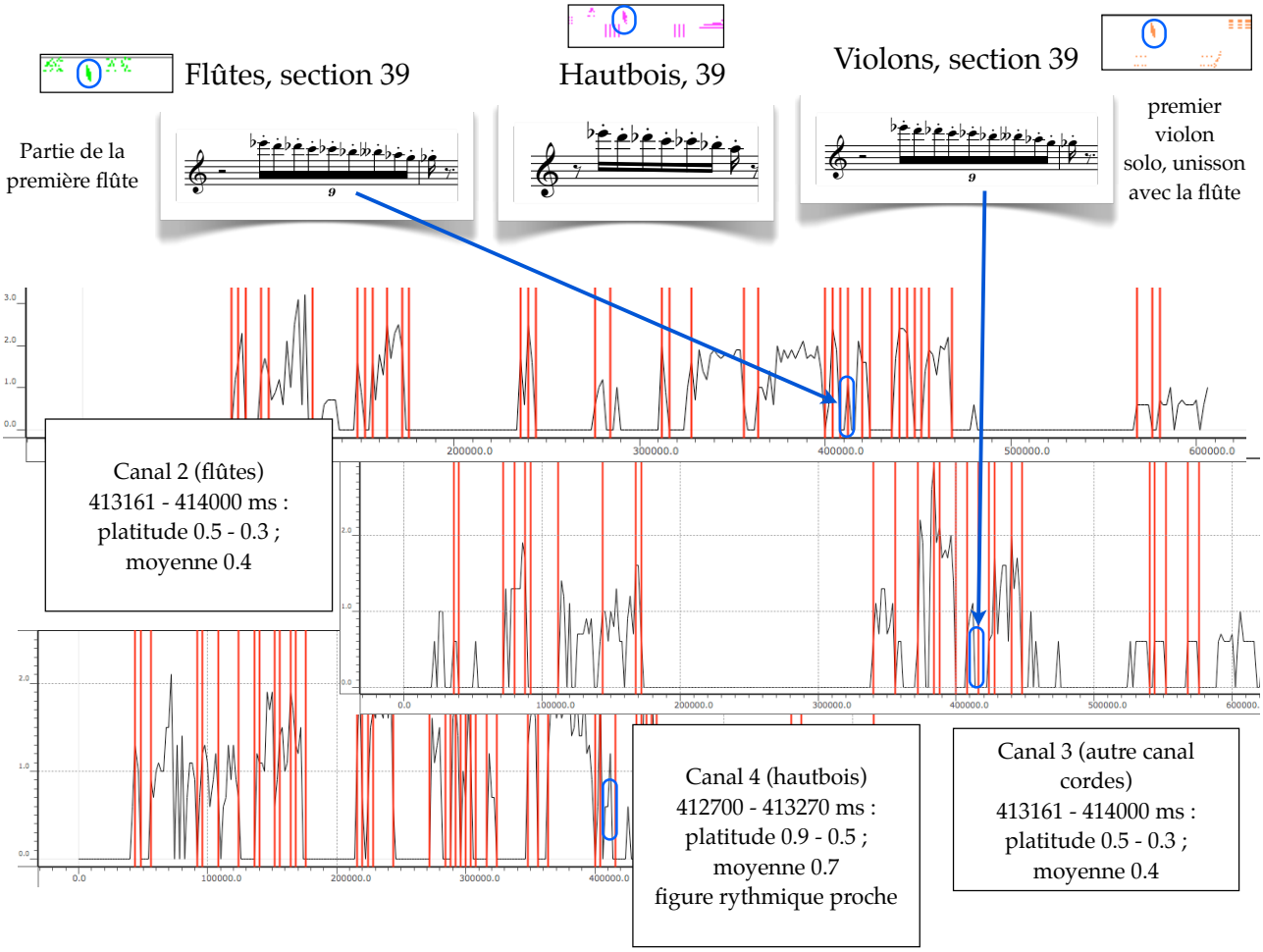


FIGURE 4.9 – Analyse à l'aide de la platitude des blocs de fig. 4.8.

Chapitre 5

Orchestration !

5.1 La méthode complète

Dans ce chapitre, nous exposons les instructions pour atteindre le but final du travail : l'orchestration d'un extrait pour piano selon un modèle orchestral donné. L'entière méthode peut être schématisée dans deux grandes parties : analyse et réalisation de l'orchestration. Les outils techniques permettant la réalisation de la première partie, relative à la segmentation (liée à l'analyse en dehors des connaissances musicales *a priori*), ont déjà été partiellement traités dans le chapitre 3 et 4. La partie concernant la réalisation n'a pas encore été implémentée ; cependant, nous donnons ici les étapes à suivre, et nous en présentons un exemple réalisée manuellement.

5.1.1 Partie première : analyse

Un petit résumé des étapes à suivre :

1. Segmentation de la pièce pour piano à orchestrer, selon inconstance et platitude.
2. Segmentation de la pièce orchestrale utilisée comme modèle pour l'orchestration, selon inconstance et platitude.
3. Comparaison des valeurs d'inconstance et après de platitude entre pièces orchestrales et pianistiques. En correspondance des valeurs les plus proches, nous prenons les séquences orchestrales comme modèle pour celles pour piano. Différentes sections avaient déjà été comparées ; l'étape suivante sera de réaliser une comparaison complète d'une façon complètement automatique, éventuellement à l'aide des DTW, *dynamique time warping*, des fonctions qui permettent une comparaison entre les lignes.

5.1.2 Partie deuxième : réalisation

Les séquences orchestrales sont de deux types : lignes (pour chaque instrument ou section instrumentale ; représentées par les canaux du fichier midi), segmentées comme des séries temporelles, et ensemble des lignes (groupes de canaux midi).

Si deux lignes de la même partition ont des valeurs de platitude ou bien d'inconstance assez proches entre elles, alors elles correspondent à la même figure musicale ; si elles sont proches dans le temps, elles font parties du même bloc.

Prenons les cas suivants :

- Dans l'extrait orchestral, il y a le même nombre de lignes que dans le morceau pianistique. Dans ce cas l'équivalence est simple : nous assignons les notes du piano à l'instrument utilisé dans la pièce orchestrale. Si les notes sont hors du registre de l'instrument orchestral, ou ne sont pas dans un bon registre par respect à l'intensité souhaitée dans la pièce à orchestrer, il serait possible de changer l'octave ou choisir un instrument de la même section le plus proche possible à l'instrument dans le modèle ; par exemple nous remplacerions un cor anglais par un hautbois. Nous pourrions écrire une table pour cet objectif. (Le doublement à l'unisson de la même section, parmi tous les premier violons, rentre dans cette catégorie).
- Dans l'extrait orchestral, la ligne pianistique est réalisée avec un ensemble de lignes. Peuvent se présenter des instruments qui jouent simultanément et à l'unisson (et donc dans ce cas nous nous trouvons dans le premier point de notre description), ou des instruments différents de la même section, ou encore des instruments de sections différents.
  Si les instruments jouent à l'unisson ou en distance d'octave, aucun problème ne se présente. Si, et il s'agit du cas le plus général, nous avons des intervalles différents, il faut garder ces intervalles. Il s'agira donc d'ajouter des répétitions des lignes. Il est clair que nous pouvons produire des changements harmoniques dans cette opération d'adjonction de lignes.
- L'enrichissement harmonique est beaucoup plus évident dans le cas où, en correspondance avec les lignes présentes dans le morceau pour piano, nous aurions, dans le modèle orchestral, d'autres lignes. Dans ce cas, nous pouvons tout de même utiliser ces lignes "ajoutées", en respectant les intervalles du modèle. Nous considérons par exemple d'avoir la séquence Do Ré Do Fa dans la pièce pour piano, et Ré Mi Ré Sol au basson dans le modèle orchestral, et, simultanément, une ligne de l'hautbois avec La Sol. L'intervalle entre le Ré et le La est un quinte ; il faut garder cet intervalle : dans la pièce orchestrée nous aurons Do Ré Do Fa avec des noirs au basson, et une ligne Sol Fa (en intervalle de quinte par respect au Do) de l'hautbois (fig. 5.1).

Le produit de ce travail ne sera toutefois qu'une version intermédiaire, qui nécessitera quelques perfectionnements par le compositeur, par exemple dans l'écriture *divisi* ou *non divisi*, selon les possibilité des cordes ; néanmoins, il est possible d'indiquer des contraintes à respecter aussi sur les possibilités de combinaison avec les cordes vides.

5.2 Bartók à la Stravinsky

Prenons maintenant des extraits de la pièce *Allegro Barbaro* de Béla Bartók, à orchestrer selon le modèle du *Sacre*, utilisé dans notre analyse. En regardant les graphiques de la platitude pour *Allegro Barbaro* (fig. 5.5) et celle du canal 0 du *Sacre*, correspondant aux cordes sans *pizzicato* (fig. 5.4), nous pouvons noter des régions de coïncidences des valeurs. Un autre résultat d'équivalence est obtenu en comparant les mesures 156-159 de la pièce de Bartók (fig. 5.9) avec un autre extrait du *Sacre* (fig. 5.10). Dans les deux cas, les résultats confirment l'équivalence visuelle entre les lignes dans la représentation graphique des fichiers midi. Nous avons orchestré deux courts extraits de *Allegro Barbaro*. Dans le premier cas, entre la partition de la pièce pour piano et celle du *Sacre* nous n'avons pas de lignes ajoutées ; dans le deuxième il y en a, et la méthode à suivre pour orchestrer se révèle plus compliquée. Nous exposons maintenant une explication plus détaillée de ces deux exemples.

5.2.1 Premier extrait

Le premier extrait (fig. 5.2) est un cas très simple, car nous avons une parfaite coïncidence des lignes avec les modèle orchestral considéré, le début des *Augures printaniers*, section 13 dans la partition du *Sacre* (fig. 5.3). Il s'agit donc seulement de distribuer les notes parmi les

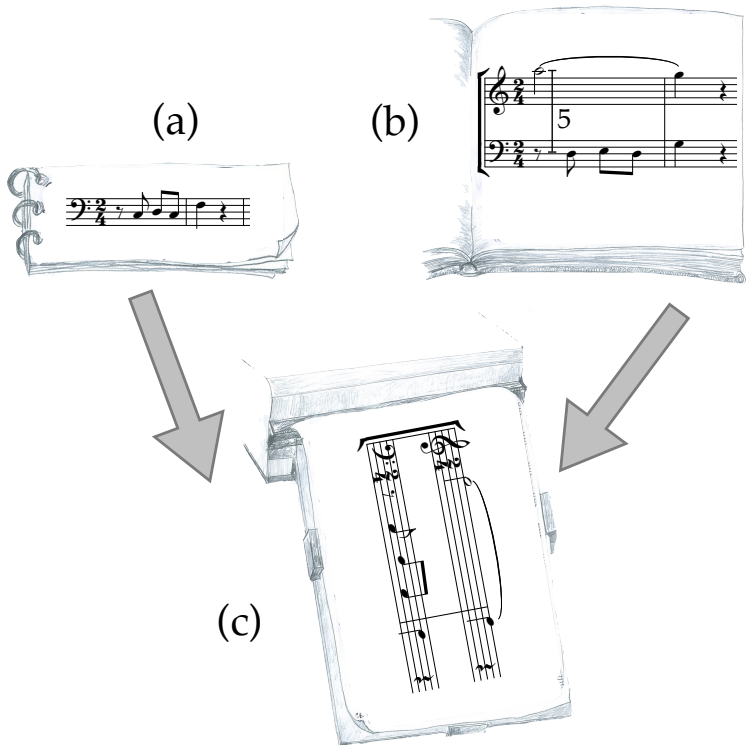


FIGURE 5.1 – Si dans le modèle (b) nous avons des lignes secondaires par respect aux lignes maîtres (celles proches de la ligne à orchestrer représentée dans (a)), nous les ajouterons dans la pièce orchestrée (c), en respectant les intervalles du modèle (à l'aide, donc, d'une transposition du modèle), dans ce cas, d'une quinte. (Dessins de Maria Mannone)

instruments du modèle. Nous orchestrons alors. Dans le premier accord de la main droite du pianiste, le La supérieur ira aux seconds violons ; le Fa#, étant hors du registre des violons, sera joué par les altos, et ainsi de suite. À la quatrième mesure de la pièce à orchestrer, nous commençons à écouter une nouvelle voix, la séquence La La La Sol La (avec un rythme régulier de noirs suivi d'une blanche), en ayant, à la main gauche, une ligne correspondante Fa# Fa# Fa# Do# Fa#. Cet élément a un correspondant dans l'entrée des cors, à la deuxième mesure du modèle (fig. 5.3). Si la même figure musicale dans le modèle est jouée par huit cors, dans notre orchestration quatre en sont suffisantes pour jouer Fa# Fa# (octave) et La La (octave) en sons d'effet. La séquence mélodique Fa# Fa# Fa# Do# Fa# dans le morceau de Bartók est noté avec des croches, car elle est intégrée dans l'accompagnement pianistique ; toutefois, il est possible de l'assimiler au rythme des noirs, en appuyant la même ligne plus aiguë, et en utilisant le même dessin rythmique pour tous les cors. La dynamique choisie est celle du morceau pour piano. Les articulations et les indications d'archet, étant déjà bien choisis, sont celles de Stravinsky. Pour ce qui concerne les cordes doubles, l'indication *non divisi* ne peut toutefois être utilisée que pour les violons. Une indication générale et précise du choix de la corde ne peut être écrite qu'en considérant beaucoup de contraintes sur les cordes multiples ; dans le cadre futur d'une orchestration automatique, ces types de choix seront sûrement laissées aux compositeurs. Enfin, nous obtenons le morceau orchestré de fig. 5.6.

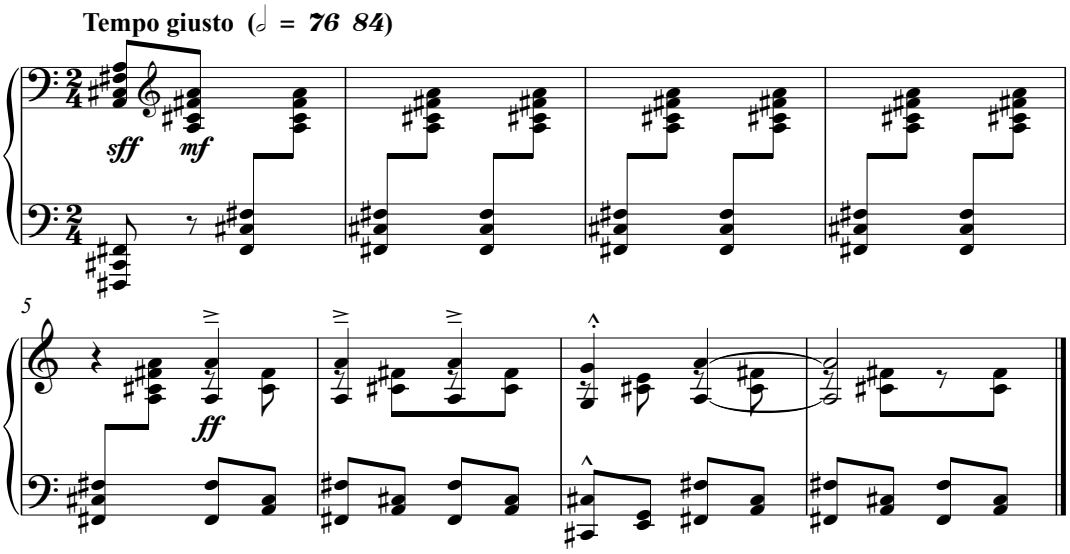


FIGURE 5.2 – *Allegro Barbaro* de Béla Bartók, début.

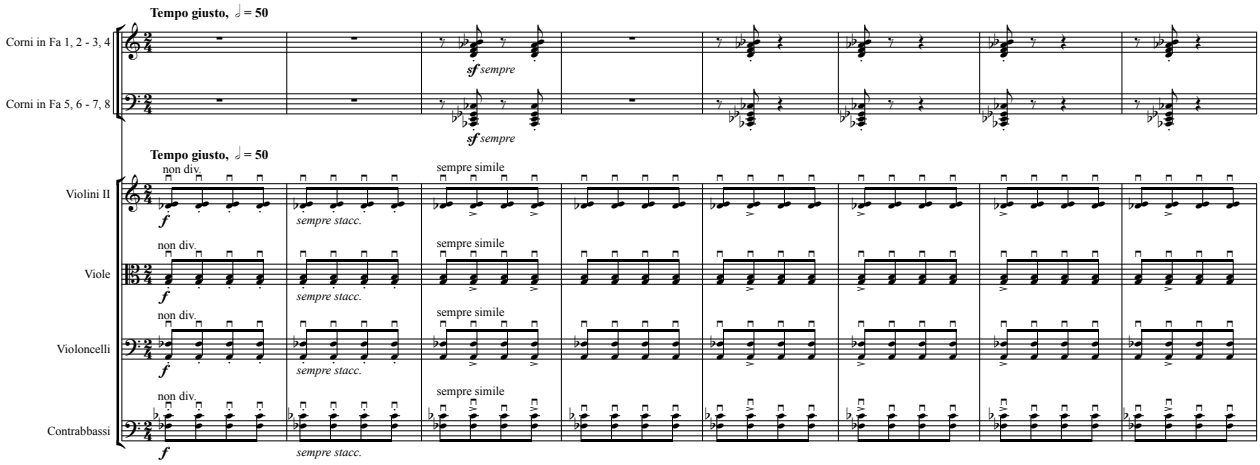


FIGURE 5.3 – *Sacre du printemps*, *Les augures printaniers*, section 13 dans la partition. Ici, et dans les figures à suivre, nous gardons la notation italienne des noms des instruments.

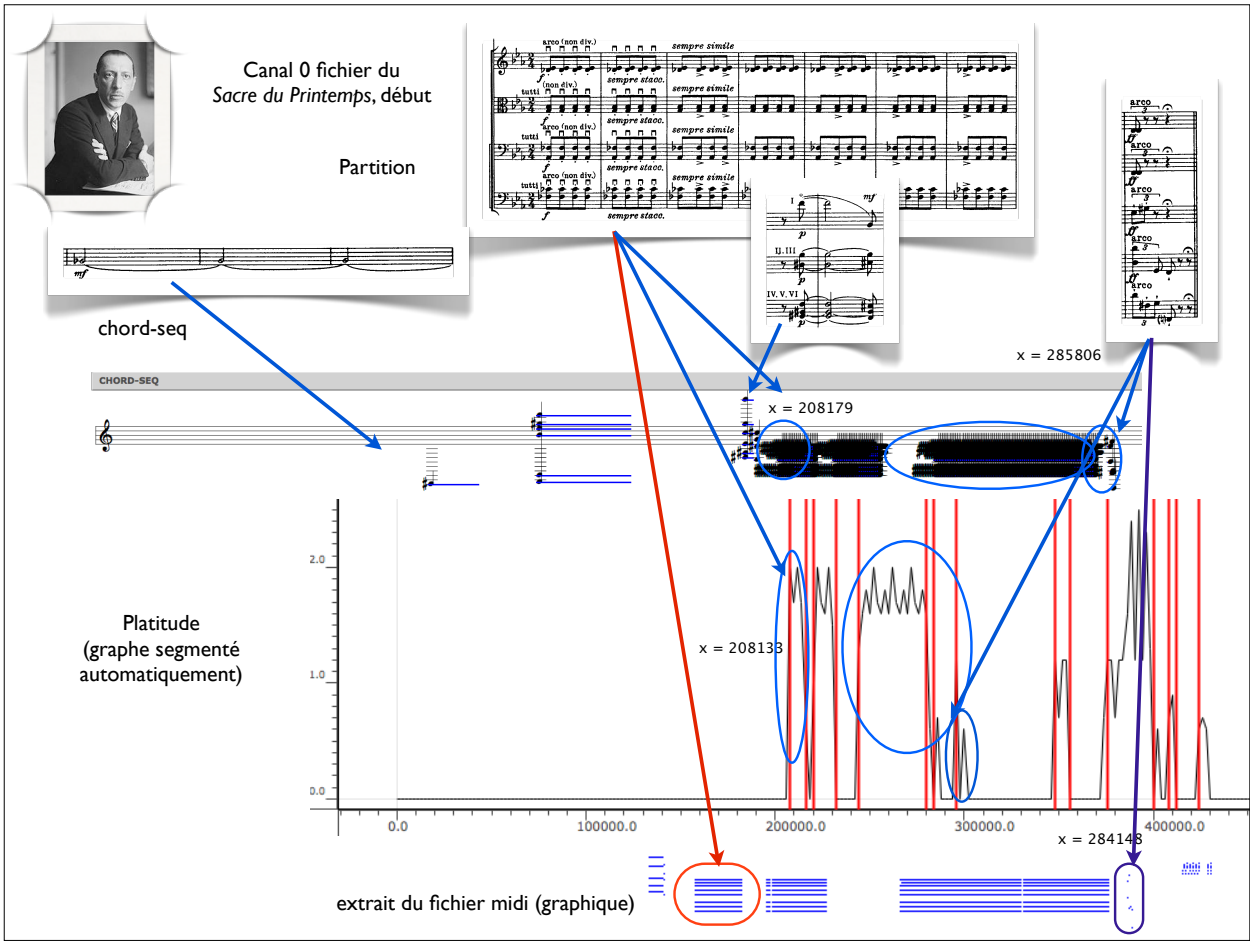


FIGURE 5.4 – Courbe des valeurs de platitude du canal 0 du fichier midi (le début) du *Sacre*, correspondant aux cordes sans *pizzicato*, segmentée automatiquement. Indications des correspondances dans la partitions des séquences proches à celles bartókiennes.

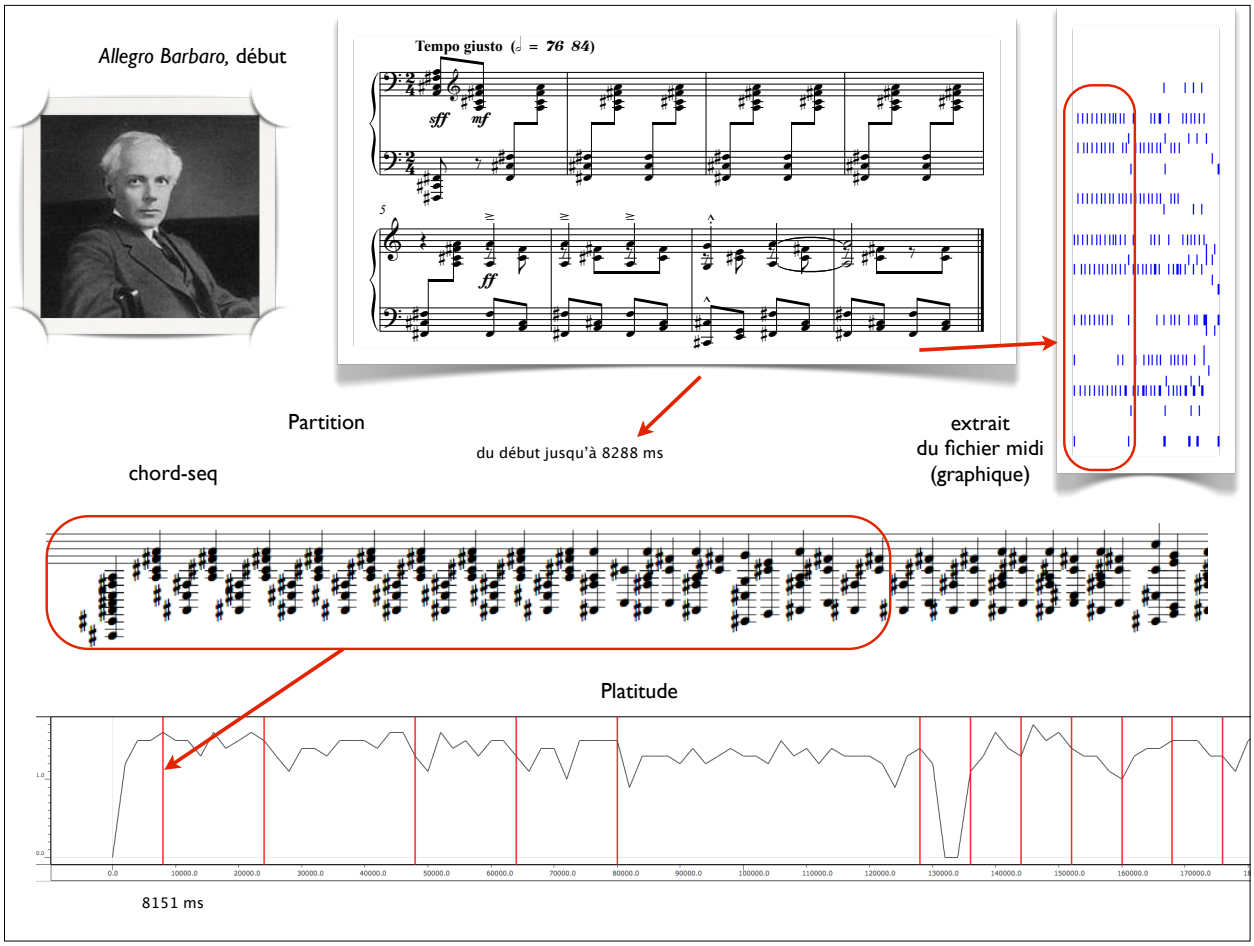


FIGURE 5.5 – Courbe des valeurs de platitude du début de *Allegro Barbaro* de Béla Bartók, segmentée automatiquement, et identification de la première séquence dans la partition.

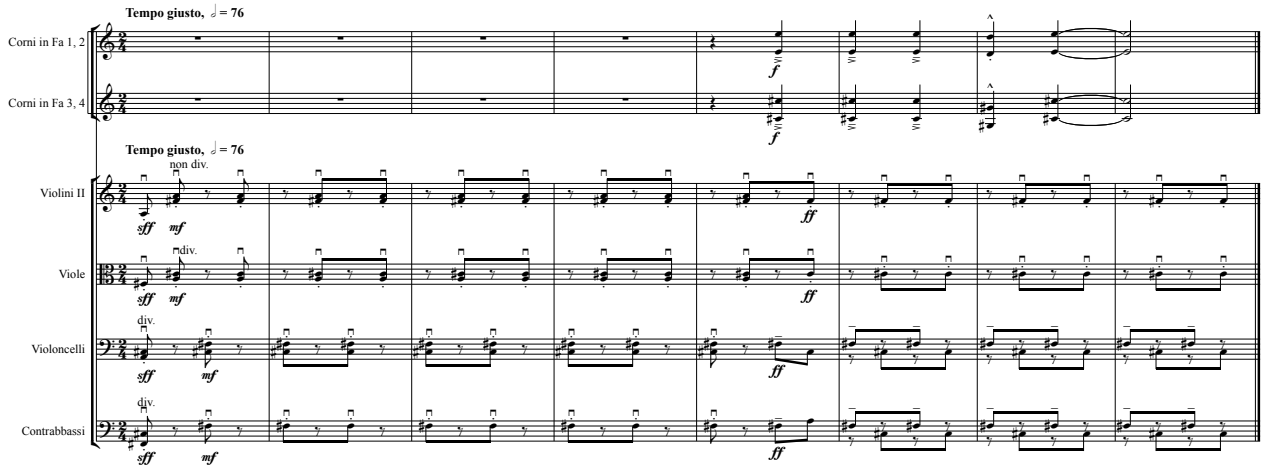


FIGURE 5.6 – Orchestration du début de *Allegro Barbaro* (fig. 5.2), selon le modèle orchestral des *Augures printaniers* du *Sacre du printemps* (5.3). La technique utilisée est expliquée dans le paragraphe 5.2.1.

5.2.2 Deuxième extrait

Le deuxième exemple, mesures 156 - 159 de *Allegro Barbaro* (fig. 5.7), a été orchestré d'une façon un peu plus complexe. Le modèle orchestral (selon la section 28 du *Sacre*, fig. 5.8), par rapport à la pièce pour piano, doit être schématisé en deux type des lignes : voix maîtres et voix secondaires.

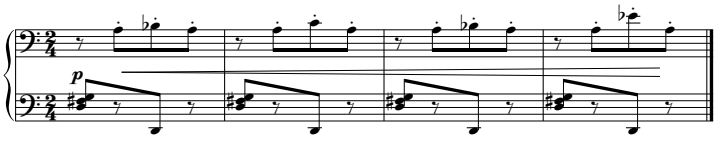


FIGURE 5.7 – *Allegro Barbaro* de Béla Bartok, extrait (mesures 156 - 159).

Les voix maîtres sont les plus proches des lignes du piano. Les voix secondaires, qui ne se retrouvent pas dans la partition pour piano, doivent lui être ajoutées pour réaliser une orchestration d'effet équivalent au modèle choisi.

Dans le cas considéré, nous retrouvons un dessin équivalent au La Si♭ La, La Do La, La Si♭ La, La Mi♭ La du piano, avec des croches en *staccato*, parmi les notes de la flûte Si♭ Do Si♭ Mi♭ ... La mélodie de la première flûte est doublée par la deuxième flûte et la flûte alto, au moins d'une partie de décoration : au lieu de Si♭ Do avec des croches, on a Si♭ Do La♭, avec une croche et deux demi croches. Le La♭ demi croche est en distance de tierce (majeure) avec le Do. Dans notre orchestration, nous allons reproduire cette décoration en écrivant La Si♭ Sol, avec une croche et deux demi croches, et une distance de tierce entre les dernières deux notes. Par rigueur, il faudrait écrire La Si♭ Sol♭ pour avoir une tierce majeure ; mais nous ne voulons pas écrire un Sol♭ qui ne soit pas dans l'harmonie des notes principales. Nous pourrions cependant écrire Fa#, il s'agit du même son pour former une tierce majeure, mais au niveau formel il s'agirait d'une quatrième diminuée. Nous gardons alors un intervalle plus lié à la tonalité qu'aux exacts nombres de demi tons.

Même raisonnement pour la flûte alto, avec des sons d'effet une octave au dessus de la première ligne, et une quatrième au dessus des notes écrites. Si la première flûte a une articulation *staccato*, la deuxième flûte et l'alto sont notées comme *legato* ; cette différence sera maintenue dans la pièce orchestrée (fig. 5.11).

En ce qui concerne la clarinette *piccolo* en Mi♭, les clarinettes en Si♭ et la clarinette basse en Si♭, nous avons des lignes sans aucune équivalence au niveau de la pièce pour piano : il s'agit donc des lignes secondaires. Nous ajouterons ces lignes à notre pièce pour orchestre, en réalisant une transposition. Nous gardons l'intervalle entre note d'attaque pour chaque ligne et note d'attaque de la voix maitre, et Nous appliquons cet intervalle à la nouvelle partition.

Pour les bassons et le contrebasson, le dessin du modèle a été gardé, mais avec les notes de la pièce pour piano Fa# et Sol (basson) et Ré (contrebasson).

Une intervention plus importante concerne la partie pour trombone. La ligne mélodique La Si♭ La, La Do La... de Bartók est originalement écrite dans un registre grave. Des notes autour du Do central, un rythme avec des croches, une intensité moyenne (dans ce cas nous gardons plus le ***mf*** - ***f*** du modèle que le ***p*** du piano, pour faciliter les flûtes dans le registre aigu), se retrouvent dans la partie du trombone. Nous avons, alors, attribué les notes de la mélodie au trombone. Cette mélodie a été doublée par les violoncelles, en *pizzicato* (donc en gardant l'articulation du modèle).

Nous assignerons aux timbales des notes jouées au piano par la main gauche (Fa# Ré). Les

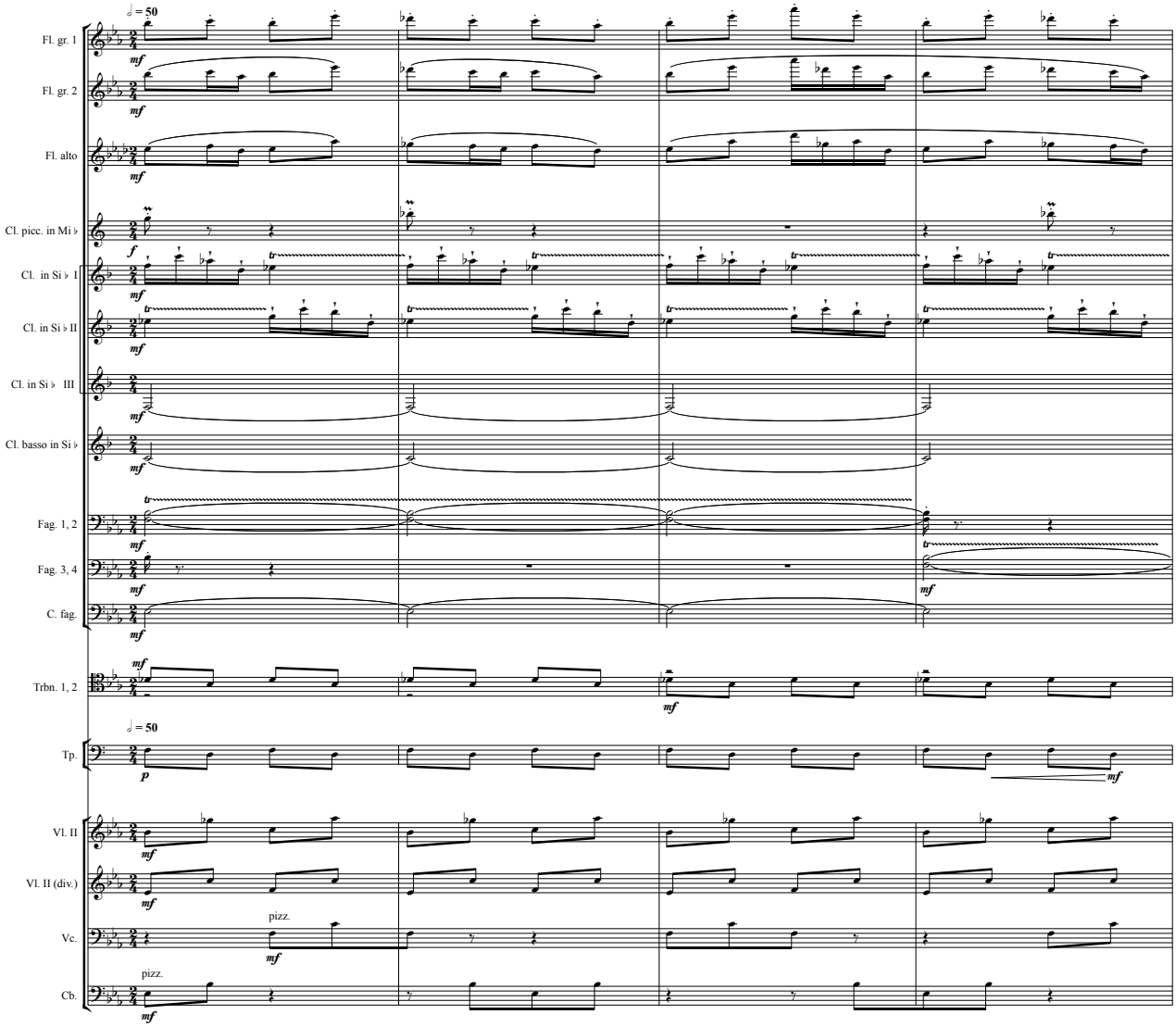


FIGURE 5.8 – Section 28 de la partition du *Sacre du printemps*.

Ré sont doublés par les contrebasses.

Les violons jouent une séquence tirée du modèle orchestral, donc une ligne secondaire, transposée en gardant les intervalles originales entre ligne des violons et la ligne maitre des flûtes.

Enfin, nous obtenons le morceau orchestré de fig. 5.11.

En général, il faut avant tout construire une structure des intervalles verticales et horizontales du modèle. La structure verticale concerne les rapports entre les notes d'attaque pour chaque voix par respect aux notes d'attaque de la ligne maitre ou bien des lignes maitres. La structure horizontale est l'ensemble des suites des intervalles pour chaque voix. Nous obtenons alors une structure qui permet de manipuler et transposer la structure orchestrale. L'ajout des lignes

secondaires, d'une manière générale, enrichira l'harmonie de la pièce finale ; dans le cas d'une pièce dissonante cela ne dérange pas, mais pour des morceaux avec une harmonie très consonante, cela pourrait bouleverser l'effet de la pièce originale. Toutefois, le problème peut être facilement résolu en considérant, parmi les modèles orchestraux, un grand nombre de partitions très variées, permettant toujours de trouver automatiquement les séquences les plus proches à la pièce à orchestrer.

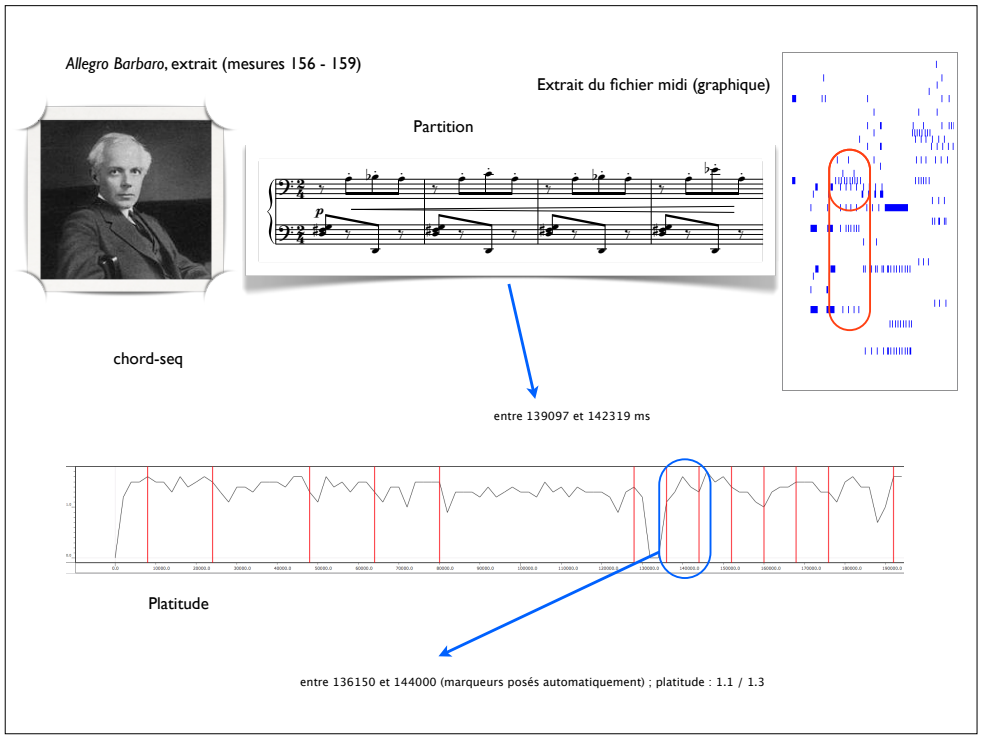


FIGURE 5.9 – Courbe des valeurs de platitude du début de *Allegro Barbaro* de Béla Bartók, segmentée automatiquement (paramètres segmentation : 4, 4, 0.1). Le dessin dans le graphique du fichier midi est similaire auquel de l'extrait dans fig. 5.10 ; le calcul de la platitude confirme cette constatation, et donc nous adopterons cette orchestration (fig. 5.11).

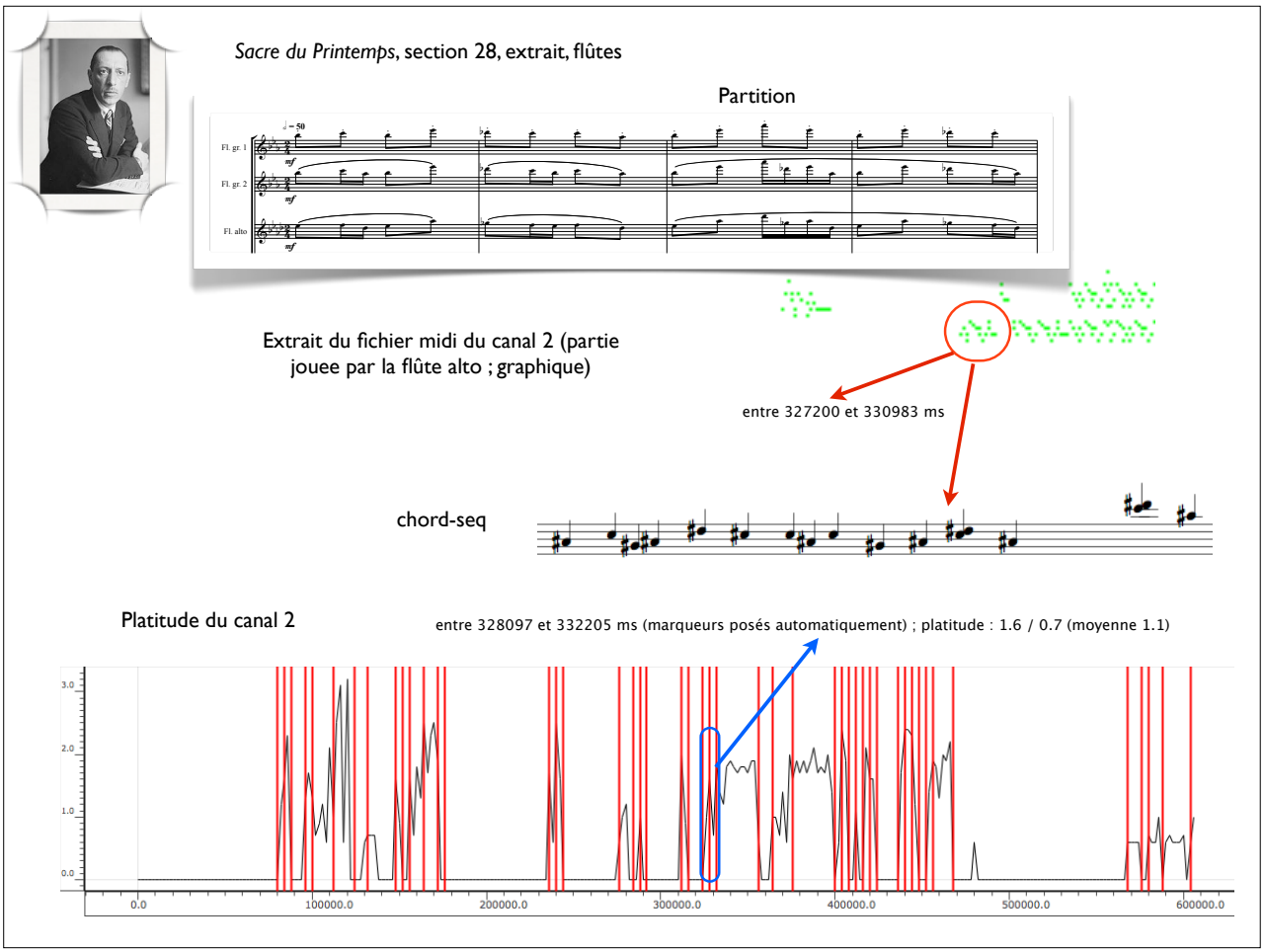


FIGURE 5.10 – Courbe des valeurs de platitude du canal 2 du fichier midi du *Sacre*, correspondant aux flûtes, segmentée automatiquement (paramètres de segmentation : 2, 2, 0.4) ; détail du calcul de la platitude d'un extrait.

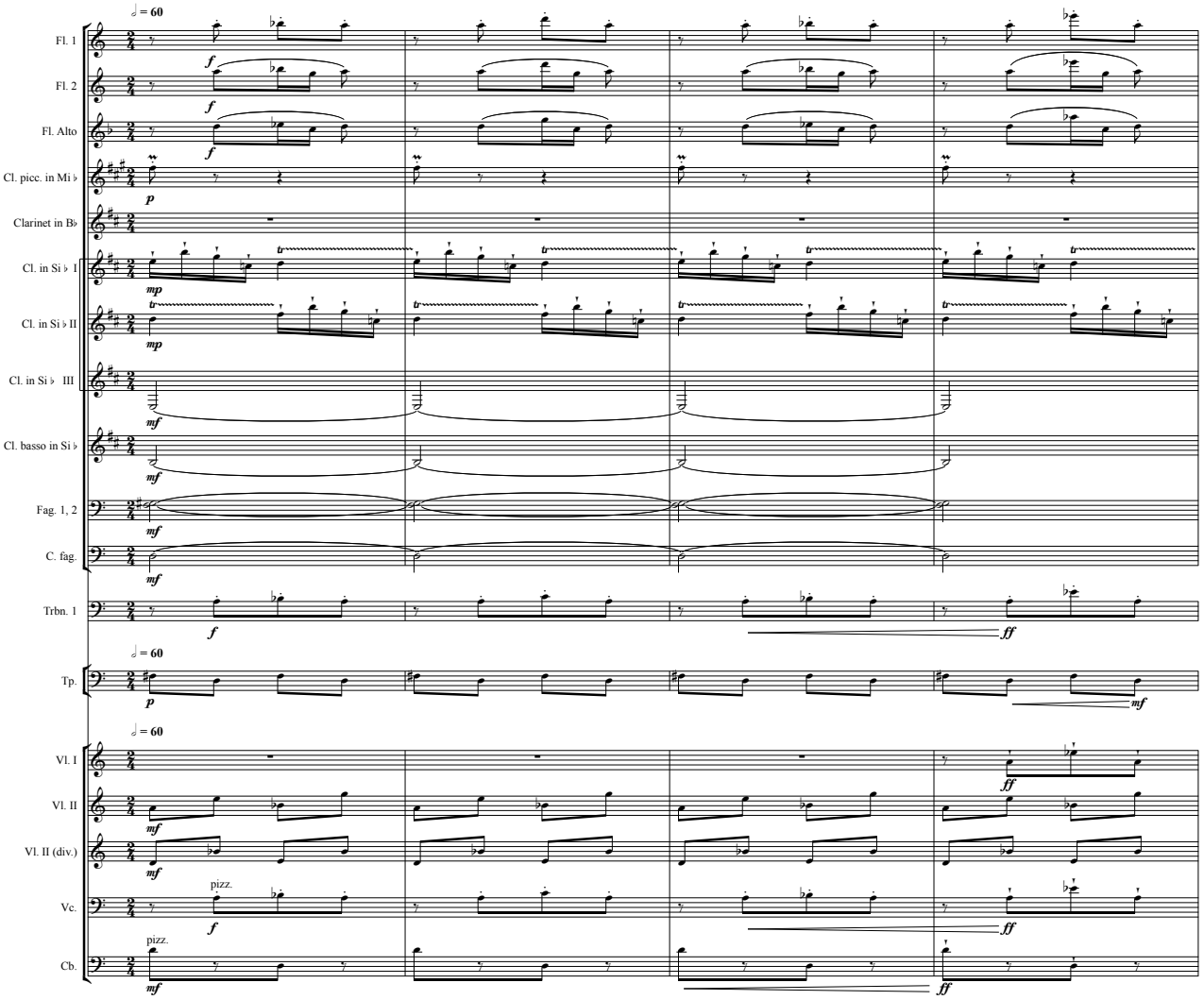


FIGURE 5.11 – Orchestration d'un extrait de *Allegro Barbaro* (fig. 5.7), selon le modèle orchestral de la section 28 du *Sacre du printemps* (fig. 5.8). La technique utilisée est expliquée dans le paragraphe 5.2.2.

Conclusion

L'orchestration d'une pièce musicale est partie intégrante du processus de création artistique. Toutefois, il y a des techniques qui peuvent être schématisées, expliquées, et automatisées. Nous prenons ici l'exemple d'étudiants de Composition qui apprennent à orchestrer en lisant un grande nombre de partitions de grands auteurs, à la recherche des structures communes, avant de construire leur propre vocabulaire, leur propre style.

L'objectif de début de ce travail de stage est celui d'orchestrer une pièce pour piano, sur le modèle d'une grande pièce orchestrale ou, en général, d'un répertoire des pièces orchestrales, selon une idée graphique de représentations des lignes musicales comme des lignes graphiques, dessinées.

La technique à suivre est de repérer les lignes (vraiment au niveau du dessin) de la pièce pour piano dans la pièce orchestrale après une comparaison graphique ; dans le cas de coïncidence ou bien de similarité, appliquer le choix d'instruments et la distribution entre voix et sections de la pièce orchestrale à la pièce pour piano. Il s'agit d'une méthode originale inspirée par l'analyse des partitions proposé par S. Sciarrino. Cette méthode fait abstraction de tout ce qui concerne l'analyse musicale traditionnelle, en faveur d'une approche graphique et facile à comprendre même par les non-musiciens.

Un problème s'est immédiatement posé : il s'agissait de la difficulté d'une analyse basée sur la comparaison graphique, très simple à réaliser par l'humain, mais malheureusement beaucoup plus compliqué à réaliser par l'ordinateur.

Un parcours possible à suivre était d'abord d'analyser les partitions, d'analyser les pièces à orchestrer, et, après une comparaison, de réaliser l'orchestration. Le centre du présent travail concerne la segmentation des séries temporelles, un problème intéressant dans le domaine de l'informatique, en raison de la grande quantité des possibles applications interdisciplinaires. La méthode suivie a, donc, été de segmenter la partition orchestrale, pour trouver des figures, des dessins isolés, à l'aide des techniques de segmentation des séries temporelles. Il n'existe pas un algorithme qui soit optimal pour tous les cas ; il faut choisir chaque fois le meilleur selon les paramètres d'intérêt dans l'analyse à réaliser. Dans le cas ici proposé, les programmes tirés de la littérature n'étaient pas satisfaisantes. Nous avons donc essayé d'appliquer pour la première fois à la musique des algorithmes inspirés des concepts mathématiques de l'*inconstance* dans le cas de l'étude des hauteurs, et de la *platitude* des coefficients de Fourier dans le cas de l'évaluation des régularités rythmiques. Chaque différente méthode a permis de trouver des marqueurs qui, appliqués à la partition, ont donné segmentations. La comparaison des ces segmentations automatiques avec celles réalisées à la main, a confirmé la fiabilité de ces méthodes, en donnant des indications sur le choix des paramètres pour améliorer l'analyse. Les paramètres qui caractérisent les séquences de chaque ligne de la partition permettent aussi de comparer plusieurs lignes entre elles, afin d'identifier les blocs des figures musicales proches. Cette application se révèle d'importance capitale dans le cadre de l'étude de l'orchestration. Cette comparaison *interne* à la pièce orchestrale, qui complète la partie d'analyse, doit être accompagnée par une autre comparaison,

cette fois *externe*. Si les séquences obtenues par segmentation automatique d'une pièce pour piano trouvent un équivalent dans une segmentation du modèle, nous prenons ce dernier pour la réalisation orchestrale du premier segment.

En suivant ce parcours, deux brefs morceaux de *Allegro Barbaro* de Béla Bartók ont été orchestrés selon le modèle de deux séquences du *Sacre du printemps* de Igor Stravinsky, étant donné la variété des figures musicales présentes dans la partition. De plus, nous fêtons cette année le centenaire de la première création, dans le 1913, du ballet au Théâtre des Champs-Elysées à Paris. L'analyse du rythme est complémentaire à celle des hauteurs ; une étude complète nécessitera de quelques paramètres pour quantifier les variations d'intensité, et seulement une combinaison des paramètres (inconstance, platitude, quantificateurs d'intensité) pourrait donner une caractérisation numériquement précise des figures musicales. Il est vrai que nous pouvons visualiser plusieurs informations contenues dans la partition d'un coup d'œil à l'aide des graphiques en 3D avec temps, intensités et hauteurs [19] ; nous rencontrons, toutefois, l'inconvénient de la difficulté technique d'analyser automatiquement les lignes des graphiques, raison pour laquelle ont été utilisées des techniques de segmentation dans une seule dimension, en considérant un seul paramètre à la fois.

La prochaine étape importante à réaliser est l'automatisation de l'attribution des notes de la pièce pianistique à la structure de la séquence orchestrale, selon une méthode algorithmique (chapitre 5), avec l'ajout, selon le modèle, des lignes secondaires à celles de la pièce à orchestrer.

Les objectifs à rejoindre à court terme sont liés à une complète automatisation de la première partie (analyse). La pièce orchestrale et la pièce pour piano devraient être comparées entièrement et, toujours d'une façon automatique, nous pourrons, donc, obtenir l'indication des séquences proches. L'autre objectif, cette fois à long terme, est l'implémentation de la deuxième partie (réalisation). Sans aucun doute, l'ambitieux projet d'orchestration automatique, ici présenté, nous a donné plusieurs difficultés au niveau de la réalisation, car chaque phase du processus implique la résolution soit de problèmes classiques rencontrés en informatique sans pour autant trouver une solution universelle (comme celui de la segmentation), soit des questions nouvelles, comme l'idéation d'un algorithme pour adapter le modèle à la pièce à orchestrer.

Malgré les premiers résultats obtenus très encourageants, le problème est encore bien loin d'être résolu. Même en négligeant les petits soucis techniques dont on a parlé d'une façon très détaillée (paragraphes 4.4 et 3.3.4), il faudrait introduire une amélioration importante, qui pourrait être réalisée pendant d'éventuelles travaux successifs.

Le but était de parcourir les étapes d'apprentissage d'un étudiant de Composition qui apprend à orchestrer. Après la phase d'analyse des partitions, et après la phase d'écriture "à la façon de", l'étudiant commence à ré-élaborer les modèles appris, à la recherche d'un style d'écriture personnel. La phase de ré-élaboration, manquante dans le modèle proposé, peut bien être projetée à l'aide de recombinaisons automatiques, et des outils comme le facteur Oracle, déjà utilisé dans le domaine de l'improvisation et de la ré-élaboration des motifs et des séquences harmoniques, à appliquer dans l'enrichissement et dans la variation des lignes secondaires tirées du modèle orchestral. A ce jour, le produit de cette méthode ressemble à un bloc de marbre auquel on aurait donné une première forme, dans l'attente de la touche finale du maître.

Remerciements

Face au grand nombre de personnes, qu'il faudrait remercier, de celles qui m'ont encouragée à me consacrer à la science et à la musique ensemble, de celles qui m'ont encouragée, quelques années auparavant, à choisir le parcours de la Musique et après aussi de la Science, je ne donnerai que quelques noms reliés au présent travail de stage et à cette année.

Impossible de ne pas citer en premier mon directeur de stage. Je souhaite remercier M. Carlos Agon, guide compétent dans l'informatique, et personne gentille et sensible. Je souhaite remercier M. Moreno Andreatta, qui m'a encouragée à me candider pour le complexe et fascinant parcours ATIAM. Je souhaite remercier le M. Karim Haddad, artiste savant et avisé avec qui j'ai eu le plaisir d'être voisin de bureau ; M. Gérard Assayag, chef de l'équipe de représentations musicales. Je souhaite remercier M. Jean Bresson, magicien dans l'environnement OpenMusic, qui m'a aidée dans des outils nécessaires pour le déroulement du travail.

Je souhaite remercier tous ceux avec qui j'ai parlé de mon projet d'orchestration automatique, et qui, avec leur mélange d'appréciation, d'objections et de questions, m'ont stimulée à réfléchir et à perfectionner mon travail : M. Mikhail Malt, Mme Michèle Castellengo, les compositeurs M. Yan Maresz et Carmine Cella.

Je souhaite remercier tous les professeurs de la formation ATIAM, toujours disponibles aux explications.

Je souhaite remercier deux mes professeurs italiens, mon maître de Composition M. Marco Betta, et mon professeur de Physique M. Giuseppe Compagno, qui m'ont encouragée à étudier à l'IRCAM.

Je souhaite remercier grandement le Maître Pierre Boulez, avec qui j'ai eu l'honneur de parler, et dont tous les ircamiens sont idéalement disciples.

Enfin, je souhaite remercier ma famille et en particulier mes parents, qui m'ont soutenue aussi du côté affectif, pendant une période plutôt difficile de transition entre les couleurs intenses du Sud et les brumes du Nord.

Annexe A

Codes pour le calcul de l'inconstance et de la platitude

Nous reportons les codes pour le calcul de l'inconstance (section 3.1) et de la platitude (4.1).

A.1 Code pour l'inconstance

```
(defun get-lower-point (list)
  (let ((rep (car list)))
    (loop for item in (cdr list) do
          (if (> (om-point-v item) (om-point-v rep)) (setf rep item))) rep))

(defun compute-angle (p1 p2)
  (let* ((a (abs (- (om-point-v p2) (om-point-v p1))))
         (b (- (om-point-h p2) (om-point-h p1)))
         (r (sqrt (+ (sqr a) (sqr b)))))
    (if (minusp b)
        (- pi (asin (/ a r))) (asin (/ a r)))))

(defun cross-product (p1 p2 p3)
  (- (* (- (om-point-h p2) (om-point-h p1)) (- (om-point-v p3) (om-point-v p1)))
     (* (- (om-point-h p3) (om-point-h p1)) (- (om-point-v p2) (om-point-v p1)))))

(defun point-distance (p1 p2)
  (let* ((a (abs (- (om-point-v p2) (om-point-v p1))))
         (b (abs (- (om-point-h p2) (om-point-h p1)))))
    (sqrt (+ (sqr a) (sqr b)))))

(defun mysort (p1 list)
  (sort list #'(lambda (x y) (or (> (second x) (second y))
                                 (and (= (second x) (second y))
                                  (< (point-distance (first x) p1)
                                  (point-distance (first y)  p1)))))))
```

```
(defmethod enveloppe-convexe ((points list))
  (let* ((max (get-lower-point points))
         (rep (list (list max (* 2 pi)))) stack)
    (loop for item in points
          for i = 0 then (+ i 1) do
          (unless (om-points-equal-p max item)
            (push (list item (compute-angle max item)) rep)))
    (setf rep (loop for item in (mysort max rep) collect (first item)))
    (push (first rep) stack)
    (push (second rep) stack)
    (loop for i from 2 to (- (length rep) 1) do
          (let ((cross-p (cross-product (second stack) (first stack) (nth i rep))))
            (cond
             ((= cross-p 0) (pop stack) (push (nth i rep) stack))
             ((> cross-p 0) (push (nth i rep) stack))
             (t (loop while (and (<= cross-p 0) (> (length stack) 2)) do
                      (pop stack)
                      (setf cross-p (cross-product (second stack) (first stack)
                      (nth i rep))))
                (push (nth i rep) stack))))) (append stack (list (car stack)))))

(defun list2points (listx listy)
  (loop for x in listx for y in listy collect (om-make-point x y)))

(defun length-curbe (points)
  (let ((rep 0))
    (loop for p1 in points
          for p2 in (cdr points) do
          (setf rep (+ rep (point-distance p1 p2)))) rep))

(defun cauchy-crofton (points)
  (/ (* 2 (length-curbe points)) (length-curbe (append (enveloppe-convexe points) ))))
```

A.2 Code pour la platitude

```
(defun ecartype (list)
  (let* ((n (length list))
         (moyenne (/ (reduce '+ list) n)))
    (sqrt (/ (loop for item in  list sum
                   (sqr (- item moyenne))) n))))

(defun flatitude1 (list period)
  (let ((coeff (norm-coeff-list list period)))
  (ecartype (cdr coeff))))

(defmethod! norm-coeff ((A list) (period integer) (i integer))
            :doc "get the i-eme fourier coeff"
           (let ((rep (loop for k in A sum
```

```
                (exp (/ (* -2 (complex 0 1) pi k i) period)))))
          (sqrt (+ (sqr (imagpart rep)) (sqr (realpart rep))))))

                (defmethod! norm-coeff-list ((A list) (period integer))
(loop for te from 0 to (- period 1) collect (norm-coeff A period te)))

                (defmethod! bpf-coeff ((A list) (period integer))
         (when (listp (car A)) (setf A (car A)))
         (simple-bpf-from-list  '(0 1) (norm-coeff-list A period) 'bpf 2))
```

13

Annexe B

Codes de segmentation

Nous reportons ici les codes écrits en Common Lisp pour segmenter, en partant de pseudo-codes présent en littérature [12], cités dans la section 2.3, et ainsi un très simple originel utilisé dans ce travail de stage B.1. Pour être compilés, ces algorithmes nécessitent des fonctions de régression linéaires écrites dans la section B.3.

B.1 Segmentation à l'aide du calcul des moyennes

Dans le paragraphe 3.3.1, nous avons défini une fonction très simple, originale, et utilisée pour segmenter les graphiques reportées dans les annexes. Une fenêtre qui glisse (sans superposition pour éviter de doubler les mêmes résultats) calcule les valeurs de moyenne, va le comparer avec les précédent, et indique un marqueur si l'écart de la moyenne précédent est supérieur à un certain niveau établi. Le code est le suivant.

```
(defun finestra_medie (seriey taglia passo errore)
(let ((i 0))
  (loop while (< i (- (- (length seriey) 1) taglia))
        collect
          (let* ((sub1 (subseq seriey i (+ i taglia)))
                 (sub2 (subseq seriey (+ i taglia) (+ i (+ taglia 2)))))
            (setf i (+ i passo))
            (if (> (variazione (calcolo_medie sub1) (calcolo_medie sub2)) errore)
            (progn (finestra_medie sub2 taglia passo errore)
              (setf mark i))
         (finestra_medie sub2 taglia passo errore))))))

(defun calcolo_medie (serieNum)
(/ (apply '+ serieNum) (length serieNum)))

(defun variazione (num1 num2)
(expt (- num1 num2) 2))
```

Les marqueurs trouvés, peuvent être appliqués aux objets graphiques BPF, pour obtenir un graphique segmenté dans l'objet *bpf-markers*, réalisé spécialement par le développer OpenMusic Jean Bresson.

B.2 Codes écrits à partir de la littérature

En ayant complétée la segmentation, il faut trouver les points d'abscisses de l'ensemble de départ pour poser des marqueurs, afin de délimiter les régions avec des gestes isolés. La fonction pour isoler les abscisse est la suivante.

```
(defun lunghezza-personalizzata (l)
 (if (atom l) 1 (length l)))

(defun bla(e)
 (cond ((null e) nil)
       (t (cons (lunghezza-personalizzata (car e)) (bla (cdr e))))))

(defun calc-onsets (start list)
 (cons start (loop for item in list collect (prog1 (+ start item)
 (setf start (+ start item))))))
```

B.2.1 Douglas-Peucker

```
(defun douglas-peucker-points (list epsilon)
  (let* ((trans (mat-trans list))
         (seriex (first trans))
         (seriey (second trans))
         (d-max 0)
         (index 0))
    (loop for i from 2 to (- (length list) 1) do
          (let ((d (distancePointSegmentK i seriex seriey)))
              (when (> d d-max)
              (setf index i)
              (setf d-max d))))
     (if (>= d-max epsilon)
         (let* ((list1 (subseq list 0 index))
                (list2 (subseq list index )))
           (append (douglas-peucker-points list1 epsilon)
           (douglas-peucker-points list2 epsilon)))
       (cons (car list) (last list)))))

       (defun distancePointSegmentK (i seriex seriey)
(abs (/ (- (retta i seriex seriey) (nth i seriey))
(sqrt (+ 1 (expt (b seriex seriey) 2))))))
```

B.2.2 Bottom-up

L'algorithme de Bottom-Up commence par l'approximation la plus fine, pour agrandir les segments jusqu'à rejoindre une certaine valeur d'erreur.

```
(defun remove-ieme (l i)
  (append (subseq l 0 i) (subseq l (+ i 1) )))
```

```
(defun bottom-up (seriex seriey epsilon)
   (let ((seg-ts nil)
         costo-fusione-list
         themin)
    (loop for i from 0 to (- (length seriex) 1) by 2 do
          (setf seg-ts (cons (list (nth i seriey) (nth (+ i 1) seriey))  seg-ts ))
          )
   (setf seg-ts (remove-if #'(lambda (x) (null (second x))) (reverse seg-ts)))

   (setf costo-fusione-list  (loop for i from 0 to (- (length seg-ts) 1) collect
                                   (costo-fusione i seg-ts seriex)))
   (setf themin (apply #'min costo-fusione-list))
    (loop while (and (> (length seg-ts) 2) (< themin epsilon)) do
            (progn
              (setf index (position themin costo-fusione-list))
              (print (list "ee" index (length costo-fusione-list)))
              (if (= index (- (length costo-fusione-list) 1)) (setf index (- index 1)))
              (setf (nth index seg-ts) (append (nth index seg-ts)
              (nth (+ index 1) seg-ts)))
              (setf seg-ts (remove-ieme seg-ts (+ index 1) ))
              (setf (nth index costo-fusione-list)
              (costo-fusione index seg-ts seriex))
(print "q")
              (setf costo-fusione-list (remove-ieme costo-fusione-list (+ index 1) ))
              (setf themin (apply #'min costo-fusione-list))
              ))
    seg-ts ))

(defun costo-fusione (i  seriey seriex)
    (let* ((seg (append (nth i seriey) (nth (+ i 1) seriey)))
           (intervallo-x (subseq seriex i (+ i (length seg))))) (sigma intervallo-x seg)))
```

B.3 Régression linéaire

Nous reportons ici des fonctions pour la régression linéaire, utilisées dans l'annexe B. La régression est du type $y = a + bx$. Les paramètres a et b, dans l'éq. B.1, obtenues par le méthode des minimum carrés (ici N est le nombre total des points),

$$a = \frac{\sum_i y_i \sum_i x_i^2 - \sum_i x_i \sum_i x_i y_i}{N \sum_i x_i^2 - \left(\sum_i x_i\right)^2}, \quad b = \frac{N \sum_i x_i y_i - \sum_i x_i \sum_i y_i}{N \sum_i x_i^2 - \left(\sum_i x_i\right)^2} \tag{B.1}$$

sont :

```
(defun a (seriex seriey)
  (/ (- (* (somma seriey) (somma-quadrati seriex))
        (* (somma seriex) (somma-xy seriex seriey)))
     (- (* (length seriey) (somma-quadrati seriex))
        (somma2 seriex) )))
```

```
(defun b (seriex seriey)
  (/ (-  (* (length seriey) (somma-xy seriex seriey))
         (* (somma seriex)  (somma seriey)) )
     (-  (* (length seriey)   (somma-quadrati seriex))
         (somma2 seriex) )))
```

et l'erreur sur la régression équivaut à $\sigma = \frac{[y_i-(a+bx_i)]^2}{N}$, où x_i et y_i sont les valeurs d'abscisse et d'ordonné des points des séries initiales, devient :

```
(defun sigma (seriex seriey) (/ (F seriex seriey) (length seriey)))

  (defun F (seriex seriey)
  (let ((termine 0))
    (do ((i 0 (1+ i))) ((>= i (length seriey)))
      (setq termine (+ termine (expt  (- (nth i seriey) (+ (a seriex seriey) (
      * (b seriex seriey) (nth i seriex)))) 2))))  termine))
```

où :

```
(defun somma (L) (apply '+ L))

(defun somma2 (L) (expt (somma L) 2))

(defun somma-xy (L1 L2)  ; il faut que list1 et list2 aient la même longueur
  (let ((sum 0)) (do ((i 0 (1+ i)))  ((>= i (length L1)))
                   (setq sum (+ sum (* (nth i L1) (nth i L2))))) sum))

(defun somma-quadrati (L) (somma-xy L L))
```

Si nous n'utilisons pas d'autres index temporelles que l'ordre d'apparition des hauteurs, nous pouvons peut utiliser le code suivant :

```
(defun seriex (seriey) (loop for x from 0 to (- (length seriey) 1) collect x) )
```

Pour trouver l'ordonnée d'un point x_p appartenant à la ligne droite de régression :

```
(defun retta (xp seriex seriey)  (+ (a seriex seriey) (* (b seriex seriey) xp) ))
```

Segmento, donc ensemble discret des valeurs d'ordonné calculés à l'aide des paramètres a et b, correspondants aux points d'abscisse compris entre x_0 et x_1 :

```
(defun segmento (x0 x1 seriex seriey)
  (loop for i from x0 to x1 collect (retta i seriex seriey)))
```

Bibliographie

[1] S. Adler. *Lo studio dell'orchestrazione*. EDT, 2002.

[2] J.-P. Allouche and L. Maillard-Teyssier. Incostancy of finite and infinite sequences. *Theoretical Computer Science*, 412 :2268–2281, 2011.

[3] E. Amiot. Gammes bien réparties et transformée de fourier discrète. *Math. and Sci. hum. Mathematical Social Sciences*, 178 :95–117, 2007.

[4] D. Barry, M. Gainza, and E. Coyle. Music structure segmentation using the azimugram in conjunction with principal component analysis. *Audio Engineering Society, Convention Paper*, October 5-8 2007.

[5] H. Berlioz. *Grand traité d'instrumentation et d'orchestration*. Schonenberger, Paris, 1843.

[6] G. Berry. Les algorithmes, cœur de l'informatique. Collège de France, 25 janvier 2008.

[7] J. Bloit, N. Rasamimanana, and F. Bevilacqua. Modeling and segmentation of audio descriptor profiles with segmental models. *Pattern Recognition Lett., Elsevier - in press*, doi : 10.1016/j.patrec.2009.11.003, 2009.

[8] G. Carpentier. *Approche computationnelle de l'orchestration musicale - Optimisation multicritère sous contraintes de combinaisons instrumentales dans de grandes banques de sons*. PhD thesis, UPMC, décembre 2008.

[9] A. Casella and V. Mortari. *La tecnica dell'orchestra contemporanea*. Ricordi, 1974.

[10] A. Cont, S. Dubnov, and G. Assayag. On the information geometry of audio streams with applications to similarity computing. *Audio, Speech, and Language Processing, IEEE Transactions on*, 2011.

[11] S. Dada. Polyphonie orchestrale et évolution de la musique orientale : le cas des mugams symphoniques chez fikret amirov (1922-1984). Master's thesis, Université Sorbonne Paris IV, 2012.

[12] D. Hart E. Keogh, S. Chu and M. Pazzani. Segmenting time series : A survey and novel approach. *Machine Perception Artificial Intelligence*, 57, 2004.

[13] P. Esling. *Analyse multi-objective des séries temporelles*. PhD thesis, UPMC, Décembre 2012.

[14] P. Esling and C. Agon. Time series analysis. *ACM Computing Surveys (CSUR)*, 45(12), 2012.

[15] J. Titanmaki J. Himberg, J. Korpiaho and H. Toivonen. Time series segmentation for context recognition in mobile devices. *Proceedings of the 1st IEEE International Conference on Data Mining*, pages 203–210, 2001.

[16] C. Koehclin. *Traité de l'orchestration*. Edition Max Eschig, 1943.

[17] D. Lemire. A better alternative to piecewise linear time series segmentation. *arXiv :cs/0605103v8*.

[18] D. Lewin. Intervalic relations between two collections of notes. *Journal of Music Theory*, 3 :298–301, 1959.

[19] M. Mannone. *Dalla Musica all'Immagine, dall'Immagine alla Musica - Relazioni matematiche fra composizione musicale e arte figurativa*. ISBN : 88-97284031. Compostampa, 2011.

[20] M. Mannone. Characterization of the degree of musical non-markovianity. *arXiv :1306.0229 [physics.data-an]*, 2013.

[21] M. Mannone, S. Lo Franco, and G. Compagno. Comparison of non-markovianity criteria in a qubit system under random external fields. *Phys. Scr. T*, 153(014047), 2013.

[22] D. Rafailidis, A. Nanopoulos, Y. Manolopoulos, and E. Cambouropoulos. Detection of stream segments in symbolic musical data. *Proceedings of the 9th International Conference on Music Information Retrieval*, pages 83–88, September 14-18 2008.

[23] N. Rimsky-Korsakov. *Principi di Orchestrazione*. Rugginenti Editore, 1912.

[24] S. Sciarrino. *Le figure della musica da Beethoven a oggi*. Ricordi, 1998.

[25] H. Shatkay. Approximate queries and representations for large data sequences. *Data Engineering, Proceedings of the twelfth International Conference*, pages 536–545, 1996.

[26] K. Vasko and H. Toivonen. Estimating the number of segments in time series data using permutation tests. *Proceedings of the 2002 IEEE International Conference on Data Mining, IEEE Computer Society*, pages 466–473, 2002.

[27] I. Xenakis. *Musique Formelles*. La Revue Musical - Paris, 1963.

[28] K. Iwamoto Y. Tanaka and K. Uehara. *Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle*, volume 58. Machine Learning, 2005.