# Modélisation de structures rythmiques

Benoit MEUDIC
Mémoire de DEA ATIAM
Université d'Aix-Marseille II

Laboratoire d'accueil : Ircam - Centre Georges Pompidou
Equipe : Représentations Musicales
Responsable de stage : Gérard Assayag

15 janvier 2001

# Table des matières

# Table des figures

# Première partie

# Introduction

Ce rapport présente les recherches que j'ai effectué au sein de l'équipe Représentations Musicales de l'Ircam dans le cadre de mon stage de DEA Atiam. L'équipe Représentations Musicales s'intéresse particulièrement aux aspects discrets et symboliques des structures musicales (notamment leur notation solfégique) et à leurs traductions en abstractions manipulables sur un ordinateur.

Le travail proposé consistait à étendre les possibilités du logiciel de quantification rythmique Kant, en particulier de manière à pouvoir analyser et représenter des structures rythmiques présentant une pulsation et une métrique régulières.

Le problème sera analysé dans une première partie, en même temps que nous présenterons l'état des recherches actuelles. Dans une deuxième partie, nous détaillerons les extensions apportées, après s'être interrogé sur le rôle de la segmentation dans la quantification, notion fondamentale à la base du logiciel Kant. Nous développerons en particulier quelques méthodes de segmentation, basées sur le coefficient d'auto-corrélation ou bien sur le concept de marquage.

Je tient à remercier toute l'équipe de Représentations Musicales, Gérard Assayag, Carlos Agon, Olivier Lartillot, Charlotte Truchet, Mauro Lanza et Moreno Andreatta pour la gentillesse et la disponibilité qu'ils m'ont accordée tout au long du stage. Je remercie par ailleurs Pascal Henriot et Catherine Dichtel pour les différents moments passés avec eux pendant ce stage.

## Deuxième partie

# La notation rythmique automatique : passage du nombre au symbole musical

# Chapitre 1

# La reconnaissance de rythmes par la machine

La notation rythmique proportionnelle existe depuis le quatorzième siècle. La longévité de ce système de notation pourrait sans doute s'expliquer par le fait qu'elle est une représentation assez naturelle de notre perception de la pulsation et du rythme. Mais l'apparente facilité que l'on a à extraire des informations musicales telles la pulsation et le rythme d'un flux sonore cache l'extrême complexité du système perceptif mis en jeu. Cette complexité explique les difficultés que l'on rencontre lorsqu'on cherche à construire un modèle informatique capable de comprendre et représenter le rythme suivant la notation traditionnelle.

Après avoir présenté les motivations qui nous poussent à établir un modèle d'extraction automatique du rythme, nous nous appuierons sur les faiblesses des outils commerciaux actuels (quantificateurs traditionnels) pour décrire les avancées de la recherche dans les différents domaines concernés.

## 1.1 L'intérêt des modèles informatiques

Une des particularités de la musique par rapport à d'autres domaines comme les langues ou les arts plastiques est que le temps tient une place essentielle dans son appréhension. On ne saurait concevoir de musique sans qu'elle soit réalisée dans le temps. De plus, le temps joue un rôle très grand dans la compréhension qu'on a de la musique. On ne peut analyser une musique sans parler du temps dans lequel elle s'inscrit.

Notre perception du temps se fait suivant un mécanisme actif très complexe qui transforme progressivement les informations reçues de manière à leur donner un sens. En fait, nous effectuons une mesure du temps. La mesure la plus courante est celle qui divise le temps en secondes. Sans mesure, il ne nous serait pas possible de mémoriser ou reproduire une distance temporelle supérieure à quelques secondes. La manipulation du temps discrétisé en secondes intervient tous les jours de notre vie quotidienne. Le temps d'un café, d'un cours ou d'un voyage sont exprimés selon une structure temporelle immuable (nombre de minutes, d'heures ou de journées). Dans ces activités quotidiennes, le temps est une dimension aussi banale qu'une autre.

Par contre, lorsque le temps devient un sujet principal d'étude, lorsqu'il s'agit de l'organiser, de le structurer de manière créative, musicale, d'en extraire un rythme, il est (à priori) nécessaire de décrire le nouveau sens qu'on lui donne par un nouveau langage. Ce langage doit mettre l'accent sur l'information que l'on veut transmettre. Dans le domaine musical, la relation de proportion qu'entretiennent deux ou plusieurs valeurs temporelles a beaucoup d'importance, et constitue la base du langage rythmique. Comme il nous est difficile de percevoir des relations de proportion complexes, les valeur temporelles sont souvent exprimées comme premiers multiples ou sous-multiples d'une unité temporelle de référence (pulsation). Cette représentation se rapproche de notre processus de perception du temps qui analyse le plus souvent les durées relativement à celles précédemment perçues ou mémorisées. Pour cela, nous approximons les durées de manière à pouvoir les comparer. Ainsi, deux durées assez proches (par exemple 120 et 100 millisecondes) seront perçues comme égales. Nous opérons une quantification. Ce processus doit être modélisé pour être compris par les machines qui à priori n'effectuent pas cette opération.

En effet, si nous mémorisons et reproduisons des rythmes en référence à une unité temporelle (pulsation) proche de la seconde, l'ordinateur ou les instruments de mesure se réfèrent plus souvent à la milliseconde. L'enregistrement, la synthèse, et le codage de données musicales se fait, pour ce qui est du rythme, par la donnée de valeurs temporelles absolues (durées en ms). Le rythme lui-même n'est pas codé, ce sont les occurrences temporelles et les durées qui sont codées.

Ainsi, pour que la machine fournisse et comprenne des informations rythmiques, indispensables aux musiciens, il est nécessaire de créer un modèle permettant d'effectuer la correspondance entre des durées absolues (exprimées en millisecondes) et une notation proportionnelle. Lorsqu'il est nécessaire d'approximer les durées pour faire apparaître des rapports de proportion, on effectue une

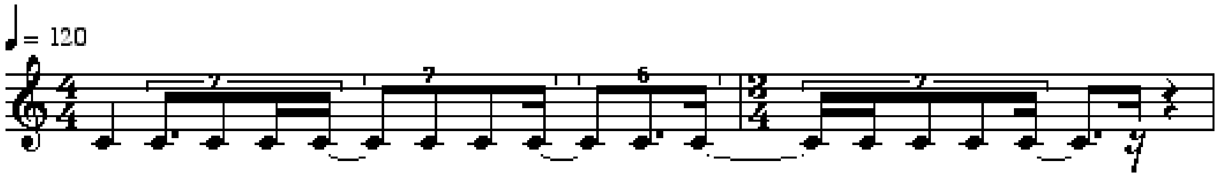


FIG. 1.1 – une mauvaise quantification
Les durées sont (en millisecondes) :
476 237 115 135 174 135 155 240 254 118 112 118 138 476

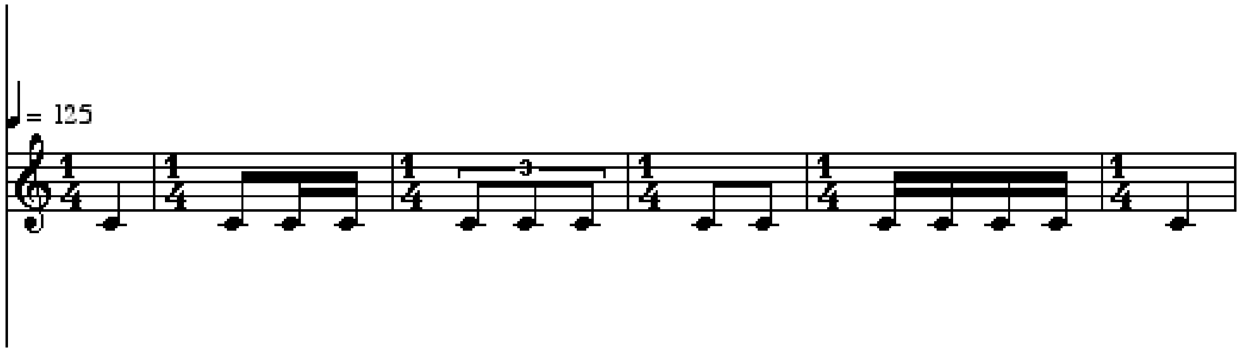


FIG. 1.2 – une meilleure quantification

quantification. Cette étape est couramment effectuée par les logiciels du commerce (éditeurs graphiques, séquenceurs, logiciels de gravure), mais les résultats sont rarement satisfaisants.

## 1.2 Les quantificateurs traditionnels

Etant donnée une suite de durées, le processus de quantification le plus simple consiste à choisir une unité temporelle arbitraire ou non, puis à approximer toutes les durées à quantifier par les valeurs les plus proches d'une grille métrique dont la résolution est un sous-multiple de l'unité choisie. Tous les quantificateurs traditionnels sont basés sur ce processus. Par ailleurs, ils est souvent nécessaire de leur fournir plusieurs paramètres de manière à limiter leur marge d'erreur. Voici par exemple une étape de quantification opérée à l'aide de quantify, un quantificateur traditionnel dans l'environnement Open-Music.

Présentation des paramètres :

Hormis la liste de durées à quantifier, les paramètres principaux sont une valeur de tempo et une liste de signatures (nombre de pulsations par mesure ex. ((3 4) (7 8) (1 4) ... ) ). Ces paramètres suffisent normalement à réaliser une quantification puisque le tempo fournit l'unité temporelle nécessaire à l'établissement de la grille métrique. La valeur <max/> rajoute une contrainte sur la division maximale d'une pulsation, le paramètre <forbid> fournit une liste de subdivisions que l'on interdit (ou impose) dans une pulsation.

Une quantification réalisée sur le flux de durées suivant (en prenant un tempo de 120 à la noire, équivalent à une pulsation de 500 ms) : 476 237 115 135 174 135 155 240 254 118 112 118 138 476 donne pour résultat la partition de la figure 1.1. Le résultat attendu serait celui de la figure 1.2. On peut alors se demander d'ou vient la différence entre les deux résultats.

## 1.3 Les défauts des quantificateurs traditionnels

La mauvaise quantification de la figure 1.1 est due au fait que les approximations effectuées n'ont pas été faites dans la perspective de trouver une structure musicale, mais dans la perspective d'approximer le moins possible les durées. La représentation de la figure 1.1 est sans doute plus exacte que celle de la figure 1.2, et donc plus proche de l'enregistrement initial. Seulement, cette représentation ne correspond pas forcément à ce que notre oreille analyserait musicalement si cette séquence était jouée. On pourrait dire que le grain de définition de cette représentation n'est pas assez grand pour qu'une information plus proche de notre perception soit représentée. La contrainte du quantificateur (faire le moins d'approximations) n'a alors pas de sens musical. Pourtant, notre perception du rythme est sans doute liée à cette notion de "faire le moins pour obtenir le plus". La représentation recherchée est en effet la plus simple que l'on puisse imaginer (dans le langage de la notation traditionnelle) de façon à ce que les informations les plus sensibles à notre oreille ne soient pas perdues. Tout le problème est donc de définir le rapport entre la quantité d'information représentée et la complexité de la représentation.

Un autre critique des quantificateurs traditionnels concerne leur manque de souplesse par rapport aux méthodes à employer selon les types de séquences musicales à quantifier. En effet, il est souvent nécessaire de rentrer un assez grand nombre de paramètres pour cerner le résultat attendu, ce qui n'est pas forcément désirable, surtout dans le cas de reconnaissance automatique de structures rythmiques. Parmi les quantificateurs traditionnels, il n'existe pas de méthode assez générale pour n'avoir quasiment aucune opération manuelle à effectuer quel que soit le type de séquence.

Par ailleurs, une fois la quantification effectuée et la pulsation supposée, il serait nécessaire pour trouver un rythme d'émettre des hypothèses par des méthodes appropriées sur différentes structures possibles, ce que les quantificateurs traditionnels ne font pas. En effet, ils sont limités (à moins que l'utilisateur ne rentre à la main ses contraintes) aux structures les plus simples (4 temps par mesure le plus souvent).

# Chapitre 2

# Les différentes approches

Le domaine de la reconnaissance automatique de structures musicales est tellement vaste que les recherches sont le plus souvent effectuées en vue d'un objectif assez précis ne couvrant qu'une partie du problème.

On peut distinguer deux principaux sous-domaines de recherche :

1. la recherche d'une pulsation
2. la modélisation d'un rythme, étant donnée une pulsation.

De plus, pour chaque domaine, le modèle est le plus souvent restreint à un type de donnée musicale (signaux acoustiques, fichiers Midi).

Cependant, certains modèles n'ont pas pour objectif d'extraire une pulsation ou un rythme d'une séquence, mais visent à quantifier directement les données sans étapes intermédiaires. C'est le cas par exemple de certains quantificateurs non traditionnels, dont l'objectif est le plus souvent de transcrire en partition un enregistrement issu d'une interprétation, présentant des écarts (nuances, rubato accélérations) par rapport à la partition originale. Ces quantificateurs s'inspirent souvent du domaine de l'intelligence artificielle pour appliquer à la musique certains concepts.

Ainsi, le quantificateur [Desain Honing 89] est basé sur un réseau à satisfaction de contraintes sans apprentissage. Chaque cellule du réseau correspond à l'une des durées à quantifier. Des cellules représentant des durées voisines interagissent entre elles par l'intermédiaire d'une fonction d'interaction qui les modifie de manière à ce que les durées aient "la meilleure proportion possible". Une fois le réseau stabilisé, les durées quantifiées sont envoyées en sortie du réseau.

Un autre quantificateur [Longuet higgins 87] est un modèle symbolique basé sur une connaissance minimale du mètre. Plusieurs pulsations sont testées de la façon suivante : Une pulsation calée sur le début de la séquence à quantifier est subdivisée en deux ou trois de façon à trouver la meilleure correspondance avec les onset des premières durées. L'algorithme itère ensuite le processus, positionnant séquentiellement les pulsations jusqu'à aboutir à la fin de la séquence. Eventuellement, suivant la subdivision choisie, la pulsation peut être modifiée au cours de l'itération. Le résultat est une suite de pulsations subdivisées en deux ou trois. Plusieurs pulsations sont testées, et la meilleure suite de subdivisions est retenue.

Ces deux algorithmes sont assez représentatifs des méthodes de quantification disponibles car ils illustrent deux différentes approches caractéristiques des problèmes posés en intelligence artificielle.

Pour mieux comprendre leur qualités et défauts, les modèles doivent être décrits sous différents points de vue, ce qui a été fait dans [Desain 93] pour le cas des deux quantificateurs dont nous venons de parler.

S'ils permettent de mesurer le potentiel que peut apporter le domaine de l'intelligence artificielle à la modélisation rythmique, ces modèles ne peuvent garantir des résultats satisfaisants dans tous les cas de figure. Le moyen de se rapprocher des modèles presque parfaits est de diviser le champ d'étude pour se pencher sur des problèmes plus spécifiques.

Nous allons décrire deux types d'approche :

- la recherche d'une pulsation
- la modélisation d'un rythme, étant donnée une pulsation.

## 2.1 La recherche d'une pulsation

La pulsation est une donnée subjective que nous formulons mentalement lorsque nous sommes dans une situation d'écoute. Une façon de la modéliser serait de chercher des méthodes qui se rapprocheraient de notre perception, c'est à dire qui prendraient en compte les derniers évènements perçus pour modifier ou conforter notre attente du prochain battement.

Ces modèles, décrits en première section, doivent donc pouvoir être implémentés en temps réel (même si ils ne sont pas toujours présentés tels quels). Dans une deuxième section, nous décrirons les autres approches.

### 2.1.1 Les algorithmes réalisables en temps réel

Nous allons décrire deux des approches les plus importantes :

- la première consiste à simuler la pulsation par des oscillateurs qui s'adaptent suivant les informations musicales auxquels ils sont soumis.
- la deuxième consiste à analyser les derniers évènements musicaux reçus par la méthode du coefficient d'auto-corrélation de manière à en déduire une pulsation.

#### La modélisation de la pulsation par banc d'oscillateurs

Edward Large [Large 94] nous propose de considérer dans la séquence musicale la seule suite d'inter-onsets (durées séparant deux attaques successives). Sa méthode s'oppose aux modèles de réseaux de neurones à temps discret qui ne prennent pas en compte l'information temporelle. Pour cela, il considère un oscillateur isolé, ayant une période d'oscillation par défaut, qu'il couple à un autre oscillateur représentant le train d'impulsions de la séquence à analyser. L'oscillateur corrige alors sa période selon la méthode du gradient descendant, de manière à minimiser l'écart entre sa pulsation et la période déduite de la nouvelle impulsion. Si cet écart est trop important il ne tient pas compte de la dernière impulsion. Plusieurs valeurs de couplage sont testées puis les résultats sont analysés par des "diagrammes régime" qui représentent la force de couplage des deux oscillateurs en fonction du rapport de couplage qui les unit (nombre de périodes d'un oscillateur pour une période de l'autre oscillateur couplé).

Ce modèle s'avère assez efficace, mais son implémentation reste assez difficile à effectuer dans un environnement qui ne serait pas adapté au temps réel.

De plus, plusieurs problèmes restent à résoudre :

- plusieurs oscillateurs différents peuvent réagir en même temps aux mêmes impulsions. Il reste alors à les relier entre eux de manière à faire ressortir une pulsation.
- les paramètres musicaux autres que les inter-onset ne sont pas pris en compte dans le modèle, et en particulier les accents (dynamique).

Eric Scheirer [Scheirer 97] propose un modèle adapté à plusieurs types de données sonores, et en particulier aux signaux acoustiques (principale différence avec le modèle précédant). Pour cela, il remarque que l'enveloppe seule des signaux contient suffisamment d'informations sur le rythme pour en extraire une pulsation (des signaux bruités sont analysés de la même façon que le signal d'origine, donc l'information est contenus dans l'enveloppe). Cela permet de réduire la taille en mémoire de la séquence à analyser, qui devient inférieure à celle qu'aurait la même séquence en représentation midi. L'algorithme utilisé commence par diviser le signal à l'aide d'un banc de 6 filtres, puis détermine l'enveloppe de chacun des 6 signaux. Chaque filtre est relié à plusieurs oscillateurs. Certains oscillateurs vont réagir au signal si leur période correspond aux impulsions suivant les règles d'algorithmes spécifiques. Plusieurs pulsations vont être extraites, représentant différents niveaux rythmiques.

Ces deux algorithmes présentent un avantage par rapport à ceux basés sur le coefficient d'auto-corrélation : ils sont sensibles à la phase de la pulsation, c'est à dire au moment dans le temps où on perçoit un battement. Cette information est pourtant indispensable pour rechercher un rythme dans une étape suivante.

#### La modélisation de la pulsation à l'aide du coefficient d'auto-corrélation

Le coefficient d'auto-corrélation est une mesure assez efficace des périodicités. Depuis l'article de Judith Brown [Brown 1993], il a été à la base de plusieurs modèles de détection de pulsation.

Judith Brown [Brown 1993] propose d'extraire d'une séquence d'inter-onset une valeur donnant l'emplacement de la mesure (groupe de pulsations) et éventuellement la pulsation par l'analyse des coefficients d'auto-corrélation calculés sur l'ensemble de la séquence. Pour cela, elle interprète le ou les pics prédominants (répondants à certains critères) du graphe d'auto-corrélation comme étant des positions d'articulation qu'elle relie aux notions de mètre. Nous verrons que les pics prédominants sont assez difficiles à repérer et qu'ils ne correspondent pas toujours aux critères énoncés par cette méthode. Toutefois, elle montre le potentiel que l'on peut tirer d'une mesure mathématique pour modéliser un phénomène perceptif.

De façon à repérer des variations de pulsation, Tanguiane [Tanguiane 94] nous propose d'utiliser la méthode de résolution de variables pour formater les données avant de les analyser par le coefficient. Cette méthode consiste à régler le grain de définition des durées réparties sur une échelle temporelle. Des durées légèrement différentes seront donc reconnues par le coefficient comme appartenant à la même périodicité. Par exemple, les durées 10 et 11 réparties dans l'échelle d'unité 1 : 1 0 0 0 0 0 0 0 0 0 (10 unités) et 1 0 0 0 0 0 0 0 0 0 0 (11 unités) seront transformées en 1 1 0 0 0 0 0 0 0 0 (10 unités mais deux 1) et 1 1 0 0 0 0 0 0 0 0 0 (11 unités mais deux 1) ce qui revient à baisser la précision temporelle. Les pics du graphes d'auto-corrélation mettant en évidence la répétition 10 puis 11 seront alors beaucoup plus significatifs.

Desain [Desain - Siebe de Vos] a adopté une méthode, simulation proche de notre perception en temps réel qui utilise un système de fenêtrage : A chaque battement de pulsation supposé, une fenêtre est disposée sur les derniers évènements de la séquence, dans laquelle un calcul de coefficients d'auto-corrélation est effectué. Des ces coefficients est déduite une pulsation, qui détermine la valeur de déplacement de la fenêtre dans la séquence, à partir de laquelle on calculera une nouvelle valeur de pulsation.

Ces algorithmes serviront de base à une méthode de segmentation que nous décrirons en 4.3.3.

### 2.1.2 Les algorithmes non temps réel

Une autre façon d'envisager la recherche d'une pulsation est de considérer l'ensemble des éléments de la séquence dans le même temps, de manière à détecter ceux qui prédominent par rapport aux autres. En effet, le fait de ne plus avoir la contrainte du temps réel permet de comparer un élément à son précédent, mais aussi à son suivant, et de cette manière permet de dire qu'un élément prédomine localement. Les accents de la séquence peuvent ainsi être trouvés, et par suite la pulsation si l'on suppose qu'elle marque les temps forts.

Emilios Cambouropoulos présente ainsi dans une théorie de la structure musicale [Cambouropoulos 98] une mesure de l'accentuation des éléments d'une séquence musicale au format midi. Cette mesure est réalisée par l'intermédiaire de marquages. Les marquages sont des poids qui sont attribués aux éléments suivant l'importance qu'ils ont suivant certains critères. Les marquages utilisés par Cambouropoulos sont de deux types : les marquages "Grouping Identity-Change" (un intervalle est marqué si le suivant est différent) et les marquages "Identity-Change Rule" (un intervalle est marqué si le suivant est plus petit). Les marquages s'appliquent aux hauteurs, inter-onsets, durées et intensités. A chaque marquage correspond un poids attribué aux éléments de la séquence vérifiant le marquage. Plusieurs marquages combinés entre eux donnent pour résultat une courbe de poids (un poids par élément de la séquence) dont les pics peuvent être interprétés comme étant les accents de la séquence. On fait ensuite correspondre à cette courbe la grille métrique pour laquelle la somme des poids des accents coïncidant avec la grille est la plus grande. La pulsation correspond alors au pas de la grille sélectionnée.

Plusieurs améliorations sont ensuite possibles : pondérer les marquages avant de les combiner entre eux, raffiner les marquages (Cambouropoulos préconise d'introduire des poids proportionnels à la différence de taille entre deux intervalles de valeurs, ou bien des poids directement proportionnels à la taille des intervalles). Enfin, en complément de la courbe d'accentuation décrite dans la section précédente, Cambouropoulos propose une méthode de plus haut niveau qui repère des groupements rythmiques (patterns). Les éléments positionnés au début d'un groupement sont marqués, puis la courbe de pattern est ajoutée à la courbe d'accentuation. La courbe finale devrait présenter des pics plus marqués.

Cette méthode sera à la base d'une méthode de segmentation que nous décrirons en 4.3.4.

## 2.2 La recherche d'un rythme

Le rythme se réfère souvent à une organisation d'évènements dans le temps, de façon à ce qu'ils se combinent de façon perceptive en groupes, et qu'ils induisent le sentiment d'un mètre ([Cooper Meyer 60], [Lerdhal et Jackendoff 83]). Ainsi, tout comme la pulsation, le rythme revêt un aspect subjectif.

Dans cette partie, nous supposeront connue une pulsation.

Goto [Goto 95] propose une méthode originale pour détecter les groupements de pulsation : il suppose que les débuts de mesure se font le plus souvent aux changements d'accord dans la séquence à analyser. Sa méthode peut s'appliquer aux signaux acoustiques car il n'est pas nécessaire de connaître les fréquences des accords, puisqu'on s'intéresse uniquement aux changements d'accords. Pour étudier la métrique de la séquence, goto suppose que les mesures comportent quatre temps (temps fort, temps faible alternés), et que les changements d'accord arrivent plus fréquemment en début de mesure qu'en milieu de mesure. Chaque changement d'accord est alors classé suivant ces hypothèses. Cette méthode est intéressante car elle se rapproche des méthodes utilisant les marquages qui seront développées dans la partie 4.3.4.

Emilios Cambouropoulos [Cambouropoulos 98] propose une méthode de catégorisation de la séquence. Il s'agit de regrouper les patterns similaires entre eux. Deux patterns sont similaires lorsque leur distance est inférieure à une certaine valeur. Parmi les différentes classes de catégories, on choisit celle qui compte le moins de recouvrements.

## 2.3 Synthèse

Au vu des différentes recherches, il apparaît assez clairement que pour obtenir des modèles fiables et résistants de détection rythmique, il faut réduire les objectifs en sous-problèmes, auxquels on pourra répondre par le recoupement de plusieurs méthodes. Ainsi, la reconnaissance d'une pulsation constitue un problème à part entière, et fera l'objet du développement d'une méthode de segmentation spécifique au chapitre 4.3.3.

Par ailleurs, nous pouvons distinguer deux types d'approches, pouvant être complémentaires l'une de l'autre :

- La recherche d'une structure rythmique du plus bas niveau vers le plus haut niveau, en commençant par la recherche de la pulsation, puis du mètre. Cette méthode est proche de notre perception du rythme en temps réel.(une méthode utilisant l'auto-corrélation sera décrite en 4.3.3)
- La recherche d'une structure rythmique en commençant par le plus haut niveau (découpage en grandes parties) pour parvenir éventuellement au niveau de la pulsation. Cette méthode ne peut être implémentée en temps réel, mais permet des analyses rythmiques de haut niveau plus fines que la précédente.(une méthode utilisant les marquages sera décrite en 4.3.4)

# Troisième partie

# La librairie Kant

# Chapitre 3

# Présentation de la librairie

Initialement implémenté sur Patchwork, KANT [Agon 94] est un modèle de quantification rythmique où une segmentation non périodique d'un flux de durées est effectuée sans connaissance a priori d'une pulsation.

## 3.1 Les objectifs initiaux

L'objectif initial auquel devait répondre la librairie Kant était de trouver des structures complexes présentes dans la musique contemporaine, adaptées aux nombreux changements de métrique et de tempo de la notation des compositeurs d'aujourd'hui. La philosophie serait donc de considérer la quantification comme une étape de composition, dans laquelle le compositeur puisse s'exprimer, plutôt que comme un procédé de reconnaissance d'une structure rythmique déjà existante et répondant à un style bien défini.

Nous verrons que des extensions permettent cependant à ce logiciel de réaliser l'analyse de structures musicales plus classiques, et de reconstituer les rythmes présents dans les séquences musicales à quantifier issues d'une partition.

Le stage a consisté à compléter le portage de Kant sur Open-Music, commencé auparavant par mes soins [meudic 99], à proposer de nouvelles méthodes de reconnaissance de la segmentation métrique, à permettre à l'utilisateur de proposer ses méthodes dynamiquement, et de manipuler aisément les structures hiérarchiques de segmentation.

## 3.2 Le processus de quantification dans Kant

La première étape du processus de quantification est une segmentation de l'axe de temps sur lequel ont étés positionnées séquentiellement les éléments de la séquence musicale (au format de fichier midi) à quantifier. La segmentation est une étape clef dans le processus de quantification. Elle détermine à la fois la forme rythmique globale (groupement des rythmes en mesures) de la séquence quantifiée, et la valeur de la pulsation qui sera utilisée pendant l'étape de quantification.

Cette segmentation permettra, par le biais de dilatations et de compressions, de corriger le flux de durées de manière à respecter une contrainte qui apparaîtra dans la notation finale : chaque durée repérée par une barre de segmentation sera placée en début de mesure.

La deuxième étape du processus consiste à déterminer une pulsation (unité de temps) qui soit un diviseur commun " approximatif " des durées séparant les barres de segmentation posées en première étape. Le diviseur commun est nécessairement approximatif car pour qu'il ait de l'intérêt, sa valeur ne doit pas être trop faible. Par exemple, nous dirons que 5 est un diviseur commun approximatif de 11, 15 et 19.

La méthode utilisée calcule la pulsation par un algorithme d'approximation équivalent à l'algorithme développé à l'Ircam pour l'extraction de la fondamentale virtuelle d'un spectre discret exprimé comme une liste de fréquences (voir description et figure 3.2). L'analogie utilisée fait correspondre la liste de fréquences à la liste de durées et la valeur de la fondamentale virtuelle à la valeur de la pulsation.

Description de l'algorithme :

Deux paramètres en entrée :

- values (durées des différents groupes)

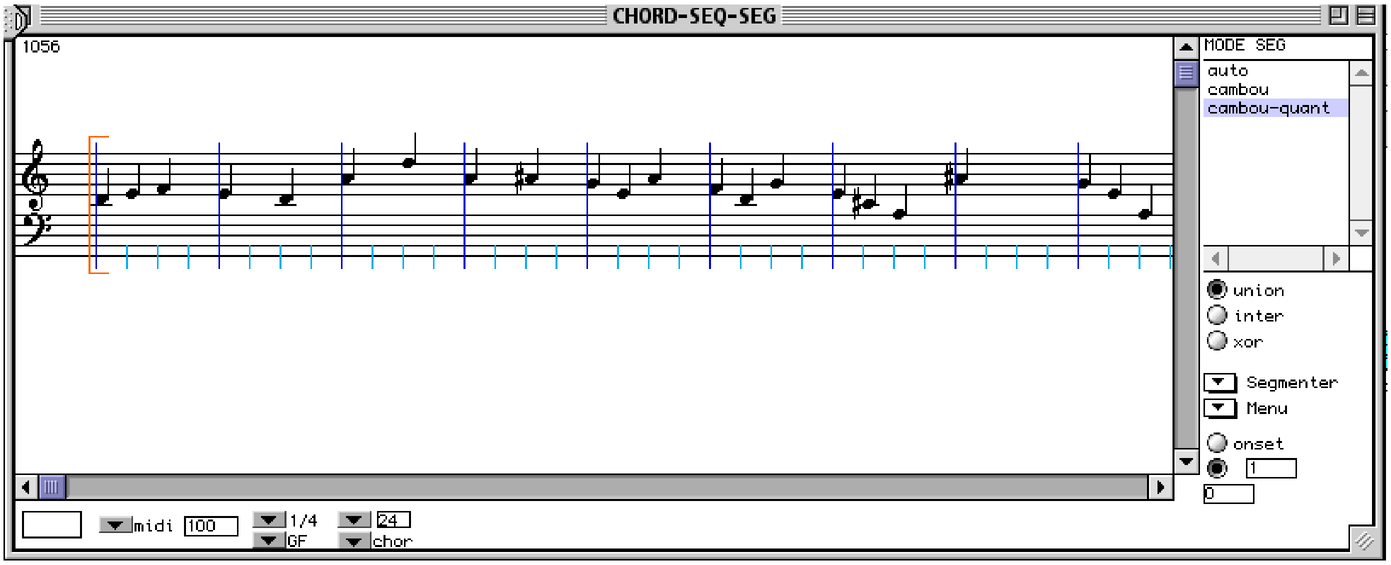


FIG. 3.1 – L'éditeur Kant

- tolérance (valeur de tolérance pour les approximations)

A chaque durée de la liste correspondent deux durées extrêmes ('val-below et 'val-above) définissant un intervalle de tolérance (défini à l'aide du paramètre 'tolérance), centré sur la durée en cours.

L'algorithme opère une récursion sur la liste de durées et calcule deux pulsations à chaque itération et pour toutes les subdivisions envisageables (en nombre de pulsations) des durées extrêmes de l'intervalle de tolérance courant. Ces deux pulsations (gcd-min et gcd-max) seront renvoyées en paramètre de l'itération suivante. Dans le cas où à un moment de l'itération, aucune subdivision n'est envisageable, l'algorithme revient en arrière et lance la récursion sur une nouvelle subdivision. Dans le cas où l'algorithme parcoure toute la liste, la valeur de pulsation retenue sera une moyenne des deux pulsations extrêmes de la dernière itération.

A l'issue de la deuxième étape, nous aurons donc déterminé une pulsation suivant laquelle pourront s'exprimer approximativement les durées des différents groupes formés par les segmentations. De façon à ce que la pulsation devienne un sous-multiple parfait de tous ces groupes de durées, nous allons ensuite modifier leur taille par des opérations de compression et de dilatation. Ainsi, les groupes de durées apparaîtront sous la forme de mesures dans la partition finale.

Enfin, lors d'une troisième étape, une quantification que nous appellerons " quantification locale " est effectuée, à l'aide d'un quantificateur traditionnel. Les emplacements des barres de mesures (définis par la segmentation) sont respectés pendant la quantification grâce aux compressions-dilatations effectuées précédemment.

## 3.3 La version 1.1

La version 1.1 développée au cours de l'année 1999 était une première version minimale de la librairie "Kant" sous OpenMusic, reprenant en grande partie la version initiale développée par Carlos Agon sous Patchwork. Elle comporte des méthodes de segmentation très simples permettant de quantifier des séquences monodiques. Ces méthodes repèrent le plus souvent des maximum locaux parmi les listes des paramètres Midi de la séquence musicale à quantifier (hauteurs, onsets, durées, vélocités). Si elles sont très simples, elles permettent de déterminer certaines structures musicales dont les évènements importants (donc interprétés ici comme des débuts de mesure) correspondent à des durées (des hauteurs, des intensités) généralement plus longues (ou plus courtes) que la moyenne. D'autres segmentations consistent à disposer sur la séquence musicale une grille (chaque barreau correspondant à une barre de segmentation) dont le pas peut être modifié par l'utilisateur de façon à caler les barreaux sur des évènements de la séquence que l'on veut faire correspondre avec des barres de mesure. Enfin, il est possible de combiner ces segmentations entre elles à l'aide des opérateurs ensemblistes habituels : union, intersection et leurs complémentaires.

```lisp
(defun agcd (values tolerance)
  "approximate gcd with backtracking"
  (labels
    ((grid-above (val) (* val (1+ tolerance)))
     (grid-below (val) (- val (* val tolerance)))
     (gcd-try (values gcd-min gcd-max)
       (when (<= gcd-min gcd-max)
         (if (null values)
           (/ (+ gcd-min gcd-max) 2.0)
           (let* ((val-below (grid-below (first values)))
                  (val-above (grid-above (first values)))
                  (quo-min (ceiling (/ val-below gcd-max)))
                  (quo-max (floor (/ val-above gcd-min))))
             (loop for quotient from quo-min upto quo-max
                   for gcd-interval =
                   (gcd-try (rest values)
                            (max gcd-min (/ val-below quotient))
                            (min gcd-max (/ val-above quotient)))
                   when gcd-interval do (return-from gcd-try gcd-interval)))))))
    (gcd-try values .1 (grid-above (first values)))))
```

FIG. 3.2 – algorithme du gcd

# Chapitre 4

# Présentation de la version 2.0

La version 2.0 développée au cours du stage reprend la version 1.1 à laquelle ont été rajoutées plusieurs fonctionnalités (principalement sur le contrôle des paramètres midi) et des méthodes de segmentation, visant à reconnaître des structures rythmiques usuelles (de métrique simple et de tempo constant sur plusieurs mesures) et/ou des structures rythmiques plus particulières (comportant des changements de métrique et de tempo).

## 4.1 Quels outils pour quels buts ?

Nous allons distinguer deux types d'utilisation du logiciel Kant :

- La recherche de structures rythmiques supposées usuelles (métrique simple et tempo constant).
- La recherche de structures rythmiques plus complexes (nombreux changements de métrique et de tempo, fréquents en musique contemporaine).

Il est nécessaire d'opérer cette distinction car suivant les objectifs fixés, différentes méthodes peuvent être utilisées certaines pouvant donner lieu à un traitement spécifique selon l'objectif visé..

Par exemple, si l'on recherche des structures supposées classiques (tempo constant, métrique classique), une approche peut être d'utiliser tout d'abord une segmentation pour reconnaître une pulsation. Les barres de segmentation seront alors interprétées comme marques d'une unité métrique. L'étape suivante serait de quantifier la séquence pour finalement faire ressortir un rythme à l'aide d'une nouvelle segmentation qui serait alors interprétée comme marque de groupements métriques (et non plus comme marque d'une unité métrique).

## 4.2 La phase de segmentation

La segmentation est, comme nous l'avons déjà vu, l'étape déterminante du processus de quantification de Kant. Nous devons donc nous interroger sur les différentes manières d'utiliser les segmentations pour cerner les résultats qu'on peut en attendre.

Déjà, une segmentation peut être interprétée de plusieurs façon lors de l'étape de quantification. Dans tous les cas, la segmentation représente un point d'articulation. Ensuite, selon son importance, l'articulation peut correspondre à une marque de pulsation, de mesure, d'archi-mesure (marque de changement de tempo), de demi-mesure etc... Plusieurs niveaux hiérarchiques apparaissent et feront l'objet d'un traitement différent.

### 4.2.1 L'interprétation des segmentations

Pendant l'étape de quantification, les segmentations servent à définir une valeur de pulsation, et sont dans le même temps interprétées comme marques de mesure.

On peut alors se demander si il est indispensable de réaliser ces deux opérations en même temps, puisqu'elles correspondent à deux objectifs différents :

- trouver une pulsation
- trouver des groupements métriques

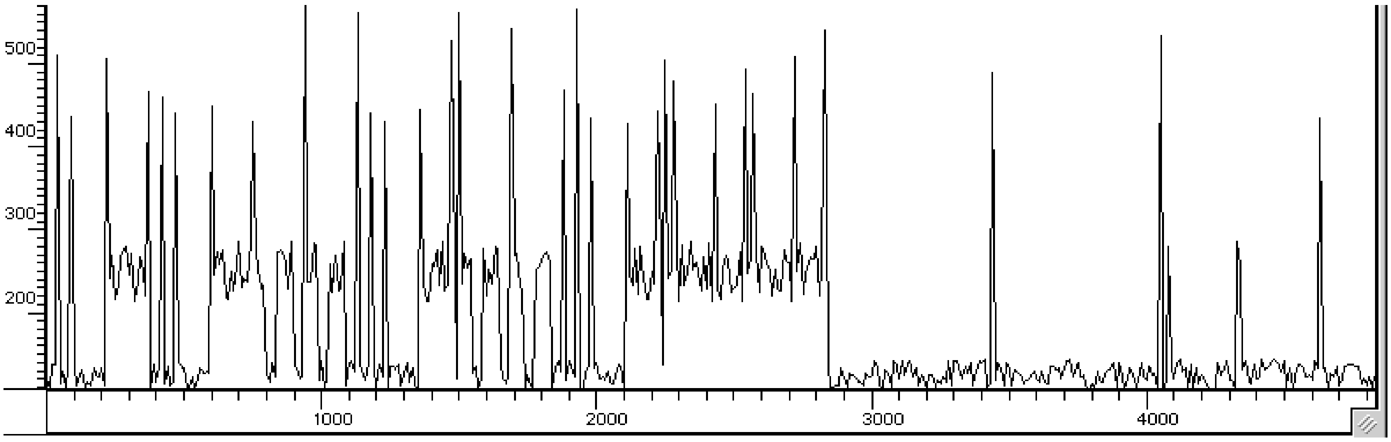


FIG. 4.1 – Les durées avant d'être filtrées par la fonction make-regular. La valeur de chaque durée est représentée en ordonnée

En supposant que l'objectif soit uniquement de trouver une valeur de pulsation, il n'est pas indispensable que seules les mesures soient marquées, les demi-mesures et les pulsations pouvant l'être aussi. De plus, il n'est pas toujours évident de faire ressortir un groupement en mesures, parfois même il n'existe pas de groupements, bien qu'une pulsation soit présente. Selon les objectifs (recherche de structure métrique comportant une pulsation régulière ou recherche de structure plus irrégulière), il faut donc pouvoir donner une signification aux segmentations qui ne limite pas les choix laissés à l'algorithme de quantification lors de la recherche d'une pulsation.

La version 2.0 de Kant permet dorénavant d'attribuer deux types aux segmentations (segmentation de mesure ou de pulsation).

### 4.2.2 Les paramètres musicaux pris en compte pour la segmentation

Si la quantification ne s'opère que sur les intervalles séparant deux éléments consécutifs (inter-onset), la segmentation se fait sur une séquence dont les paramètres sont tous ceux d'un fichier midi (hauteurs, onsets, durées, vélocités, channels). Ainsi, il est possible de prendre en compte les différents éléments qui contribuent à notre perception du rythme. Une option supplémentaire du logiciel permet de séparer les voix (en utilisant l'information du paramètre channel du fichier midi) de manière à simplifier les informations à prendre en compte.

Enfin, une fonction ("make-regular") permet d'opérer une première quantification sur les données de manière à segmenter une séquence plus régulière donc plus simple.

Description de la fonction :

Le principe de cette quantification est d'approximer les durées données en entrée par les valeurs d'une grille. Ainsi, deux durées initialement proches seront égales à la suite de la quantification (voir figures 4.1 et 4.2). La grille est formée d'une grille de pas variable couplée à une grille de pas constant. L'association des deux grilles permet de balayer tout l'espace des durées à quantifier (grille de pas constant), tout en adoptant une résolution plus fine au niveau des faibles valeurs (grille à pas variable linéairement). Au lieu de faire une simple approximation de chaque durée par la valeur de la grille la plus proche, l'algorithme forme des groupes centrés sur des valeurs de grille calculées au début de la quantification. Pour chaque valeur de grille, on définit un intervalle de tolérance dont la taille varie en proportion de la valeur de la grille. Toutes les durées comprises dans l'intervalle de tolérance feront partie du groupe. Certaines durées peuvent donc appartenir à plusieurs groupes. La tâche suivante sera donc de regrouper ou séparer les différents groupes, de manière à approximer toutes les durées d'un groupe par une même valeur. Pour cela, deux filtres sont utilisés. Le premier retient deux groupes ayant certaines durées en commun celui dont la moyenne de ses valeurs est la plus proche de sa valeur (la valeur caractéristique est la position sur la grille à partir de laquelle a été formé le groupe). Le second filtre rassemble en un seul groupe les groupes non filtrés et partageant certaines mêmes valeurs. au final, nous disposons donc de plusieurs groupe de valeurs, chaque groupe ayant une valeur caractéristique qui sera l'approximation des durées du groupe.

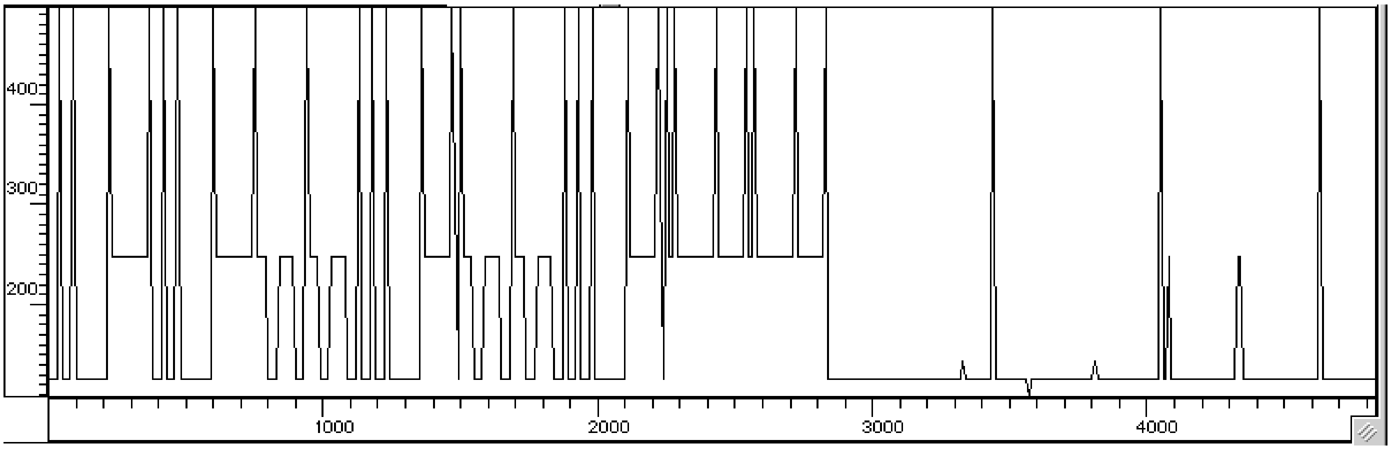


FIG. 4.2 – Les durées après avoir été filtrées par la fonction make-regular. La valeur de chaque durée est représentée en ordonnée

### 4.2.3 Le problème des tempi différents

Les segmentations sont utilisées par le quantificateur pour déterminer une valeur constante de pulsation. Lorsqu'une séquence présentant des fortes variations de tempo est à quantifier, il faut donc procéder par étapes, c'est à dire découper cette séquence en sous-séquences (archi-mesures) de tempo constant, qui seront chacune segmentées puis quantifiées séparément. Ces variations de tempo peuvent être repérées par une première segmentation.

### 4.2.4 Le problème de la levée

La levée est un aspect métrique de la partition. Il s'agit de déterminer quel est la position du mètre sur laquelle commence la première mesure. Une règle de traitement particulier des segmentations a alors été introduit : la première segmentation de la séquence détermine la position de début de la deuxième mesure, sauf si la segmentation est située sur le premier élément de la séquence.

### 4.2.5 La manipulation des segmentations

Lorsqu'on veut mettre en évidence (ou introduire) certaines structures complexes non régulières (accélérations, décélérations, déformations), il est possible de déplacer les segmentations à la main. Suivant les options choisies, les déplacements pourront être :
- constants pour toutes les barres de segmentations sélectionnées, ce qui revient à translater les segmentations
- proportionnels à une barre de segmentation de référence, ce qui revient en supposant que les barres de segmentations soient placées à pas réguliers, à augmenter ou à diminuer ce pas
- non linéaires, exprimés par l'exponentielle de leur distance par rapport à une barre de segmentation de référence, ce qui permet de mettre en évidence une accélération ou une décélération

Ces transformations, combinées les unes aux autres, peuvent donner lieu à des déformations assez complexes. Par exemple, la figure 4.3 montre une segmentation qui met en valeur un rétrécissement des mesures.

## 4.3 Les méthodes de segmentation

### 4.3.1 La segmentation à pas régulier

Cette méthode, relativement simple à implémenter, s'est révélée très utile pour segmenter des séquences qui ne comportaient presque pas de déformations par rapport à une séquence supposée parfaite (image midi d'une partition). Elle consiste à repérer les éléments de la séquence situés sur une valeur multiple d'une durée, en référence à une position temporelle dans la séquence. Un paramètre de l'algorithme est une marge d'erreur, ce qui permet aux éléments ne correspondant pas exactement à une valeur multiple d'être repérés.

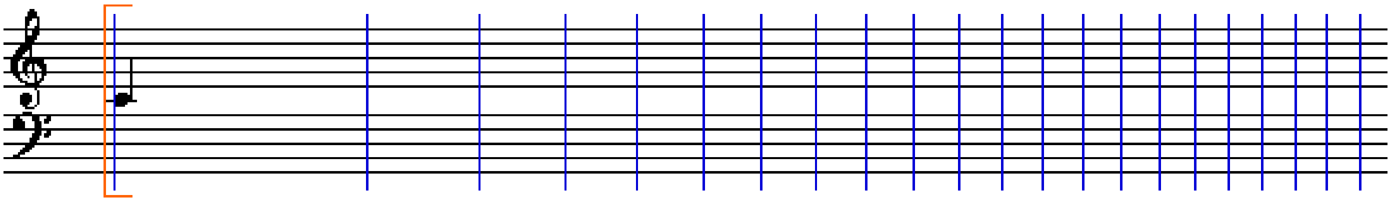

FIG. 4.3 – Exemple de segmentation dont les barres se rapprochent de plus en plus, faisant penser à une accélération.

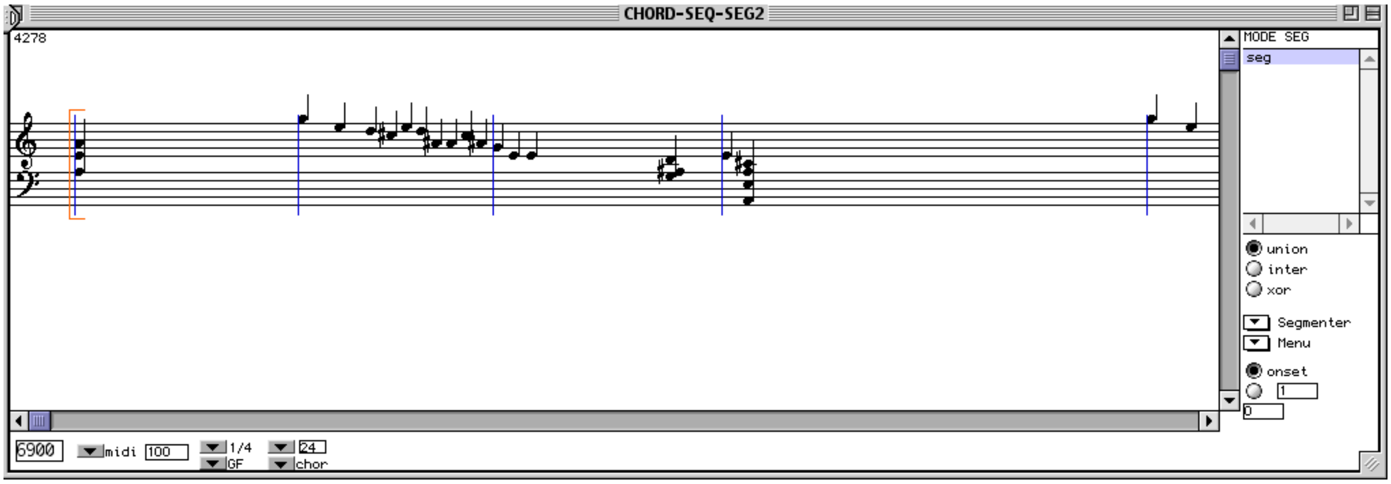


FIG. 4.4 – Exemple de segmentation à pas régulier

Par exemple, la figure 4.4 nous montre une séquence musicale segmentée de cette manière. Le pas proposé par l'algorithme pour la segmentation est la distance séparant les deux premières barres de segmentation (en supposant qu'elles aient été disposées par l'utilisateur au préalable). L'algorithme essaie alors de poursuivre cette segmentation en conservant le même pas. Si il ne trouve pas de note sur le premier multiple du pas, il cherche des notes sur le deuxième multiple puis le troisième et ainsi de suite. Par exemple, sur la figure 4.4, l'algorithme a placé deux barres de segmentation (traits verticaux) en plus des deux initialement disposées (les deux premières barres à gauche), puis il y a un vide (il n'y a plus d'éléments de la séquence), puis il place une barre sur le multiple suivant (barre verticale tout à droite de la figure). La marge d'erreur permet aux barres de ne pas tomber exactement sur un multiple du pas initial (on peut voir sur la figure que les distances séparant les barres sont légèrement différentes).

### 4.3.2 Les segmentations définies par l'utilisateur

Si l'on recherche des structures peu courantes, l'utilisateur peut définir ses propres méthodes de segmentation à l'aide des fonctions habituelles d'Open-Music. Cette option permet de personnaliser la quantificateur et de tester plusieurs variations d'une même méthode de segmentation avant de déterminer celle qui sera utilisée dans l'étape de quantification.

Pour cela, l'utilisateur définit un patch dans l'environnement Open-Music (figure 4.5), qu'il connecte à l'éditeur de quantification (boîte rectangulaire représentant la séquence de notes reliée par un trait à la nouvelle méthode "mypatch"). Dans le menu de l'éditeur de quantification apparait le nom de la nouvelle méthode : "user-method" (nom par défaut). Pour définir la nouvelle méthode, l'utilisateur crée une fonction par l'intermédiaire d'une abstraction de patch (sorte de lambda-fonction définie visuellement). La figure 4.6 montre l'intérieur du patch abstrait. D'autres fonctions (utilisant le principe de marquage) interviennent pour sélectionner les onset de la séquence musicale à quantifier.

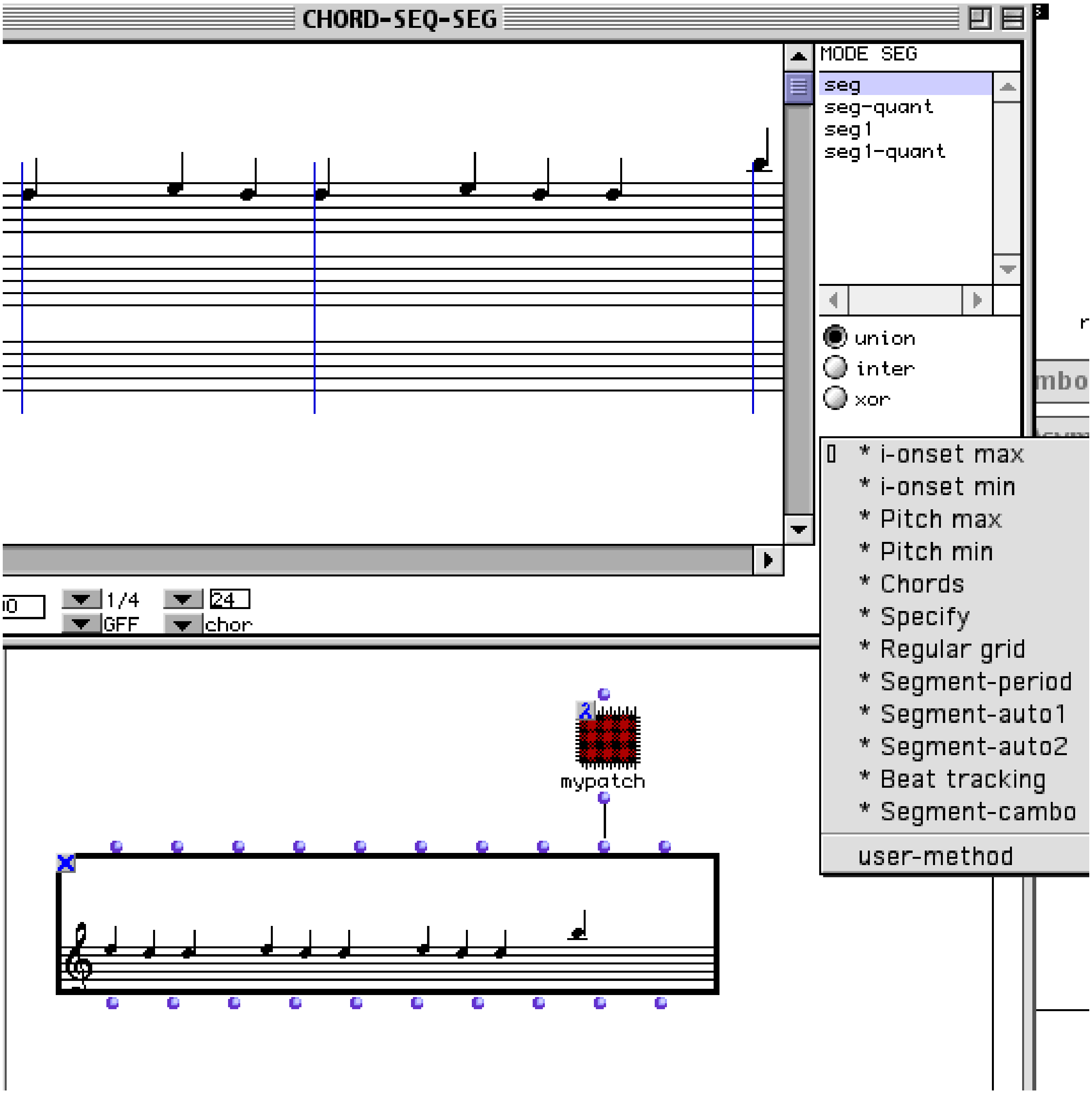


FIG. 4.5 – L'intégration de nouvelles méthodes de segmentation

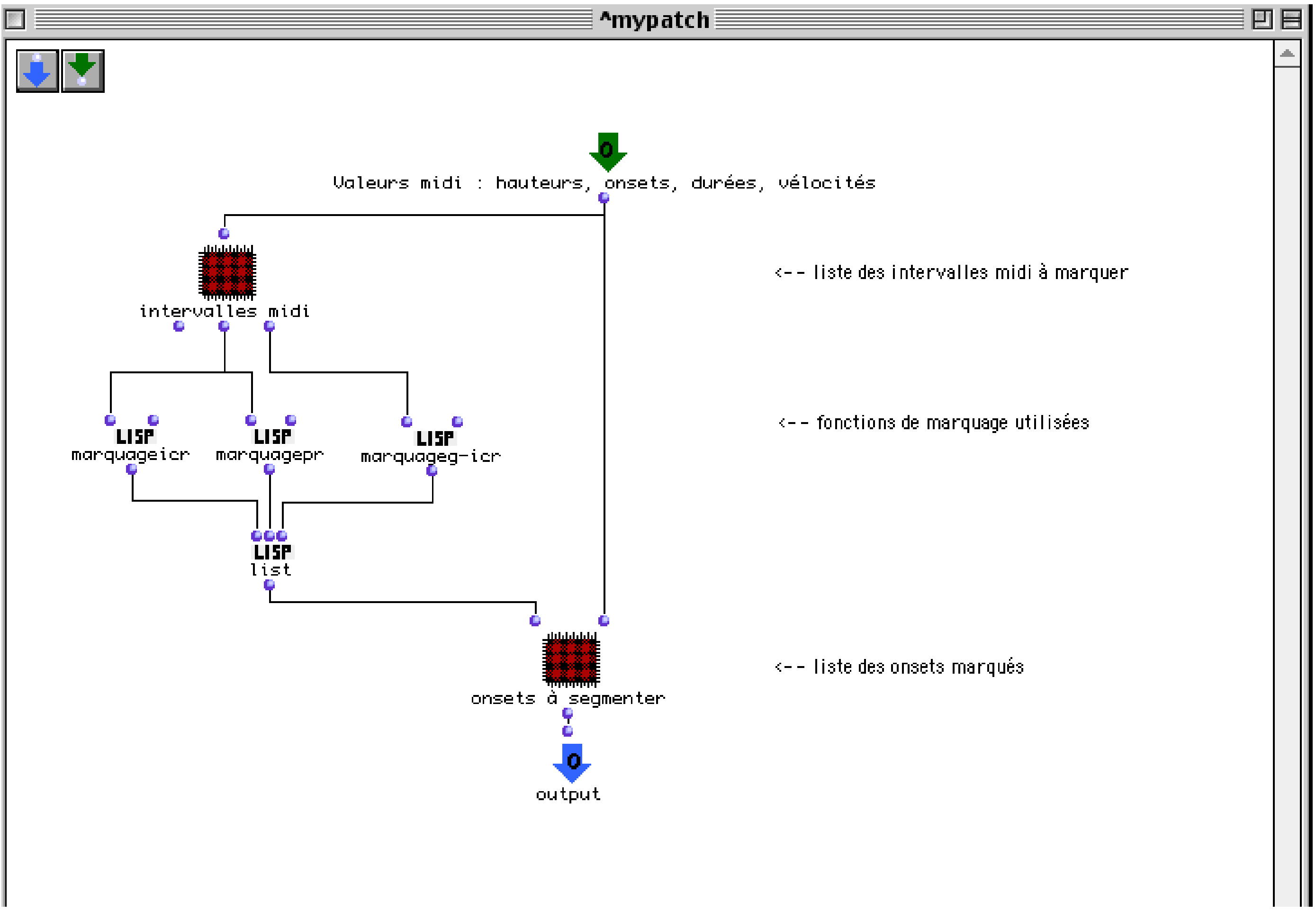


FIG. 4.6 – Un patch définissant une segmentation

### 4.3.3 La méthode du coefficient d'auto-corrélation

La méthode employée ici utilise le coefficient d'auto-corrélation pour déterminer une segmentation de la séquence en pulsations. L'objectif est donc de reconnaître une structure rythmique à pulsation constante. Les formules utilisées sont issues d'un article de Peter Desain [Desain Siebe]. Les calculs se font sur la suite d'inter-onset de la séquence.

Le coefficient :

> Le coefficient d'auto-corrélation permet de mettre en valeur des régularités. De façon générale, il s'exprime par la formule :
>
> $$A[m] = \sum_{n=0}^{N-1} x[n]x[n+m] \tag{4.1}$$

Etant donné une suite X(n), le principe est de comparer deux sous-suites extraites x[0..N] et x[m..m+N] de même longueur N, décalées de m éléments, afin de déterminer leur corrélation A[m]. Plus la valeur de A[m] sera grande et plus la corrélation entre les deux sous-suites est forte. Généralement, on se donne une sous-suite de référence x[0..N] que l'on compare à plusieurs autres sous-suites x[m..m+N] en faisant varier m afin de déterminer celles qui lui sont le plus corrélées. Les coefficients A[m] obtenus pour chaque valeur de m peuvent alors être visualisés dans un graphe que nous appellerons graphe de corrélation.

La méthode de segmentation peut se décomposer en quatre étapes :

- calcul des paramètres donnés à la fonction d'auto-corrélation
- calcul des coefficients d'auto-corrélation
- interprétation des pics de chaque graphe d'auto-corrélation
- extraction d'une courbe de tempo associée à la séquence et segmentation aux valeurs coïncidant avec une pulsation

L'automatisation des étapes de calcul a été déterminée à la suite de nombreux tests, de manière à séparer les informations propres à la séquence étudiée des informations liées au choix des paramètres de calcul. Le but serait de disposer d'une méthode entièrement automatique de reconnaissance de tempo. On suppose que la séquence comporte à tout moment une pulsation, qui peut éventuellement varier dans le cas par exemple d'un fichier midi issu d'une interprétation.

Les paramètres calculés lors de la première étape sont :

- l'unité dans laquelle on exprime les durées
- la valeur supposée de la pulsation de départ

Pour effectuer le calcul, les durées doivent en effet être disposées sur une grille temporelle dont le pas détermine la précision du calcul. Si le pas est trop fin (par exemple la milliseconde), la taille de la grille sera trop grande pour que les calculs se fassent rapidement (moins d'une minute). Inversement, si le pas est trop grand, les calculs seront faussés par l'imprécision. L'unité qui a été choisie est le quart de la plus petite durée présente dans la séquence. On choisit comme unité 10 millisecondes est trop petite. La valeur supposée de pulsation de départ qui est choisie est la durée la plus présente dans la séquence. Comme certaines durées peuvent être très proches l'une de l'autre sans être égales, une première quantification est effectuée à l'aide de la fonction "make-regular" décrite dans la section précédente.

Les deuxièmes, troisièmes et quatrièmes étapes constituent une boucle qui est répétée sur toute la séquence. A chaque itération, une fenêtre de taille proportionnelle à la valeur de pulsation attendue est disposée sur la séquence de manière à calculer une nouvelle valeur de pulsation à l'aide des éléments sélectionnés.

Les coefficients d'auto-corrélation sont alors calculés.

Pour chaque graphe d'auto-corrélation trouvé, les pics localement maximaux de taille supérieure à tous les pics les précédant sont sélectionnés. Une liste de pulsations possibles est donc extraite. Souvent, les pulsations sont multiples les unes des autres et représentent différents niveaux rythmiques de la séquence, mais cette interprétation est difficile à automatiser à cause de certaines exceptions.

Par exemple, la figure 4.7 montre un graphe d'auto-corrélation. Ici, seul le deuxième pic sera sélectionné car les pics suivants lui sont tous inférieurs.

Enfin, on recherche parmi la liste des pulsations celle qui est la plus probable en fonction de la dernière pulsation trouvée. Cette méthode permet de conserver des valeurs à peu près égales de pulsation plutôt que de passer brutalement d'une pulsation à sa valeur moitié. Une valeur de tolérance permet de considérer une pulsation comme continuation possible de l'ancienne même si elles présentent un écart. Une accélération ou décélération progressive peut ainsi être mise en valeur.

On réitère ensuite ces trois étapes sur une nouvelle fenêtre de la séquence séparée de l'ancienne par la valeur de pulsation trouvée.

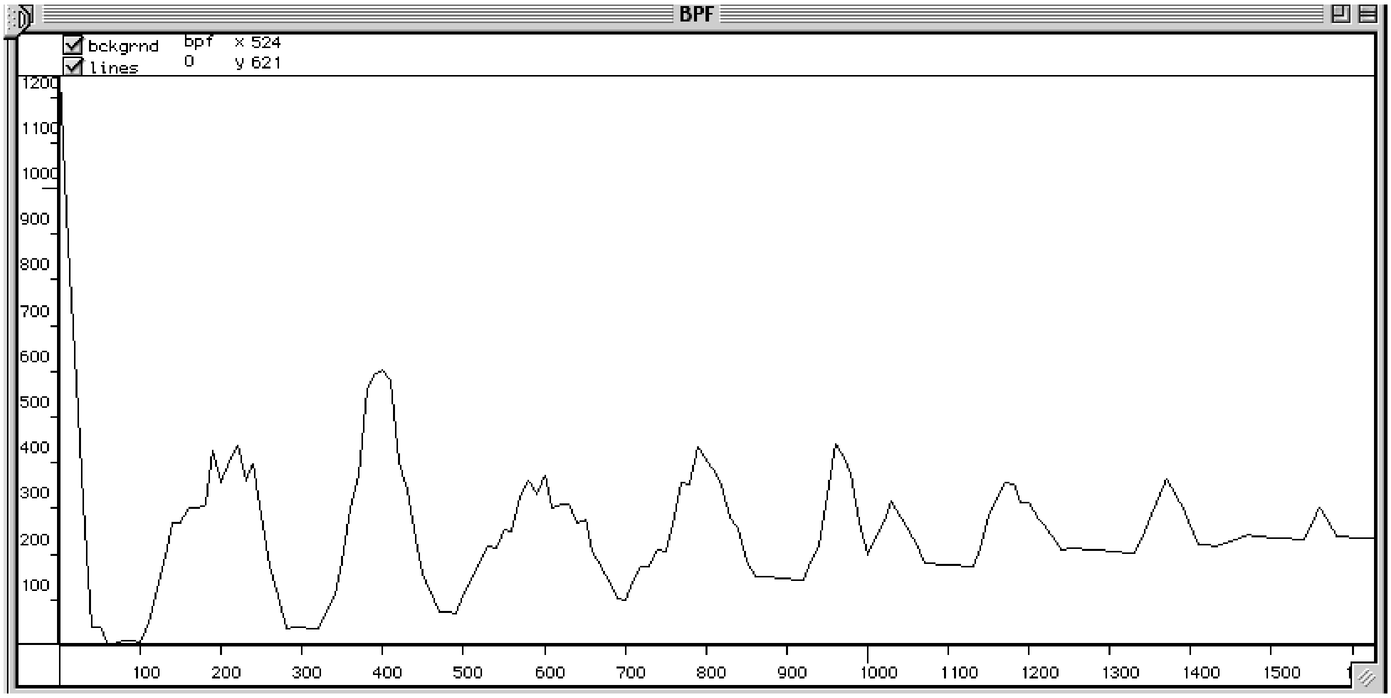


FIG. 4.7 – Un graphe d'auto-corrélation

### 4.3.4 La segmentation par marquages

Cette méthode de segmentation est issue de [Cambouropoulos 98]. Elle a pour objectif de repérer les accents rythmiques de la séquence. Pour cela, un procédé utilisant la notion de marquage est utilisé. La notion de marquage est présente dans plusieurs théories, notamment dans la théorie du rythme de Pierre Lusson [LUSSON 86], dont on parlera en dernière partie. Le principe général est de considérer la séquence à segmenter comme un ensemble d'éléments (pour nous, ce seront les intervalles entre les notes ou les accords successifs du fichier midi). Chacun des éléments répond à plusieurs propriétés (pour nous les intervalles de hauteurs, d'onsets, de durées et d'intensités). Un poids (marquage) est ensuite attribué à ces éléments suivant que certaines de leurs propriétés répondent ou non à un critère (pour nous, par exemple, les intervalles suivis d'un intervalle plus petit seront marqués). Il est possible de faire plusieurs marquages, puis de les combiner (en faisant la somme des poids) de manière à obtenir une courbe de poids (un poids par élément) représentative de la séquence pour les marquages utilisés. Eventuellement, chaque marquage peut être pondéré de manière à ce qu'il ait plus d'importance dans le résultat final.

Plusieurs essais ont été effectués. Certains sont concluants, d'autres mettent en évidence des accents placés aux mauvais endroits. (voir les exemples de la section 4.5)

De plus, un problème de cette théorie est qu'il est nécessaire d'analyser les hauteurs des musiques tonales selon la place qu'elles occupent dans la gamme et non dans l'échelle chromatique (par exemple, en do majeur, l'intervalle mi-fa (1/2 ton) et l'intervalle fa-sol (1 ton) doit être analysés de la même façon).

## 4.4 La phase de quantification

La quantification est locale (elle se fait pour chaque pulsation), car les compressions-dilatations effectuées précédemment ont placé les barres de segmentation sur les temps d'occurrence des pulsations. Les segmentations ne seront donc pas modifiées par la quantification. Cette étape recherche le rythme d'une sous-séquence dont la taille ne peut pas varier et dont la longueur est la durée d'une pulsation.

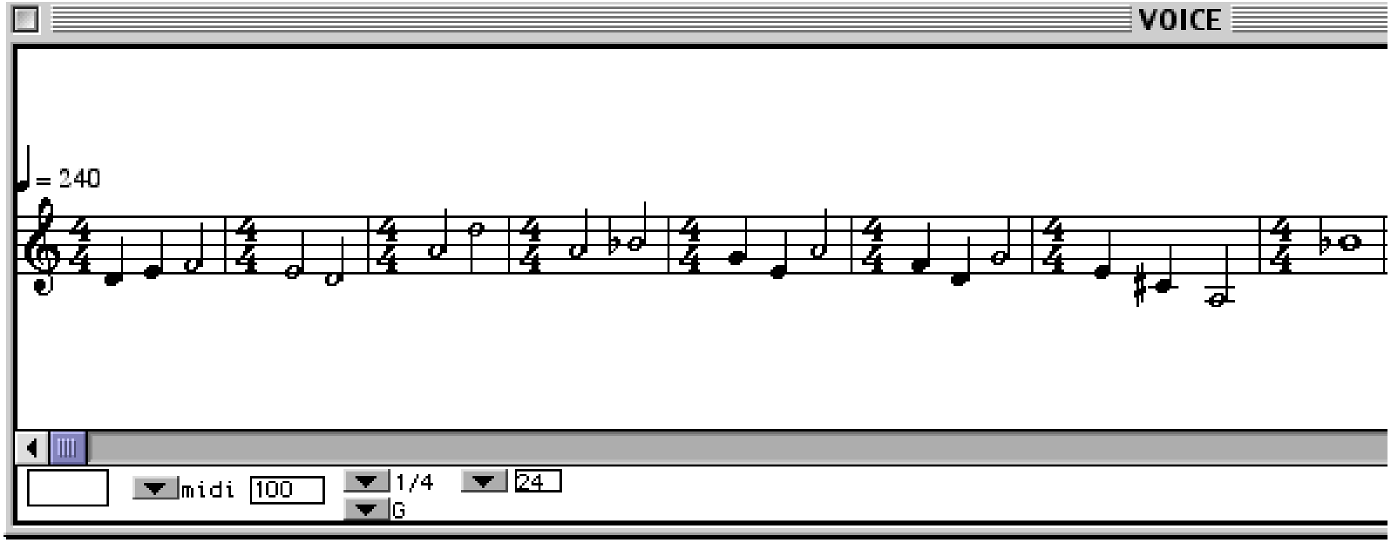


FIG. 4.8 – Un rythme peu élégant

### 4.4.1 Le contrôle au niveau de la pulsation

Le seul contrôle possible se fait par l'intermédiaire des entrées du module quantify, qui permet d'interdire ou de forcer certaines subdivisions rythmiques. La version 2.0 permet de spécifier ces contraintes au niveau de la segmentation (contraintes appliquées à toute la séquence), de la mesure (contraintes appliquées à toutes les pulsations) et de la pulsation.

Enfin, à l'issue de la quantification, il est possible d'adopter une notation à la croche, à la noire ou à la blanche, de manière à simplifier la lecture de la partition finale. Ainsi, la figure 4.9 peut être préférée à la figure 4.8.

## 4.5 Exemples de quantifications dans l'environnement Open-Music et description de fonctions Lisp

Aujourd'hui, Kant est implémenté sur OpenMusic, successeur de Patchwork. Open-Music est un environnement visuel de programmation musicale, destiné aux compositeurs [Agon 98]. Il est écrit en CLOS (Common Lisp Object System, extension objet de CommonLisp) qui est un puissant langage fonctionnel et objet. Il a pour but l'intégration de divers modèles de programmation dans un même langage :

- le modèle fonctionnel
- le modèle objet
- le modèle par contraintes
- le modèle visuel

Ses principales caractéristiques sont :

- la réflexivité (il contient une représentation de lui-même).
- la méta-programmation, qui permet de considérer les classes comme objets à part entière, de façon à pouvoir définir de nouveaux " types " de classes.
- une gestion de la dualité entre le temps de calcul d'un objet musical et le temps d'exécution d'une séquence musicale.
- une interface présentant à l'utilisateur des objets musicaux prédéfinis permettant de gérer à la fois le son, les paramètres midi et la notation musicale.

La librairie Kant consiste en un ensemble de fonctions écrites en Clos définissant les opérations de segmentation et de quantification pouvant être effectuées sur une séquence musicale au format midi. Ces fonctions sont manipulables par l'intermédiaire d'un éditeur graphique, lui-même écrit en Clos. Cet éditeur est une classe (chord-seq-seg) héritant de la classe de l'éditeur graphique de séquences midi (l'éditeur chord-seq).

La librairie Kant est donc représentée graphiquement par l'éditeur 'chord-seq-seg.

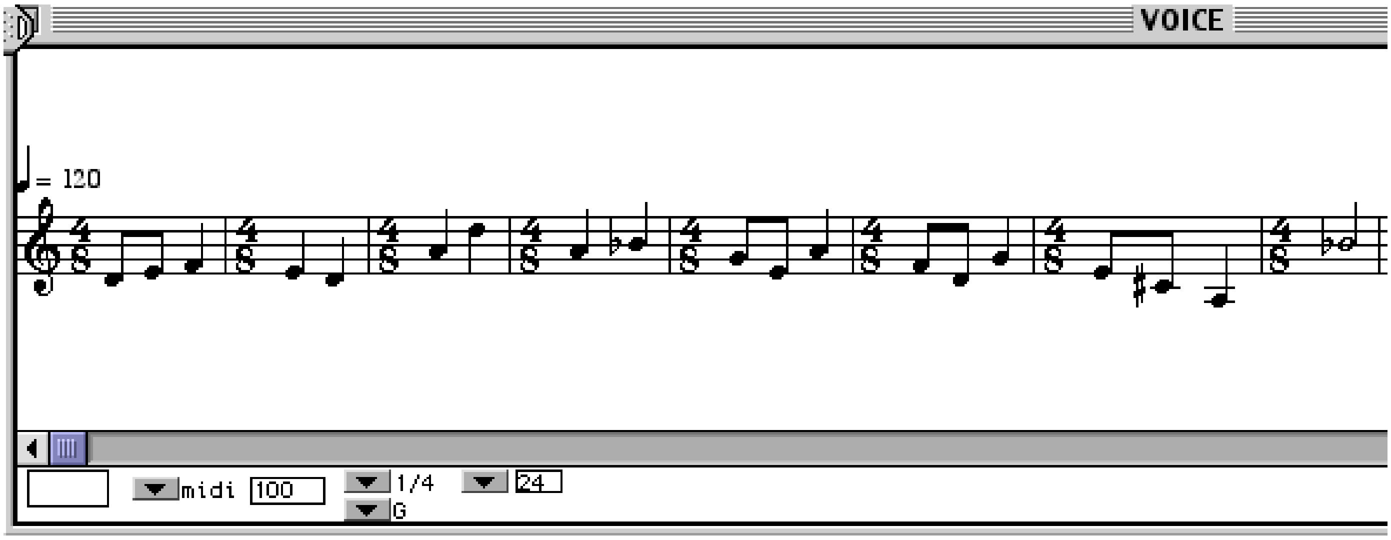


FIG. 4.9 – Un rythme plus satisfaisant

Ce chapitre présente une description de l'éditeur graphique chord-seq-seg en remplacement de celle que j'ai développé pendant l'année 1999. Les modifications et ajouts que j'ai effectué pendant mon stage de Dea seront marqués en gras, et une brève présentation des nouvelles fonctions Lisp sera effectuée.

### 4.5.1 Exemples de segmentations et quantifications

**exemple de quantification satisfaisante effectuée avec la méthode d'auto-corrélation**

La figure 4.10 montre une segmentation dont le pas (correspondant à une pulsation) a été déterminé par la méthode du coefficient d'auto-corrélation. Seule la voix du haut (définie par le numéro de channel 1 du fichier midi source) a été prise en compte. Une fois le pas déterminé, la méthode de segmentation a pas régulier décrite en 4.3.1 a été utilisée. Les segmentations correpondent plus à des valeurs de pulsation qu'à des barres de mesure, mais l'objectif ici est de déterminer la pulsation. La quantification résultante (mise en place des barres de segmentation à chaque valeur de pulsation) est présentée en figure 4.11. La notation finale est présentée dans la figure 4.12. Les barres de mesure n'ont pas une grande signification puisqu'elles marquent la pulsation et non les groupements rythmiques. Cependant, on peut observer que la séquence est segmentée correctement.

**exemple de quantification peu satisfaisante effectuée à l'aide de marquages**

La figure 4.13 montre une segmentation de la même séquence que précédemment, mais cette fois-ci les barres de segmentation sont disposées suivant la position des accents déterminés à l'aide des marquages définis par Emiliios Cambouropoulos. Seule la voix du haut (définie par le numéro de channel 1 du fichier midi source) a été prise en compte. La quantification de cette séquence puis la représentation en partition sont assez peu satisfaisantes (figures 4.14 et 4.15). En effet, les accents mis en valeur par les marquages ne sont pas disposés aux endroits attendus. Cela provient de la configuration de la séquence qui présente des écarts d'intervalles localement maximaux aux positions non accentués (ce qui est contraire aux résultats attendus par Cambouropoulos).

**exemple de quantification satisfaisante effectuée à l'aide de marquages**

La même méthode de segmentation (figure 4.16) par marquage est appliquée à toute la séquence (et non plus seulement à la seule voix du haut). Les résultats (figures 4.17 et 4.18) sont plus satisfaisants. L'interprétation de ces différences avec l'exemple précédant reste difficile.

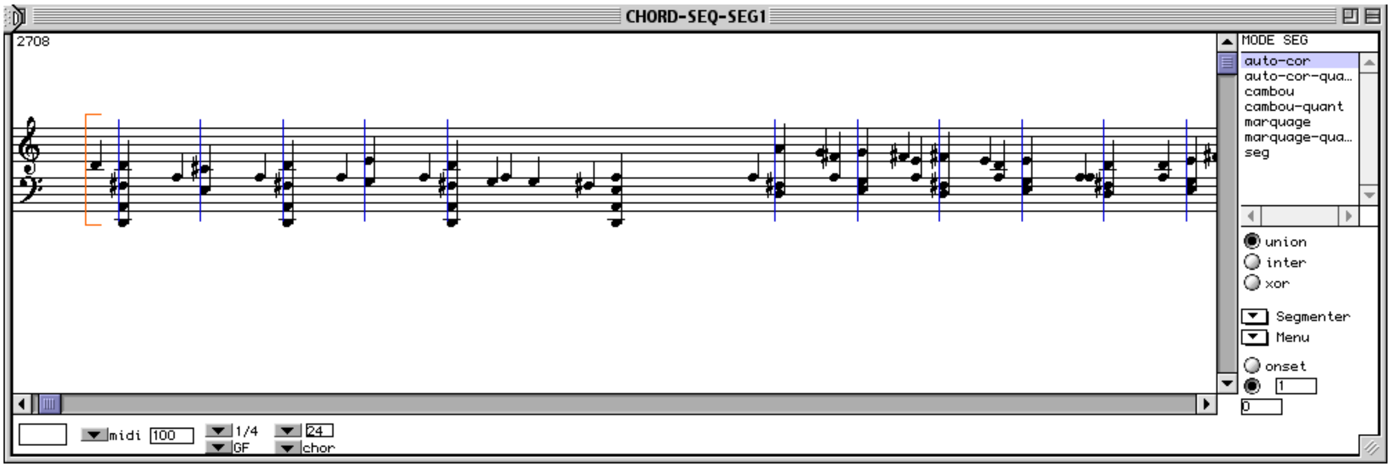


FIG. 4.10 – La segmentation effectuée à l'aide du coefficient d'auto-corrélation

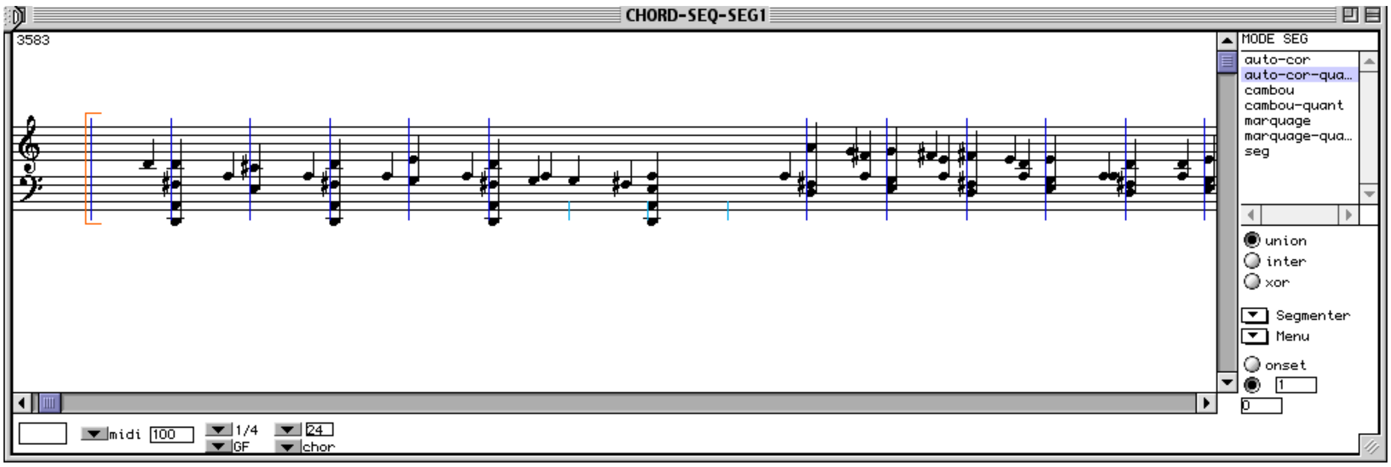


FIG. 4.11 – La quantification effectuée à l'aide du coefficient d'auto-corrélation

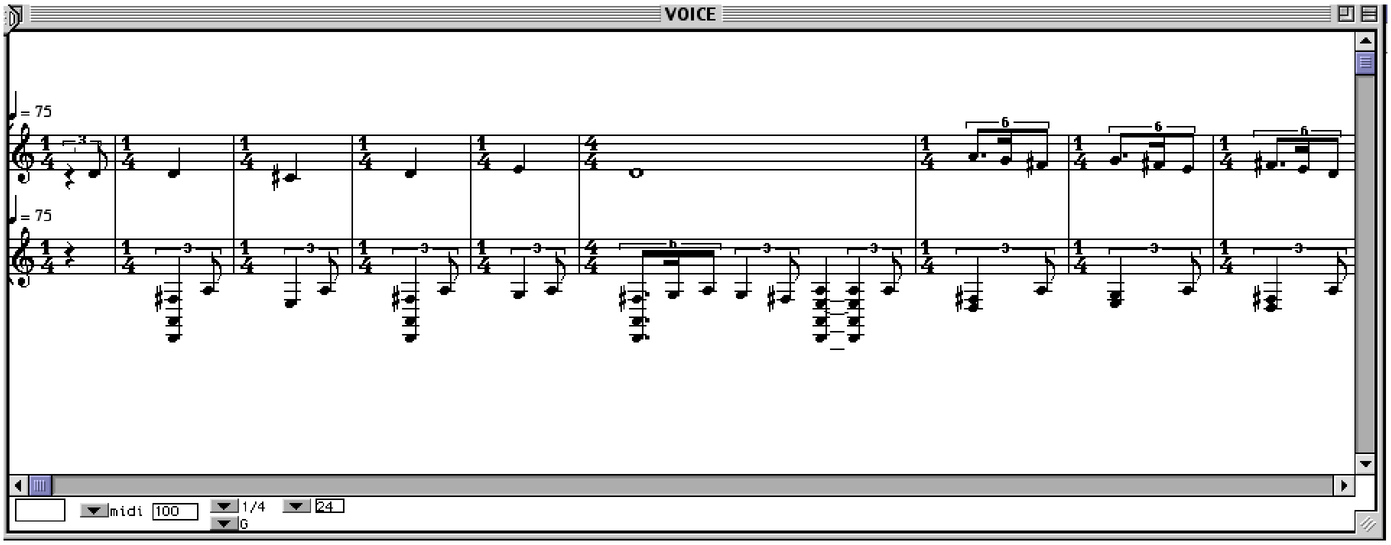


FIG. 4.12 – La partition écrite à l'aide du coefficient d'auto-corrélation

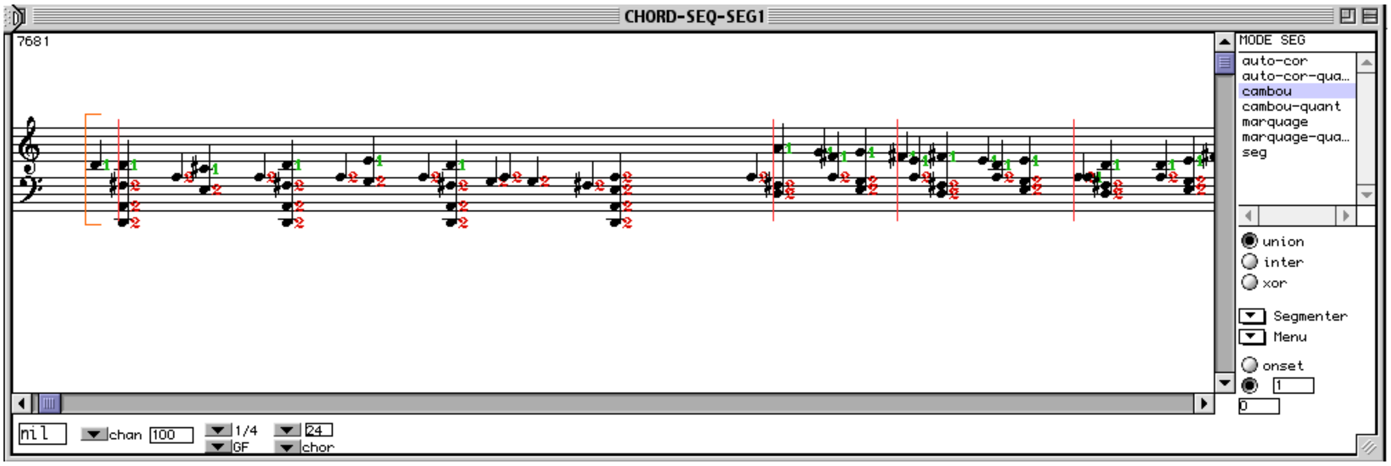


FIG. 4.13 – La segmentation peu satisfaisante effectuée à l'aide de marquages

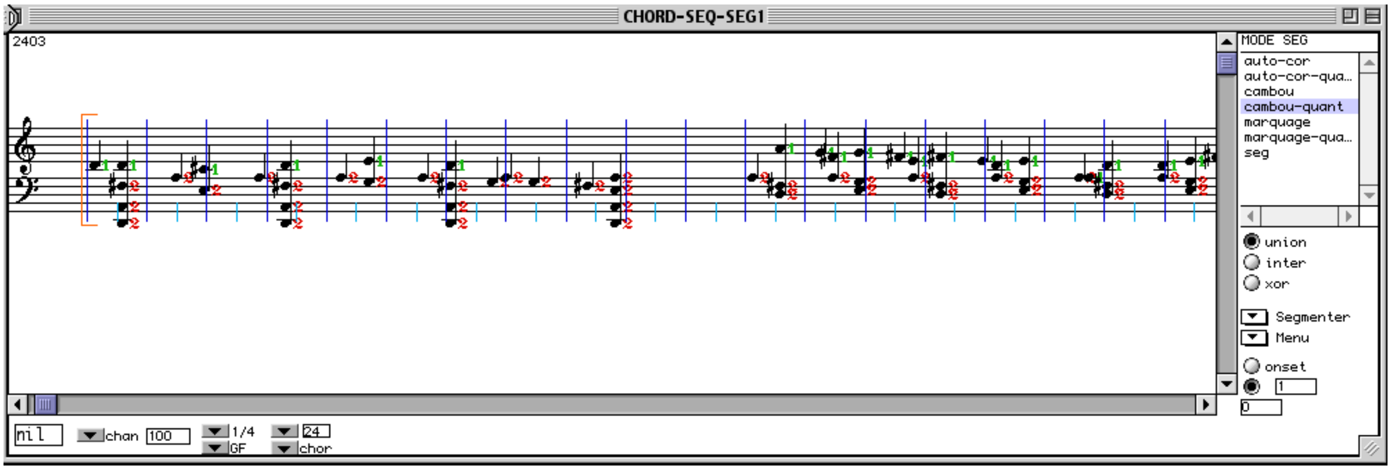


FIG. 4.14 – La quantification peu satisfaisante effectuée à l'aide de marquages

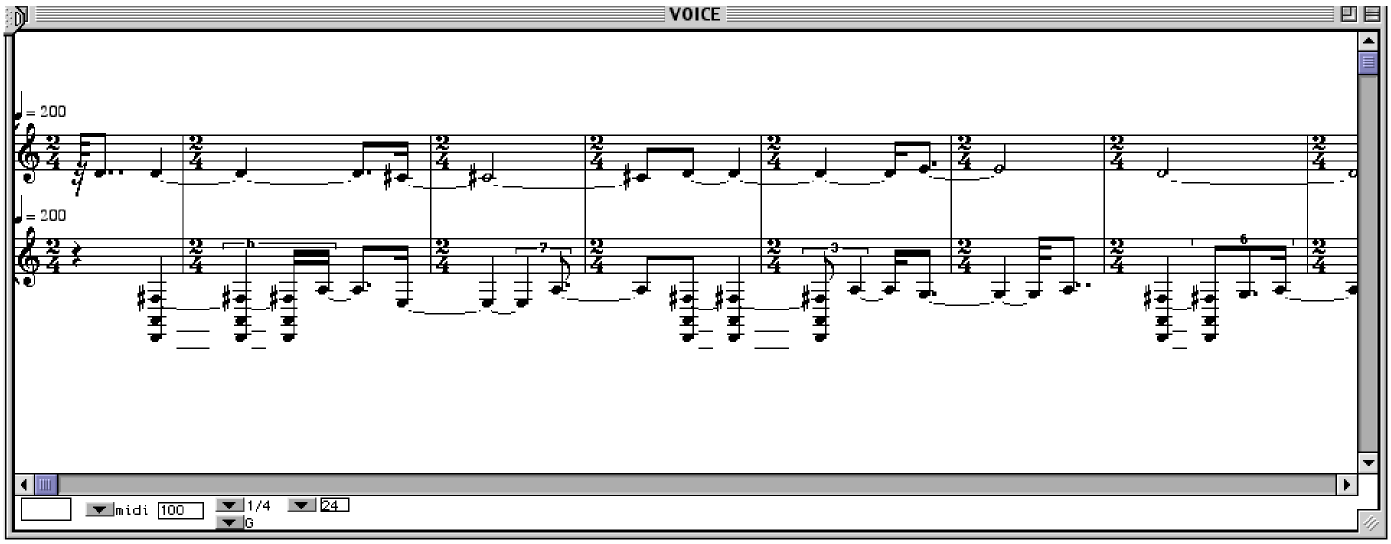


FIG. 4.15 – La partition peu satisfaisante écrite à l'aide de marquages

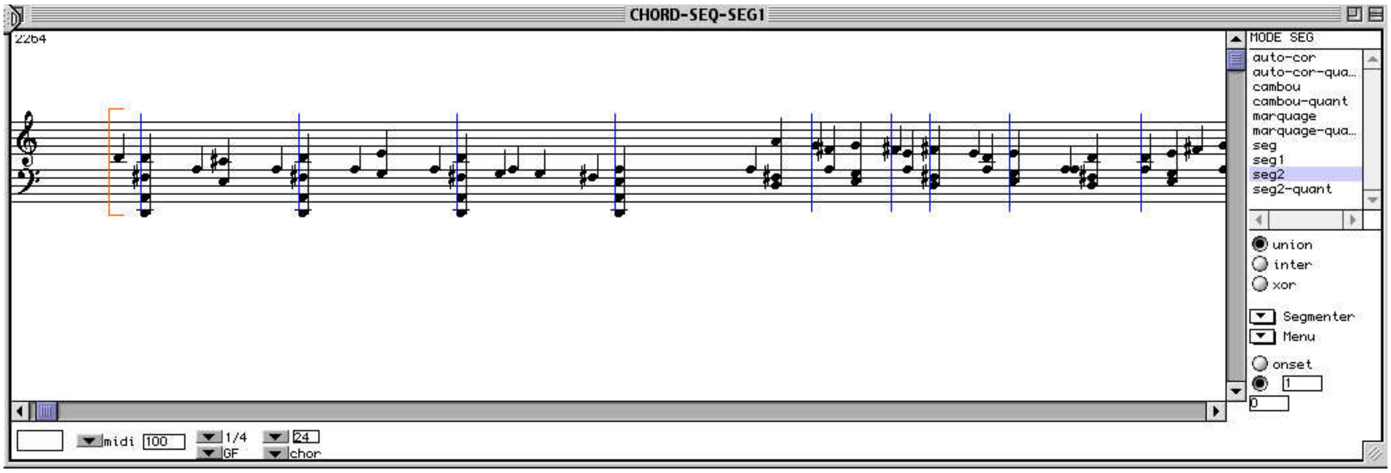


FIG. 4.16 – La segmentation satisfaisante effectuée à l'aide des marquages

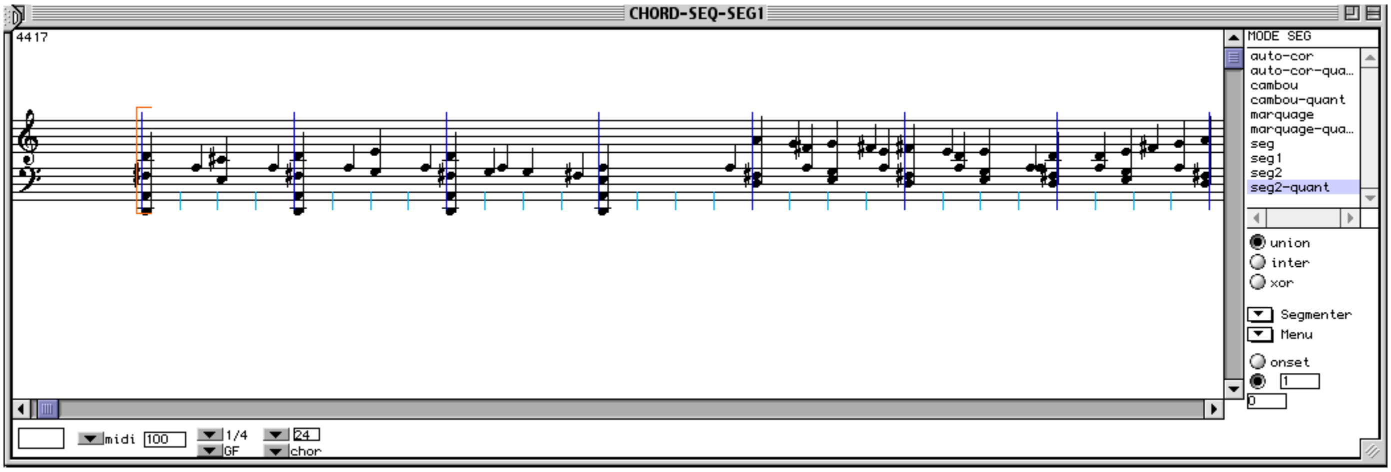


FIG. 4.17 – La quantification satisfaisante effectuée à l'aide des marquages

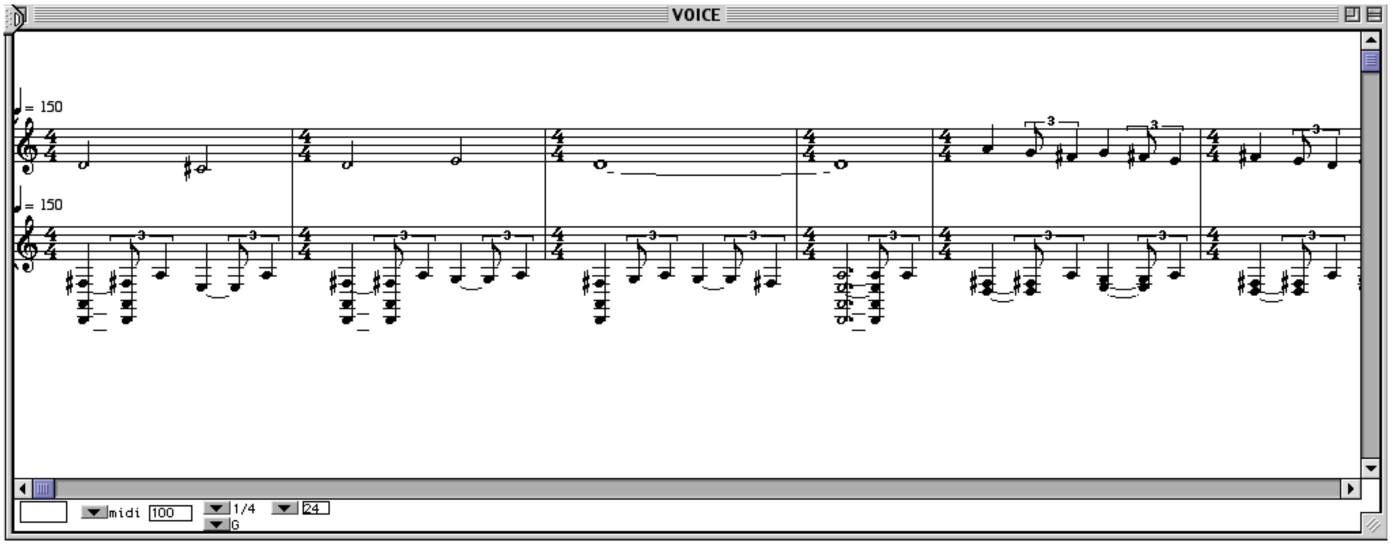


FIG. 4.18 – La partition satisfaisante écrite à l'aide de marquages

**exemple de quantification où plusieurs interprétations de la courbe de tempo sont possibles**

Une méthode de recherche de courbe de tempo est ici utilisée (figure 4.19) pour segmenter une sonate facile de Mozart enregistrée sur un synthétiseur Yamaha (donc accessible directement en fichier midi). L'algorithme d'extraction de pulsation est décrit à travers cet exemple. (l'algorithme correspond à la troisième étape de la méthode de segmentation par le coefficient d'auto-corrélation décrite en section 4.3.3.)

Les résultats issus de l'interprétation des graphes d'auto-corrélation sont les suivants (ils sont exprimés en millisecondes) :

((190) (60 190) (140 240 440) (160 440) (190 410) (220 400))

Chaque sous-liste correspond à une ou plusieurs pulsation(s) supposée(s). Dans notre exemple, la première pulsation supposée est 190 millisecondes, puis après avoir avancé sa fenêtre de 190 millisecondes dans la séquence, l'algorithme trouve deux pulsations possibles : 60 et 190 millisecondes. Il choisit alors la plus proche de la précédente, donc ici 190 millisecondes, puis avance de nouveau sa fenêtre de 190 millisecondes. Cette fois-ci, trois pulsations sont possibles : 140, 240 et 440 millisecondes. Comme aucune des trois n'est proche de 190, l'algorithme choisit la plus grande, donc 440 millisecondes. ensuite, l'algorithme se stabilise sur cette valeur : 440, 410 puis 400 millisecondes. Cette valeur de pulsation est correcte.

La figure 4.19 montre la représentation de ces valeurs de pulsation (par les notes des aigus) dans la séquence enregistrée.

### 4.5.2 Description de quelques fonctions Lisp

**La fonction 'remove-lie'**

Cette fonction élimine les liaisons rythmiques entre deux éléments d'un arbre (arbre rythmique) situés sur une même branche.

Par exemple, l'arbre ( ? ((4//4 (1 (1 (1 2.0 1 1.0)) 1 1)))) comporte deux liaisons 2.0 et 1.0 qui sont rattachées par l'algorithme à la note précédente. On obtient finalement l'arbre : ( ? ((4//4 (1 (1 (3 2)) 1 1))

```
(defmethod! remove-lie ((liste list))
   :indoc '("tree")
   :icon 137
   :initvals '((? ((4//4 (1 (1 (1 -2 1 1)) 1 1)) (4//4 (1 (1 (1 2 1 1)) -1 1)))))
   :doc "renvoie le tree avec les liaisons présentées de facon plus conviviale"
   (list (car liste)
         (remove-lie1 (cadr liste))))

(defun remove-lie1 (liste)
```

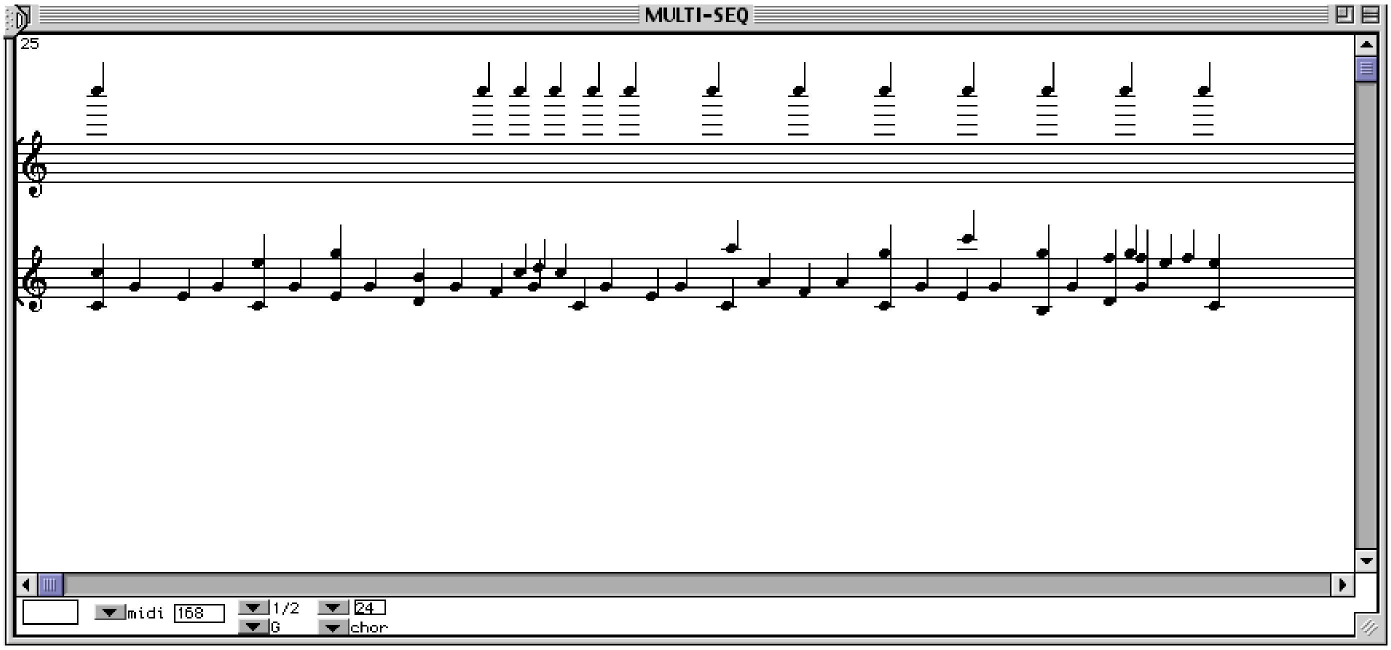


FIG. 4.19 – Le suivi de pulsation pour une sonate facile de Mozart

```
(cond ((null liste) liste)
      ((and (numberp (car liste))
            (numberp (cadr liste))
            (not (integerp (cadr liste))))
       (remove-lie1 (cons (+ (car liste) (abs (round (cadr liste))))
                          (cddr liste))))
      ((listp (car liste))
       (cons (cons (caar liste) (list (remove-lie1 (cadar liste))))
             (remove-lie1 (cdr liste))))
      (t (cons (car liste) (remove-lie1 (cdr liste))))))
```

**La fonction 'method-seg'**

Cette fonction fait le lien entre une méthode de segmentation et la position finale des barres de segmentation dans la séquence.

En effet, toutes les méthodes de segmentation (les méthodes internes à l'éditeur comme les méthodes définies par les utilisateur par l'intermédiaire des lambda-patch (voir section 4.3.2)) sont définies par une lambda-expression. Quelque soit la méthode de segmentation, elle est appliquée à la séquence par la fonction 'method-seg'.

Le moteur de segmentation ne connaît donc pas vraiment la fonction de segmentation, car il l'applique sans en connaître son contenu. Certaines procédures doivent alors être suivies.

Tout d'abord, les méthodes de segmentation peuvent être associée dans une liste à un nom (une chaîne de caractères) qui leur sera attribué dans le menu de l'éditeur de segmentation. La fonction 'méthod-seg' doit donc reconnaître la nature de ses arguments (ce peut être : une liste de méthodes de segmentation, la liste d'une méthode et de son nom, une fonction (la méthode seule), ou toute combinaison de ces types d'arguments.

Enfin, à son appel, la fonction 'method-seg' ne connait pas encore la valeur des arguments des méthodes de segmentation à appliquer. En effet, suivant le contexte, les méthodes doivent pouvoir avoir des arguments différents, soit rentrés par l'utilisateur, soit calculés automatiquement sur le contexte de la séquence à segmenter. La solution choisie est de dire que lorsqu'une méthode de segmentation a des paramètres, une fonction de même nom que ces paramètres est crée pour chacun d'entre eux. Lors de l'appel de la méthode de segmentation, 'method-seg' appliquera les fonctions-paramètres pour calculer leur valeur suivant le contexte.

Ensuite, 'method-seg' présentera une fenêtre de dialogue dans laquelle l'utilisateur pourra éventuellement modifier la nouvelle valeur du paramètre. Enfin, connaissant la valeur de ses paramètres, la méthode de segmentation sera appliquée.

```
(defmethod method-seg ((self omcss-seg) abstract)
   (let* ((obj (object (view-container self)))
          (l-identifiant (l-identifiant-seg
                           (l-obj-seg-nom
                              (selection-item (list1-seg
                                                 (seg-view (view-container self))))
                                              (Lsegmentation obj))))
          (seg nil)
          (new-val nil)
          (func (if (listp abstract)
                    (if (functionp (car abstract)) (car abstract) (cadr abstract))
                    abstract))
          (nom (if (listp abstract)
                   (if (functionp (car abstract)) (cadr abstract) (car abstract))
                   "")))
     (unless (or (not (equal (length l-identifiant) 1))
                 (not (null (tempo (car (Lsegmentation-panel obj))))))
       (if (> (length (arglist func)) 1)
         (let* ((arg (cdr (arglist func)))
                param)
           (mapcar #'(lambda(a)
                       (if (or (functionp a) (and (symbolp a)
                                                  (fboundp a)
                                                  (symbol-function a)))
                         (set a (apply a (list (make-chord-seq-seg-area
                                                 (view-container self)))))
                         (set a "")))
                   arg)
           (setq param (dialog-method nom arg))
           (unless (or (equal ':closed param) (null param))
             (setq new-val (filter-atom (apply func (cons (make-chord-seq-seg-area
                                                          (view-container self))
                                                         param)) '(nil))))))
         (setq new-val (apply func (list (make-chord-seq-seg-area
                                          (view-container self))))))
       (unless (null (remove 'nil new-val))
         (setq seg (obj-seg-identifiant (car l-identifiant)
                                        (Lsegmentation-panel obj)))
         (if (listp (car new-val))
           (progn (setf (lpoids seg) (cadr new-val)) (setq new-val (car new-val))))
         (setf (Lseparation seg)
               (classer-sep-not-equal
                (append (Lseparation seg)
                        (mapcar #'(lambda(a) (make-obj-sep a seg (color seg)))
                                (valeur->unite obj new-val)))))
         (setf (Lsegmentation-panel obj)
               (cons seg (del-seg (Lsegmentation-panel obj) (car l-identifiant))))
         (action-save-seg (view-container self))
         (update-panel (panel (view-container self)))))))
```

## 4.6 Critique et solutions à court terme

Le principe de segmentation s'est montré très utile pour intégrer des méthodes pourtant très différentes (méthodes de marquage, méthodes basées sur le coefficient d'auto-corrélation). De ce point de vue, la librairie Kant constitue un environnement très ouvert à l'expérimentation.

Certains résultats peuvent être faussés du fait que le module 'quantify ne répond pas toujours aux attentes que l'on peut formuler. En particulier, les contraintes rythmiques (interdiction ou imposition de certains rythmes au niveau de la pulsation) ne sont pas toujours respectées. De plus, certains décalages progressifs de rythmes aboutissent à une notation complètement absurde par rapport aux résultats attendus. Ce problème provient des approximation effectuées lors de la conversion de la pulsation en tempo (nombre de pulsations à la seconde). Pour éviter ces décalages, seules certaines pulsations devraient être choisies : les diviseurs de 60000, soit (1 2 3 4 5 6 8 10 12 15 16 20 24 25 30 32 40 48 50 60 75 80 96 100 120 125 150 160 200 240 250 300 375 400 480 500 600 625 750 800 1000 1200 1250 1500 1875 2000 2400 2500 3000 3750 4000 5000 6000 7500 10000 12000 15000 20000 30000). Cet ensemble de pulsations est trop restreint, il faudrait envisager d'autres solutions.

Par ailleurs, il serait assez peu coûteux de développer un environnement de manipulation de marquages, suffisamment ouvert pour que l'utilisateur puisse y intégrer ses propres méthodes. A cet environnement pourraient s'ajouter des fonctions spécifiques de manipulation des lignes de poids issues des marquages, de manière à disposer d'informations suffisamment riches pour en extraire des profils rythmiques. Ces fonctions, ainsi qu'un vocabulaire, ont été définies par Pierre Lusson [LUSSON 86] dans la théorie du rythme.

## Quatrième partie

# Travaux futurs - Perspectives

Parmi les différentes approches que nous avons vu, certaines n'ont pas été introduites dans la librairie Kant. Il s'agit des modèles basés sur les oscillateurs. Il faudrait étudier si l'environnement Open-Music peut permettre l'implémentation de ces méthodes, qui pourraient alors compléter les approches déjà introduites (marquages, coefficient d'auto-corrélation), pour constituer un modèle rythmique qui résiste au plus grand nombre de cas de figures possible.

D'autres approches seraient envisageables. Par exemple, nous pourrions nous demander si les récentes avancées des techniques de la linguistique, et en particulier l'analyse paradigmatique ([Ruwet 66]), ne pourraient pas, par l'analyse des relations entre les unités musicales, fournir des informations à partir desquelles on dégagerait un modèle du rythme. Un premier problème est que l'approche de Ruwet ne propose pas de théorie computationnelle. On pourrait alors se pencher sur des théories voisines [Wolff 98].

# Annexe A

# Bibliographie


[Agon 98] A. Agon.
" OpenMusic : Un langage visuel pour la composition musicale assistée par ordinateur "
Thèse, IRCAM-Paris VI, 1998.

[Agon 94] Agon, A.
" KANT : Une critique de la quantification pure. "
Mémoire de DEA Informatique, Université de Paris XI, 1994.

[AAFR 94] Agon C., Assayag G., FinebergJ.,Rueda C.
" Kant : a Critique of Pure Quantification "
ICMC 94. Aarhus, 1994.

[Brown 1993] Judith C Brown
" Determination of the meter of musical scores by autocorrelation "
Journal of the Acoustical Society of America 94 (4) 1953-1957 October 1993

[BrownPuckette 89]
" Calculation of a narrowed autocorrelation fonction "
Journal of the Acoustical Society of America 85 1595-1601

[Cambouropoulos 98] Cambouropoulos E.
" Towards a General Computational Theory of Musical Structure "
The university of Edinburgh, Faculty of Music and Department of Artificial Intelligence, 1998.

[Cooper Meyer 60]
" The rhytmic structure of music "
Chicago : university of Chicago Press.

[Dannenberg 87] Dannenberg R B and Mont-Reynaud B
" Following an improvisation in real time "
Proceedings of the 1987 International Computer Music Conference. Computer Music Association.

[Desain - Siebe de Vos] Peter Desain, Siebe de Vos .
" Auto-correlation and the study of Musical Expression "

[Desain Honing 89] Desain, P., and Honing, H. (1989)
" Quantization of musical time : a connectionist approach. "
Computer Music Journal 13(3)

[Desain Honing 92] Desain, P., and Honing, H. (1992)
" The quantization problem : traditional and connectionist approaches. "
In M. Balaban, K. Ebcioglu, and O. Laske (eds.), Understanding Music with AI : Perspectives on Music Cognition.
448-463. Cambridge : MIT Press.

[Desain 93] Desain, P (1993)
" A connectionist and a traditional AI quantizer, symbolic versus sub-symbolic models of rythm perception "
In contemporary music review, 9, 239-254.

[Desain Honing 94] Desain P and Honing H
" Rule-based models of initial beat induction and an analysis of their behavior "
In proceedings of the 1994 International Computer Music Conference 80-82.
San Francisco :International Computer Music Association

[Goto Muraoka 95]
"A real time beat tracking system for audio signals."
In Proceedings of the 1995 ICMC (pp. 171-174). Banff.

[Goto 95]
"Real time beat tracking for drumless audio signals : Chord change detection for musical decisions."
Speech Communication 27(3-4), 311-335
http :www.etl.go.jp/ goto/PROJ/bts.html

[Honing 93] Honing, H. (1993)
" Issues in the representation of time and structure in music "
Contemporary Music Review, 9, 221-239.

[Large 94] Edward W.Large, John F.Kolen
" Resonnance and the Perception of musical meter "
Connection Science, 6 (1) 177-208

[Lerdahl Jackendoff 83]
" A generative theory of tonal music "
Cambridge : MIT press.

[Longuet higgins 87]
" Mental Process "
Cambridge, Mass. MIT Press.

[LRD 93] M. Laurson, C. Rueda, J. Duthen.
" The Patchwork Reference Manual "
IRCAM, 1993

[LUSSON 73] Pierre Lusson
" Notes préliminaires sur le rythme "
Cahiers de Poétique Comparée, Vol. I, n°1,1973, pp.30-54.

[LUSSON 86] Pierre Lusson
" Place d'une théorie générale du rythme parmi les théories analytiques contemporaines "
Analyse Musicale n° 2, pp. 44-51, 1986.

[Mazzola 97] Guerino Mazzola
" inverse performance theory "
gbm@presto.pr.net.ch

[Meudic 99]
" Organisation rythmique dans un environnement d'aide à la composition"
Mémoire de fin d'étude de l'IIE.

[Palmer Krumhansl 90]
" Mental representation of musical meter "
Journal of Experimental Psycology : Human Perception and Performance.

[Rosenthal 92]
" Emulation of human rhythm perception "
Computer Music journal 16, 64-76

[Ruwet 66]
" Méthodes d'analyse en musicologie "
Revus belge de musicologie, (20) :65-90, 1966

[Scheirer 97]
" Tempo and beat analysis of acoustic music signals "
Journal of Acoustical Society of America 103(1) 588-601

[Scheirer 2000]
" Music listening systems "
Phd, avril 2000, Program in Arts and Sciences, MIT
http ://sound.media.mit.edu/ eds/thesis/

[Tanguiane 94]
" A principle of correlativity of Perception and its application to Music Recognition "
Music Perception, 1994, vol11 n°4 465-502

[Wolff 98]
" Parsing as information compression by multiple alignment, unification and search : SP52 "
School of electronic Engineering and computer Systems, university of Wales.
http ://www.sees.bangor.ac.uk/ gerry/

# Annexe B

# Description détaillée de la version 2.0 pour son utilisation

## B.1 L'environnement graphique

Description des entrées de l'éditeur chord-seq :
- la référence de la classe chord-seq
- les hauteurs des notes
- les onsets (voir définition dans le glossaire musical)
- les durées
- les intensités

L'éditeur chord-seq-seg :

L'éditeur de quantification chord-seq-seg dispose par rapport à chord-seq d'une vue supplémentaire apparaissant sur la droite de l'éditeur, qui permet de définir et manipuler des segmentations. De plus, l'éditeur compte deux entrées-sorties supplémentaires : la liste des méthodes de segmentation définies hors de l'éditeur et une liste complète des paramètres définissant les segmentations. Cette dernière liste permet par ailleurs d'effectuer des sauvegardes.

Description de la vue :

La liste des segmentations définies par l'utilisateur se situe en haut à droite. La segmentation sélectionnée (en surbrillance) est visualisée sur le panel de l'éditeur. Pour sélectionner plusieurs segmentation, cliquer sur les segmentations du menu en maintenant la touche 'shift appuyée.

Des " radio-boutons ", situés sous la liste, permettent de visualiser plusieurs segmentations en même temps. Le choix de la combinaison (union, intersection ou complémentaire de l'intersection) se fait en cliquant sur le " radio-boutton " correspondant.

Les actions du menu " segmenter " permettent de placer des barres de segmentation sur la séquence. Les méthodes proposées seront détaillées dans la section B.2.2.

Les actions du menu principal " menu " permettent de réaliser les opérations de création, sauvegarde, copie et suppression des segmentations d'une part, et de rechercher une pulsation d'autre part.

Les deux " radio-bouton " du bas permettent de choisir l'unité selon laquelle s'expriment les valeurs des barres de segmentation. La valeur d'une barre de segmentation est définie par sa position dans l'échelle de temps (onset). Par défaut, l'unité est la milliseconde. Le " radio-bouton " 'onset' contraint les barres à se situer sur l'onset d'une note du panel.

On peut distinguer deux étapes dans la manipulation de l'éditeur. La première consiste à créer une/des segmentation(s), et à segmenter la séquence de l'éditeur. La deuxième étape consiste à rechercher une ou plusieurs valeurs de pulsation à partir de cette segmentation et à quantifier la séquence.

## B.2 Première étape : segmenter

### B.2.1 Réaliser des opérations sur les segmentations dans l'éditeur

Créer une segmentation

Touche 'n' , ou Menu 'edit' - 'new-seg'

Visualiser une segmentation

Sélectionner un nom de la liste des segmentations.

Modifier une segmentation

Double-click sur un nom de la liste des segmentations.

Les paramètres que l'on peut modifier sont :

le nom de la segmentation

la couleur des séparations

le tempo attribué à la segmentation si la phase de quantification a été effectuée

les paramètres du module quantify d'OpenMusic

Sauvegarder une segmentation

Sélectionner une (ou plusieurs avec la touche shift) segmentation(s) dans la liste.

Ensuite, menu 'edit' - 'save' (ou touche 's').

Copier une segmentation

Sélectionner une (ou plusieurs avec la touche shift) segmentation(s) dans la liste.

Ensuite, menu 'edit' - 'copy' (ou touche 'c').

Supprimer une segmentation

Sélectionner une (ou plusieurs avec la touche shift) segmentation(s) dans la liste.

Ensuite, touche 'd' ou menu 'edit' - 'delete'.

Combiner des segmentations entre elles

Il est possible de combiner plusieurs segmentations pour en former une nouvelle.

Pour cela, sélectionner une (ou plusieurs avec la touche 'shift' et/ou 'pomme') segmentation(s) de la liste avec la souris, puis suivant le type de combinaison désiré, cliquer sur le radio-bouton 'union' (union des segmentations), 'inter' (intersection des segmentations) ou 'xor' (complémentaire de l'intersection des segmentations).

La sauvegarde de la nouvelle segmentation s'effectue ensuite par la touche 's' ou par le menu 'edit' - 'save'.

Ajouter une/des séparation(s) à une segmentation

1) à la main

Appuyer sur 'alt' puis cliquer sur l'emplacement désiré.

2) à l'aide des méthodes

Choisir une méthode de segmentation dans le menu 'segmentation'.

La liste des méthodes disponibles est donnée en B.2.2

Il est possible de définir une méthode de segmentation externe à l'éditeur par le biais d'un patch OpenMusic :

Procédure :

Créer un patch dans OpenMusic avec une entrée, une sortie.

L'entrée sera un chord-seq.

La sortie sera une liste contenant les onsets des positions désirées des barres de séparations.

Le patch pourra donc contenir des fonctions effectuant le calcul de cette liste en fonction des critères de l'utilisateur.

Exemple ci-dessous : une fonction qui recherche les inter-onset localement maximum ou minimum du chord-seq. Un paramètre (input1) sera demandé lors de l'utilisation de la méthode dans l'éditeur de quantification. Il permet de choisir entre les inter-onset maximaux et minimaux. (par défaut).

Une fois le patch défini, il faut l'abstraire.

L'introduire ensuite dans une liste avec éventuellement d'autres abstractions, puis connecter la liste à l'entrée 'Labstract' de l'éditeur de quantification.

Pour préciser le nom de l'abstraction qui devra apparaître dans la liste 'segmentation', faire une liste de l'abstraction et du nom voulu avant de l'associer à la liste des autres abstractions.

Un exemple est disponible dans la partie 4.5.

Ajouter une/des séparation(s) à une segmentation dans une certaine zone.

Il est possible de définir des zones dans la séquence étudiée sur lesquelles s'appliqueront les méthodes de segmentation.

Une première façon d'opérer est de déplacer les délimiteurs pour encadrer la zone voulue. Cette méthode ne permet pas de définir des zones disjointes, et il est assez contraignant de déplacer les délimiteurs à chaque définition de nouvelle zone. Leur principal rôle est d'ailleurs d'encadrer l'ensemble de la séquence à étudier.

Une deuxième méthode permet de définir des zones disjointes une fois que une ou plusieurs séparations aient été placées. Pour sélectionner une zone délimitée par deux séparations, il faut double-cliquer dans cette zone. Pour y ajouter des zones disjointes, double-cliquer dans les zones désirées en maintenant appuyée la touche shift.

Il apparaît alors un trait pointillé au-dessus de chaque zone sélectionnée. Les méthodes de segmentation s'appliqueront à ces nouvelles zones.

Pour supprimer la sélection d'une zone, double-cliquer sur la zone en maintenant la touche shift appuyée.

Sélectionner une/des séparations

Cliquer sur la séparation. Elle devient noire si elle est sélectionnée.

Pour sélectionner plusieurs séparations, utiliser la touche 'shift'.

Pour sélectionner l'ensemble des séparations, double-cliquer sur une séparation.

Enlever une/des séparations

Sélectionner la/les séparations, puis appuyer sur la touche '<–' (supprim)

Manipuler les séparations

1) avec la souris

Sélectionner une (ou plusieurs avec la touche shift) séparation avec la souris (une séparation devient noire lorsqu'elle est sélectionnée).

Pour déplacer les séparations sélectionnées, cliquer sur une séparation en déplaçant la souris.

Pour réaliser des compressions et des dilatations sur en ensemble de séparations sélectionnées, appuyer sur la touche 'ctrl' et déplacer la souris.

Pour réaliser une translation sur en ensemble de séparations sélectionnées, appuyer sur la touche 'pomme' et déplacer la souris.

Pour réaliser une accélération ou décélération sur en ensemble de séparations sélectionnées, appuyer sur la touche 'pomme' + 'ctrl' et déplacer la souris.

2) à l'aide du clavier

Après avoir sélectionné une/des séparations, il est possible de la/les déplacer avec les touches de déplacement du curseur (gauche et droite).

Déplacer les délimiteurs

Un délimiteur détermine la zone de la séquence à quantifier.

On le déplace avec la souris après avoir cliqué dessus.

Sinon, cliquer sur la zone de la séquence désirée en maintenant les touches 'ctrl' + 'alt' enfoncées pour le délimiteur gauche (début de séquence à quantifier) et 'alt' + shift' pour le délimiteur droit (fin de séquence à quantifier).

### B.2.2 Présentation des méthodes de segmentation

Certaines méthodes de segmentation sont prédéfinies dans le menu 'segmentation' :

- Les méthodes "i-onset-max" et "i-onset-min" placent des séparations sur les inter-onset localement maximaux ou minimaux. Le paramètre, 'smooth est un entier qui permet de lisser plus ou moins finement la courbe des inter-onsets avant la recherche des maxima. Plus sa valeur est grande, plus le lissage est grossier. La valeur par défaut est 1.
- Les méthodes "pitch-max" et "pitch-min" recherchent les hauteurs localement maximales ou minimales. Il n'existe pas pour l'instant de paramètre de contrôle.
- La méthode "chord" effectue un simple repérage des groupes de notes ayant exactement le même onset.

- La méthode "specify" offre la possibilité de repérer un élément particulier de la séquence. Par exemple, pour poser une barre de séparation sur toutes les hauteurs de 6000 midicents, il faut sélectionner le bouton ’midic, le bouton ’=, puis mettre la valeur 6000 dans la première case de paramètre.
- La méthode "regular grid" permet de placer des séparations à intervalles réguliers. Ces intervalles sont précisés dans le champ ’pas de la fenêtre de dialogue.
- La méthode "segment period" permet de placer des séparations si possible à intervalles réguliers uniquement sur les onset de la séquence. Ces intervalles sont précisés dans le champ ’pas de la fenêtre de dialogue.
- La méthode "segment-auto1" permet de placer des séparations si possible à intervalles réguliers uniquement sur les onset de la séquence pour un pas défini à l’aide de la méthode du coefficient d’auto-corrélation.
- La méthode "segment-auto2" est une variante de la méthode précédente dans laquelle le calcul du coefficient d’auto-corrélation est effectué sur des inter-onset non répartis dans le temps. Elle offre parfois de meilleurs résultats que la précédente.
- La méthode "beat-tracking" permet de placer des séparations tous les intervalles de temps définis par la courbe de tempo issue des calculs réalisés avec les coefficients-d’auto-corrélation.
- La méthode "segment-cambo" place des barres de segmentation suivant la courbe d’accents issue des marquages préconisés par Cambouropoulos.

Il est par ailleurs possible de définir ses propres méthodes de segmentation.

La procédure est la suivante :

Créer un patch dans OpenMusic avec une entrée, une sortie. L’entrée sera un chord-seq. La sortie sera une liste contenant les onsets des positions désirées des barres de séparations. Le patch pourra donc contenir des fonctions effectuant le calcul de cette liste en fonction des critères de l’utilisateur. Abstraire ce patch. L’introduire dans une liste, avec éventuellement d’autres abstractions, puis connecter la liste à l’entrée ’Labstract’ de l’éditeur de quantification. Pour préciser le nom de l’abstraction qui devra apparaître dans la liste ’segmentation’, faire une liste de l’abstraction et du nom voulu avant de l’associer à la liste des autres abstractions. Un exemple est disponible dans la partie 4.5.

## B.3 Deuxième étape : rechercher une pulsation et quantifier

### B.3.1 Rechercher une pulsation dans l’éditeur

Sélectionner une segmentation dans la liste de noms.
Pour lancer la recherche appuyer sur la touche ’t’, ou par le menu ’edit’ - ’tempo’.
Choisir les paramètres par défaut (case en haut à gauche cochée par défaut), ou remplir les champs proposés.
Choisir le tempo désiré parmi ceux qui sont proposés.
Une nouvelle segmentation apparaît dans la liste des segmentations, de nom :
" nom de la segmentation sélectionnée " + " -quant "
La nouvelle segmentation comporte deux types de séparations :
Les grandes barres verticales représentent l’emplacement des futures barres de mesure.
Les petites barres verticales représentent la pulsation à l’intérieur de chaque mesure.
La position des séparations de la nouvelle segmentation ne peut plus être modifiée.
Il est possible de modifier le tempo trouvé et les paramètres de quantification dans une fenêtre de dialogue qui est activée par un double-click sur le nom de la nouvelle segmentation ou appuyer sur o (open) après avoir sélectionné une ou plusieurs barres de segmentation. Une trop grande modification de tempo peut entraîner d’importants changements dans la séquence étudiée.

### B.3.2 Quantifier dans l’éditeur

La quantification se fait automatiquement.
Elle utilise le module quantify.
Pour affiner l’étape de quantification, cliquer deux fois sur la segmentation de la liste ou appuyer sur o (open) après avoir sélectionné une ou plusieurs barres de segmentation. Les entrées habituelles du module quantify sont affichées et peuvent être modifiées.
Pour visualiser le rythme obtenu, connecter la sortie ’self’ de l’éditeur au module voice d’OpenMusic (ou poly si la séquence comporte plusieurs voix) :

## B.4 Glossaire des termes utilisés dans la description de la librairie Kant

Abstraction

Il est possible de définir une méthode de segmentation externe à l'éditeur par le biais d'un patch OpenMusic.

Procédure :

Créer un patch dans OpenMusic avec une entrée, une sortie.

L'entrée sera un chord-seq.

La sortie sera une liste contenant les onsets des positions désirées des barres de séparations.

Le patch pourra donc contenir des fonctions effectuant le calcul de cette liste en fonction des critères de l'utilisateur.

Abstraire ce patch.

L'introduire dans une liste, avec éventuellement d'autres abstractions, puis connecter la liste à l'entrée 'Labstract' de l'éditeur de quantification.

Pour préciser le nom de l'abstraction qui devra apparaître dans la liste 'segmentation', faire une liste de l'abstraction et du nom voulu avant de l'associer à la liste des autres abstractions.

Un exemple est disponible dans la partie 4.5.

Pour obtenir plus d'informations, voir aussi la partie Ajouter une/des séparation(s) à une segmentation de Réaliser des opérations sur les segmentations dans l'éditeur

Couleur

Il est possible d'attribuer une couleur aux séparations verticales d'une segmentation.

Pour cela, double-cliquer sur le nom de la segmentation faisant partie de la liste en haut à droite de l'éditeur.

Cliquer sur le bouton choisir de la nouvelle fenêtre.

Délimiteurs

Les délimiteurs permettent la segmentation d'une partie restreinte de la séquence.

Il existe deux délimiteurs, un délimiteur de début de séquence (crochet placé au début de la séquence à étudier), et un délimiteur de fin de séquence (crochet placé en fin de la séquence étudiée).

Par défaut, ils sont placés au début et à la fin de la séquence de manière à ce que la segmentation se fasse sur l'ensemble de la séquence.

Pour restreindre la zone de segmentation, on peut les déplacer à l'aide de la souris, ou par l'intermédiaire du clavier :

'ctrl'+'alt'+click pour le délimiteur de début

'shift'+'alt'+click pour le délimiteur de fin

Durée

Lors de la phase de quantification, les durées sont légèrement modifiées avec les onsets de manière à respecter les rapports de valeur initiaux dans chaque nouvelle mesure, et les contraintes d'emplacement des barres verticales (séparations, futures barres de mesure).

Hauteurs

Les hauteurs de la séquence peuvent servir à placer des barres de séparation (méthodes Pitch-min et Pitch-max du menu 'segmentation').

Inter

Le bouton 'inter' de l'éditeur spécifie le mode de combinaison des séparations lorsque plusieurs segmentations sont sélectionnées en même temps (à l'aide de la touche shift) .

Si le bouton inter est sélectionné, seule l'intersection entre les séparations de toutes les segmentations sélectionnées est affichée.

I-onset

Les inter-onset sont les distances qui séparent deux onsets consécutifs d'une séquence. Ils ne correspondent pas forcément aux durées qui peuvent se chevaucher au contraire des inter-onset.

Labstract

L'entrée 'Labstract de l'éditeur permet d'ajouter aux méthodes de segmentation existantes des méthodes définies par l'utilisateur au moyen de patchs abstraits.

Voir abstraction.

Ldur

voir durées

Lmidic

Les hauteurs ne sont pas prises en compte dans la phase de quantification.

Seule les méthodes 'pitch-max et 'pitch-min dépendent des hauteurs.

Elles pourront cependant être utilisées par des méthodes de segmentation plus sophistiquées définies par l'utilisateur ou dans une prochaine phase de développement de l'éditeur.

Mesure

Les mesures de la structure rythmique finale sont définies par l'emplacement des traits verticaux (séparations) placés lors de la phase de segmentation de la séquence.

La correspondance entre les barres de séparations et les mesures du rythme final est exacte.

Méthodes de segmentation

Une méthode de segmentation est un critère d'emplacement des barres de séparation.

Par exemple, une méthode peut être de placer des barres de séparation sur les éléments dont la durée est localement maximale (méthode 'dur-max' du menu segmentation de l'éditeur).

Les éléments ainsi repérés seront tous situés en début de mesure à l'issue de la quantification.

Mode

Le mode (affiché en haut de la liste des segmentations) permet d'utiliser l'éditeur pour manipuler les segmentations (mode seg) ou pour ajouter/enlever des notes (mode chord).

Pour passer d'un mode à un autre, appuyer sur la touche m ou aller dans le menu 'edit' puis choisir 'change mode'.

! En mode chord, il est impossible de manipuler les séparations.

! En mode seg, il est impossible d'ajouter ou d'enlever des notes.

Nom

Chaque segmentation est définie par son nom unique placé dans la liste en haut à gauche de l'éditeur.

Il est possible de modifier ce nom en faisant un double-click sur le nom de la liste, puis en modifiant le champ nom de la fenêtre de dialogue.

En cas de conflit de nom, un message demandera la confirmation du nouveau nom. en cas de confirmation, la segmentation portant déjà le nom sera écrasée.

Onset

Le champ 'onset de l'éditeur (en bas à droite) impose une contrainte sur les séparations (barres verticales déterminant l'emplacement des futures barres de mesures) : lors d'un ajout ou d'un déplacement, si le champ est sélectionné, les séparations se placeront uniquement sur les onset des éléments de la séquence.

Le champ situé en dessous du champ onset détermine le pas d'une grille virtuelle sur laquelle iront se placer les séparations en cas d'ajout ou de déplacement. Par défaut, ce pas a une valeur de 1 ms (valeur minimale).

Quantification

La quantification intervient après l'étape de segmentation.

Elle débute par la recherche d'une pulsation dans chaque zone définie par les séparations.

Rythme

Le rythme final obtenu à l'issue de la quantification peut être visualisé à l'aide du module voice d'OpenMusic.

Pour cela, connecter la sortie 'self' de l'éditeur au module voice d'OpenMusic :

Sauvegarde

La sauvegarde des segmentations dans l'éditeur se fait par l'intermédiaire du menu 'edit'+'save' ou par la touche 's'.

L'évaluation de l'éditeur initialise toutes les segmentations précédemment crées. Pour ne pas les perdre, il faut fermer l'éditeur à l'aide de la touche 'b'.

Il est aussi possible de conserver les segmentations en faisant une copie de la sortie 'Lseg' de l'éditeur, ou en faisant une copie de l'éditeur lui-même. Pour cela, appuyer sur les touches 'alt' et 'shift' en cliquant sur la sortie 'chord-seq-seg' (sauvegarde de tout l'éditeur) ou la sortie 'Lseg' (sauvegarde des emplacements des barres de séparation.

Segmentation

Une segmentation est un découpage de la séquence étudiée en prélude à une quantification rythmique. Les contraintes posées par ce découpage donne un puissant contrôle sur l'étape de quantification. Le découpage correspondra au découpage en mesures de la séquence quantifiée.

Séparation

Une séparation matérialise un découpage de la séquence étudiée en deux parties.

Elle est visualisée par un trait vertical dans l'éditeur.

Pour ajouter une séparation, faire 'alt' puis cliquer à l'endroit de la séquence désiré.

Ou utiliser le menu segmentation.

Séquence

La séquence étudiée peut être un ensemble de notes ou d'accords, rentrées à la main ou par les entrées habituelles d'un chord-seq.

Signature

Lorsqu'un tempo est choisi, le nombre de pulsation de chaque nouvelle mesure est déterminé.

Pour les visualiser, double cliquer sur le nom de la nouvelle segmentation. Le champ 'signature' de la fenêtre de dialogue contient la liste des signatures de toute la séquence étudiée.

Tempo

Une fois les barres de séparation placées, la recherche d'un tempo se fait en appuyant sur la touche 't' ou par l'intermédiaire du menu 'edit' + 'tempo'.

Une liste de tempi est alors proposée. Pour modifier la valeur du tempo une fois qu'il a été trouvé, double-cliquer sur le nom de la nouvelle segmentation, puis corriger le champ 'tempo'. Il est conseillé de ne pas faire de correction trop importante pour limiter les changements des onsets et durées par rapport aux valeurs initiales.

Tree

Une fois le tempo choisi, la quantification se fait automatiquement. Le résultat est une structure rythmique, disponible dans la sortie 'tree' de l'éditeur.

Pour ajuster les paramètres de quantification, double-cliquer sur le nom de la segmentation puis corriger les paramètres de la fenêtre de dialogue.

Les paramètres sont ceux du module quantify d'Open-Music :

Max/

Forbid

Precis

Union

Le bouton 'union' de l'éditeur spécifie le mode de combinaison des séparations lorsque plusieurs segmentations sont sélectionnées en même temps.

Ici, l'union (au sens mathématique) entre les séparations de toutes les segmentations sélectionnées est affichée.

Xor

Le bouton 'xor' de l'éditeur spécifie le mode de combinaison des séparations lorsque plusieurs segmentations sont sélectionnées en même temps.

Ici, le complémentaire de l'intersection entre les séparations de toutes les segmentations sélectionnées est affichée.