École des Hautes Études en Sciences Sociales, Paris, France
Centre d'Analyse et de Mathématique Sociales
(CAMS, UMR 8557 CNRS – EHESS)

*en collaboration avec l'*Ircam, Équipe Représentation Musicale

Donner à OMax le sens du rythme : vers une improvisation plus riche avec la machine

Laurent Bonnasse-Gahot

Juillet – Septembre 2010

Rapport interne

Ce travail de 3 mois se place dans le cadre du projet soutenu par l'ANR
IMPROTECH, Technologies et musiques improvisées, (ANR-09-SSOC-068)
piloté par M. Chemillier (CAMS, EHESS)

Table des matières

1 Introduction

1.1 Cadre général

Conçu et développé à l'Ircam (Assayag *et al.*, 2006), le logiciel *OMax* [1] propose d'établir une interaction musicale entre l'homme et la machine dans le cadre d'une improvisation. Son nom provient des deux logiciels de programmation musicale OpenMusic [2] et Max/MSP [3], tous deux conçus à l'Ircam et particulièrement utilisés dans le domaine de l'informatique musicale. L'idée de les associer est née après un premier prototype de logiciel pour l'improvisation conçu par Marc Chemillier avec un modèle de grammaire harmonique (Chemillier, 2001), et dont l'interface était réalisée par Carlos Agon directement dans OpenMusic.
Dans la suite des travaux sur la simulation stylistique menés par l'équipe Représentation Musicale à l'Ircam (Dubnov et Assayag, 1998; Assayag *et al.*, 1999), OMax est capable d'apprendre le style d'un improvisateur humain grâce à une représentation basée sur l'algorithme d'oracle des facteurs développé par Allauzen *et al.* (1999) et étendu dès 2002 à un contexte musical (voir Poirson, 2002; Assayag et Dubnov, 2004). Le logiciel construit un modèle du jeu du musicien au fur et à mesure qu'il le capte. Il est ensuite possible de naviguer à l'intérieur de cette représentation en empruntant des chemins différents de celui pris par le musicien, ce qui conduit à générer des improvisations originales dans l'esprit du jeu de l'artiste. Un dialogue inédit entre le musicien et son clone virtuel peut alors se mettre en place. Si OMax peut fonctionner de façon autonome, le système est généralement contrôlé par un utilisateur, qui peut orienter le logiciel dans sa mémoire et agir sur la fidélité des phrases générées (introduisant plus ou moins de surprise dans le jeu de la machine). Depuis sa création en 2004, le logiciel OMax a interagi avec de nombreux musiciens contemporains comme Mike Garson, Cécile Daroux, Médéric Collignon, Bernard Lubat, etc.

Dans une version ancienne du système, il fonctionnait selon deux modes : un mode totalement libre ('free-mode'), et un mode avec accompagnement ('beat-mode') (Assayag *et al.*, 2006). Ces deux cas correspondent à deux situations extrêmes se situant de part et d'autre d'une situation réelle d'improvisation entre musiciens. Dans le premier mode, aucune mise en place rythmique ne relie l'homme et la machine, tandis que dans le second, le rythme est imposé par la machine. Dans ce dernier cas, le tempo est fixé

1. `http://omax.ircam.fr/`
2. `http://recherche.ircam.fr/equipes/repmus/OpenMusic/`
3. `http://www.cycling74.com`

et la pulsation délivrée de façon métronomique, ce qui enlève inévitablement une partie de la richesse musicale de l'interaction entre musiciens et contraint le musicien dans son expression. Comme le dit Bernard Lubat, "le rythme, c'est la danse, c'est la narration de l'heureux-bondi. C'est un rebondi qui ne rebondit jamais toujours pareil" [4]. Trop contraignant, le mode 'beat' d'OMax est ainsi à l'arrêt depuis 2007. Le but de ce projet est justement de relancer cette facette d'OMax, grâce au développement d'un algorithme qui détecte la pulsation en temps réel. Cette pulsation, directement extraite de l'improvisation en cours, viendra remplacer la pulsation métronomique imposée par le système, ce qui permettra une interaction musicale plus riche avec la machine.

1.2 Objectifs

Le présent projet vise à implémenter un algorithme de suivi de tempo en Max dans le but d'une intégration avec le logiciel OMax. Ce suivi de tempo permettra lors de l'écoute de segmenter les phrases jouées en fonction de la pulsation, puis de générer des phrases rythmiquement en place avec l'improvisation en cours. À noter que cette utilisation de la pulsation était déjà implémentée dans le mode 'beat' d'OMax. La différence ici est que la pulsation ne sera plus imposée par la machine mais directement captée dans le jeu du musicien.
Ce cadre impose quelques contraintes fondamentales dans le choix de la méthode à utiliser. La musique étant improvisée, le logiciel ne peut connaître à l'avance ce qui va suivre : l'algorithme doit donc être causal, c'est-à-dire qu'il ne peut utiliser que l'information passée, et doit être capable de prédire l'arrivée de la pulsation suivante. Par ailleurs, il doit fonctionner en temps réel avec une latence la plus faible possible, de sorte que les phrases générées soient effectivement en rythme avec le jeu du (ou des) musicien(s).

Deux points majeurs sont à considérer pour développer un algorithme de suivi de tempo. Il faut d'abord être capable d'estimer le débit moyen (*ie* le tempo global), puis de suivre les variations de tempo au cours du temps. Dans le cadre de notre projet, l'utilisateur peut donner des indications précieuses à l'algorithme de suivi de tempo par une battue manuelle du tempo sur quelques temps. Ceci donne à l'algorithme non seulement le tempo initial mais également l'emplacement des pulsations, ce qui peut éviter par exemple que le logiciel ne se synchronise à contretemps. Ainsi, l'objectif principal se

4. Propos recueillis dans le cadre du séminaire de Marc Chemillier, *Modélisation des savoirs musicaux relevant de l'oralité*, EHESS, mai 2010

résume à suivre les variations de tempo, tout en restant synchronisé en phase avec la musique, la question (difficile) de savoir à quel débit global se placer n'étant pas ici un problème.

Si les algorithmes de suivi de tempo actuels connaissent de bonnes performances sur des styles musicaux caractérisés par un rythme particulièrement marqué et un tempo quasi constant (la pop, le rock, la techno, le reggae, le rap, etc.), les performances se dégradent sérieusement pour le jazz et le classique, lesquels se distinguent par une plus grande complexité rythmique et/ou des variations de tempo plus importantes. OMax étant principalement destiné à une utilisation en contexte jazz/musiques improvisées, une attention particulière doit être portée pour s'assurer que l'algorithme développé est capable de suivre ce type de musique-là.

Enfin, la réaction du suivi de tempo en l'absence d'entrées est un dernier point important à considérer ici. Chez l'homme, une fois la sensation de la pulsation établie, celle-ci peut continuer à exister dans l'esprit du sujet même en l'absence de stimuli externes. Des musiciens peuvent ainsi s'arrêter de jouer le temps de quelques pulsations (un 'break'), puis reprendre ensemble de façon synchrone. De même, l'objet développé devra être capable d'assurer la continuité de la pulsation même dans les cas où aucun événement n'est capté par le système.

En résumé, pour une utilisation appropriée au cadre d'OMax, l'algorithme choisi devra :

(1) être causal
(2) être utilisable en temps réel
(3) être capable de suivre des variations de tempo
(4) faire face à des rythmes complexes
(5) assurer une continuité en l'absence d'entrée

2 État de l'art

Le but de cette section n'est pas de faire une revue exhaustive des travaux sur la perception du rythme. Ce thème de recherche fait l'objet de nombreux travaux dans de multiples disciplines comme la psychophysique, l'informatique, les neurosciences computationnelles, la neuroimagerie, l'éthologie, etc. C'est pourquoi seuls les points décisifs pour le présent projet sont abordés.

2.1 Quelques données chez l'homme

L'aptitude à percevoir le rythme d'une source musicale est une caractéristique humaine universelle. Même un individu sans formation musicale peut battre du pied la pulsation sous-jacente à une musique sans le moindre effort. Des études récentes montrent même que, dès la naissance, un enfant perçoit la pulsation (Winkler *et al.*, 2009). Plus précisément, il est non seulement sensible à la périodicité d'une séquence musicale, mais développe également des attentes quant au moment d'apparition de la pulsation suivante. Chez l'adulte, dans le cadre de la perception d'une séquence de tons, Snyder et Large (2005) ont montré que l'activité induite de la bande gamma prédit l'arrivée d'un ton et persiste lorsque les tons attendus sont omis.

Plusieurs données issues des neurosciences ou de la psychophysique nous permettent de préciser quels indices acoustiques présents dans le signal musical interviennent dans la perception de la pulsation. Tout d'abord, la perception du rythme et celle de la hauteur semblent relever de modules en grande partie indépendants (Peretz et Coltheart, 2003). Suite à des lésions cérébrales, une personne peut perdre l'aptitude à discriminer les hauteurs tout en gardant la perception du rythme. La situation inverse est également rencontrée (voir Peretz et Zatorre, 2005, pour une revue). Au niveau comportemental, Snyder et Krumhansl (2001) ont montré qu'enlever l'information mélodique et harmonique de ragtimes joués au piano n'a que peu d'impact sur l'aptitude des sujets à trouver et à suivre la pulsation (voir également Scheirer, 1998). Ce résultat est d'autant plus intéressant qu'il n'est pas intuitif. En effet, si l'on considère la série de notes d'égales longueurs 'do-re-do-re...', un sujet perçoit préferentiellement la pulsation une note sur deux (rythme binaire), alors que pour la série 'do-re-mi-do-re-mi...', la pulsation est préferentiellement portée par une note sur trois (rythme ternaire). En pratique, dans les morceaux considérés par les auteurs de cette étude, les résultats des expériences montrent que le motif rythmique n'est pas aussi ambigu que pour cet exemple simple (celui-ci ne consistant qu'en une série de croche, seule l'information mélodique est disponible).
Cette dissociation rythme/hauteur a une importance capitale ici puisqu'elle implique que l'on peut, en première approximation, ne considérer que les attaques des notes sans s'occuper de l'information de hauteur, ce qui simplifie le problème. D'autres travaux ont montré que l'intensité d'une note mais aussi sa durée jouent un rôle dans notre perception du rythme. Dans une séquence alternant une note longue et une note plus courte (♩. ♪ ♩. ♪ par exemple), la pulsation sera généralement portée par la note longue (Parncutt, 1994).

2.2 Revue des algorithmes existants

La recherche sur l'estimation et le suivi de tempo constitue un domaine de recherche particulièrement actif. Ce thème est particulièrement stimulé par le développement de la recherche d'information musicale ('music information retrieval', MIR), qui vise à développer des méthodes de recherche, de segmentation, de classification et d'indexation de signaux musicaux. Depuis quelques années, dans le cadre de la conférence annuelle donnée par l'ISMIR (International Society for Music Information Retrieval [5]), un concours (MIREX : Music Information Retrieval Evaluation eXchange [6] ; voir Downie, 2008) est organisé dans le but de proposer des méthodes d'évaluation communes entre laboratoires et de comparer ainsi les algorithmes les plus récents. Deux tâches concernent l'analyse du rythme : l'estimation de tempo (aussi appelée 'tempo induction' dans la littérature anglophone) et le suivi de tempo ('beat tracking'). Le premier cas concerne l'estimation, à partir de son signal audio, du débit global des pulsations d'un morceau musical présentant peu de variations (voir Gouyon *et al.*, 2006, pour une revue). Le deuxième cas vise à trouver l'emplacement des pulsations dans un signal musical audio (ce qui permet donc également de suivre les variations de tempo) (voir McKinney *et al.*, 2007; Davies *et al.*, 2009). Si les travaux présentés à cette conférence dominent l'état de l'art dans le domaine, ils ne sont pas forcément appropriés au cadre de notre projet. En effet, ces travaux ont principalement des applications en ingénierie, par exemple la recherche de morceaux de musique en fonction du tempo choisi (pour le cinéma, la publicité, la radio, un DJ, etc.). Ce type d'applications ne présentent pas les mêmes contraintes que dans notre cas (voir Section 1.2) : en particulier, les contraintes de temps réel et de causalité, indispensables ici, ne sont pas importantes dans ces cas-là, ce qui permet d'obtenir de meilleurs résultats.

Il convient ainsi d'analyser les algorithmes existants en fonction des objectifs et contraintes imposés par le présent cadre. Pour une revue détaillée des algorithmes existants, le lecteur pourra se référer à Gouyon et Meudic (2003), Hainsworth (2003, p. 46–57), Gouyon et Dixon (2005), et Collins (2004).
Le fonctionnement commun à la grande majorité des modèles peut se résumer en la description de deux étapes majeures :

- La première étape consiste à fournir au système une liste d'événements. Dans le cas où l'entrée provient de données MIDI, aucun traitement particulier n'est à faire. On dispose directement du temps d'attaque (*onset*

5. `http://www.ismir.net/`
6. `http://www.music-ir.org/mirex/`

times), et éventuellement de la hauteur et de l'amplitude (vélocité) des notes. Si le système se base sur un signal audio, il s'agit d'extraire du signal des informations symboliques de ce type-là, en particulier les temps d'attaques. La suite du traitement est alors similaire à un algorithme disposant de données MIDI.

- La seconde étape consiste à trouver l'emplacement des pulsations grâce à un algorithme qui cherche une périodicité dans les événements considérés auparavant. Cette étape peut être éventuellement précédée d'une estimation de tempo.

Les modèles existants diffèrent principalement par la méthode utilisée pour chaque étape. Si, à l'origine, la plupart des modèles n'utilisaient que des informations de type MIDI, la majorité des modèles actuels sont capables de traiter directement un signal audio (différentes méthodes existent pour extraire d'un signal audio les attaques des événements, voir Bello, 2003, pour une revue). L'étape cruciale dans le cadre de notre projet est la seconde étape. De nombreuses méthodes ont été développées (voir par exemple les classifications proposées par Hainsworth, 2003; Gouyon et Meudic, 2003). Les différences les plus fondamentales entre ces modèles concernent les questions de causalité et de temps réel. Comme on l'a évoqué plus haut, la plupart des algorithmes développés ces dernières années ne sont pas causals – or notre projet nécessite un algorithme causal qui fonctionne en temps réel (avec une latence la plus faible possible). Sur les 33 algorithmes répertoriés par Hainsworth (2003, Table 2.1, p.49), seuls 8 sont causals (Goto, 2001; Jensen et Andersen, 2003; Klapuri *et al.*, 2006; Large et Kolen, 1994; McAuley, 1995; Seppänen, 2001b; Scheirer, 1998; Toiviainen, 1998). Goto (2001) suppose un tempo quasi constant et une signature du temps spécifique (4/4), ce qui proscrit l'utilisation d'un tel modèle dans le présent contexte. Le modèle de Scheirer (1998) est assez complexe et computationnellement exigeant, ce qui rend son utilisation en temps réel difficile. Le modèle de Klapuri *et al.* (2006) combine les qualités des algorithmes développés par Scheirer et Goto. Divers tests montrent que ce modèle surpasse les modèles de Scheirer (1998) et Seppänen (2001b) (voir Klapuri *et al.*, 2006; Seppänen, 2001a), et se montre particulièrement compétitifs face à d'autres modèles comme celui de Dixon (2001, 2007) (qui a obtenu les meilleurs résultats lors du concours MIREX 2006) (voir Klapuri *et al.*, 2006; Davies *et al.*, 2009). Enfin, les modèles de McAuley (1995), Large et Kolen (1994) et Toiviainen (1998) sont similaires, et abordent la question de la perception du rythme par une méthode d'oscillateurs adaptatifs (ainsi que les diverses évolutions du modèle de Large ; voir Large, 1995; Large et Jones, 1999; Large, 2000; Large et Palmer, 2002). Deux paramètres essentiels décrivent ces oscillateurs : leur phase et leur période,

qui peuvent s'adapter pour permettre à l'oscillateur de rester synchronisé avec l'entrée musicale (représentée sous forme d'événements discrets). En l'absence de stimuli, l'oscillateur continue à produire une pulsation au tempo en cours, sauf pour le modèle de McAuley (1995) qui revient à une période par défaut, et que l'on proscrit par conséquent (voir contrainte de continuité, Section 1.2). Les deux modèles restants diffèrent principalement par le mécanisme choisi pour adapter leurs paramètres internes. Lorsqu'un nouvel événement arrive (une note, représentée par son temps d'attaque), le modèle de Large adapte instantanément ses paramètres (phase et période) selon une méthode de descente de gradient ; le modèle de Toivianen, largement inspiré de celui de Large, introduit une fonction d'adaptation qui retarde les modifications apportées aux paramètres internes, ce qui permet de tenir compte de la durée des notes (qui, comme on l'a vu Section 2.1, est un paramètre potentiellement important dans la perception de la pulsation). Une lacune de ce type de modèle est qu'il n'est pas capable de trouver les bonnes valeurs initiales de la phase et surtout de la période, mais, comme annoncé précédemment, ce point-là n'est pas un problème ici vu que l'utilisateur d'OMax peut donner ces valeurs (en pressant une touche du clavier en rythme, ou au moyen d'un capteur externe). Signalons également que ces modèles ont dès le départ été utilisés et testés dans le cadre d'improvisations (voir Large et Kolen, 1994; Large, 1995; Toiviainen, 1998).

2.3 Applications musicales

Dès les années 80 (voir par exemple les experiences de Koechlin où une basse est générée en temps réel en accompagnant le pianiste Bobby Few ; Koechlin, 2010), diverses applications ont été développées dans le cadre de l'improvisation en interaction avec une machine. Le 'Continuateur' conçu par Pachet est similaire au mode libre d'OMax (Pachet, 2002a,b). Le système est capable de capter le jeu d'un musicien, d'en faire un modèle, puis de générer des phrases à partir de cet apprentissage (d'où le nom de 'continuateur'). Comme dans le mode libre d'OMax, aucun mécanisme de perception rythmique ne permet au programme de se synchroniser avec le musicien. À l'opposé, le programme GenJam, développé par Biles (2002) est plus proche du mode 'beat' d'OMax : la pulsation est automatiquement générée par le logiciel, qui délivre un accompagnement sur lequel un musicien improvise. Après écoute, le système génère des phrases issues du jeu du musicien mais plus ou moins modifiées, cette transformation s'opérant grâce à un algorithme génétique.
Plus proche de notre but, signalons l'accompagnateur jazz décrit par Toiviainen (1998). Là encore, la machine joue un accompagnement suivant une

grille de jazz et un musicien soliste improvise dessus. L'intérêt majeur du système est qu'il est capable de suivre les variations de tempo de l'instrumentiste, donnant ainsi un aspect plus humain à la performance musicale, grâce à un algorithme basé sur un oscillateur adaptif évoqué dans la section précédente et décrit dans le même article. Un suivi de tempo a également été employé dans le projet *Rameses 2000* [7] du saxophoniste de jazz Steve Coleman (en collaboration avec Gérard Nouno, compositeur et chercheur à l'Ircam). Cette œuvre a fait l'objet d'une performance présentée au théâtre des Bouffes du Nord à Paris. Des capteurs placés sur la batterie envoient à la machine les temps d'attaques, décodés en temps réel par un algorithme basé sur l'oscillateur de Large, ce qui permet de synchroniser le jeu des musiciens avec les interventions de l'ordinateur. Plus orienté vers la musique pop/rock, le projet B-Keeper (Robertson et Plumbley, 2007) utilise également un algorithme de suivi de tempo (que l'on peut voir comme une version simplifiée de l'algorithme de Large) pour synchroniser en temps réel le jeu d'un batteur avec un séquenceur musical. L'entrée du système se base sur le signal audio d'un microphone captant les frappes sur la grosse caisse de la batterie.

3 Modèles

3.1 Introduction

Il ressort de la revue effectuée en section 2.2 que les modèles d'oscillateurs de Large et Toiviainen semblent satisfaire pleinement les conditions imposées par le cadre de notre projet (Section 1.2). Ils sont causals, fonctionnent en temps réel, peuvent suivre des variations importantes et s'adapter à des rythmes complexes (ce point restant à tester sur le matériel musical qui nous intéresse), et assurent une continuité de pulsation en l'absence d'événement musical extérieur. Ajoutons à cela qu'ils sont computationnellement particulièrement simples, ce qui permet de ne pas surcharger le travail du processeur, le laissant libre d'effectuer d'autres tâches (apprentissage, calcul de nouvelles séquences, etc.). De plus, les paramètres de ces modèles sont faciles à comprendre et à interpréter, ce qui les rend a priori plus simples à contrôler. Un des atouts de ces modèles réside également dans leur plausibilité à la fois neurobiologique et comportementale. Sans être directement neurobiologiquement plausibles, ces modèles peuvent s'envisager dans un cadre plus réaliste (Large et Snyder, 2009), et sont par ailleurs compatibles avec les données de neuroimagerie les plus récentes (Snyder et Large, 2005; Grahn, 2009). Plusieurs études psychophysiques sur la battue du rythme ont égale-

7. `http://mac-texier.ircam.fr/works/work/7487/`

ment montré que le modèle de Large permet de rendre compte de plusieurs données comportementales (Large et Jones, 1999; Large et Palmer, 2002). On peut se demander l'intérêt d'une telle propriété dans le cadre de ce projet, pour lequel seul un objectif de performance pourrait être attendu. Avoir un comportement proche de l'humain est pourtant primordial dans le cadre d'une interaction homme-machine en temps réel. Imaginons par exemple que, suite à un changement de tempo, la machine fasse une erreur sur la prédiction de la pulsation suivante. Si cette erreur va dans la même direction que celle que ferait un être humain, elle sera moins perturbante dans un contexte musical qu'une erreur complètement décorrélée du comportement humain.

La notion d'erreur n'est d'ailleurs pas simple à définir de façon précise. L'étude de Dixon *et al.* (2006) est particulièrement intéressante sur ce point, et donne une motivation supplémentaire pour considérer l'implémentation d'un modèle à base d'oscillateurs. Les auteurs ont demandé à des sujets de noter la qualité de la correspondance entre un morceau musical et une piste marquant la pulsation selon différentes conditions. Une piste correspond à la pulsation 'objective', où les temps correspondent précisément aux temps d'attaques de notes présentes dans l'extrait musical. D'autres pistes évaluées correspondent à des versions 'lissées' de cette piste, de sorte que le tempo varie de façon plus souple que dans la version originale. Dans ce cas, la pulsation ne correspond pas forcément exactement à l'emplacement d'une note (ce qui est le cas lorsqu'on demande à des auditeurs de battre la pulsation). Le résultat de l'étude montre que les sujets préfèrent les versions plus 'souples' (*ie* moins irrégulière), et cet effet est plus marqué chez les musiciens. Or, de nombreux modèles de suivi de tempo génèrent une pulsation correspondant à la présence d'une attaque de note, ce qui n'est pas le cas pour les modèles à oscillateurs considérés ici. Ces derniers, en fonction des paramètres, peuvent générer une pulsation plus ou moins souple à l'image de la pulsation battue par un musicien.

3.2 Le modèle de Large

Le but ici est de donner une synthèse des différentes versions du modèle à oscillateurs développé par Large au fil des années (Large et Kolen, 1994; Large, 1995; Large et Jones, 1999; Large et Palmer, 2002). Le modèle considère un oscillateur auto-entretenu décrit par une fonction $o(t)$ du temps t. En l'absence de stimuli externes, elle est périodique de période p, et le modèle génère une pulsation régulière toutes les p millisecondes. Cette fonction

a d'abord été définie dans Large et Kolen (1994) et Large (1995) selon :

$$o_{95}(t) = K_{95}(\phi(t)) \tag{1}$$

où

$$K_{95}(\phi(t)) = 1 + \tanh(\gamma(\cos(2\pi\phi(t)) - 1)) \tag{2}$$

Les versions suivantes du modèle (Large et Jones, 1999; Large et Palmer, 2002) ont adopté la forme (ici normalisée à 1) :

$$o_{99}(t) = K_{99}(\phi(t)) \tag{3}$$

avec

$$K_{99}(\phi(t)) = \frac{1}{\exp\gamma} \exp\left(\gamma \cos(2\pi\phi(t))\right) \tag{4}$$

Dans ces expressions, la fonction $\phi(t)$ est également périodique de période p. On l'appelle la *phase de l'oscillateur*, et on appelle *période de l'oscillateur* la valeur du paramètre p. La période caractérise le temps requis pour compléter un cycle d'oscillation, tandis que la phase est une fonction "linéaire par morceaux", se reproduisant à des intervalles de longueur p, et définie au temps t, $-p/2 \leq t < p/2$, par $\phi(t) = t/p$.

La sortie de l'oscillateur $o(t)$ modélise l'attente du modèle par rapport à la venue d'un événement. Cela signifie qu'à l'arrivée d'un événement, la période et la phase vont être modifiées en fonction de cette attente, c'est-à-dire de la forme "en cloche" de la courbe K représentant la dépendance de o par rapport à ϕ (cf. Fig. 1). Cette attente est maximale à $t = t_x$, pour lequel $\phi(t) = 0$, qui correspond au moment où le modèle génère une pulsation, puis décroît de part et d'autre de cet événement selon un paramètre noté γ dans les expressions ci-dessus. La fonction $o(t)$ peut être vue comme un 'champ récepteur temporel', c'est-à-dire une fenêtre de temps pendant laquelle le système s'attend à recevoir un stimulus (un événement extérieur).

La Figure 1 propose une représentation graphique de la relation de dépendance entre o et ϕ pour ces deux fonctions et pour différentes valeurs de γ. Ce paramètre contrôle la largeur du champ récepteur temporel : plus γ est grand, plus l'attente est focalisée autour du temps t_x. Si un événement arrive à $t < t_x$, cet événement est en avance par rapport à l'attente de l'oscillateur ; inversement, si un événement arrive à $t > t_x$, il est en retard par rapport à cette attente. On peut écrire :

$$\phi(t) = \frac{t - t_x}{p}, \qquad t_x - \frac{p}{2} \leq t < t_x + \frac{p}{2} \tag{5}$$

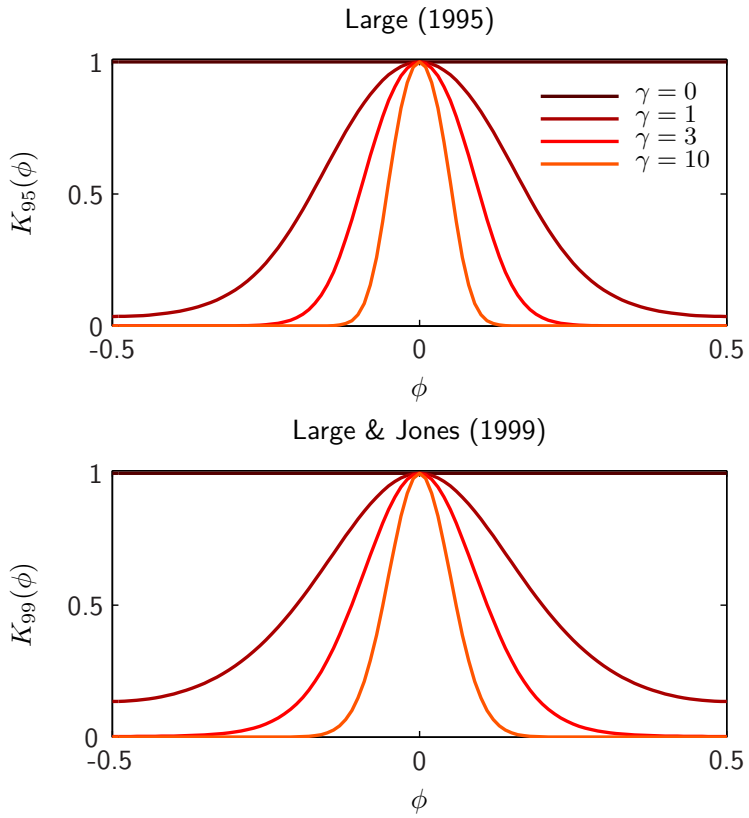

FIGURE 1 – Champ récepteur temporel de l'oscillateur en fonction de la phase, pour différentes valeurs de γ.

La correction à apporter à la phase lors de l'arrivée d'un événement dépend de la *pente* de la courbe Fig. 1 (c'est-à-dire de la dérivée de K) : nulle si l'événement coïncide avec la pulsation (maximum de la courbe en t_x), forte au voisinage de ce point, puis faible pour les points plus éloignés qui ne tombent pas à proximité de la pulsation.

Dans les implémentations présentées par la suite, l'algorithme est discrétisé avec un pas de temps δt (taux de rafraîchissement), fixé à 5 ms. En l'absence d'événement extérieur, la période ne change pas. D'après l'expression linéaire 5 ci-dessus, étant donnée une valeur $\phi(t)$ de la phase au temps t, la phase s'écrit au temps suivant $t + \delta t$:

$$\phi(t + \delta t) = \phi(t) + \frac{\delta t}{p} \tag{6}$$

dans le cas où t et $t + \delta t$ sont dans le même intervalle de longueur p centré autour de t_x. Dans le cas où $t + \delta t$ tombe dans l'intervalle suivant (centré en

$t_x + p$), un terme -1 est ajouté dans l'expression (6) donnant $\phi(t + \delta t)$.

Pour être capable de suivre les variations présentes dans la source musicale, l'oscillateur doit être capable d'adapter sa phase et sa période. Lorsqu'un événement arrive au temps t^*, la phase effectue un saut au temps $t^* + \delta t$, selon l'égalité (comme précédemment avec un terme -1 additionnel si t^* et $t^* + \delta t$ ne sont pas dans le même intervalle) :

$$\phi(t^* + \delta t) = \phi(t^*) + \Delta\phi \tag{7}$$

avec

$$\Delta\phi = \frac{\delta t}{p_{t^*}} - \eta_\phi \, F(\phi(t^*)) \tag{8}$$

où p_{t^*} désigne la valeur de la période au temps $t \leq t^*$; cette période est mise à jour pour $t = t^* + \delta t$ selon :

$$p_{t^*+\delta t} = p_{t^*} + \Delta p \tag{9}$$

où

$$\Delta p = \eta_p \, p_{t^*} \, F(\phi(t^*)) \tag{10}$$

Pour les dates ultérieures $t \geq t^* + \delta t$, et avant de rencontrer un nouvel élément, la phase est mise à jour selon l'équation (6) (absence de stimuli), mais la période considérée est évidemment la nouvelle période $p_{t^*+\delta t}$. La pente de la droite représentant ϕ est donc modifiée une fois effectué le saut défini par (7).

Les deux constantes η_p et η_ϕ représentent respectivement la force de couplage du suivi de la période et celle de la phase. $F(\phi)$ est une fonction qui quantifie la correction de la phase. Elle est proportionnelle à la dérivée par rapport à ϕ de la sortie $K(\phi)$ de l'oscillateur [8], de sorte que les changements s'effectuent dans le voisinage de la pulsation prédite (la date t_x). La taille de ce voisinage est alors contrôlée par le paramètre γ. Notons que la fonction F est du même signe que la phase ϕ, et que la courbe représentée Fig. 2 est celle de l'opposée $-F$. En effet, lorsque ϕ est négatif, $-F$ est positif, donc la phase est augmentée ; lorsque ϕ est positif, $-F$ est négatif, donc la phase

8. Pour chacun des modèles, cette dérivée a pour expression :

$$\frac{dK_{95}(\phi)}{d\phi} = -2\pi\gamma \, \mathrm{sech}^2 \left[\gamma(\cos(2\pi\phi) - 1)\right] \sin(2\pi\phi)$$

et

$$\frac{dK_{99}(\phi)}{d\phi} = -2\pi\gamma \, \frac{1}{\exp\gamma} \exp\left[\gamma \cos(2\pi\phi)\right] \sin(2\pi\phi)$$

est diminuée. Cela veut dire que si un événement arrive avant la date de la pulsation prédite, ϕ est négatif, donc le saut défini par $-F$ dans l'équation (8) est positif.

La différence essentielle entre les diverses versions du modèle de Large se situe dans la forme spécifique de cette fonction F (voir Fig. 2 pour une illustration numérique), qui dépend du choix de la fonction de sortie de l'oscillateur. Dans Large et Kolen (1994) et Large (1995), la correction de phase F s'écrit :

$$F_{95}(\phi(t)) = \frac{1}{2\pi}\,\mathrm{sech}^2\left[\gamma(\cos(2\pi\phi(t)) - 1)\right]\sin(2\pi\phi(t)) \tag{11}$$

Par la suite (Large et Jones, 1999; Large et Palmer, 2002), cette fonction F prend la forme suivante :

$$F_{99}(\phi(t)) = \frac{1}{2\pi\exp\gamma}\exp\left[\gamma\cos(2\pi\phi(t))\right]\sin(2\pi\phi(t)) \tag{12}$$

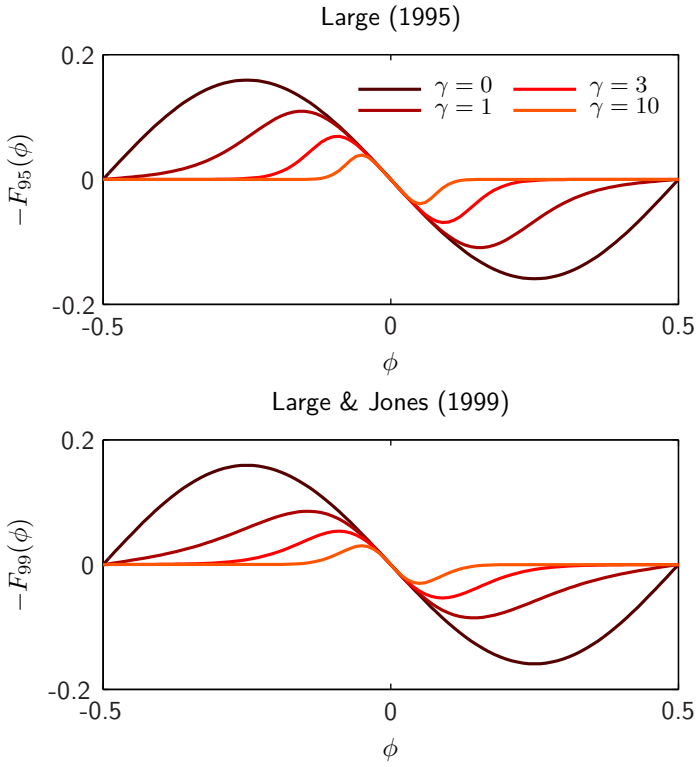

FIGURE 2 – Fonction de correction de phase, pour différentes valeurs de γ.

Ces deux fonctions ont la même valeur pour $\gamma = 0$ (égale à $1/2\pi \sin(2\pi\phi)$), et présentent, pour $\gamma > 0$, le même comportement qualitatif. Un événement qui arrive avant t_x ($\phi < 0$) diminue la période et augmente la phase, alors qu'un événement qui arrive après t_x entraîne une correction opposée. La largeur γ du champ récepteur contrôle le domaine d'application de ces modifications. Plus γ est grand, plus l'événement devra être proche de celui attendu pour affecter la phase et la période de l'oscillateur. Tous les événements qui tombent à l'extérieur du champ récepteur de l'oscillateur sont ignorés.

La Figure 3 propose une illustration simple du comportement du modèle. L'oscillateur part d'une période égale à 500 ms et reçoit une série d'événements régulièrement espacés de 400 ms. Chaque impulsion qu'il reçoit entraîne une modification de la phase et de la période. Au bout de quelques secondes, l'oscillateur se retrouve en phase avec le train d'événements, et délivre une pulsation dont la période est égale à celle de la source extérieure.

Il n'y a aucune raison a priori de préférer l'une ou l'autre fonction de correction de phase. Les deux ont donc été implémentées dans le cadre de ce projet, puis testées et comparées quantitativement afin de déterminer laquelle des deux donne les meilleurs résultats. Cette étude comparative est présentée dans la partie *Résultats et Discussion* (Section 4). À noter également que l'intensité des événements (vélocité des notes MIDI par exemple) peut être prise en compte dans le modèle (il suffit pour cela de multiplier le terme de droite des Éqs. 10 et 8 par un terme représentant l'intensité relative). Cette possibilité a été testée ici, mais, n'améliorant pas les résultats, elle a finalement été abandonnée (une autre motivation sera présentée en Section 4.1). Signalons enfin un aspect très intéressant du travail de Large, mais qui, après test, n'a finalement pas été retenu non plus : la possibilité de coupler plusieurs oscillateurs à des périodes différentes. Par exemple, dans le cas d'une division du temps binaire, on peut utiliser deux oscillateurs dont les périodes ont un rapport 1 : 2 entre elles ; dans le cas ternaire, ce rapport devient 1 : 3. Une particularité de ce modèle est qu'il ignore les événements qui 'tombent' hors du champ récepteur de l'oscillateur. Coupler plusieurs oscillateurs permet a priori d'avoir une meilleure stabilité grâce à une utilisation plus complète de l'information provenant des événements reçus. Le matériel qui nous intéresse ici, et que nous présenterons en Section 4.1, est issu d'improvisations jazz, un style souvent caractérisé par un rythme swing. Le phrasé swing est souvent noté par des croches, avec la mention 'swing feel', ou 'triple feel', qui indique qu'une succession de deux croches doit être interprétée comme la première et la troisième croche d'un triolet. En pratique, le swing n'est cependant ni binaire ni ternaire. Plusieurs travaux ont montré que le rapport de durées

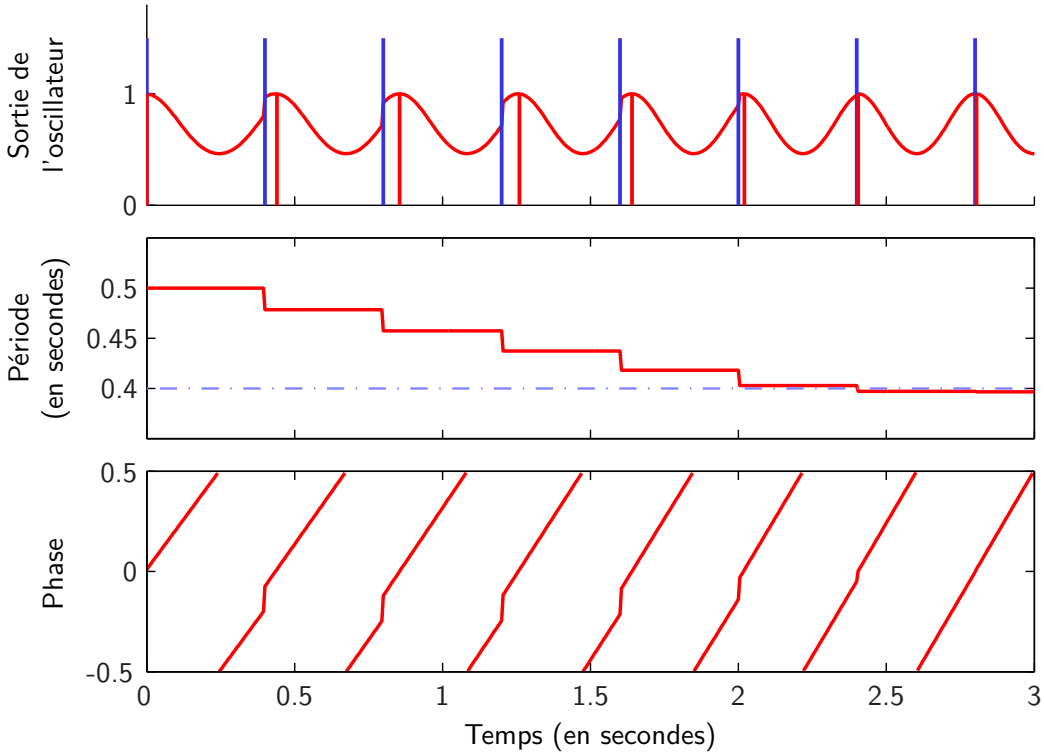

FIGURE 3 – Synchronisation d'un oscillateur avec une source externe (train d'événements régulièrement espacés, avec une période égale à 400 ms, représentés par les barres verticales bleues). La sortie de l'oscillateur est représentée par la courbe rouge ; les barres verticales rouges indiquent la pulsation générée par le modèle. Le modèle utilisé ici est celui de Large (1995) (cf. Éqs. (1) et (11), avec pour paramètres : $\gamma = 0.3$, $\eta_p = 0.3$ et $\eta_\phi = 0.8$). L'oscillateur, dont la période est initialisée à 500 ms, se synchronise en phase et en fréquence avec les événements.

entre deux notes consécutives exécutées dans un contexte swing dépend du musicien, du tempo, et peut varier, à l'intérieur du même morceau, entre musiciens et pour un même musicien (Cholakis, 1995; Friberg et Sundström, 2002). Le rapport n'étant pas un entier, coupler un oscillateur qui suit la pulsation avec d'autres oscillateurs à des niveaux sous-harmoniques (entiers) rend finalement le système instable, d'où l'abandon de cette possibilité et l'absence d'une présentation dans ce rapport des détails techniques.

3.3 Le modèle de Toiviainen

Le modèle proposé par Toiviainen (1998) se base sur l'oscillateur adaptatif développé par Large (1995). L'idée principale de l'auteur est de tenir compte de la durée des notes (en pratique approximée en considérant le temps entre attaques) dans l'adaptation des paramètres de l'oscillateur. Pour cela, l'ajustement des paramètres s'effectue dans le temps, a posteriori et graduellement, de sorte qu'une note plus longue a plus d'impact qu'une note plus courte, comme c'est généralement le cas chez l'homme (voir Section 2.1). La pulsation est alors préférentiellement portée par les notes les plus longues. Ceci permet par ailleurs d'éviter une situation qui peut poser problème au modèle de Large, à savoir une succession rapide de plusieurs notes (cas d'un trille par exemple) – voir Toiviainen (1998, Fig. 4).

Le modèle est ici présenté de façon à souligner les similitudes et les différences avec le modèle de Large, introduit à la section précédente. Dans le modèle de Large (1995), lorsqu'un événement est reçu au temps t^*, les modifications apportées à la phase et à la période sont instantanément prises en compte au pas de temps suivant. À cet instant $t = t^* + \delta t$, l'expression donnant la phase (7) et (8) peut être écrite de la façon suivante :

$$\phi(t) = \phi(t^*) + \epsilon(t) - \eta_\phi \, F_{95}(\phi(t^*)) \tag{13}$$

avec

$$\epsilon(t) = \frac{t - t^*}{p_{t*}} \tag{14}$$

où p_{t^*} désigne la valeur de la période au temps $t \leq t^*$. À cet instant $t = t^* + \delta t$, la période est également mise à jour selon (9) et (10) que l'on peut réécrire :

$$p_t = p_{t^*} \Big[1 - \eta_p \, F_{95}(\phi(t^*)) \Big] \tag{15}$$

De façon à conserver une valeur de la phase comprise entre -0.5 et 0.5, un terme -1 est ajouté dans l'expression (13) si nécessaire. Lorsque la phase est nulle ($\phi = 0$), le système génère une pulsation.

Dans le modèle de Toiviainen (1998), le mécanisme d'adaptation de la phase et de la période est similaire, la différence essentielle étant qu'au lieu d'appliquer les changements de façon instantanée, ces modifications sont faites de façon progressive. Ainsi, pour un temps $t > t^*$, et avant de rencontrer un nouvel événement, la phase s'écrit :

$$\phi(t) = \phi(t^*) + \epsilon(t) - 2\pi F_{95}(\phi(t^*))\Theta(\epsilon(t)) \tag{16}$$

où, comme précédemment, $\epsilon(t)$ est déterminé par l'équation (14), et où $F_{95}(\phi(t^*))$ quantifie l'erreur entre la pulsation prédite et l'arrivée de l'événement à t^*. $\Theta(\epsilon(t))$ est une fonction positive du temps qui permet d'effectuer les modifications de façon graduelle au cours du temps. Cette fonction, dont on donnera plus tard l'expression analytique, se base sur la fonction Γ, définie de la façon suivante :

$$\Gamma\left(\epsilon(t)\right) = \frac{\alpha^3}{2}\epsilon(t)^2 e^{-\alpha\epsilon(t)} \tag{17}$$

À noter que toutes ces fonctions dépendent de $\epsilon(t)$, qui donne le temps écoulé depuis l'instant t^*, normalisé par la période au temps t^*. Le paramètre α détermine la vitesse à laquelle l'adaptation a lieu. Plus α est grand, plus l'adaptation a lieu rapidement après la venue de l'événement. Autrement dit, plus α est petit, plus les notes longues auront un poids élevé par rapport à des notes courtes, qui au contraire n'auront que peu d'influence sur l'adaptation des paramètres. Une représentation graphique de la fonction Γ est proposée en Figure 4, pour différentes valeur de α.

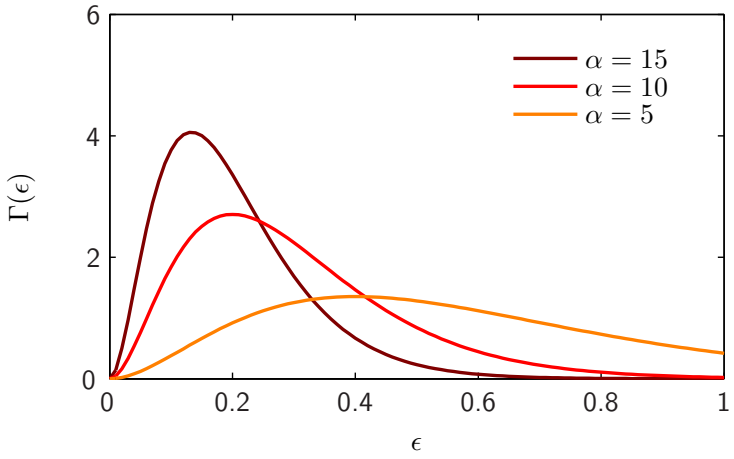

FIGURE 4 – Représentation de la fonction $\Gamma(\epsilon)$ pour différentes valeurs du paramètre α.

La période est mise à jour lorsqu'un nouvel événement arrive, au temps t^{**}. La modification apportée à la période dépend de la quantité $\epsilon(t^{**})$, autrement dit du temps passé depuis l'événement arrivé à t^* :

$$p_{t^{**}} = p_{t^*}\left[1 - 2\pi F_{95}(\phi(t^*))\,\theta(\epsilon(t^{**}))\right]^{-1} \tag{18}$$

où la fonction $\theta(\epsilon)$ s'écrit comme une combinaison linéaire de la fonction Γ

définie ci-dessus par l'équation (17) et d'une fonction G :

$$\theta(\epsilon) = \eta_s \Gamma(\epsilon) + \eta_l G(\epsilon) \tag{19}$$

Γ représente l'adaptation à court terme (rapide), et G l'adaptation à long terme (lente) définie par l'expression suivante (il s'agit en fait d'une intégrale de Γ) :

$$G(\epsilon(t)) = 1 - \left(\frac{\alpha^2}{2} \epsilon(t)^2 + \alpha \epsilon(t) + 1 \right) e^{-\alpha \epsilon(t)} \tag{20}$$

Ces fonctions sont choisies de sorte qu'après suffisamment de temps, la quantité d'adaptation ne dépend pas de la valeur de α :

$$\lim_{\epsilon \to \infty} G(\epsilon) = 1 \tag{21}$$

Les deux constantes η_s (s pour *short-term*) et η_l (l pour *long-term*) représentent ainsi la force de l'adaptation à court terme et à long terme, respectivement. Toiviainen (1998) montre que, dans le cadre de son modèle, ces deux composantes sont nécessaires au suivi de tempo.

L'équation (18) est très similaire au mécanisme d'adaptation présenté dans la section précédente (voir Eq. (15)). On peut remarquer dans le terme de droite le changement de signe et l'apparition de l'exposant -1 par rapport à l'équation décrivant l'adaptation de la période dans le modèle de Large. Ceci vient du fait que le modèle de Toiviainen (1998) considère la vitesse de la phase (dérivée de la phase par rapport au temps, inversement proportionnelle à la période). Dans le but de souligner de façon plus explicite le lien avec les équations décrivant le modèle de Large, la modification apportée à la période est ici présentée. Le lien entre les deux modèles apparaît plus clairement si l'on se rappelle que les quantités $(1-x)^{-1}$ et $(1+x)$ sont égales au premier ordre en x, pour x petit devant 1. Comme précédemment, si, à t^*, l'événement arrive avant la pulsation ($t^* < t_x$), l'erreur $-F_{95}(\phi(t^*))$ est positive : la période est donc diminuée. La différence majeure est que, plutôt que de mettre à jour la période immédiatement, on attend l'arrivée de l'événement suivant à t^{**}, de sorte que cette modification dépend de la durée entre deux événements. Comme annoncé, une note longue aura donc plus d'impact qu'une note brève.

La fonction $\Theta(\epsilon(t))$, qui intervient dans l'équation (16) donnant l'ajustement de la phase après réception d'un événement, est définie comme une combinaison linéaire de la fonction G et d'une autre fonction H (en fait, Θ

n'est autre que l'intégrale entre 0 et $\epsilon(t)$ de la fonction $\theta(\epsilon)$ [9]). Dans l'implémentation numérique, la forme analytique de $\Theta(\epsilon(t))$, présentée ci-dessous, est directement utilisée. On peut écrire :

$$\Theta(\epsilon(t)) = \eta_s G(\epsilon(t)) + \eta_l H(\epsilon(t)) \tag{22}$$

avec $G(\epsilon(t))$ définie par l'équation (20), et $H(\epsilon(t))$ par :

$$H(\epsilon(t)) = \epsilon(t) + \left(\frac{\alpha}{2}\epsilon(t)^2 + 2\epsilon(t) + \frac{3}{\alpha}\right) e^{-\alpha\epsilon(t)} - \frac{3}{\alpha} \tag{23}$$

Comme précédemment, de façon à conserver une valeur de la phase comprise entre -0.5 et 0.5, un terme -1 est ajouté dans l'expression (16) si nécessaire, et lorsque la phase est nulle ($\phi = 0$), le système génère une pulsation. En pratique, le temps est discrétisé par pas de temps de 5 ms.

3.4 Paramètres

Les deux variables principales du modèle de Large sont la phase et la période. Si ces quantités ont une interprétation évidente, il faut cependant souligner qu'il s'agit de variables internes. Lorsqu'une pièce musicale est interprétée ou improvisée par un musicien, celle-ci présente de nombreuses variations temporelles. Ces déviations peuvent provenir d'une intention expressive de la part du musicien, mais aussi être dues au bruit inhérent à l'exécution motrice. Considérer les variations de tempo au cours du temps à travers la mesure des intervalles entre pulsations peut alors induire en erreur quant à la variabilité réelle de la période interne. On s'attend à ce que cette dernière soit plus souple, plus lente à se modifier que la période apparente. Les variations globales de tempo doivent être suivies grâce à une adaptation relativement lente de la période, tandis que les 'micro-déviations', locales, sont traitées grâce à des variations de phase, plus rapides. Ceci est cohérent avec la littérature psychophysique sur le suivi de tempo chez l'homme : des travaux montrent en effet que la correction de la phase est rapide et subconsciente, alors que la correction de la période est lente et en grande partie consciente (Repp, 2005, pour une revue). On s'attend ainsi à choisir une valeur plus faible de la constante η_p par rapport à η_ϕ. De plus, grâce à un contrôle de ces valeurs, on peut obtenir un suivi plus ou moins souple de la pulsation, ce qui permet, comme on l'a vu en Section 3.1, de s'accorder aux préférences des auditeurs (*cf* les travaux de Dixon *et al.*, 2006). On peut

9. Ceci vient du fait que l'inverse de la période correspond à la vitesse de la phase, c'est-à-dire à la dérivée par rapport au temps de la phase. Voir Toiviainen (1998) pour plus de détails.

retrouver cette idée dans les propos du batteur Stéphane Galland : “S'il y a variation de tempo, elle doit se faire avec une certaine douceur, s'il y a un angle ce n'est pas une variation de tempo.” [10].

L'autre paramètre fondamental du modèle est la largeur γ du champ récepteur temporel. Lorsque γ est grand, le champ récepteur est étroit, c'est-à-dire l'attente est focalisée autour d'un point dans le temps, et seuls les événements proches de ce point sont pris en compte dans l'adaptation de la phase et de la période. Un champ récepteur large permet ainsi de s'adapter rapidement à des changements de tempo abrupts, mais entraîner une forte instabilité en présence d'un grand nombre d'événements qui ne tombent pas sur le temps. Inversement, un champ récepteur étroit permet de faire face à une rythmique complexe, en particulier à des syncopes, mais peut passer à côté d'une note importante (sur le temps), ce qui rompt le suivi de tempo. Aucune solution parfaite n'existe à ce problème : un compromis doit être réalisé en fonction du style de musique considéré.
Pour fixer les idées, considérons une série d'événements qui commence par des impulsions régulièrement espacées par un intervalle de 750 ms, puis continue par des impulsions séparés par une durée de 533 ms. Si l'on suppose que les premiers événements portent la pulsation, la suite peut être interprétée de deux façons : (1) une augmentation de tempo importante, passant de 80 bpm (‘battements par minute’) à 112.5 bpm, si ces événements sont considérés comme des noires ; (2) une faible diminution de tempo (de 80 bpm à 75 bpm), les événements étant alors perçus comme des triolets de noires. En fonction de la largeur de son champ récepteur, l'oscillateur converge vers l'une ou l'autre solution. Cette situation est illustrée par les Figures 5 et 6.
Signalons enfin que Large propose dans ses différents travaux un mécanisme d'adaptation de ce paramètre γ : le focus attentionnel augmente (*ie* γ croît) lorsque la synchronisation est bonne, et diminue (*ie* γ décroît) lorsqu'elle se dégrade. En pratique, dans le cadre qui nous intéresse, ce mécanisme entraîne une forte instabilité. En effet, lorsque la variabilité des événements augmente, la largeur du champ récepteur augmente (Large et Jones, 1999), ce qui, comme on l'a discuté, entraîne une perte du suivi de tempo dans le cas de rythmes complexes. Cette propriété n'a donc finalement pas été retenue ici.

10. Propos extraits de l'entretien de Stéphane Galland avec Gilbert Nouno (Nouno, 2008, p.131–141)

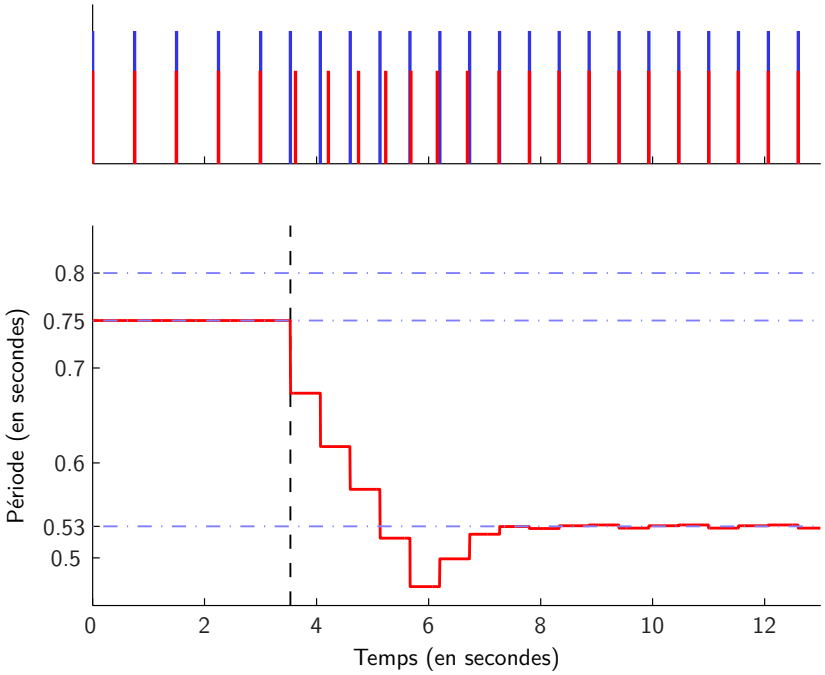

FIGURE 5 – Comportement de l'oscillateur face à une situation ambiguë : cas où le champ récepteur est large (*ie* γ petit). Les événements extérieurs sont représentés en bleu ; les barres rouges représentent la pulsation marquée par le modèle. Le pointillé noir indique le moment où la période des événements est modifiée. Le changement de tempo (ici une augmentation du bpm) est important, et les événements sont interprétés comme des noires. Le modèle utilisé ici est celui de Large (1995) ($\gamma = 0.2$, $\eta_p = 0.7$, $\eta_\phi = 1.0$).

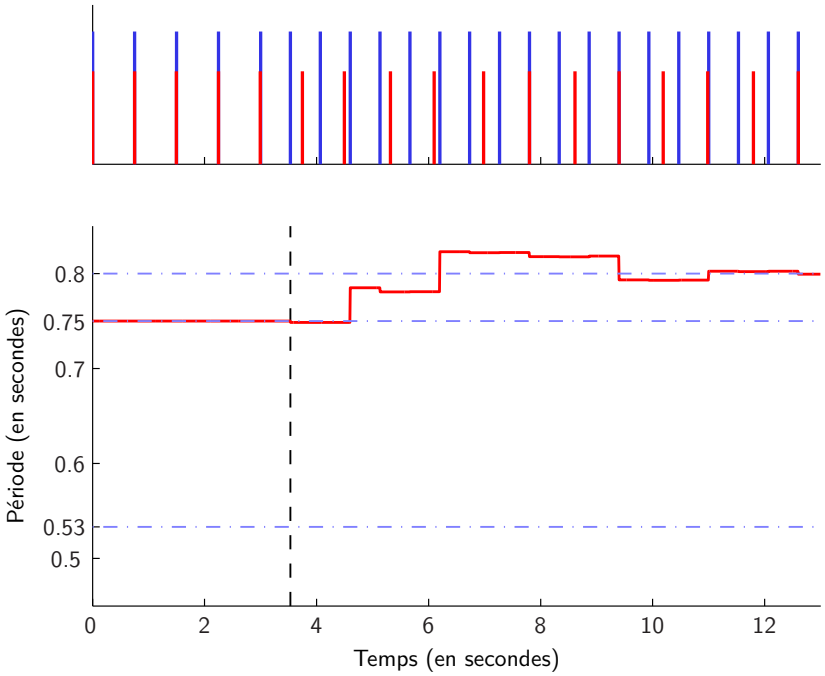

FIGURE 6 – Comportement de l'oscillateur face à une situation ambiguë : cas où le champ récepteur est étroit (*ie* γ grand). Les événements sont les mêmes que dans la figure précédente. Le changement de tempo (ici une diminution) est faible, et les événements sont interprétés comme des triolets de noires. Le modèle utilisé ici est celui de Large (1995) ($\gamma = 2.2$, $\eta_p = 0.7$, $\eta_\phi = 1.0$).

4 Résultats et discussion

4.1 Données MIDI

Quatre morceaux sont utilisés pour tester les modèles considérés. Ces données, enregistrées lors d'un concert à Uzeste en novembre 2007, ont été captées dans le format MIDI depuis le clavier de Bernard Lubat. Trois morceaux sont issus des albums *Chansons enjazzés* et *Scatrap jazzcogne* de la compagnie Lubat : 'St-Just Blues', 'J'aime pour la vie', et 'D'ici d'en bas' ; le quatrième est une reprise du célèbre standard de jazz 'Night in Tunisia', composé par Dizzy Gillespie. Dans le but d'évaluer les performances des algorithmes choisis ici, la pulsation a été extraite de chacun des morceaux par une battue en temps réel (seules les 40 premières secondes sont considérées). Cette battue a été effectuée 10 fois par 2 personnes différentes. Les données recueillies ont alors été moyennées pour obtenir l'emplacement moyen des pulsations. Vu le tempo élevé de 'Night in Tunisia', ce morceau a été battu à la blanche. Pour chaque extrait, le tempo moyen obtenu, exprimé en battements par minute (bpm), est le suivant :

- St-Just Blues : 64 bpm
- D'ici d'en bas : 99 bpm
- J'aime pour la vie : 88 bpm
- Night in Tunisia : 228 bpm

Pour plus de lisibilité, seuls les résultats obtenus sur 'St-Just Blues' sont détaillés par la suite. La Figure 7 présente un piano-roll de cet extrait, accompagné des pulsations (construites à l'aide de moyennes comme décrit précédemment). Une représentation graphique similaire est disponible pour les trois autres extraits en Annexe A, p.38.

Avant de présenter les résultats obtenus par les différents modèles, il est intéressant d'analyser l'information présente dans ces données MIDI, indépendamment d'un algorithme particulier. Comme on l'a vu en Section 2.1, deux informations sont a priori utiles pour trouver la pulsation : l'intensité d'une note (ici donnée par la vélocité MIDI) et sa durée (ici approximée par l'intervalle entre attaques). Pour ne pas fausser l'analyse, les événements associés à un intervalle entre attaques inférieur à 50 ms sont exclues (notes d'un accord). Ces événements ont été répartis en trois groupes, grâce à l'algorithme des k-moyennes ("k-means clustering algorithm") : ceux qui arrivent avant la pulsation ($\phi < 0$), ceux qui se situent autour de la pulsation ($\phi \approx 0$), et enfin ceux qui tombent après la pulsation ($\phi > 0$). La Figure 8 (Gauche) présente la vélocité moyenne pour chacun de ces groupes. Contrairement à ce

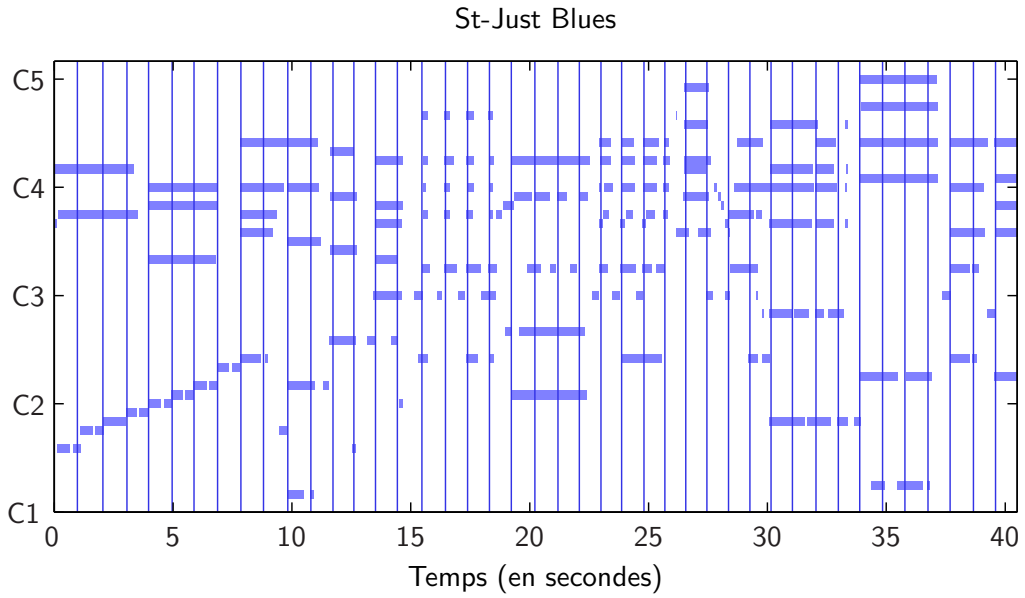

FIGURE 7 – Piano-roll des 40 premières secondes de 'St-Just Blues'. Les barres verticales indiquent l'emplacement des pulsations (construites à l'aide de moyennes comme décrit dans le texte).

à quoi on pourrait s'attendre, la vélocité des événements présents autour de la pulsation est plus faible que celle des autres événements (test de Student : p=0.037). Ceci explique le constat effectué en section 3.2 : la prise en compte de la vélocité ne permet pas de mieux suivre la pulsation. En revanche, comme on peut le voir clairement sur la Figure 8 (Droite), l'intervalle entre attaques est significativement plus grand (p=4.7e-6) pour $\phi \approx 0$ que pour $\phi \neq 0$. Ce résultat motive l'utilisation de l'algorithme de Toiviainen (1998), qui permet de prendre en compte la durée des événements dans le suivi de la pulsation (voir section 3.3).

4.2 Optimisation des paramètres

L'objectif de cette section est double : il s'agit d'abord d'optimiser les paramètres de chacun des modèles pour obtenir les meilleures performances possibles sur les quatre morceaux présentés ; ceci permet en retour de comparer les algorithmes entre eux. L'évaluation des algorithmes s'effectue ici grâce à une comparaison entre la pulsation délivrée par un modèle et la frappe humaine (présentée dans la section précédente). Plusieurs méthodes existent pour évaluer les performances d'un algorithme de suivi de tempo, et cette question n'est pas simple (voir par exemple Goto et Muraoka, 1997;

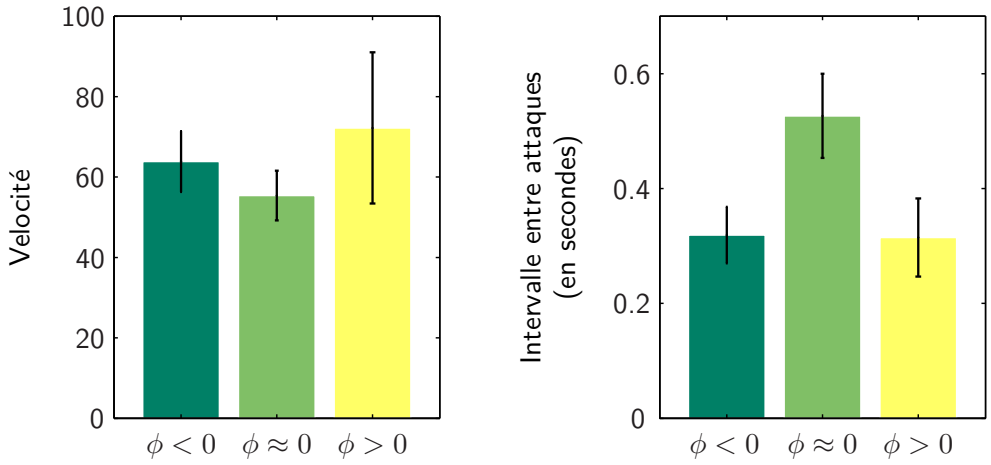

FIGURE 8 – (Gauche) Intensité (vélocité MIDI) d'un événement en fonction de sa position par rapport à la pulsation. (Droite) Durée (approximée par l'intervalle entre attaques) d'un événement en fonction de sa position par rapport à la pulsation. Dans les deux cas, les barres d'erreurs indiquent l'intervalle de confiance à 95%.

Davies *et al.*, 2009). Parmi les nombreuses mesures existantes, trois ont été ici retenues : la F-Mesure, la précision de Cemgil, et la mesure de continuité de Klapuri (voir Davies *et al.*, 2009, pour les détails mathématiques). La F-Mesure est souvent utilisée dans l'évaluation de recherche d'information. Elle combine la précision (proportion des pulsations générées qui sont correctes) et le rappel (proportion des pulsations correctes par rapport au nombre total de pulsations). Une détection est considérée correcte si la pulsation estimée se situe à moins de 50 ms de l'annotation humaine. La précision de Cemgil se base sur une fonction d'erreur gaussienne qui, plutôt que de porter un jugement binaire correct/incorrect, quantifie la performance d'une estimation en mesurant la distance entre celle-ci et la 'vraie' pulsation la plus proche. Enfin, la mesure de continuité de Klapuri donne le rapport entre la longueur de la plus longue série de pulsations correctes et la longueur totale. Ainsi, si une seule erreur se produit au milieu de la séquence, on obtient une valeur de 50%. Cette mesure est intéressante dans le présent contexte car elle impose à l'algorithme d'être capable de délivrer une pulsation correcte pendant une durée la plus longue possible. Comme on l'a vu, si à un moment le suivi de tempo décroche, l'utilisateur peut toujours recaler le système. Plus ce dernier est capable de suivre la pulsation sur une longue durée, plus l'utilisateur est libre d'utiliser ce temps pour contrôler d'autres aspects de l'interaction avec le musicien (la génération de nouvelles phrases par exemple).
Ces trois mesures de la qualité du suivi de la pulsation donnent un nombre

entre 0 et 100.

Pour un modèle donné, chaque mesure a été utilisée pour optimiser les paramètres du modèle. Dans le but d'éviter un surapprentissage (une situation pour laquelle les paramètres choisis ne fonctionnent que sur le morceau servant de base à l'optimisation), la fonction de coût utilisée pour cette optimisation se base sur la moyenne du critère d'évaluation calculé sur les quatre morceaux. Ceci permet d'obtenir des paramètres robustes qui donnent de bons résultats indépendamment d'un morceau précis. Cette optimisation a été effectuée sous MATLAB grâce à un algorithme de recuit simulé (Kirkpatrick *et al.*, 1983). À chaque fois, le modèle est initialisé de la façon suivante : à $t = 0$, la période est égale à la période moyenne des quatre premières pulsations, et la phase est telle que l'attente du modèle corresponde à la première pulsation annotée. La Figure 9 présente les résultats obtenus (moyenne sur les quatre extraits utilisés), et la Table 1 détaille les résultats pour chacun des morceaux. De façon à avoir des points de comparaison, deux cas extrêmes sont également présentés. Le premier donne la performance moyenne, pour chaque mesure, des annotations individuelles humaines par rapport à la pulsation moyenne utilisée comme référence. Le second cas considère une pulsation délivrée de façon métronomique, dont la phase et la période initiales sont identiques à la situation dans laquelle les modèles sont testés.

Le premier constat que l'on peut faire est que les trois algorithmes obtiennent en moyenne de bonnes performances, et celles-ci sont à peu près équivalentes entre ces modèles, même si le modèle de Large (1995) est en général légèrement meilleur que les autres. Si l'on regarde le détail des résultats pour chaque morceau (Table 1), on peut voir que les performances sont quasi parfaites pour trois des quatre morceaux : pour un même jeu de paramètres, chaque modèle est capable de suivre la pulsation de façon correcte sur l'ensemble de l'extrait considéré. Les trois modèles ont par contre des difficultés avec l'extrait de "D'ici d'en bas". Ceci est dû à un passage particulièrement chargé (en termes d'événements), qui correspond à un solo de la main droite se cumulant avec l'accompagnement main gauche (voir piano-roll Fig.A, p.38).

On peut dégager de cette optimisation les paramètres suivants :

- Large (1995) : $\gamma = 5.2$, $\eta_p = 0.11$, et $\eta_\phi = 0.50$
- Large (1999) : $\gamma = 7.1$, $\eta_p = 0.15$, et $\eta_\phi = 0.53$
- Toiviainen (1998) : $\gamma = 2.2$, $\alpha = 7.4$, $\eta_l = 0.08$, et $\eta_s = 0.23$

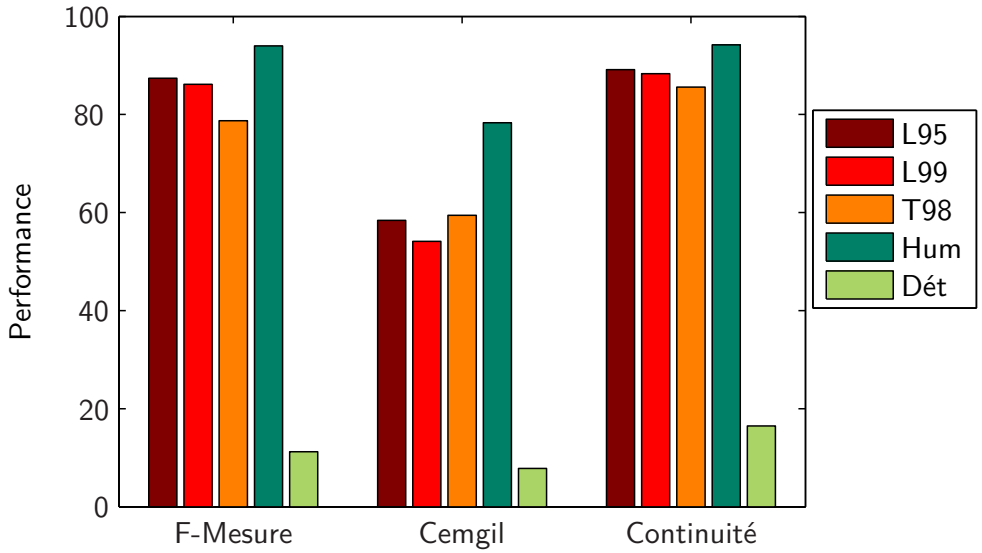

FIGURE 9 – Résultats de l'optimisation, pour chacune des trois mesures de performance choisies. Les barres de couleurs chaudes correspondent aux performances des trois modèles testés (L95 : Large, 1995 ; L99 : Large, 1999 ; T98 : Toiviainen, 1998). 'Hum' désigne les performances de chaque essai humain par rapport à la moyenne, et 'Dét' indique les performances obtenues par une pulsation déterministe, métronomique, calée sur la première pulsation annotée, et de période égale à la période initiale des modèles.

	F-Mesure				Cemgil				Continuité			
	sjb	dici	jm	nit	sjb	dici	jm	nit	sjb	dici	jm	nit
L95	97.4	40.3	97.1	96.2	70.7	30.3	69.3	52.0	100	60	98.1	98.5
L99	97.4	45.4	95.1	93.2	69.3	19.7	72.1	55.0	100	56.7	98.1	98.5
T98	71.8	65.5	87.4	90.2	41.0	29.6	82.5	80.8	97.5	48.3	98.1	98.5
hum	96.7	92.2	88.9	98.1	80.5	75.8	72.0	85.0	99.2	90.2	89.8	97.6
dét	18.4	17.1	7.7	1.5	11.5	11.6	6.2	1.9	15.4	8.3	36.5	5.9

TABLE 1 – Détails des résultats de l'optimisation, pour chacune des trois mesures de performance choisies, et pour chacun des morceaux (sjb : St-Just Blues ; dici : D'ici d'en bas ; jm : J'aime pour la vie ; nit : Night in Tunisia). L'optimisation a été effectuée sur la moyenne des performances sur les quatre extraits.

Comme on pouvait s'y attendre, les paramètres optimaux des deux versions du modèle de Large sont proches. La différence principale porte sur le paramètre γ : celui-ci est plus grand pour la version la plus récente du modèle, ce qui s'explique par la forme du champ récepteur temporel de ces deux versions (voir Fig. 1). Pour ne pas être perturbé par les événements qui tombent loin de la pulsation, γ doit être plus grand dans la version 99 par rapport à la version 95.
Dans le cas du modèle de Toiviainen (1998), le champ récepteur est plus large (*ie* γ est plus petit) que pour le modèle de Large. Ceci est dû au mécanisme d'adaptation propre au modèle de Toivianen. Les événements de courte durée ayant peu d'influence sur l'adaptation de l'oscillateur, le champ récepteur n'a pas besoin d'être aussi étroitement focalisé autour de la pulsation attendue pour que le système conserve sa stabilité.

4.3 Implémentation en Max/Msp

Dans le but d'une utilisation en temps réel et d'une intégration avec OMax, les modèles de Large et de Toiviainen ont été implémentés dans l'environnement Max. Des deux versions du modèle de Large, la version la plus ancienne (Large, 1995) a été retenue, au vu les meilleures performances qu'elle a obtenues (voir Section précédente). L'annexe B fournit une documentation détaillée de l'objet implémenté. Celui-ci possède deux entrées. La première reçoit principalement des bang correspondant aux événements extérieurs (issus de données MIDI, ou extraits d'un signal audio grâce à un algorithme de détection des attaques). On peut arrêter le suivi de tempo en envoyant la commande `stop` dans cette entrée. Celle-ci permet également de changer les paramètres de l'objet depuis l'extérieur, grâce à l'envoi d'un message contenant le nom de l'algorithme suivi des paramètres souhaités. Dans le cas de l'algorithme de Large, ces paramètres sont, dans l'ordre, γ, η_ϕ, et η_p ; pour Toiviainen, γ, α, η_l, et η_s. Par exemple, l'envoi du message `large 3 0.5 0.1` permet à l'objet d'utiliser le modèle de Large, avec pour paramètres $\gamma = 3$, $\eta_\phi = 0.5$, et $\eta_p = 0.1$. La deuxième entrée reçoit des 'bang', qui permettent à l'utilisateur de contrôler la phase et la période de l'objet. Ce 'bang' peut être généré suite à une frappe de la barre espace par exemple. Les deux premières frappes permettent de démarrer le suivi de tempo. Chaque fois que l'objet reçoit un bang dans cette entrée, la phase est remise à 0. La durée entre deux frappes consécutives détermine la période de l'oscillateur (sauf si cette durée est inférieure à 200 ms ou supérieure 2000 ms ; autrement dit, le tempo doit se situer entre 30 bpm et 300 bpm).

En sortie, l'objet envoie un bang pour indiquer la pulsation.

La Figure 18 (Annexe B, p. 40) présente l'interface graphique développée en Max pour contrôler l'objet de suivi de tempo. Cette interface permet de choisir l'algorithme utilisé, et d'ajuster les paramètres. Elle permet également de faire entendre un son lorsqu'une pulsation est générée. L'utilisateur peut contrôler le volume sonore de ce clic, ainsi que le type de son utilisé. La pulsation est également marquée visuellement grâce à un disque de couleur qui flashe au moment de la pulsation. Par défaut, les algorithmes utilisent les paramètres présentés à la section précédente (Section 4.2).

En pratique, l'objet fonctionne de façon satisfaisante sur les quatre morceaux MIDI testés ici, en particulier lorsqu'on utilise l'algorithme de Large. À noter qu'à l'utilisation, le système fonctionne effectivement en temps réel, et consomme très peu de ressources processeur. À l'écoute, aucune latence n'est perceptible.
La Figure 10 présente un extrait de quelques secondes de 'St-Just Blues', annoté de la pulsation présentée en Section 4.1 (voir Fig. 7) et de la pulsation délivrée par l'objet max (utilisant l'algorithme de Large). Seules les deux premières pulsations ont été données à la machine (par deux frappes consécutives de la barre espace). La pulsation générée par le système est très proche de la battue humaine. Pour ce même essai, la figure suivante (Fig. 11) compare les variations au cours du temps de l'intervalle entre pulsations dans le cas humain et dans le cas de la machine (quantité inversement proportionnelle au tempo). Les variations de tempo sont effectivement suivies par le système. Enfin, toujours pour ce même extrait, la Figure 12 donne l'histogramme de l'écart entre chaque pulsation générée par l'objet max et la pulsation battue. On peut voir que le système génère une pulsation en général légèrement en avance (erreur négative) par rapport à la pulsation battue. Ceci est peut-être dû à une latence lors de la captation de la frappe humaine. L'erreur moyenne est d'environ 30 ms, ce qui est faible, au sens où elle est du même ordre de grandeur que le seuil différentiel pour percevoir un changement de tempo (entre 2,5 et 5% de la période, soit ici entre 24 et 47 ms ; voir Friberg et Sundberg, 1995).

Dans l'ensemble, le suivi de tempo est assuré pendant plusieurs dizaines de secondes voire quelques minutes. Il arrive que le suivi manque une ou deux pulsations, suite par exemple à une syncope appuyée, mais il se recale souvent de lui-même. L'utilisateur peut bien sûr agir sur le système en donnant l'emplacement d'une ou plusieurs pulsations. Plus rarement, le suivi 'décroche' complètement. Ceci arrive sur des passages particulièrement char-

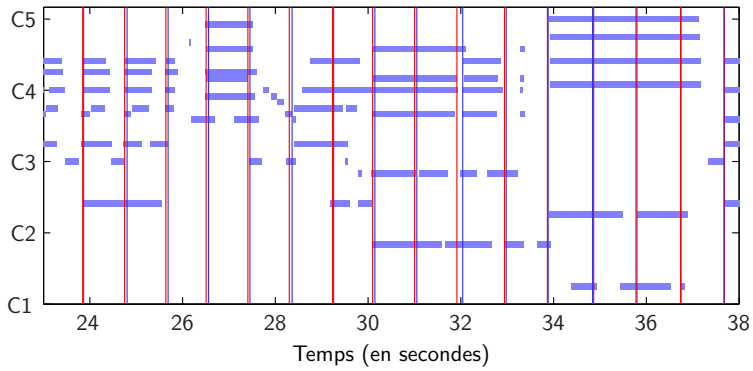

FIGURE 10 – Piano-roll d'un extrait de quelques secondes de 'St-Just Blues', accompagné de la pulsation battue par l'homme (barres verticales bleues) et de celle délivrée en temps réel par notre objet Max (en rouge).

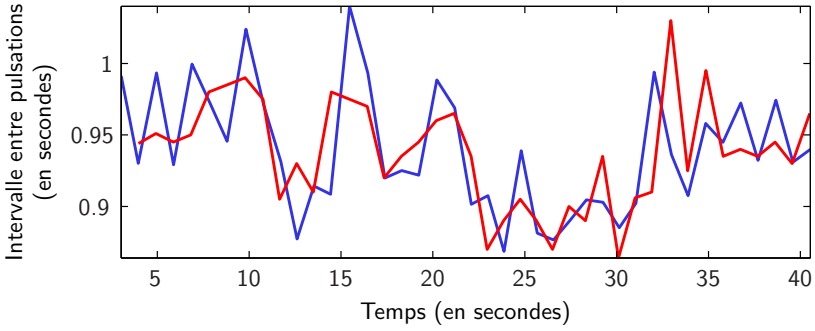

FIGURE 11 – Durée entre deux pulsations consécutives (intervalle entre pulsations) dans le cas de la battue humaine (en bleu) et dans le cas de notre objet Max (en rouge), pour le morceau 'St-Just Blues'.

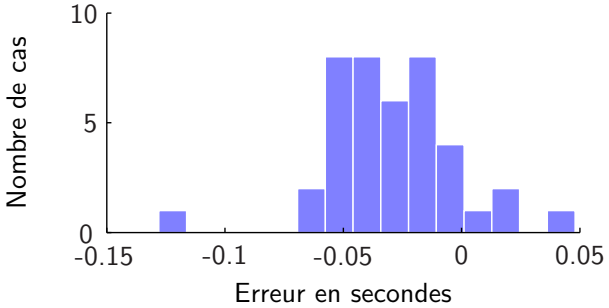

FIGURE 12 – Histogramme de l'écart, en secondes, entre la pulsation délivrée par notre objet Max et la battue humaine, prise comme référence (sur 'St-Just Blues').

gés, lorsque le musicien se lance dans un solo main droite tout en assurant l'accompagnement main gauche. Dans le cadre où la captation de la pulsation sert à générer des phrases rythmiquement en place avec la pulsation, ce type de cas ne devrait pas poser de problème a priori, puisque le musicien, engagé dans son solo, a la parole. Les moments où ce dernier accompagne sont plus faciles à suivre, et c'est justement ce type de moment où OMax peut se mettre à improviser. Cependant, il serait profitable d'avoir un suivi le plus stable et fiable possible, y compris lorsque le musicien improvise un solo. Ceci permettrait, comme on le discutera dans la dernière partie, de segmenter les phrases captées en fonction de la pulsation, pour pouvoir ensuite les recombiner différemment tout en tenant compte de la pulsation. D'où l'idée de suivre le rythme à partir du jeu de la batterie, assise rythmique d'une formation jazz typique, ce qui pourrait permettre non seulement de générer des phrases en rythme, mais également de capter le jeu d'un ou plusieurs musiciens accompagné de l'information de la pulsation.

4.4 Tests sur des enregistrements audio de batterie

Bien que les données utilisées jusqu'à présent soient sous format MIDI, l'objet de suivi de tempo développé n'a en fait besoin que des temps d'arrivée d'événements. On peut donc envisager de l'utiliser sur du signal audio, capté en temps réel lors d'une performance musicale. Il suffit pour cela de placer un algorithme de détection des attaques entre le signal audio et l'objet de suivi de tempo. Différentes méthodes existent pour détecter les attaques dans un signal audio (voir Bello, 2003, pour une revue). Compte tenu des brefs délais impartis par le présent projet, nous avons directement utilisé l'objet Max/Msp `bonk~`, développé par Miller Puckette pour PureData (voir Puckette *et al.*, 1998), et porté sur Msp par Ted Apel et Barry Threw [11]. Cet objet prend en entrée un signal audio et envoie au fur et à mesure de la lecture un 'bang' chaque fois qu'une attaque est détectée.

Le matériel audio utilisé ici est constitué de trois accompagnements de batterie, appelés ici 'J'aime pour la vie', 'Night in Tunisia', et 'D'ici d'en bas'. Les deux premiers viennent d'une session de travail avec Bernard Lubat à Uzeste en mai 2005. Le dernier, également joué par Lubat, est issu d'une maquette de ses chansons enjazzées datant de juillet 2003.

Les tests suivants ont été effectués en utilisant les paramètres par défaut (définis Section 4.2). Sur 'J'aime pour la vie' et 'Night in Tunisia' (ce dernier,

11. Objet disponible à l'adresse : `http://crca.ucsd.edu/~tapel/software.html`

au tempo particulièrement rapide, est battu à la blanche), le suivi est particulièrement bon, puisqu'en donnant seulement les premières pulsations, le système est capable de générer une pulsation en place sur l'ensemble de ces deux morceaux. Ici, l'algorithme de Toiviainen (1998) semble le plus efficace, mais aucun test quantitatif n'a été effectué. Le Figure 13 présente la pulsation générée par le système (configuré avec l'algorithme de Toiviainen) sur le morceau 'J'aime pour la vie', en relation avec le signal audio et la sortie de l'objet `bonk~` (attaques détectées). On peut voir que les notes sont plus longues sur les temps, ce qui peut expliquer la réussite de ce modèle dans ce cas-là.
L'accompagnement 'D'ici d'en bas' est plus difficile à suivre. Le suivi est tout de même satisfaisant en utilisant l'algorithme de Large. En revanche, le modèle de Toiviainen ne réussit pas à détecter convenablement la pulsation. La Figure 14 montre la pulsation générée (en utilisant l'algorithme de Large) et les attaques détectées. On peut voir la raison pour laquelle l'algorithme de Toiviainen échoue ici : la pulsation démarre sur un roulement, c'est-à-dire un enchaînement d'attaques faiblement espacées dans le temps. Étant donné que dans le modèle de Toiviainen la pulsation est préférentiellement portée par les notes longues, le système tend alors à se désynchroniser, ou à battre la mesure à contretemps.

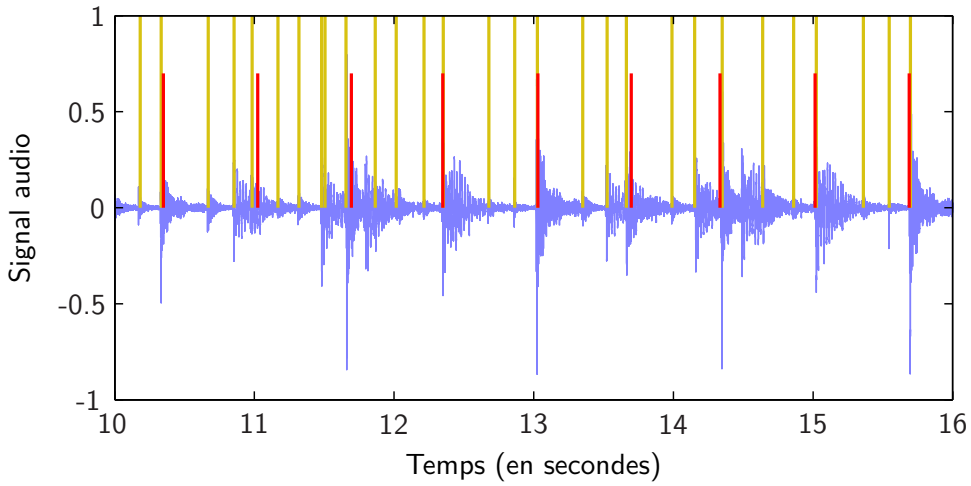

FIGURE 13 – Signal audio (en bleu) d'un extrait de l'accompagnement de batterie 'J'aime pour la vie'. En jaune les événements détectés par l'objet `bonk~`, en rouge la pulsation délivrée par notre objet de suivi de tempo. L'algorithme utilisé est celui de Toiviainen (1998).

Il faut noter que les parties de batterie utilisées sont assez complexes.

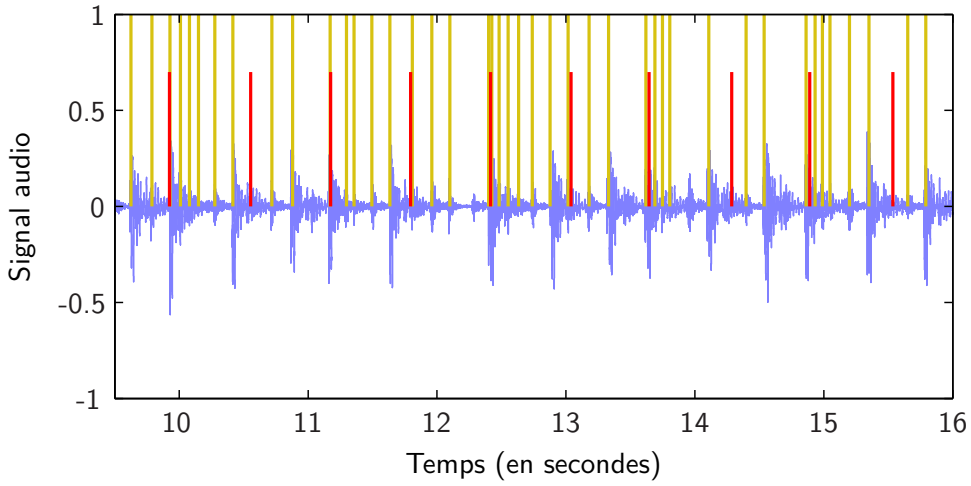

FIGURE 14 – Signal audio (en bleu) d'un extrait de l'accompagnement de batterie 'D'ici d'en bas'. En jaune les événements détectés par l'objet `bonk~`, en rouge la pulsation délivrée par notre objet de suivi de tempo. L'algorithme utilisé est celui de Large (1995).

Même si le système s'en sort bien, on pourrait encore gagner en stabilité grâce à un algorithme de détection des attaques qui filtre certains types de frappes, comme la caisse claire, dont le comportement, souvent syncopé, peut rendre le suivi plus difficile. Plus simplement, en situation de concert, un micro pourra directement capter les frappes sur la grosse caisse et/ou la charleston par exemple, contrairement aux enregistrements utilisés ici où toute la partie de batterie est utilisée.

5 Conclusion

Comme on l'a vu, le mode 'beat' d'OMax ne pouvait fonctionner qu'en imposant aux musiciens sa propre pulsation, délivrée à la façon d'un métronome. Cette utilisation d'OMax, trop contraignante, a du coup été mise de côté depuis quelques années. L'objectif de ce projet était de relancer ces travaux en développant un algorithme capable de détecter la pulsation dans une source musicale. Le contexte musical considéré (improvisations jazz) impose plusieurs contraintes : l'algorithme choisi doit être causal, utilisable en temps réel, et capable de suivre des variations de tempo. Il doit par ailleurs faire face à des rythmes complexes, et assurer une continuité en l'absence d'entrée. Une revue de la littérature sur les algorithmes de suivi de tempo a naturellement conduit à considérer les modèles à oscillateurs adaptatifs (Large, 1995; Large

et Jones, 1999; Toiviainen, 1998), qui, en plus de répondre aux exigences de notre projet, ont l'avantage d'être relativement simple à comprendre et de présenter une certaine plausibilité.

Les premiers tests, effectués sous MATLAB, ont montré que ces modèles à oscillateurs arrivent effectivement à détecter la pulsation des extraits musicaux considérés ici. Ces tests ont été réalisés sur du matériel MIDI capté sur scène directement depuis le clavier de Bernard Lubat. La pulsation délivrée par la machine a été comparée à la pulsation battue par l'homme. Les performances obtenues sont très satisfaisantes. Dans le but d'une utilisation en temps réel et d'une intégration avec OMax, nous avons alors implémenté les algorithmes de Large (1995) et Toiviainen (1998) dans l'environnement de programmation Max. En pratique, l'objet fonctionne effectivement en temps réel, sans latence perceptible. L'utilisateur peut agir sur cet objet en lui indiquant les premières pulsations grâce à une battue. Une fois lancé, l'objet Max développé détecte effectivement la pulsation, et est capable de suivre les variations de tempo de façon similaire à l'homme. En cas de décrochage, l'utilisateur peut à tout moment intervenir pour recaler le suivi.
En plaçant un algorithme de détection des attaques (l'objet `bonk~` développé par Puckette *et al.*, 1998) entre un signal audio et notre objet de suivi tempo, nous avons également pu tester ce dernier sur des enregistrements de batterie. Là encore, la pulsation est délivrée en temps réel de façon tout à fait satisfaisante.

Ces résultats positifs permettent d'envisager l'intégration de cet objet de suivi de tempo dans le logiciel OMax. Les performances particulièrement bonnes obtenues sur les enregistrements de batterie dessinent la place que ce suivi de tempo pourra prendre dans l'improvisation avec la machine en situation de scène. Un ou plusieurs microphones placés sur la batterie enregistrent les frappes sur la grosse caisse et/ou la charleston. Le signal audio ainsi capté est utilisé pour détecter la pulsation, grâce à notre objet. Le système enregistre également le jeu d'autres musiciens, un clavier par exemple. La pulsation détectée peut alors servir de référence pour annoter le jeu du musicien que l'on cherche à modéliser. Grâce à cette information, OMax peut alors construire de nouvelles phrases liées à une pulsation. Lors de la génération, la pulsation détectée en temps réel permet à la machine de développer un solo en place rythmiquement avec les autres musiciens, même si le tempo a changé. Une telle utilisation d'OMax, couplée à des connaissances harmoniques, ouvre ainsi la voie à une interaction musicale inédite, expressive et créative entre l'homme et la machine.

A Piano-rolls

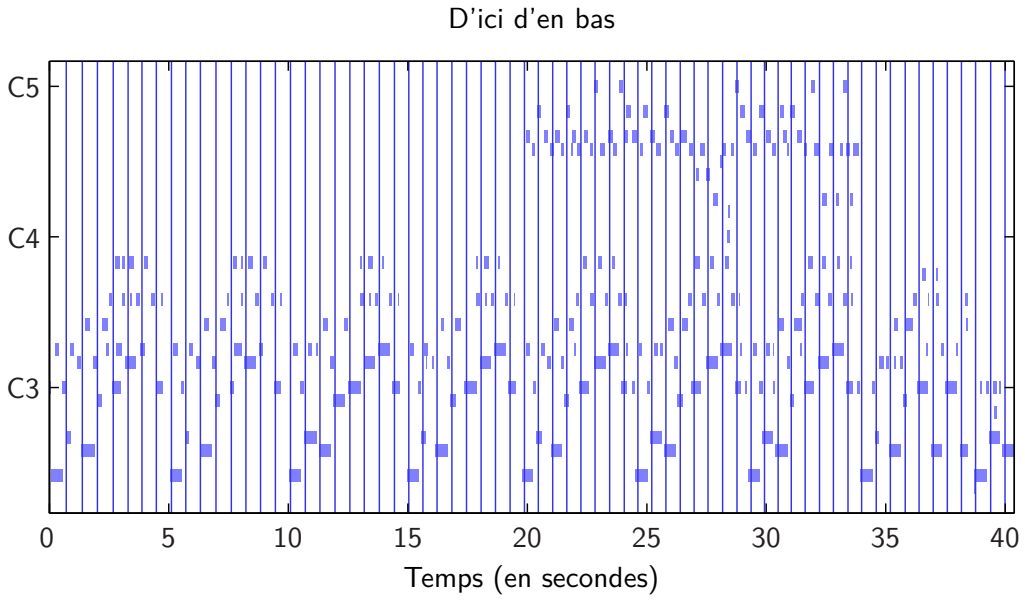

FIGURE 15 – Piano-roll et pulsation des 40 premières secondes de "D'ici d'en bas"

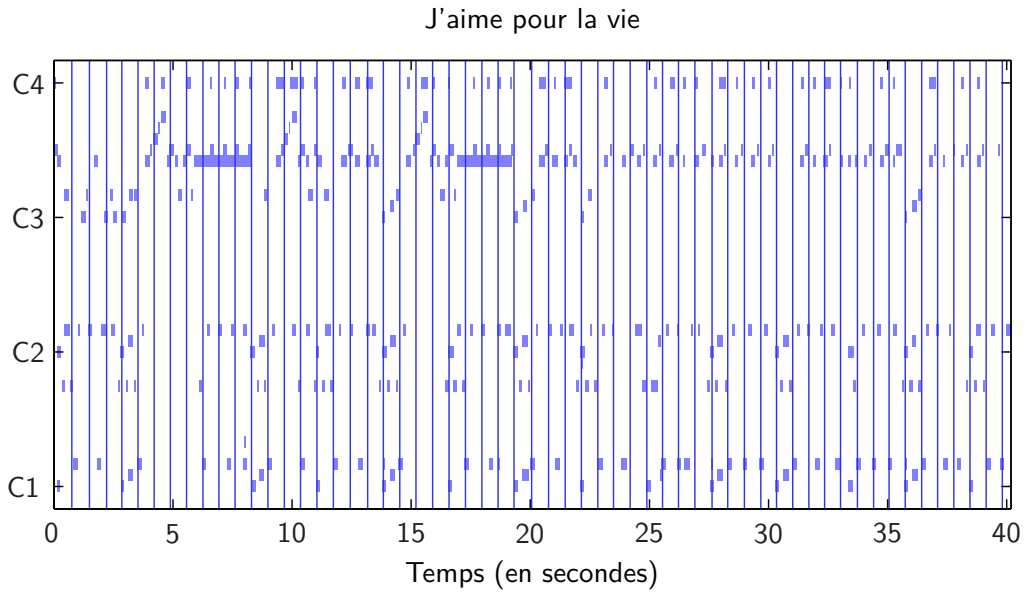

FIGURE 16 – Piano-roll et pulsation des 40 premières secondes de "J'aime pour la vie"

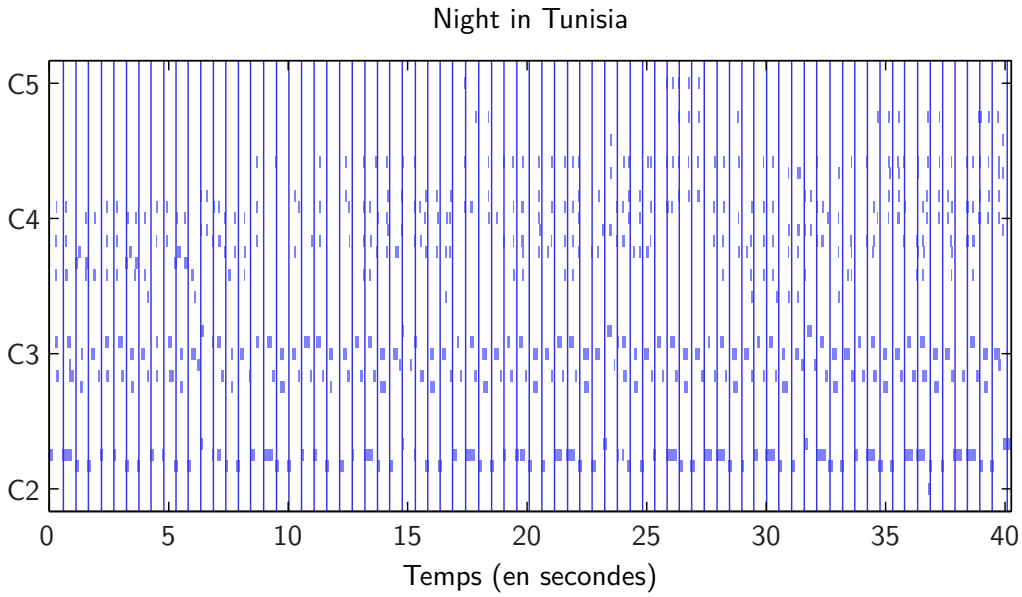

FIGURE 17 – Piano-roll et pulsation des 40 premières secondes de “Night in Tunisia”

B Documentation de l'objet MAX/MSP

beat-tracker
Détecte en temps réel la pulsation d'une source musicale

Description
L'objet `beat-tracker` permet de détecter la pulsation présente dans une source musicale. Il reçoit dans son entrée de gauche des événements sous la forme de bang. Chaque événement peut être issu de données MIDI ou extrait d'un signal audio grâce à un algorithme de détection des attaques. En sortie, l'objet envoie un bang en temps réel pour indiquer la pulsation. Pour réaliser ce suivi de tempo, l'objet utilise un oscillateur adaptatif capable de modifier ses paramètres pour rester synchroniser en phase et en fréquence avec la source musicale. L'utilisateur peut choisir entre deux mécanismes d'adaptation différents : celui développé par Large (1995), et celui proposé par Toiviainen (1998). Ce choix s'effectue grâce à l'envoi d'une commande dans la première entrée indiquant le type d'algorithme à utiliser avec ses paramètres. Le suivi de tempo peut être arrêté à tout moment en envoyant la commande `stop` dans la première entrée.

La deuxième entrée permet à l'utilisateur d'amorcer (ou de resynchroniser) le suivi de tempo en envoyant un ou plusieurs bang. Chaque bang reçu met la phase à 0. La durée entre deux bang consécutifs détermine la période du modèle. Seules les durées comprises entre 20 ms et 2000 ms sont acceptées (soit un tempo compris entre 30 bpm et 300 bpm).

Messages

Nom	*Type*	*Opt*	*Description*
`bang`			Chaque `bang` envoyé dans l'entrée gauche est interprété comme un événement.
`stop`			La commande `stop` envoyée dans la première entrée permet d'arrêter le suivi de tempo.
`large`	algorithme et ses paramètres (3) [list]		La commande `large` suivie des paramètres du modèle indique à l'objet qu'il doit utiliser l'algorithme de Large (1995). Les paramètres doivent être présentés dans l'ordre suivant : largeur γ du champ récepteur, force de couplage η_ϕ du suivi de la phase, et force de couplage η_p du suivi de la période. Par défaut, l'objet utilise `large 5.2 0.50 0.11`
`toiviainen`	algorithme et ses paramètres (4) [list]		La commande `toiviainen` suivie des paramètres du modèle indique à l'objet qu'il doit utiliser l'algorithme de Toiviainen (1998). Les paramètres doivent être présentés dans l'ordre suivant : largeur γ du champ récepteur, vitesse de l'adaptation α, forces d'adaptation à long terme η_l et à court terme η_s. Par défaut, l'objet utilise `toiviainen 2.2 7.4 0.08 0.23`
(entrée 2) `bang`			Chaque `bang` reçu dans la 2e entrée indique à l'objet la pulsation. La phase est forcée à 0. La durée entre deux `bang` consécutifs détermine la période du modèle. Seules les durées comprises entre 20 ms et 2000 ms sont acceptées (soit un tempo compris entre 30 bpm et 300 bpm).

Sortie
En sortie, l'objet génère un `bang` pour indiquer la pulsation.

Interface graphique
Cette interface, disponible en insérant l'objet dans un `bpatcher`, permet de choisir l'algorithme utilisé, et d'ajuster les paramètres en temps réel. La Figure 18 présente une copie d'écran de cette interface. Celle-ci permet également de faire entendre un son lorsqu'une pulsation est générée. L'utilisateur peut contrôler le volume sonore de ce clic, ainsi que le type de son utilisé. La pulsation est également marquée visuellement grâce à un disque de couleur qui flashe au moment de la pulsation.

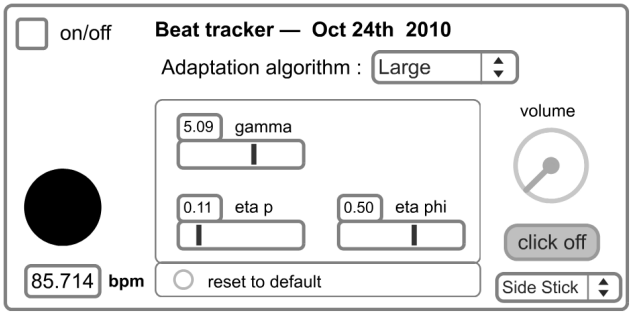

FIGURE 18 – Interface graphique de l'objet de suivi de tempo implémenté en Max.

Références

Allauzen, C., Crochemore, M. et Raffinot, M. (1999). Factor oracle : a new structure for pattern matching. *In* Pavelka, J., Tel, G. et Bartosek, M., éditeurs : *Proceedings of SOFSEM'99, Theory and Practice of Informatics*, Lecture Notes in Computer Science 1725, pages 291–306, Milovy, Czech Republic. Springer-Verlag, Berlin.

Assayag, G., Bloch, G. et Chemillier, M. (2006). Omax-ofon. *Sound and Music Computing (SMC)*.

Assayag, G. et Dubnov, S. (2004). Using factor oracles for machine improvisation. *Soft Comput.*, 8(9):604–10.

Assayag, G., Dubnov, S. et Delerue, O. (1999). Guessing the composer's mind : Applying universal prediction to musical style. *In Proceedings of the ICMC : International Computer Music Conference, Beijing*, pages 496–499.

Bello, J. (2003). *Towards the Automated Analysis of Simple Polyphonic Music : A Knowledge-based Approach*. Thèse de doctorat, Queen Mary University of London.

Biles, J. (2002). Genjam : evolutionary computation gets a gig. *In Conference on information technology curriculum*, Rochester, NY.

Chemillier, M. (2001). Improviser des séquences d'accords de jazz avec des grammaires formelles. *In JIM 2001 Journées d'informatique musicale, Imeb, Bourges*, pages 121–126.

Cholakis, E. (1995). Jazz swing drummers groove analysis. *Numerical Sound.* En ligne : `http://www.numericalsound.com`.

Collins, N. (2004). Beat induction and rhythm analysis for live audio processing : 1st year phd report. Rapport technique, University of Cambridge.

Davies, M., Degara, N. et Plumbey, M. (2009). Evaluation methods for musical audio beat tracking algorithms. Rapport technique, C4DM-TR-09-06, Queen Mary University, Centre for Digital Music.

Dixon, S. (2001). Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30:39–58.

Dixon, S. (2007). Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, 36(1):39–51.

Dixon, S., Goebl, W. et Cambouropoulos, E. (2006). Perceptual smoothness of tempo in expressively performed music. *Music Perception*, 13(3):195–214.

Downie, J. (2008). The music information retrieval evaluation exchange (2005–2007) : A window into music information retrieval research. *Acoust. Sci. & Tech.*, 29(4):247–255.

Dubnov, S. et Assayag, G. (1998). Universal classification applied to musical sequences. *In Proceedings of the ICMC : International Computer Music Conference, Ann Arbor, Michigan*.

Friberg, A. et Sundberg, J. (1995). Time discrimination in a monotonic, isochronous sequence. *Journal of the Acoustical Society of America*, 98(5):2524–2531.

Friberg, A. et Sundström, A. (2002). Swing ratios and ensemble timing in jazz performance : Evidence for a common rhythmic pattern. *Music Perception*, 19(3): 333–349.

Goto, M. (2001). An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–171.

Goto, M. et Muraoka, Y. (1997). Issues in evaluating beat tracking systems. *In Proceedings of IJCAI-97 Workshop on Issues in AI and Music*, pages 9–16.

Gouyon, F. et Dixon, S. (2005). A review of automatic rythm description systems. *Computer Music Journal*, 29(1):34–54.

Gouyon, F., Klapuri, A., Dixon, S., Alonso, M., Tzanetakis, G., Uhle, C. et Cano, P. (2006). An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech and Language Processing*, 14(5):1832–1844.

Gouyon, F. et Meudic, B. (2003). Towards rhythmic content processing of musical signals : Fostering complementary approaches. *Journal of New Music Research*, 32(1):41–64.

Grahn, J. (2009). Neuroscientific investigations of musical rhythm : Recent advances and future challenges. *Contemporary Music Review*, 28(3):251–277.

Hainsworth, S. (2003). *Techniques for the Automated Analysis of Musical Audio*. Thèse de doctorat, University of Cambridge.

Jensen, K. et Andersen, T. (2003). Beat estimation on the beat. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 87–90.

Kirkpatrick, S., Gelatt, C. et Vecchi, M. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Klapuri, A., Eronen, A. et Astola, J. (2006). Analysis of the meter of acoustic musical signals. *IEEE Transactions on Audio, Speech and Language Processing*, 14(1):342–355.

Koechlin, O. (2010). De l'influence des outils numériques interactifs sur le temps musical. *Musimédiane*.

Large, E. (1995). Beat tracking with a nonlinear oscillator. *In Working Notes of the IJCAI-95 Workshop on Artificial Intelligence and Music*, volume 24031.

Large, E. (2000). On synchronizing movements to music. *Human Movement Science*, 19:524–566.

Large, E. et Jones, M. (1999). The dynamics of attending : How people track time-varying events. *Psychological Review*, 106(1):119–159.

Large, E. et Kolen, J. (1994). Resonance and the perception of musical meter. *Connection Science*, 6(1):177–208.

Large, E. et Palmer, C. (2002). Perceiving temporal regularity in music. *Cognitive Science*, 26:1–37.

Large, E. et Snyder, J. (2009). Pulse and meter as neural resonance. *Ann. N.Y. Acad. Sci.*, 1169:46–57.

McAuley, J. (1995). *Perception of time as phase : Toward an adaptive-oscillator model of rhythmic pattern processing.* Thèse de doctorat, Indiana University, Bloomington.

McKinney, M., Moelants, D., Davies, M. et Klapuri, A. (2007). Evaluation of audio beat tracking and music tempo extraction algorithms. *Journal of New Music Research*, 36(1):1–16.

Nouno, G. (2008). *Suivi de Tempo Appliqué aux Musiques Improvisées.* Thèse de doctorat, Université Paris VI.

Pachet, F. (2002a). The continuator : Musical interaction with style. *In* ICMA, éditeur : *Proceedings of ICMC*, pages 211–218, Göteborg, Sweden. ICMA.

Pachet, F. (2002b). Playing with virtual musicians : the continuator in practice. *IEEE Multimedia*, 9(3):77–82.

Parncutt, R. (1994). A perceptual model of pulse salience and metrical accent in musical rhythms. *Music Perception*, 11(4):409–464.

Peretz, I. et Coltheart, M. (2003). Modularity of music processing. *Nature Neuroscience*, 6(7):688–691.

Peretz, I. et Zatorre, R. (2005). Brain organization for music processing. *Annu. Rev. Psychol.*, 56:89–114.

Poirson, E. (2002). Simulation d'improvisations à l'aide d'un automate de facteurs et validation expérimentale. Mémoire de D.E.A., ATIAM, UPMC-IRCAM.

Puckette, M., Apel, T. et Zicarelli, D. (1998). Real-time audio analysis tools for pd and msp. *In Proceedings of the International Computer Music Conference*, pages 109–112.

Repp, B. (2005). Sensorimotor synchronization : A review of the tapping literature. *Psychonomic Bulletin & Review*, 12(6):969–992.

Robertson, A. et Plumbley, M. (2007). B-keeper : A beat-tracker for live performance. *Proceedings of the 2007 Conference on New Interfaces for Musical Expression (NIME07), New York, NY, USA*.

Scheirer, E. (1998). Tempo and beat analysis of acoustic musical signals. *J. Acoust. Soc. Am.*, 103(1):588–601.

Seppänen, J. (2001a). Computational models of musical meter recognition. Mémoire de D.E.A., Tampere University of Technology.

Seppänen, J. (2001b). Tatum grid analysis of musical signals. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 131–4.

Snyder, J. et Krumhansl, C. (2001). Tapping to ragtime : Cues to pulse finding. *Music Perception*, 18(4):455–489.

Snyder, J. et Large, E. (2005). Gamma-band activity reflects the metric structure of rhythmic tone sequences. *Cognitive Brain Research*, 24(1):117–126.

Toiviainen, P. (1998). An interactive midi accompanist. *Computer Music Journal*, 22(4):63–75.

Winkler, I., Háden, G., Ladinig, O., Sziller, I. et Honing, H. (2009). Newborn infants detect the beat in music. *PNAS*, 106(7):2468–2471.