LIVESCALER : CONTRÔLER EN LIVE L'HARMONIE D'UN MORCEAU DE MUSIQUE ÉLECTRONIQUE

Alice Rixte
Université de Bordeaux
alice.rixte@u-bordeaux.fr

RÉSUMÉ

Dans le contexte de l'Electronic Dance Music (EDM), de nombreux artistes utilisent des techniques de *DJing* pour performer leurs propres productions sur scène, se privant ainsi de l'accès à la structure interne de leurs morceaux, et en particulier de l'équivalent de leur partition : les fichiers MIDI joués par des instruments virtuels. De plus, si l'artiste remixe ou interprète sa propre production, le nombre de pistes pouvant être contrôlées simultanément est limité sans un outil adapté.

Cet article présente le logiciel LiveScaler, qui permet de contrôler en live l'harmonie et la hauteur des notes de tous les instruments virtuels d'un morceau de musique électronique. Un ensemble restreint de transformations de l'espace des hauteurs de notes, les *transformations affines*, est introduit. Ces transformations sont appliquées à tous les flux MIDI d'un morceau composé préalablement. Une implémentation utilisant Max MSP en combinaison avec Ableton Live (Max for Live) est proposée. Une attention particulière est portée aux questions de contrôle, de *mapping* et de mise en pratique dans le cadre de l'EDM.

1. INTRODUCTION

Aux origines de la *club culture*, le rôle des DJs était surtout de jouer des enregistrements musicaux pour un public, en général dansant. Avec l'évolution rapide de la technologie et en particulier l'omniprésence de l'audionumérique à partir des années 2000, de nombreux DJs sont devenus à la fois compositeurs, producteurs et performeurs de leurs propres morceaux. C'est notamment le cas des artistes évoluant dans le milieu de l'Electronic Dance Music (EDM) [1] .

Ces artistes composent le plus souvent leur musique à l'aide d'une station audionumérique (Digital Audio Workstation ou DAW) qui leur permet de combiner sampleurs, synthétiseurs et enregistrements pour créer un morceau de musique complet. Recréer en live un tel morceau, composé souvent de plusieurs dizaines de pistes, reste difficile et conditionné par les contrôles proposés par les DAW. Pour leur performance live, la plupart des artistes vont ainsi choisir entre interpréter certaines pistes spécifiques à l'aide de synthétiseurs ou sampleurs, remixer en live un morceau préparé au préalable ou utiliser des techniques de DJing leur permettant de mixer entre eux des rendus audios en leur appliquant divers effets [14],[22]. Dans ces contextes, modifier le rythme ou l'harmonie d'un morceau joué en live est difficilement réalisable dans ces contextes. En effet, cela requerrait un accès à la structure interne du morceau, ce que le DJing ne permet pas, ou bien de contrôler toutes les pistes simultanément, ce qui est précisément ce que nous proposons de faire ici.

Cet article présente LiveScaler, qui permet de modifier en live la structure harmonique d'un morceau de musique électronique [2] . En partant d'un morceau composé au préalable, LiveScaler permet d'appliquer des transformations MIDI à l'ensemble des instruments virtuels. LiveScaler conserve l'ensemble des caractéristiques du morceau initial mais agit sur la hauteur des notes, permettant de changer en live l'harmonie du morceau ou de générer de nouvelles mélodies. En particulier, l'accent est mis sur la possibilité d'utiliser LiveScaler dans un DAW, ici Ableton Live [3] , tout en minimisant les contraintes et les connaissances techniques nécessaires pour son utilisation. Dans Ableton Live, LiveScaler peut s'utiliser aussi bien dans le mode *session* que dans le mode *arrangement*, et s'ajoute à toutes les pistes MIDI que l'on souhaite être impactées par les transformations.

Cet article est organisé de la manière suivante : nous commencerons par définir les transformations affines, un ensemble restreint de transformations de l'espace des hauteurs de notes. Ensuite, nous présenterons LiveScaler, qui implémente l'application de ces transformations en live à un nombre arbitraire d'instruments virtuels. Enfin, nous décrirons la manière dont nous avons utilisé LiveScaler dans le cadre d'une performance live d'EDM . Nous terminerons par une comparaison de LiveScaler avec les travaux et outils existants.

2. TRANSFORMATIONS DE L'ESPACE DES HAUTEURS

Dans cette section nous présentons les transformations de l'espace des hauteurs que nous utilisons dans LiveScaler. L'idée principale est d'associer à chaque note une nou-

1 . *Electronic Dance Music* (EDM) est un terme parapluie regroupant de nombreux genres de musique électronique tels que la house, la techno, la trance, la drum n bass, le dubstep, etc.

2 . Une vidéo de démonstration est disponible à l'adresse suivante : youtu.be/Cn0HBgWS5Pw

3 . Une implémentation de LiveScaler avec Max For Live est disponible en libre accès sur Github : github.com/autonym8/LiveScaler

velle note, qui sera jouée en lieu et place de la note initiale par l'ensemble des instruments du morceau de musique joué. LiveScaler permet d'appliquer deux types de transformations. D'une part les *transformations périodiques sur un intervalle*, que nous présenterons après avoir défini l'espace des hauteurs de note sur lequel agissent nos transformations. D'autre part les *transformations affines*, dont nous décrirons les propriétés musicales. Enfin, nous terminerons en restreignant l'écart de hauteur entre la note initial et la note transformée afin de résoudre de potentiels problèmes de tessiture.

2.1. Espace des hauteurs

Dans cet article, nous nous plaçons dans l'espace des hauteurs linéaire. Celui-ci est obtenu à partir de la demi-droite réelle positive des fréquences, en fixant une origine arbitraire de fréquence α (par exemple 440 Hz). Ici, nous appellerons cette fréquence l'*ancre*. À partir d'une fréquence $f > 0$, on détermine une hauteur p par

$$p = \log_2(f/\alpha)$$

On obtient alors la droite réelle : $p \in \mathbb{R}$. En discrétisant cette droite, on peut obtenir une note n à partir de la hauteur p par

$$n = \lfloor \beta p \rfloor$$

avec $\beta \in \mathbb{N}^*$ le nombre de divisions de l'octave. On obtient ainsi le tempérament à division multiple β-TET [4]. En particulier, lorsque $\beta = 12$ et $\alpha = 8.1758$ Hz [5], on retrouve le codage des hauteurs MIDI.

Ainsi, pour définir un tempérament multiple, nous avons besoins de deux paramètres : l'*ancre* α et le nombre de divisions de l'octave β, que nous appelons *base*. On définit alors

$$\begin{aligned} T\langle\alpha,\beta\rangle : \mathbb{R}_+^* &\to \mathbb{Z} \\ : f &\mapsto \lfloor \beta \log_2(f/\alpha) \rfloor \end{aligned}$$

Dans cet article, nous nous intéresserons à l'*espace des hauteurs linéaires* discrétisé $\mathbb{Z} = T\langle\alpha,\beta\rangle(\mathbb{R}_+^*)$. Par abus de langage, on appellera ici la fonction $T\langle\alpha,\beta\rangle$ le *tempérament égal* d'ancre α et de base β. En pratique, α correspondra toujours à une note MIDI, et on se permettra d'écrire $\alpha = \mathrm{C}_5$ pour indiquer que la fréquence correspondant à la note C_5 est associée à 0 dans $\mathbb{Z}$.

Nous définissons alors une *transformation de gamme* comme une fonction qui à toute hauteur de note associe une nouvelle hauteur de note a priori quelconque, autrement dit une fonction de $\mathbb{Z}$ dans $\mathbb{Z}$. Cette transformation est *a priori* indépendante de la manière dont $\mathbb{Z}$ est relié à l'espace des fréquences.

2.2. Transformations périodiques sur un intervalle

Nous définissons ici un ensemble de transformations des hauteurs facilement implémentable : les transformations périodiques sur un intervalle positif $n \in \mathbb{N}$. Ces transformations s'obtiennent en donnant l'image de toutes les notes se situant dans l'intervalle i , typiquement une octave ($i = 12$), puis en répétant ce motif verticalement en additionnant (ou soustrayant) l'intervalle considéré. Plus précisément, pour une transformation $X : \mathbb{Z} \to \mathbb{Z}$, X est périodique sur l'intervalle$i$ lorsque la fonction $Y : n \mapsto X(n) - n$ est périodique de période i. Autrement dit,

$$Y(iq + n) = Y(n)$$

avec $0 \leq n < i$ et tout $q \in \mathbb{Z}$.

On peut par exemple définir une transformation qui quantiser les 12 demi-tons chromatique vers une gamme de notre choix, par exemple la gamme majeure dans la Figure 1.

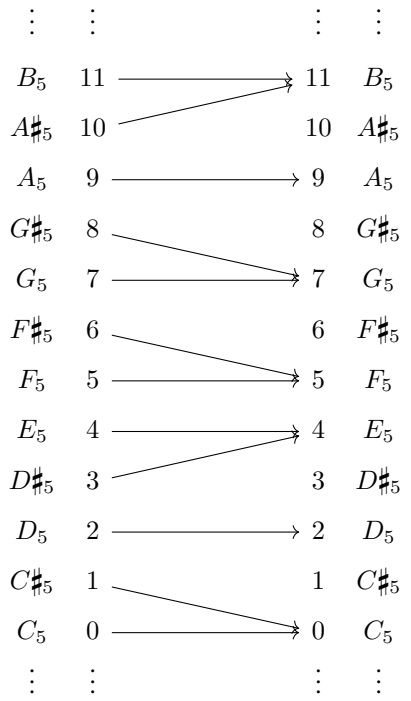

Figure 1: La quantisation vers la gamme majeure est périodique sur l'octave : le même motif est répété sur chaque octave.

2.3. Transformations affines

Nous présentons ici les *transformations affines*, c'est-à-dire les fonctions de la forme $A\langle\mu,\tau\rangle : n \mapsto \mu n + \tau$ avec μ le *mode* de la transformation affine et τ sa *transposition*. Nous allons classifier ces transformations en fonction de mu, qui détermine, en fonction du mode départ, le mode dans lequel on se trouvera après avoir appliqué la transformation.

Les transformations affines ont la propriété importante de préserver les classes de hauteur [6] (au sens de Forte

4 . TET signifie Tone Equal Temperament en anglais

5 . 8.1758 à 0 est la fréquence (ici arrondie) correpondant à la note C_{-1} et la note MIDI 0.

6 . En effet, pour toute base $\beta \in \mathbb{N}^*$, $\forall n_1, n_2 \in \mathbb{Z}, n_1 \equiv n_2 \mod \beta \implies \mu n_1 + \tau \equiv \mu n_2 + \tau \mod \beta$. Avec $\beta = 12$, on obtient le résultat pour les classes de hauteurs dodécaphoniques.

[15]) c'est-à-dire que si deux notes sont identiques à l'octave près, alors elles le seront toujours une fois la transformation affine appliquée.

La suite de cette section étudie plusieurs exemples afin de donner au lectorat un aperçu de leur expressivité. Dans l'ensemble de ces exemples, on se placera dans le tempérament égal $T\langle \mathrm{C}_5, 12\rangle$: la note C_5 correspondra ainsi à l'entier 0, $\mathrm{C}\sharp_5$ à 1, C_6 à 12, B_4 à -1, C_4 à -12, etc. Nous nous concentrerons ici sur les cas $\mu = 1$, $\mu = -1$ et $\mu = 2$.

2.3.1. Transpositions

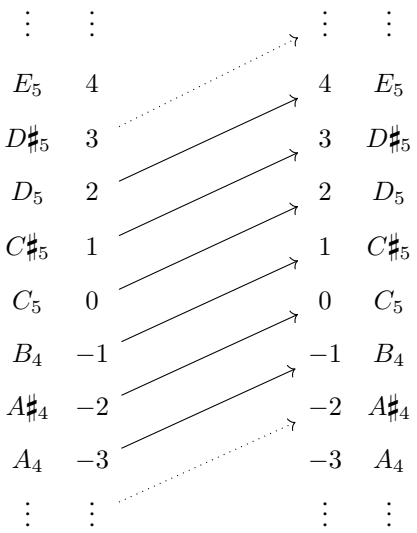

Figure 2: La transformation $A\langle 1, 2\rangle : n \mapsto n + 2$ correspond à la transposition d'un ton vers l'aigu

Lorsque $\mu = 1$, les transformations affines $A\langle 1, \tau\rangle : n \mapsto n + \tau$ permettent de représenter toutes les transpositions possibles (voir Figure 2).

2.3.2. Inversions

Lorsque $\mu = -1$, les transformations $A\langle -1, \tau\rangle : n \mapsto -n + \tau$ permettent de passer d'une gamme majeure à un mode de Mi et réciproquement (voir Figure 4). La Figure 3 illustre la manière dont la triade C_5, E_5, G_5 est transformée par $A\langle -1, 4\rangle$ en la triade E_5, C_5, A_4. De même, l'image de la triade A_4, C_5, E_5 est G_5, E_5, C_5. Comme les transformations affines préservent les classes de hauteur, on peut affirmer plus généralement que $A\langle -1, 4\rangle$ transforme l'accord C en Am et Am en C.

Remarquons de plus qu'en changeant l'ancre, en choisissant $\alpha = \mathrm{G}_4$ par exemple, l'image de l'accord G par $A\langle -1, 4\rangle$ sera Em. La Table 1 explicite les images des accords de la gamme de Do majeur par $A\langle -1, 4\rangle$ lorsque $\alpha = \mathrm{C}_n$ et des accords de la gamme de Sol majeur lorsque l'on change l'ancre pour $\alpha = \mathrm{G}_n$, avec n un entier quelconque.

Ainsi, en se basant sur l'image de l'accord majeur dont la tonique correspond à l'ancre, nous pouvons associer les

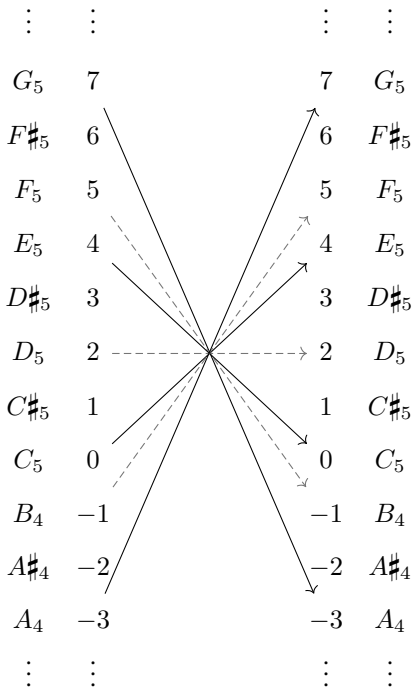

Figure 3: La transformation $A\langle -1, 4\rangle : n \mapsto -n+4$ avec $\alpha = \mathrm{C}_5$ et $\beta = 12$

Pour des raisons de lisibilité seules les flèches de la gamme de Do majeur ont été tracées. On notera la passage de l'accord C à Am et de Am à C.

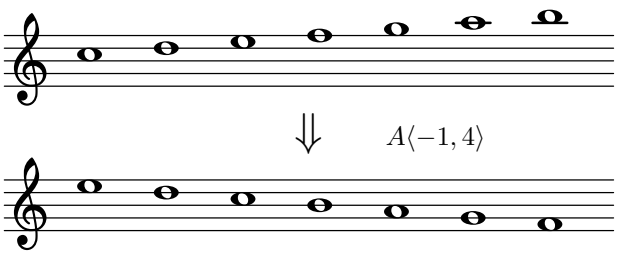

Figure 4: L'image de la gamme de Do majeur par $A\langle -1, 4\rangle$ est un mode de Mi

C	$\mapsto$	Am	G	$\mapsto$	Em
Dm	$\mapsto$	G	Am	$\mapsto$	D
Em	$\mapsto$	F	Bm	$\mapsto$	C
F	$\mapsto$	Em	C	$\mapsto$	Bm
G	$\mapsto$	Dm	D	$\mapsto$	Am
Am	$\mapsto$	C	Em	$\mapsto$	G
B^{o}	$\mapsto$	B^{o}	$\mathrm{F}\sharp^{\mathrm{o}}$	$\mapsto$	$\mathrm{F}\sharp^{\mathrm{o}}$

Table 1: Images des accords de la gamme de Do majeur par $A\langle -1, 4\rangle$ pour $\alpha = \mathrm{C}_n$ (à gauche) et de la gamme de Sol majeur pour $\alpha = \mathrm{G}_n$ (à droite), pour $n \in \mathbb{Z}$

transformations affines pour lesquelle $\mu = 1$ ou $\mu = -1$ au degré de l'image de cet accord dans la gamme. Par exemple, $A\langle -1, 4\rangle$ peut être associée au degré vi car elle associe le sixième degré mineur à l'accord majeur dont la tonique est l'ancre (Am pour C, Em pour G, ...). On obtient ainsi une notation plus intuitive que la notation mathématique, dont la table 2 donne un aperçu.

Degré	Transformation affine
I	$A\langle\ 1,\ 0\rangle$
ii	$A\langle -1, -3\rangle$
iii	$A\langle -1, -1\rangle$
IV	$A\langle\ 1,\ 5\rangle$
V	$A\langle\ 1,\ 7\rangle$
vi	$A\langle -1,\ 4\rangle$
vii	$A\langle -1,\ 6\rangle$

Table 2: Correspondances entre triades d'une gamme majeure et transformations de gamme

Remarquons que, de manière générale, les inversions affines se comportent moins bien sur les modes non naturels. Par exemple, $A\langle -1, 4\rangle(\text{G}\sharp) = \text{G}\sharp$ donc l'image de la gamme de La mineur harmonique par $A\langle -1, 4\rangle$ contient les notes C, D, E, F, G, G♯, B, ce qui ne correspond pas à un mode standard de la musique tonale. C'est une des faiblesses des transformations affines.

2.3.3. Transformation vers un mode à transposition limitée

Lorsque $\mu = 2$ ou $\mu = -2$, les transformations affines envoient n'importe quelle gamme vers une gamme apparentée à une gamme par tons (voir Figure 5). Les transformations affines peuvent donc permettre de sortir du cadre de la musique tonale.

Contrairement aux inversions et aux transpositions, cette transformation n'est pas bijective : l'image de $A\langle 2, 0\rangle$ contient exactement 6 classes de hauteurs qui correspondent aux 6 notes d'une des deux gammes par tons. Il est intéressant de noter que l'image d'une gamme majeure ou mineure naturelle par $A\langle 2, \tau\rangle$ contient les 6 notes de la gamme par tons (voir Table 3). Ce n'est pas le cas pour la gamme mineure harmonique dont l'image par $A\langle 2, \tau\rangle$ ne contient que 5 classes de hauteur.

C	↦	C	C	↦	C
D	↦	E	D	↦	E
E	↦	G♯	E♭	↦	F♯
F	↦	A♯	F	↦	A♯
G	↦	D	G	↦	D
A	↦	F♯	A♭	↦	E
B	↦	A♯	B♭	↦	G♯

Table 3: Image des gammes de Do majeur (à gauche) et Do mineur naturel (à droite) par $A\langle 2, 0\rangle$

De manière plus générale, lorsque le coefficient modal μ n'est pas premier avec β, on obtient un mode à transposition limitée dont les notes sont séparées par un intervalle de μ demi-tons. Ainsi, pour $\mu = 4$, on obtient un mode composé de 3 classes d'hauteurs, séparées par des tierces majeures (4 demi-tons).

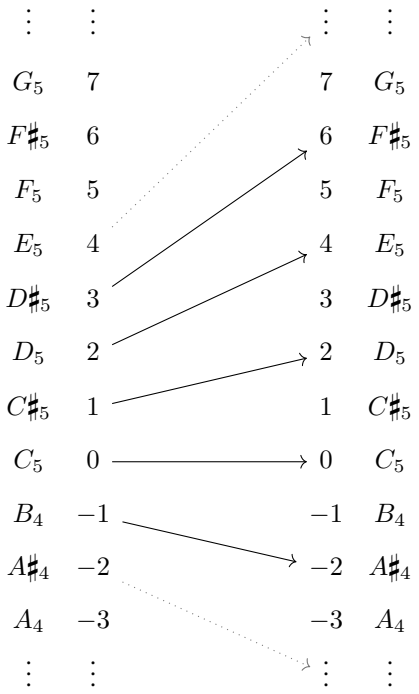

Figure 5: La transformation $A\langle 2, 0\rangle : n \mapsto 2n$ permet d'obtenir des gammes apparentées à une gamme par tons.

La Table 4 résume les différents types de transformations qu'offrent les transformations affines, ainsi que le nombre de classes de hauteur dans l'image de ces transformations [7]. Les transformations affines bijectives - leur image contient 12 classes de hauteurs - correspondent aux automorphismes $F\langle u, j\rangle$ du groupe T/I décrits par [19].

μ	Type de transformation	Classes de hauteur
-1	Inversions majeur/mineur	12
0	Constante	1
1	Transpositions	12
-2,2	Gamme par tons	6
-3,3	Tierces mineures	4
-4,4	Tierces majeures	3
-5,5	$F\langle 7, \tau\rangle$, $F\langle 5, \tau\rangle$	12
-6,6	Tritons	2
12	Octaves	1

Table 4: Classification des transformations affines en fonction de leur coefficient modal μ

7 . Le nombre de classes de hauteur dans l'image de $A\langle \mu, \tau\rangle$ est égal à $\frac{\beta}{\mu \wedge \beta}$ où $\wedge$ dénote le pgcd de deux entiers.

2.4. Restriction de l'écart de hauteur

Jusqu'ici, nous avons présenté les transformations affines en nous concentrant sur leur action sur les classes de hauteur. Dans la pratique, si nous appliquons directement $A\langle -1, 4\rangle$ à la note A_4 qui correspond au La 440Hz et à la note MIDI 69, on obtient $A\langle -1, 4\rangle(69) = -65 = \mathrm{G}_{-6}$, qui est bien trop grave pour être audible.

Afin de ne pas trop s'éloigner de la tessiture de l'instrument, ou même du spectre auditif, nous allons restreindre l'écart entre la note initiale et son image. Soit $X : \mathbb{Z} \to \mathbb{Z}$ une transformation de gamme quelconque. Définissons à partir de X une nouvelle transformation $X\langle \delta^-, \delta^+\rangle$ de sorte que

$$-\delta^- \leq X\langle \delta^-, \delta^+\rangle(n) \leq \delta^+$$

où δ^+ (resp. δ^-) est l'interval montant (resp. descendant) maximum entre la note initiale et son image.

Soit $\delta = \delta^+ - \delta^-$. Dans la plupart des cas on souhaite que $\delta = \beta = 12$, mais il peut être intéressant de choisir par exemple $\delta = 2\beta$, pour préserver le caractère montant ou descendant d'une ligne mélodique.

Soit $r = |X(n) - n| \mod \beta$ le reste de la division euclidienne de $|X(n) - n|$ par β. On pose alors

$$X\langle \delta^+, \delta^-\rangle : n \mapsto \begin{cases} n + r & \text{si } r \leq \delta^+ \\ n + r - \delta & \text{sinon} \end{cases}$$

Nous pouvons maintenant appliquer cette restriction de l'intervalle de hauteur à nos fonctions affines. On obtient alors un sous-ensemble de transformations de gamme de la forme $A\langle \mu, \tau, \delta^-, \delta^+\rangle$. Ce sont exactement ces transformations, combinées avec les tempéraments $X\langle \alpha, 12\rangle$, qui sont implémentées dans LiveScaler.

3. IMPLÉMENTATION DES TRANSFORMATIONS AFFINES : LIVESCALER

Nous allons à présent nous intéresser à l'implémentation des transformations affines que nous venons de présenter afin de pouvoir les appliquer en live à l'ensemble des flux MIDI qui composent le morceau joué.

3.1. Architecture de LiveScaler

LiveScaler fonctionne à la manière d'un orchestre dont le DJ serait le chef. Chaque piste MIDI contenant un instrument virtuel (synthétiseur, sampleur, etc.) est un instrumentiste de l'orchestre. On souhaite que sur un geste du DJ, chaque instrument virtuel interprète différemment sa partition, c'est à dire le flux MIDI qu'il reçoît. Dans le cadre de LiveScaler, le DJ envoie les paramètres d'une transformation affine à tous les instruments simultanément et ceux-ci doivent appliquer cette transformation dès qu'ils la reçoivent.

L'implémentation de LiveScaler est donc séparée en deux outils interdépendants :

1. une interface (appelée *Conductor*, en référence à l'analogie avec l'orchestre) qui récupère les entrées de l'utilisateur (ici le DJ) et les convertit en paramètres d'une transformation affine puis envoie ces paramètres à toutes les instances de *Instrument*.
2. un plug-in MIDI appelé *Instrument* qui transforme le flux MIDI entrant en appliquant à toutes les notes la transformation affine dont les paramètres ont été reçus de *Conductor*.

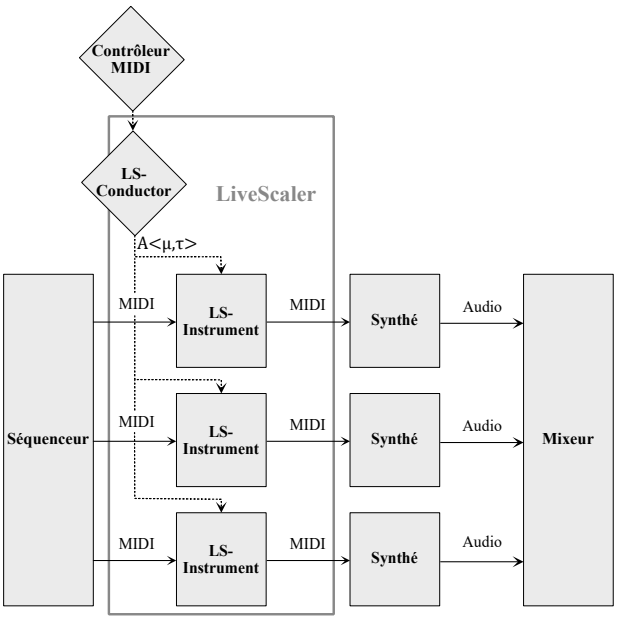

Figure 6: Architecture de Live Scaler

La Figure 6 illustre l'architecture globale de LiveScaler.

On distingue également les paramètres *locaux*, qui sont propres à chaque instrument, et les paramètres *globaux*, qui sont reçus du chef d'orchestre et donc communs à tous les instruments. Dans le cadre des transformations affines, les paramètres μ, τ, α et β sont globaux, il correspondent dans une certaine mesure à l'harmonie actuelle du morceau. Quant à δ^- et δ^+, ils sont locaux et peuvent être adaptés à la tessiture de l'instrument.

3.2. Quand appliquer les transformations ?

Lorsque *Instrument* reçoit la commande d'appliquer une nouvelle transformation de gamme, celle-ci est sensée prendre effet immédiatement et être appliquée à toutes les notes reçues jusqu'au prochain changement de gamme. Lorsque l'instrument n'est pas en train de jouer, cela ne pose aucune difficulté : il appliquera la nouvelle transformation au prochain message MIDI qu'il recevra. Il se peut en revanche que l'instrument soit déjà en train de jouer une note. LiveScaler propose 4 manières de réagir dans une telle situation :

1. *Stop* : toutes les notes en train d'être jouées sont instantanément arrêtées en envoyant un message Note-off pour chaque note en cours. L'instrument reprendra son jeu, en appliquant la nouvelle transformation, au prochain message MIDI qu'il recevra. Cette

Figure 7: Interface graphique de LiveScaler (*Conductor* à gauche et *Instrument* à droite)

option est particulièrement adaptée aux instruments dont la durée des notes est courte.

2. *Legato* : chaque note en cours est stoppée et instantanément remplacée par son image par la nouvelle transformation. Si l'instrument virtuel est paramétré sur *Legato*, alors les changements de gamme déclencheront des legatos.
3. *ReTrigger* : agit sur le même principe que *Legato* à la différence qu'un court délai est introduit entre la fin de la note en cours et la note transformée, forçant une nouvelle attaque, même si l'instrument virtuel est en mode legato.
4. *Wait* : les notes en cours continuent d'être jouées telles quelles. Si elles ne se sont pas arrêtées avant, elles seront stoppée lorsque la prochaine note sera jouée, à partir de laquelle la nouvelle transformation prendra effet.

Le choix entre ces quatre modes se fait de manière locale, deux instances de *Instrument* pourront donc réagir différemment.

3.3. Implémentation avec Max for Live

L'objectif principal de l'implémentation proposée était de pouvoir expérimenter le plus rapidement possible sur les transformations en tant que musicienne. Max for Live est une intégration du langage de progammation graphique Max MSP à Ableton Live. On peut aisément communiquer avec les différences instances du logiciel, ce qui permet dans LiveScaler à *Conductor* de contrôler les *Instruments* avec une faible latence [8] . La station audionumérique Ableton Live étant particulièrement populaire pour composer et produire de la musique dans le milieu de l'EDM, c'est donc naturellement que nous avons choisi Max for Live pour une première implémentation.

La Figure 7 montre l'interface graphique de LiveScaler. L'implémentation actuelle permet d'appliquer les transformations affines et les transformations périodiques sur un intervalle. Pour ces dernières, il suffit de renseigner manuellement l'image de chaque note de l'intervalle considéré dans un fichier pour y avoir ensuite accès via LiveScaler.

8 . En moyenne, LiveScaler introduit une latence inférieure à 1 ms.

4. ESCAPE : UNE PERFORMANCE AVEC LIVESCALER

Afin de tester la pertinence des transformations proposées, j'ai utilisé LiveScaler pour créer *Escape*, une performance live de trance psychédélique. J'ai notamment pu proposer cette performance en publique pour la soutenance de mon mémoire de master en Septembre 2019. Dans cette section, j'expliquerai la manière dont j'ai procédé pour cette mise en pratique et je donnerai mes impressions subjectives en tant que musicienne et utilisatrice de LiveScaler.

4.1. Contrôle live dans Escape

Figure 8: Contrôleurs MIDI utilisés pour la performance live : à gauche le Push 2 par Ableton (contrôle de la structure du morceau) et à droite ATOM par Presonus (contrôle de l'harmonie du morceau).

Pour *Escape*, j'utilise deux contrôleurs MIDI distincts (voir Figure 8) :

— le Push 2 par Ableton, qui est conçu spécialement pour contrôler Ableton Live. Il me permet de recréer en live la structure d'Escape en déclenchant ses différentes parties.

— l'ATOM de Presonus, composé d'une grille de 4×4 touches qui me permettent de déclencher les transformations de LiveScaler.

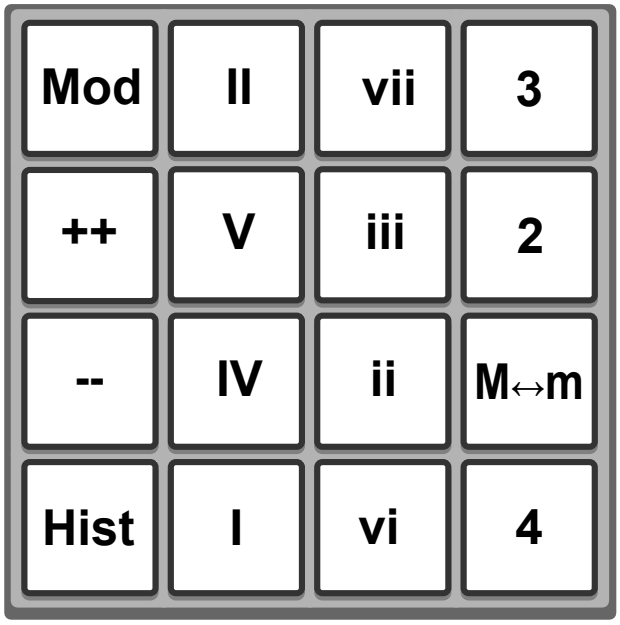

Figure 9: Mapping de LiveScaler sur un contrôleur MIDI à 4×4 touches

La figure 9 illustre la manière (le plus souvent appelée *mapping*) dont les touches du contrôleur sont associées aux transformations de LiveScaler.

Voici le détail de l'action des différentes touches :

— I, vi, IV, II, V, iii, II, vii : les deux colones centrales déclenchent instantanément les transformations décrites précédemment. Elles sont organisées par relatives mineures/majeures.

— `Hist` : LiveScaler garde en mémoire un court historique des transformation précédemment appliquées. Appuyer sur `Hist` permet de déclencher une des transformations de cet historique. Des combinaisons de la touche `Hist` et des touches `Hist`, `M↔m`, `2`, `3`, et `4` permettent de naviguer dans cet historique [9] .

— `++` , `--` : applique $\tau = \tau + 1$ (resp. $\tau = \tau - 1$). En pratique, combiner la touche $++$ (resp. $--$) avec une des transformations des colonnes centrales, on transpose cette transformation d'un demi-ton vers le haut (resp. vers le bas).

— `M↔m` : applique $\tau = \tau + 7\mu$ et $\mu = -\mu$. En pratique, combiner `M↔m` avec une des transformations centrales permet de passer d'une transposition à une inversion et réciproquement : combiner `M↔m` avec I (resp. ii, iii, IV, V, vi, vii) donnera la transformation i (resp. II, III, iv, v, VI, VII) et réciproquement.

— `2`, `3`, `4` : applique $\mu = 2\mu$ (resp. $\mu = 3\mu$, $\mu = 4\mu$). On obtient ainsi les modes à transposition limitée décrits précédemment.

— `Mod` : en combinant `Mod` avec une des transformations, on indique à LiveScaler qu'on souhaite moduler l'harmonie de notre morceau vers cette nouvelle gamme.

Ainsi, on pourrait imaginer harmoniser en live une instrumentation jouant de manière répétée sur l'accord C. Par exemple, si on veut reproduire la suite d'accord de la chanson *Summer Nights* de la comédie musicale *Grease* [10] en commençant par indiquer à LiveScaler qu'on est dans une tonalité de Ré majeur (`Mod` + II). Puis, une fois l'instrumentation lancée, on appuiera successivement tous les deux temps sur I- IV- V- IV.

Puis, arrivés au moment tant attendu de la modulation d'un demi-ton vers le haut, on indique à LiveScaler

I - IV - V - (`++` + `M↔m` + vi) - (`Mod` + `++` + I)

pour repartir joyeusement sur I- IV- V- IV, mais cette fois dans une tonalité de Mi bémol majeur. On obtiendrait ainsi la progression harmonique suivante :

... - D - G - A - G - D - G - A - B♭ - E♭ - A♭ - B♭ - A♭ - ...

4.2. Processus de composition

J'ai composé *Escape* dans le but de le jouer avec LiveScaler. C'est un morceau de trance psychédélique (psytrance) [11] composé :

— d'une mélodie minimaliste durant 2 mesures jouée par un synthétiseur dont le son se rapproche d'un métallophone éthéré

— d'une *rolling bass* classique devenue une des signatures de la *psytrance*, et de plusieurs autres basses sur une unique note pédale produisant une ligne de basse riche dans sa texture et son timbre

— d'une rythmique séquencée à l'avance indépendante de LiveScaler (pistes audio)

— de quelques effets sonores (*risers*, *downshifters* ...) eux aussi typiques de la *psytrance*, que je déclenche ponctuellement à l'aide du Push.

L'objectif était d'une part de partir d'un morceau simple, basé sur un unique accord (ici Am) et d'enrichir son harmonie et sa ligne mélodique avec LiveScaler ; d'autre part de proposer un morceau typique d'un genre de musique électronique populaire très codifié (ici la *psytrance*). Le choix du genre n'est pas anodin : je souhaitais utiliser un outil de contrôle live expérimental pour un morceau de musique qui, lui, n'a rien d'expérimental. Pour moi, il s'agit plus avec LiveScaler d'explorer de nouvelles modalités live que de nouveaux horizons musicaux.

Pour autant, LiveScaler peut tout à fait intervenir dans le processus de composition. On peut par exemple l'utiliser pour trouver des variations sur une mélodie ou une arpège en expérimentant avec les différents changements de gamme, puis consolider le MIDI une fois qu'on a trouvé

9 . Pour plus de détail, se référer au manuel de LiveScaler

10 . Si, comme pour moi, cette chanson à tendance à rester dans votre tête, je suis (presque) désolée

11 . La trance psychédélique est souvent appelée *psytrance*, le lectorat curieux pourra écouter par exemple l'album *The Gathering* (1999) du duo israëlien *Infected Mushroom*.

une idée satisfaisante. On obtient alors un processus créatif incrémental et prône à la sérendipité partant d'une mélodie ou d'une progression d'accords simple qu'on enrichit ensuite avec LiveScaler.

5. TRAVAUX CONNEXES

Les transformations affines sont directement inspirées de la théorie transformationnelle (pour une introduction généraliste du point de vue mathématique, voir [5], et du point de vue musicologique, lire [6]). La théorie transformationnelle prend ses racines dans la *Set-Theory*, qui se concentre sur la notion de classe de hauteur, c'est à dire un ensemble de notes identiques à l'octave près [15]. Elle propose une approche plus algébrique que la *Set-Theory*, en se concentrant, entre autre sur la notion de transformation entre ensembles de classes de hauteurs [18].

Les transformations affines sont particulièrement proche des automorphismes du groupe T/I proposés par David Lewin [19]. L'unique différence avec ceux-ci est que les transformations affines ne sont pas nécessairement bijectives et autorisent donc un coefficient modal qui ne soit pas nécessairement premier avec 12. Les transformations affines sont donc une vision plus appliquée (l'aspect algébrique est passé sous silence) des automorphismes proposés par Lewin et Klumpenhouwer, tout en proposant quelques transformations supplémentaires sortant du système tonal.

Plusieurs outils s'appuient plus ou moins explicitement sur les représentations de la théorie transformationnelle, en particulier OpenMusic, qui propose une aide à la composition directement inspirée de celle-ci [4], [3]. Depuis une dizaine d'années, OpenMusic essaie de concilier l'approche hors du temps (approche guidée par les demandes) de l'aide à la composition avec l'approche temps-réel propre à la performance [10], [11]. Plus récemment, Bach propose lui aussi cette approche hybride mais en partant de Max MSP, un language de programmation fondamentalement temps-réel et guidé par les données [2].

Les outils évoqués ci-dessus offrent une grande flexibilité pour appliquer des transformations potentiellement bien plus sophistiquées que les transformations affines, et sont *a priori* tous capables de le faire en live. Pour autant, cela demanderait un grand travail préparatoire de programmation et de *mapping* avant d'arriver à un résultat fluide. C'est exactement ce travail qui est fait par LiveScaler, mais sur un nombre restreint de transformations, ici jugées pertinentes. LiveScaler sacrifie donc la flexibilité dans le choix des transformations au profit d'une utilisation immédiate, et sans connaissances de progammation requises, pour faire de la musique live.

Une approche intermédiaire offrant plus de flexibilité, mais moins d'immédiateté est celle du *live coding*. Pendant une performance de *live coding*, le musicien utilise un langage de programmation dédié pour coder en live un morceau de musique [8]. En particulier le langage Tidal [21] permet de manipuler et transformer des motifs en live. Il reprend notamment les transformations de Laurie Spiegel [25] et plus particulièrement les transpositions et inversions (qui correspondent aux coefficients modaux 1 et -1 dans le paradigme des transformations affines proposé ici).

Si une des revendications initialement associées à la pratique du *live coding* était de s'affranchir des contraintes et rigidités des stations audionumériques telle qu'Ableton Live [12], l'ajout de langages de scripting ainsi que la posibilité de contrôler les stations numériques [12] avec des langages de programmation graphique haut niveau [13] semble avoir développé leur usage dans les pratiques de musique algorithmique live [13]. Ces pratiques, dans lesquelles s'incrit le présent article, permettent de combiner d'une part la flexibilité et la liberté qu'offrent un langage de programmation et d'autre part l'accès aux outils de production commerciaux utilisés par l'industrie de la musique. C'est particulièrement important dans le cadre de l'EDM, qui utilise intensivement ces outils de production sophistiqués [16].

L'idée d'utiliser un morceau composé au préalable comme matière première à laquelle on applique des transformations est particulièrement développée, et théorisée par Louis Bigo et Darrell Conklin [7]. En analysant au préalable l'harmonie d'un morceau, ils proposent ainsi de la transformer ensuite afin d'obtenir une variation du morceau. LiveScaler se démarque de cette approche par deux principaux aspects : LiveScaler ne nécessite aucune analyse préalable et se concentre sur le live. Une fois de plus la contrepartie est une moins grande flexibilité dans le choix des transformations.

Le logiciel EmoteControl [23] est sans doute celui qui se rapproche le plus de LiveScaler. Il permet un macro-contrôle en live de paramètres tels que le tempo, l'articulation, la hauteur de note, etc. En particulier, il propose une inversion du mode (qui correspond au coefficient modal -1). LiveScaler offre une bien plus grande variété de transformations de gammes et de flexibilité sur la performance live. Une collaboration serait ici particulièrement intéressante : le contrôle de l'articulation ou du timbre seraient particulièrement pertinents à ajouter à LiveScaler.

6. CONCLUSION

Nous avons présenté LiveScaler, qui propose de nouvelles modalités pour la musique électronique live, ansi qu'un jeu de transformations MIDI : les transformations affines. LiveScaler permet d'appliquer ces transformations affines en live. En particulier, cet outil peut être utilisé dans le contexte de l'EDM, proposant ainsi une nou-

12 . On peut par exemple piloter Ableton Live et FL Studio avec Python, Logic et Bitwig avec JavaScript.

13 . On trouve dans les stations audionumériques commerciales de plus en plus de langages de "*patching*" permettant de contrôler le logiciel ou de créer des plugins audio de manière modulaire : voir par exemple Max for Live pour Ableton Live, FL FlowStone pour FL Studio, The Grid pour Bitwig.

velle alternative ou un complément pour la performance live en musique électronique.

Plusieurs améliorations techniques pourraient être proposées pour améliorer LiveScaler. En particulier, il serait pertinent de le rendre compatible avec n'importe quel DAW, et pas seulement Ableton Live. Une solution sera de développer un plugin VST ou LV2 pour LiveScaler. De plus, le protocole MIDI étant contraignant, rendre LiveScaler compatible avec un protocole plus flexible tel que OSC [26], MPE (Midi Polyphonic Expression), Midi 2.0 ou encore MP [17], qui serait particulièrement adapté à cette application.

Bien que LiveScaler propose déjà la possibilité d'ajouter manuellement des transformations de gamme périodiques sur un intervalle, ce mécanisme est laborieux et mérite d'être amélioré. Utiliser des transformations sur un espace diatonique serait également intéressant et soulèverait la difficulté de connaître la tonalité dans laquelle on se trouve. Enfin, nous aimerions pouvoir agir sur d'autres paramètres que la hauteur des notes, en particulier le rythme, le timbre, ou même contrôler simultanément des transformations musicales et vidéos. C'est sur ces axes que seront concentrées nos recherches futures, tout en restant sur le même paradigme de performance live que celui proposé ici.

6.1. Remerciements

Je tenais à remercier ici David Janin et Martin Laliberté, mes encadrants de thèse, pour leur relecture et leurs nombreux conseils. Merci également à Chloé Lavrat, pour avoir pris le temps de me lire et pour nos nombreuses discussions sur le sujet, toujours très inspirantes.

7. REFERENCES

[1] Agostini, A., Ghisi, D., Giavitto, J. "Programmer en Max avec bell", *Journées D'informatique Musicale (JIM)*, 2020.

[2] Agostini, A., Ghisi, D., Giavitto, J. "Programming in style with bach", *Perception, Representations, Image, Sound, Music*, pp. 257-278, 2021.

[3] Andreatta, M., Agon, C. "Formalisation algébrique des structures musicales à l'aide de la Set-Theory : aspects théoriques et analytiques", *Journées D'Informatique Musicale*, 2003.

[4] Andreatta, M., Agon, C. "Implementing algebraic methods in OpenMusic", *ICMC*, 2003.

[5] Andreatta, M. "Calcul algébrique et calcul catégoriel en musique : aspects théoriques et informatiques", *Le Calcul De La Musique*, pp. 429-477, 2008.

[6] Andreatta, M. "Une introduction musicologique à la recherche «mathémusicale» : aspects théoriques et enjeux épistémologiques", *Circuit*, **24**, 51-66, 2014.

[7] Bigo, L., Conklin, D. "A viewpoint approach to symbolic music transformation", *Music, Mind, And Embodiment : 11th International Symposium, CMMR 2015, Revised Selected Papers 11*, pp. 213-227, Plymouth, UK, 2016.

[8] Blackwell, A., Cocker, E., Cox, G., McLean, A., Magnusson, T. "Live coding : a user's manual", MIT Press, 2022.

[9] Bresson, J., Agon, C., Assayag, G. "OpenMusic : visual programming environment for music composition, analysis and research", *Proceedings Of The 19th ACM International Conference On Multimedia*. pp. 743-746, 2011.

[10] Bresson, J., Giavitto, J. "A reactive extension of the openmusic visual programming languag", *Journal Of Visual Languages & Computing*, **25**, pp. 363-375, 2014.

[11] Bresson, J., Bouche, D., Carpentier, T., Schwarz, D., Garcia, J. "Next-generation Computer-aided Composition Environment : A new implementation of OpenMusic", *International Computer Music Conference*, 2017.

[12] Collins, N., McLean, A., Rohrhuber, J., Ward, A. "Live coding in laptop performance", *Organised Sound*, **8**, pp. 321-330, 2003.

[13] Collins, N., McLean, A. "Algorave : Live performance of algorithmic electronic dance music", *Proceedings Of The International Conference On New Interfaces For Musical Expression*, pp. 355-358, 2014.

[14] Ferreira, P. " When sound meets movement : Performance in electronic dance music", *Leonardo Music Journal*, **18**, pp. 17-20, 2008.

[15] Forte, A. "The structure of atonal music", Yale University Press,1973.

[16] Fraser, A. "The spaces, politics, and cultural economies of electronic dance music", *Geography Compass*, **6**, pp. 500-511, 2012.

[17] Goudard, V., Genevois, H. "Mapping modulaire de processus polyphoniques", *Journées D'Informatique Musicale*, 2017.

[18] Lewin, D. "Generalized musical intervals and transformations", New Heaven : Yale University Press, 1987.

[19] Lewin, D. "Klumpenhouwer networks and some isographies that involve them", *Music Theory Spectrum*, **12**, pp. 83-120, 1990.

[20] Livingstone, S., Muhlberger, R., Brown, A., Thompson, W. "Changing Musical Emotion : A Computational Rule System for Modifying Score and Performance", *Computer Music Journal*, **34**, pp. 41-64, 2010.

[21] McLean, A., Wiggins, G. "Tidal–pattern language for the live coding of music", *Proceedings Of The 7th Sound And Music Computing Conference*. pp. 331-334, 2010.

[22] Magana, J. "Performance in EDM-A Study and Analysis of DJing and Live Performance Artists", 2018.

[23] Micallef Grimaud, A., Eerola, T. "EmoteControl : an interactive system for real-time control of emotional expression in music", *Personal And Ubiquitous Computing*, **25** pp. 677-689, 2021.

[24] Morris, R. "Composition with pitch-classes", *Yale University*, 1987.

[25] Spiegel, L. "Manipulations of musical patterns", *Proceedings Of The Symposium On Small Computers And The Arts*, pp. 19-22, 1981.

[26] Wright, M. "Open Sound Control : an enabling technology for musical networking", *Organised Sound*, **10**, pp. 193-200, 2005.