博　士　論　文

タイムストレッチ技術を用いた演奏同期システムの開発

—Fixed Mediaとライブ演奏と音楽的な同期の達成に向けて—

蒋斯汀

目次

序章

本研究は、Fixed Media[1]とライブ演奏の混成による音楽作品に対する「演奏同期システム」の開発について論じるものである。

Fixed Mediaとライブ演奏の混成による音楽作品、一般的にはMixed music[2]と呼ばれる音楽分野は、メディア上に固定された電子音響音楽作品と人間によるライブ演奏（主にアコースティック楽器）が同時に演奏（再生）されることがその特徴である。しかし、Fixed Mediaの物理的・音楽的性質により、特に長いFixed Mediaを使用すると、双方の同期に様々な課題が生じることがある（第 1 章で詳述）。

このような課題に対して、音楽家はストップウォッチ、クリックトラック、コンピュータ・インタラクティブ・システムなどを補助的に用いることで、制作時に想定された同期演奏を実現してきた。しかし、これらの技術の利用は、双方の演奏時間を同期させる一方で、演奏者による自発的な演奏時間表現に制限を課すだけでなく、作曲家に対しても、表現の幅に制約を課すことになるなど、一長一短の結果を生むこととなった。

そこで筆者は、同期に関する問題の改善・解決策として、近年進化を続けているオーディオ・タイムストレッチ技術に着目した。この技術の特徴は、音声素材の再生時間（速度）とピッチを、それぞれ独立させながら制御できることである。特に近年では、コンピュータの処理能力の向上やタイムストレッチ技術のアルゴリズムの進歩により、オーディオ素材の音質を損なうことなく、リアルタイムで制御することが可能になっている。筆者は、このような技術をMixed musicのライブ演奏に導入することで、Fixed Mediaの再生時間をリアルタイムに伸縮させながら演奏者の演奏時間に同期させ、制約が課されることなく、柔軟かつ、厳密な演奏（再生）同期を実現できるのではないかと考えた。

タイムストレッチ技術を「ライブ」同期システムに導入する試みは、実のところ、これまでにも行われてきた。例えば、Christopher Raphaelの『Music Plus One』や、IRCAM（Institut de recherche et coordination acoustique/musiqueの略称）の『Antescofo』の技術は、タイムストレッチ技術とScore Following技術[3]と組み合わせて開発された代表的な開発例である。しかしこれらのシステムは、テンポやメロディが明確な音楽にはある程度の有効性が認められているものの、Mixed musicの多くがそうであるように、流動的なテンポ、単純

---

[1] テープ音楽の現代用語。メモリーや、ハードディスクなど、コンピュータ上に「固定」されている記憶媒体にメディア（音楽）が記録されているもの、いわば「テープ音楽」を指している。後に「Acousmatic music」や「Electroacoustic music」など、スピーカーから発せられる音響を聞くことに重点においた音楽形態を派生していった。本研究では、事前に作曲・録音された電子音響音楽作品をFixed Mediaと呼ぶことにする。

[2] 「Mixed music」とは、主にFixed Mediaと楽器のライブ演奏の混成による作品形態を指す（Collins and Schedel and Wilson 2013）。

[3] コンピュータが演奏者の演奏音を聴き、伴奏音源を演奏者の演奏時間に追従させる技術を指す。

明快ではないメロディ、あるいは特殊奏法[4]を取り入れているなど、作曲・演奏表現が複雑な楽曲に対しては効果を示せておらず、実用的でないのが現状である（第１章で詳述）。そのため、現在のタイムストレッチ技術は、Mixed musicの同期問題を改善・解決し、さらに複雑な創作を実現するための有用な手段であると考えられているものの、現実的に有効なシステムはまだ存在しておらず、筆者は新たなアプローチのもと、実用的なシステムを開発する必要があると感じている。本研究の目的は、昨今のタイムストレッチ技術を利用することを前提に、Mixed musicのライブ同期問題を解決・改善し得るシステムを開発することである。新しいシステムの開発の目標は以下の通りである。

１）従来の同期技術を使用することなく、比較的に長い時間（例えば３分以上）にわたってFixed Mediaと厳密な同期演奏を達成できること。

２）たとえ特殊奏法による楽器演奏や複雑で変化するリズムを伴うものであっても、作曲に制限を設けず、自由な創作を保障すること。

第１章『開発の背景』では、システム開発の背景を説明する。まず、Mixed musicの音楽的背景と本研究で取り上げる同期問題について述べる。次に、これまで使用・考案されてきた同期技術の歴史について述べ、既存の技術に存在する問題点を明らかにする。最後に、タイムストレッチ技術の可能性を提示し、新しいシステムの開発の必要性を論じる。

第２章『課題とその解決方法』では、システム開発の方法について述べる。本研究の研究目的を達成するにあたって解決すべき課題を論じ、タイムストレッチ技術を用いた新しい同期システムの開発方法を提示する。

第３章『システムの提案』では、第２章で明確にした方法を用いて、筆者が実際に構築した新しいシステムを提案する。システムの設計と各モジュールを詳細に説明し、実際に新しいシステムを構築する方法を詳述する。

第４章『システムの検証』では、新しいシステムを用いて行った演奏をもとに、検証結果について述べる。実際の演奏結果から得られたデータを分析しながら、新システムの有用性について論じる。

第５章『終章』では、新しいシステムのメリットとデメリットを明確にし、既存の問題点と今後の開発の展望について説明する。

キーワード：コンピュータ音楽、ライブエレクトロニクス、同期演奏、フィクストメディア、電子音楽、電子音響音楽、タイムストレッチ、computer music、live electronic music、

[4] 通常の操作法に従わない楽器の演奏方法。特に楽音の要素が比較的少ないで、複雑な音響変化を表現するのに用いられることが多い。

performance synchronization、fixed media、electronic music、electroacoustic music、time stretching.

第 1 章 開発の背景

1.1 音楽的背景

Mixed musicは、1950年以降、フランスやドイツにおけるテープ音楽の発展から派生した創作・演奏形態である。つまり、現代音楽[5]である。その最大の特徴は、あらかじめ作曲され、録音された音響音楽データと、楽器演奏者（特にアコースティック楽器）によるライブ演奏を同時に行うことである。Pierre SchaefferとPierre Henryの合作作品《Orphée 51》(1951)は、この創作スタイルにおける最も初期の作品であり、テープ音楽（ミュジーク・コンクレート）と人の声を組み合わせた作品である。それ以降、数多くの作曲家によって、それぞれの美的アプローチや、録音・作曲技術を駆使した試みのもとに現在までに発展してきた。例えば、Karlheinz Stockhausen（1928-2007）、Bruno Maderna（1920-1973）、Edgard Varèse（1883-1965）、Mario Davidovsky（1934-2019）、Luigi Nono（1924-1990）、Luciano Berio（1925-2003）、Jean-Claude Risset（1938-2016）、Horacio Vaggione（1943- ）、Barry Truax（1947- ）、Kaija Saariaho（1952- ）、Hans Tutschku（1966- ）、Jacopo Baboni Schilingi（1971- ）、Natasha Barrett（1972-)、今井慎太郎（1974- ）などは、この分野の代表的な作曲家として挙げられる。緻密で複雑な音響的変化の性質を持つFixed Mediaと、演奏家によるライブ演奏の融合・共鳴は、観客に斬新な聴取体現をもたらすことができる点において革新的といえよう。また、Fixed Mediaの高い保存性や、複雑な設定を必要とせず、すぐに演奏できることから、Mixed musicは現代のコンピュータ音楽領域においても主流の創作、演奏スタイルであると言える（Emmerson 2011；Manning 2013；Roads 2016）。

1.2 同期問題

本研究における『同期問題』とは、ライブ演奏において、演奏者がFixed Mediaと演奏時間の同期を試みる際に生じる様々な困難を示している。具体的な困難とは、第 1 に、Fixed Mediaの再生時間が固定されているため、演奏者が自分の演奏時間を自由に表現することができないこと。そして第 2 に、Fixed Mediaの音楽的特徴に起因する同期の難しさである。Fixed Mediaには、音の変容を追求する作曲手法とその作曲過程により、人間の聴取では明確に把握できない時間やリズムの移り変わりが多く存在する。例えば、長時間にわたって徐々に変化する音色、きわめて短い時間間隔（例えば50ミリ秒以下）で瞬時に変化するリズム、極端な加速・減速の時間変化、ランダムで不規則なリズムの変化などである（Roads 2001；2016）。もちろん、上記のような同期の問題が発生しない場合もある。例えば、作曲者が意

[5] 本稿でいう「現代音楽」とは、クラシック音楽の延長線上に存在し、主に20世紀初頭から現在に至るまでに派生した音楽のカテゴリーを総称したものを指す。

図的にFixed Mediaとの厳密な同期を要求していない作品や、変化が明確で、確実に把握し得るFixed Mediaに同期させる作品、あるいは、比較的短いFixed Media（例えば30秒以下）を用い、冒頭のみ同期させてしまえば、以降は同期が求められない作品などである。このような楽曲の場合、上記のような同期の問題というのは生じることがない。一方で、例えば、長い時間にわたって（例えば1分以上）の時間変化を含むFixed Mediaの音響と厳密な同期を要求するような場合においては、Fixed Mediaの再生時間と厳密に同期させることは難しい（Ding 2006；McNutt 2003；Rowe 1999）。本研究は、このように困難の伴う楽曲に対して有効的な「演奏同期システム」[6]の開発に関するものである。

1.3 演奏同期システムの歴史

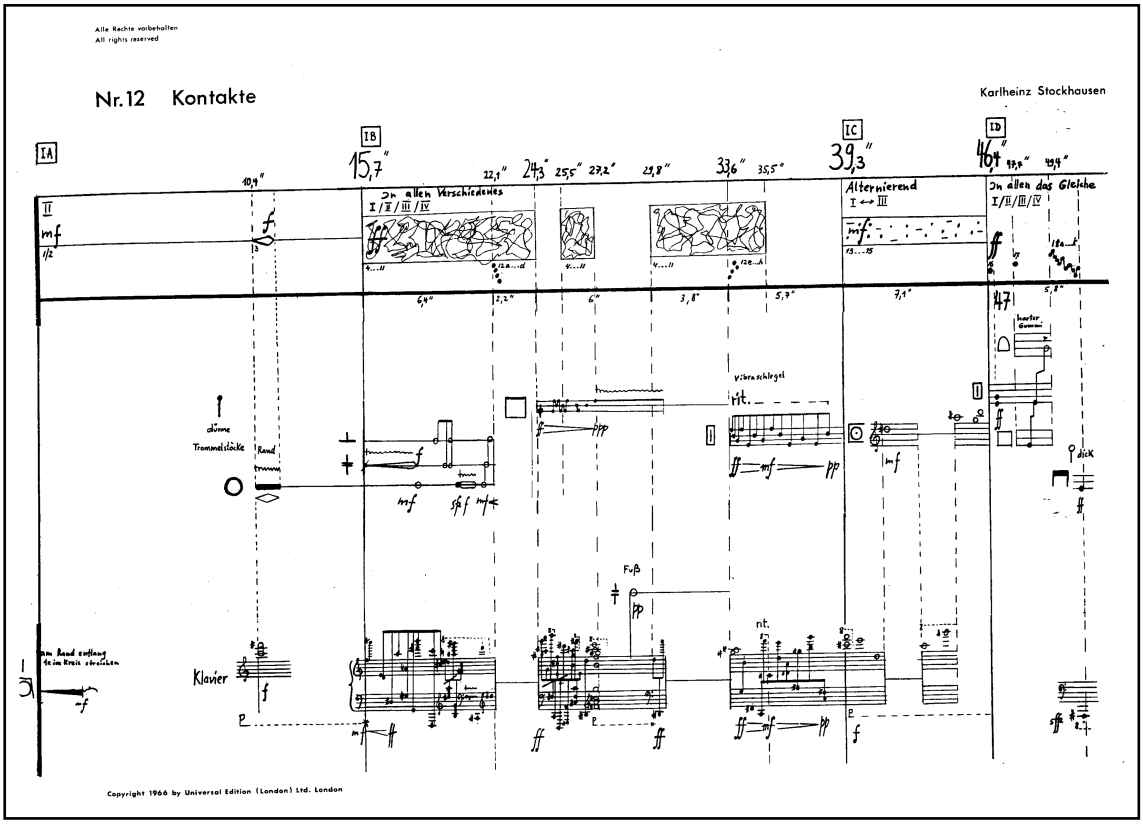
これまで使用・考案されてきた演奏同期システムを分類すると、大きく3つに分けることができる。（1）キューの送信、（2）マルチトラック即時再生システム、（3）Score Following、これら3つである。これらの技術は、それぞれ異なるアプローチに基づいて発案されているが、いずれも、演奏者や作曲者に様々な制約を課してしまうという点において同様のデメリットを秘めている。新しい演奏同期システムが最終的に目指すのは、既存の技術では得られない「無制約」を実現させることである。そのため、まずは、既存技術の長所と短所について詳しく知る必要がある。次項ではまず、既存技術の歴史とその種類・特徴について紹介する（ローズ 2001；Emmerson 2011；Stroppa 1999）。

1.3.1 キューの送信

キュー（信号）の送信は、演奏同期システムの最も伝統的な方法と考えられている。ライブ演奏中に、何らかの形で演奏者にキューが送られ、それを頼りに、Fixed Mediaの時間に同期して演奏するものである。このアプローチに着想されたのが、クリックトラックやストップウォッチを用いる手法である。1950年代の、このアプローチが生み出されてから日が浅い段階では、外部のデバイスは使われておらず、Fixed Mediaの変化を表す情報（テキスト、グラフ、音符など）を演奏者の見る楽譜上に記載することで同期演奏を図っていた。例えば、K. Stockhausenの2人の打楽器奏者と電子音楽のための《Kontakte》（1960）は、この手法を用いた初期の作品の一つである。「図例1.3.1.1」は、2人の演奏者が使用する演奏楽譜の一部である。演奏部分の上部に電子音の変化を表すグラフや文字、時間などの情報を提示することで、演奏者はライブで電子音を聴きながら記載されている情報と見比べることで、Fixed Mediaとの同期を試みることができる。しかし、この作品は電子音響（Fixed Media）と0.1秒までの正確さで同期するよう要求されるため、作曲者の意図する同期を実現するには、と

---

[6] ライブ演奏において、演奏者の時間とFixed Mediaの再生時間を合わせるための補助的な方法・技術を指す。このような手法や技術に関しては先行研究の例が少ないため、「演奏同期システム」という用語自体は筆者の考案による呼称である。



「譜例1.3.1.1」K. Stockhausen作曲による《Kontakte》の演奏楽譜（1ページ目）。
上段には電子音響の楽譜、
中段と下段はライブ演奏者の演奏パートの楽譜が記されている。

てつもなく膨大な練習時間が求められている。このような「記譜」によるキューイングは、外部デバイスを必要とせず、舞台上のセッティングも簡略化できるという利点があるため、厳密な同期を必要としない、ある程度の同期の「自由度」が認める作品にはよく用いられる方法である。しかし仮に《Kontakte》までの精度を求めるのであれば、記譜だけでは事実、不十分であるといえよう。実際の演奏でも、この方法だけでは、Fixed Mediaとの厳密な同期をとることは困難とされているのが現実である（Barrett 1997；Emmerson 2011；Truax 2016）。

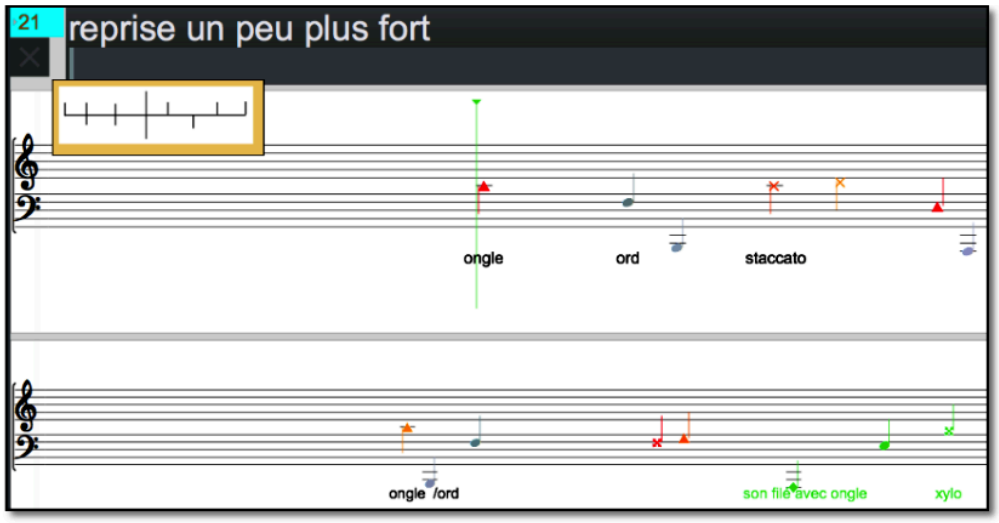
ライブ中に「ストップウォッチ」を導入する手法も、Mixed musicが発展した初期（1960年代）から使われている手法である（Chadabe 1997）。多くの場合、演奏者の演奏パートには「タイミングのカウンター」が記載されていて、その時間を参照しながら、演奏者はFixed Mediaと同期を図ることができる。例えば、Luigi Nonoの声とテープのための《La fabbrica illuminata》(1964)、Jean-Claude Rissetのフルートとテープのための《Passages》（1982）は、この手法を用いた初期の代表的作品である。電子音響の再生時間を具体的に提示できることで、前述の「記譜」によるキューイングに比べ、より精度の高い同期が実現できる。し

かし、ストップウォッチが提示される「速度」、すなわち演奏のテンポは一定であり（1秒刻みでは60BPMで、0.1秒刻みでは600BPMである）、演奏者の演奏パートでテンポの変化や、複雑なリズム表記が必要な作品では、この手法を使うことが困難とされている。

次に紹介するのは「クリックトラック」と呼ばれる方法を用いる案である。演奏者に連続的なインパルス信号[7]を送信することで、Fixed Mediaの時間に同期をさせる方法「クリックトラック」において、Emmanuel Ghent（1925-2003）は先駆者といえよう。彼は1967年に発表した論文「演奏者へのプログラムされた信号：新しい作曲のリソース（Programmed Signals to Performers: A New Compositional Resource）」の中で、自分の作品に存在する複雑な同期の問題を改善し解決するために、どのようにクリックトラックを使用したかを説明している。例えば、アンサンブル演奏におけるポリテンポの同期や、長いFixed Media作品との同期の達成などである。彼のトランペットとテープ音楽のため《Hex, An Ellipsis》（1966）はMixed musicの音楽の歴史において、クリックトラックを使用して同期演奏を実現した先駆的な作品の1つと考えられる（Ghent 1967）。クリックトラックは、基本的にテンポに合わせたインパルス音であるため、テンポレベルでの高い精度で同期することが可能である。しかしその一方で、演奏者の演奏時間が完全に「グリット」の中で拘束されてしまうこと、そして演奏者が常に与えられるインパルス（情報）について注意を払わなければならいため、複雑なリズム・テンポ変化を伴う作品には適しておらず、実際、クリックトラックに批判的な音楽家も少なくない（McNutt 2003）。

こういった「キューの送信」を活用した手法の中で、比較的新しい手法としては、「スクリーンスコア」（screen-score）が挙げられる。これは、作曲者が演奏者の楽譜をコンピュータの画面上に表示し、その上に時間の経過を示すことができる視覚的変化（カーソルやスライダーの移動、楽譜のスクロールなど）を作り、映し出すことで、演奏者がそれに合わせてFixed Mediaの時間と同期できるという手法である。スクリーンスコアという言葉は、Jonathan Bell（1982- ）氏が2016年に提出した博士論文「オーディオスコア：作曲とコンピュータ支援演奏のためのリソース（Audio-Scores: A Resource for Composition and Computer-Aided Performance）」に由来する。彼が提案した方法は、演奏者が複雑かつ細かいリズムやピッチ変化を含むライブ演奏を実現させる支援的ツール（オーディオとビジュアルを伴う）を提供するというものであった。彼はこの手法について 「小節、拍、またはプロラツィオ（Prolation）といった伝統的な概念に依存しない、正確なリズム記譜の原型を明らかにする（筆者訳）」と述べ、その利便性を強調した（Bell 2016）。「譜例1.3.1.2」は彼の作品、ハープとエレクトロニクスのための《Archipel》（2015）がシステムを使用して演奏された時のスクリーン上の抜粋である。スクリーンスコアでは、演奏者が「コンピュータの演奏」を追従・参照して同期をとるため、高精度かつ繊細なタイミングの表現力のある同期演奏が期待で

[7] 非常に短い音。音を使う以外、視覚的な信号（多くの場合、小節や拍の表示を伴う）を使用することもある。



「譜例1.3.1.2」Jonathan Bell作曲による《Archipel》
スクリーンスコア同期技術を使用した演奏楽譜の画面。

きる。しかしこの方法では、演奏者はコンピュータの時間に合わせて演奏表現をするため、とくに演奏時間に関しては自発的な演奏表現が制限されてしまう。ひいては、作曲家に対する作曲上の表現の制限に繋がることは、言わずもがな必至である。

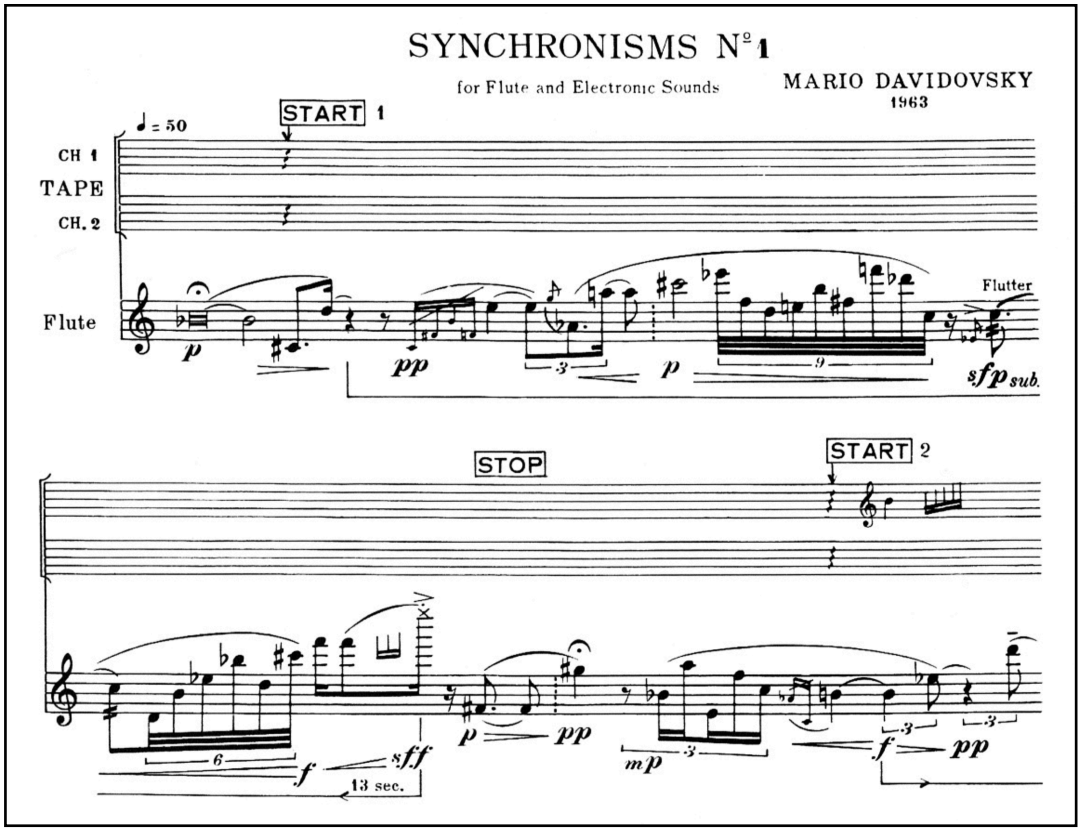
1.3.2 マルチとラック即時再生システム

一方で、演奏者の演奏時間の表現の自由とともに、演奏パートのリズム表記に対して制約なく複雑な作曲をも可能にする同期システムが「マルチトラック即時再生システム」である。この技術は、長いFixed Mediaのオーディオファイルを断片化し、ライブ演奏の演奏者のタイミングに合わせて各断片をトリガー[8]することで（各断片について必要なフェードイン / アウト処理を加える必要がある）同期を図るものである。Fixed Mediaの各断片は、演奏者が自由に表現できる時間領域が確保するために、作曲の段階の中で長めに作曲しておく必要がある（Barrett 1997；Emmerson 2011；McNutt 2003）。

短いFixed Mediaの断片を連続的にトリガーするというアイデアに関しては、Mario Davidovskyの《Synchronisms》シリーズ（「譜例 1.3.2.1」を参照）が早期の試みと言えよう。しかし比較的長い（例えば 1 分以上）Fixed Mediaを繋ぎ合わせて再生することが技術的に可能になったのは、コンピュータによるマルチトラックの再生が一般化した1990年代以降である。Kaija Saariaho、Hans Tutschku、Natasha Barrett、今井慎太郎、Jacopo Baboni Schilingiは、この演奏同期システムを用いて多くの名作を作曲した作曲家である。

マルチトラック即時再生システムにより、演奏者自身によるタイミングのコントロールによって、自由度の高い同期演奏が可能になった。そして演奏パートの作曲に対しても、ほとん

[8] オーディオファイル（Fixed Media）の再生を実行すること。

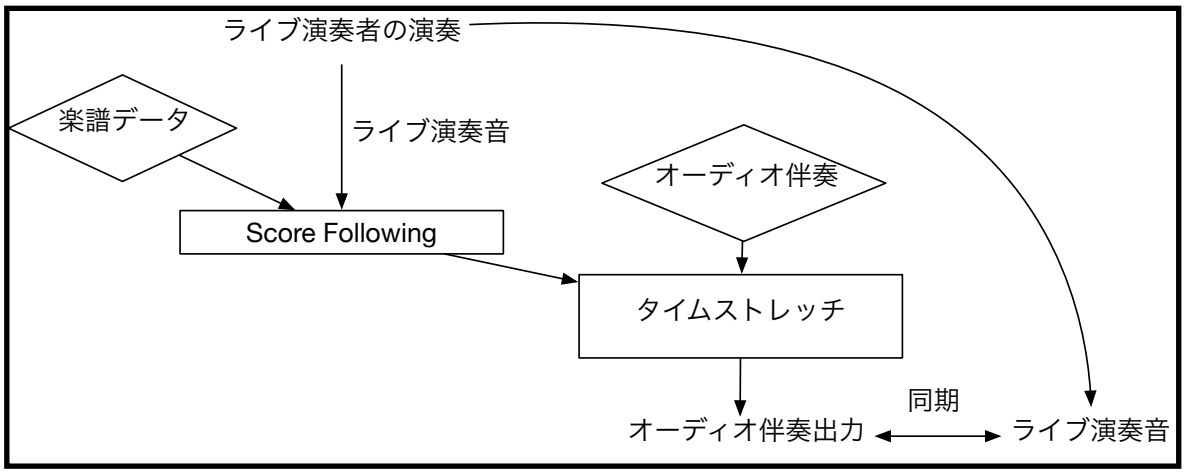


「譜例1.3.2.1」Mario Davidovsky作曲による
『フルートとエレクトロニクスのための《Synchronisms No.1》（1963）』。
楽譜の中で四角に囲まれたStartとStopはFixed Mediaの再生指示である。

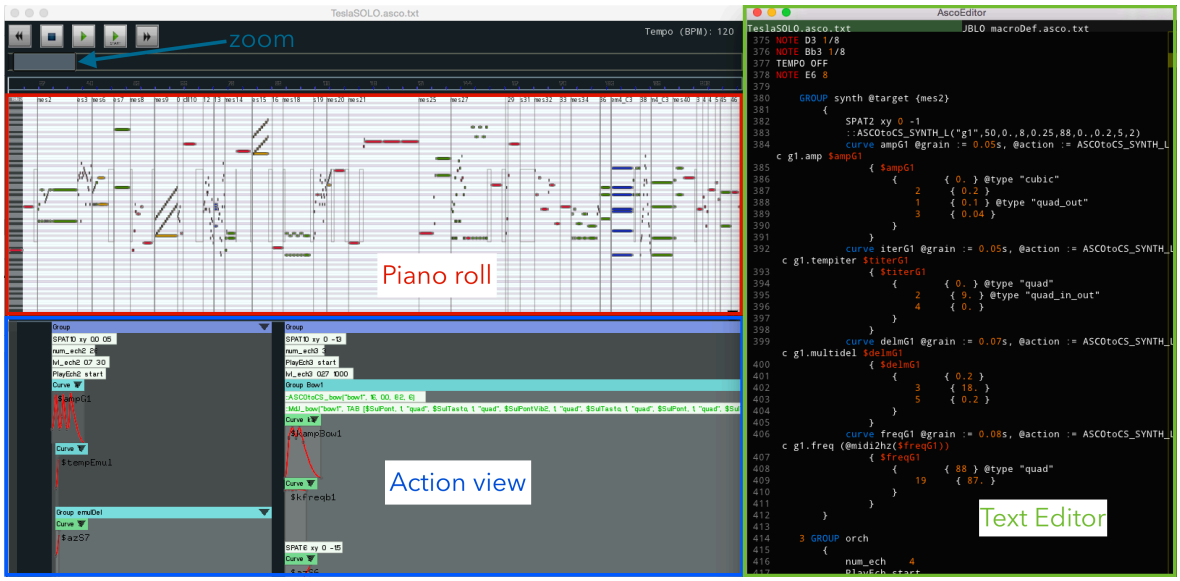
どの制限なく、複雑なリズムや音色の変化を取り入れることが可能になった。ただし、この方法を使うにはFixed Mediaの断片化が前提になるため、長い時間にわたり、連続的な変化を含むようなFixed Mediaに対しては使用することができない。この点は、Fixed Mediaパートの作曲に対する制約と言えよう。

1.3.3 Score Following

Score Followingは、演奏同期システムの中で最も複雑なアプローチに基づく手法と言える。この技術は、コンピュータ音楽の分野の中で30年以上の開発の歴史があり、一般的には、演奏者に追随するコンピュータを用いた自動伴奏システムとして開発されている（Puckette 1992）。「図例1.3.3.1」は、この技術の仕組みを図式化したものである。この技術の基本的な仕組みは、ライブ演奏中に、コンピュータが演奏者の演奏音を認識し、あらかじめコンピュータに入力されている演奏者の楽譜データと照合しながら、演奏者の演奏時間を「予測」し、演奏の同期を実現しようとするものである。オーディオデータを伴奏として利用する場合、音声素材の再生速度とピッチ、それら双方をそれぞれ独立して、別個に制御でき



「図例1.3.3.1」Score Following技術を用いた
演奏同期システムの模式図。



「図例1.3.3.2」Antescofoの使用時の画面。

るオーディオ・タイムストレッチ技術[9]が必要になる。このようなシステムは、一般にコンピュータによる自動伴奏システムと呼ばれている。例えば、IRCAMが開発した『Antescofo』[10]「図例1.3.3.2」は、このカテゴリーの代表的な開発例である。その他にも、

[9] オーディオ信号処理技術の一つ。詳しくは以下のリンクを参照。
https://ja.wikipedia.org/wiki/タイムストレッチ/ピッチシフト (accessed September 7, 2020).

[10] Antescofoホームページ
https://forum.ircam.fr/projects/detail/antescofo/ (accessed September 7, 2020).

Christopher Raphael氏の『Music Plus One』[11]や、Andrew Robertson氏とMark Plumbley氏の『B-Keeper』（Robertson and Plumbley 2007）、Antescofoの開発者・Arshia Cort氏を中心とするチームが開発したiPhone/iPadアプリ『Metronaut』[12]などが有名なものとして挙げられる。それらのシステムは、現状、クラシック音楽、ポップス、ジャズなど、コンピュータが演奏者のリズムや音程を明確に聞き取ることができる音楽領域において、高効率で自由なライブ同期を実現することが可能である[13]。しかし筆者の知る限り、Mixed musicの分野に活用された演奏報告や検証はなく、Score Following技術がMixed Mediaの同期問題の改善・解決に繋がるか否かを客観的に検証し得るデータが存在しないのが現状である。そのため、特にIRCAMのAntescofoについて、より専門的に検証を行っていた。筆者はこの技術こそ、現在のMixed musicのジャンルにおいて、作曲家の新しい音楽作品に自由に取り入れることができる唯一の技術であると捉えている。本稿では、Antescofoの実使用体験を交えながら、Score Following技術における同期問題解決の可能性について考察してみたい。

Antescofoは、IRCAMが開発したタイムストレッチを可能にする『Max』[14]の外部モジュール『SuperVP for Max』[15]と組み合わせて、プログラミング言語Maxによる同期システムとして実装することが可能である[16]。「図例1.3.3.3」は、Maxを用いた同期システムの実装例である。左側（①の部分）が演奏者の演奏音をAntescofoに入力する仕組み、右側（②の部分）はAntescofoが演奏者の演奏音を解析してタイムストレッチ技術によりFixed Mediaの再生時間を変化させる仕組みである。このように演奏者の演奏音をシステムに送信することで、伴奏音源と同期したライブ演奏を実現することができる。筆者はかつて、IRCAMのCursusコースという作曲家向けのコースに参加しており、そこでデモ音源や自分の作品に実装してテストを行った。これらのテスト結果から、筆者はAntescofoには間違いなく、クリックトラックなどの補助的な手法を必要とせずとも、柔軟なライブ同期を約束するものであると考えられるが、しかし、この技術にも課題が存在しているのもまた事実である。

---

[11] Music Plus Oneホームページ
https://music.informatics.indiana.edu/~craphael/music_plus_one/ (accessed September 7, 2020).

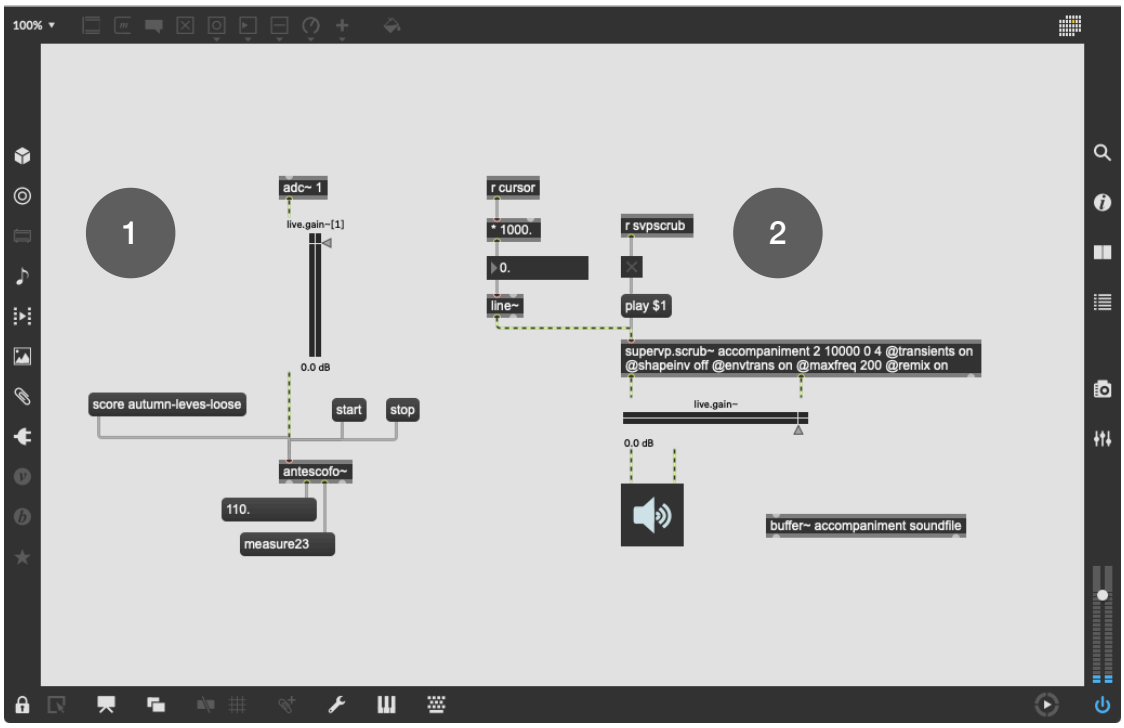
[12] Metronautホームページ
https://www.metronautapp.com (accessed September 7, 2020).

[13] Antescofoによるオーディオ伴奏を使用したライブ同期演奏の一例
https://youtu.be/YkMGtpcAA04 (accessed September 7, 2020).

[14] 音楽とマルチメディアのためのビジュアルプログラミング言語。詳細は以下を参照。
https://cycling74.com

[15] SuperVP for Maxホームページ
（https://forum.ircam.fr/projects/detail/supervp-for-max/ (accessed September 7, 2020).

[16] Antescofoによるオーディオ音源の自動伴奏システムの構築の仕方については以下のリンクを参照できる。
https://discussion.forum.ircam.fr/t/antescofo-and-supervp-for-automatic-accompaniment/1227
(accessed September 7, 2020).



「図例1.3.3.3」 Maxを用いたAntescofoの実装画面。

第１の問題点は、Machine Listening技術[17]の制約性である。Antescofoには、複雑な楽器演奏のライブ・オーディオ信号（連続和音や装飾音など）を高い精度で識別できる、これまでで最も先駆的なMachine Listening技術が搭載されている。しかし、微分音による変化、音高が不明瞭な楽器の特殊奏法による変化、複雑なリズムの変化など、明確には聞き取れない微細で複雑な音色の変化の中には、現段階でも、ライブ演奏で高精度に聞き取ることが困難なものも多くある。もちろん、システムは明確に「認識」できるような音楽作品で用いる場合において、システムは高い安定性を持って、使用することができるが。複雑な演奏の中での音響的な変化を重視する現代音楽の作曲では、この技術を高い安定性で使用することは難しい。そのため、どうしても作曲の表現範囲に制約が生じる。

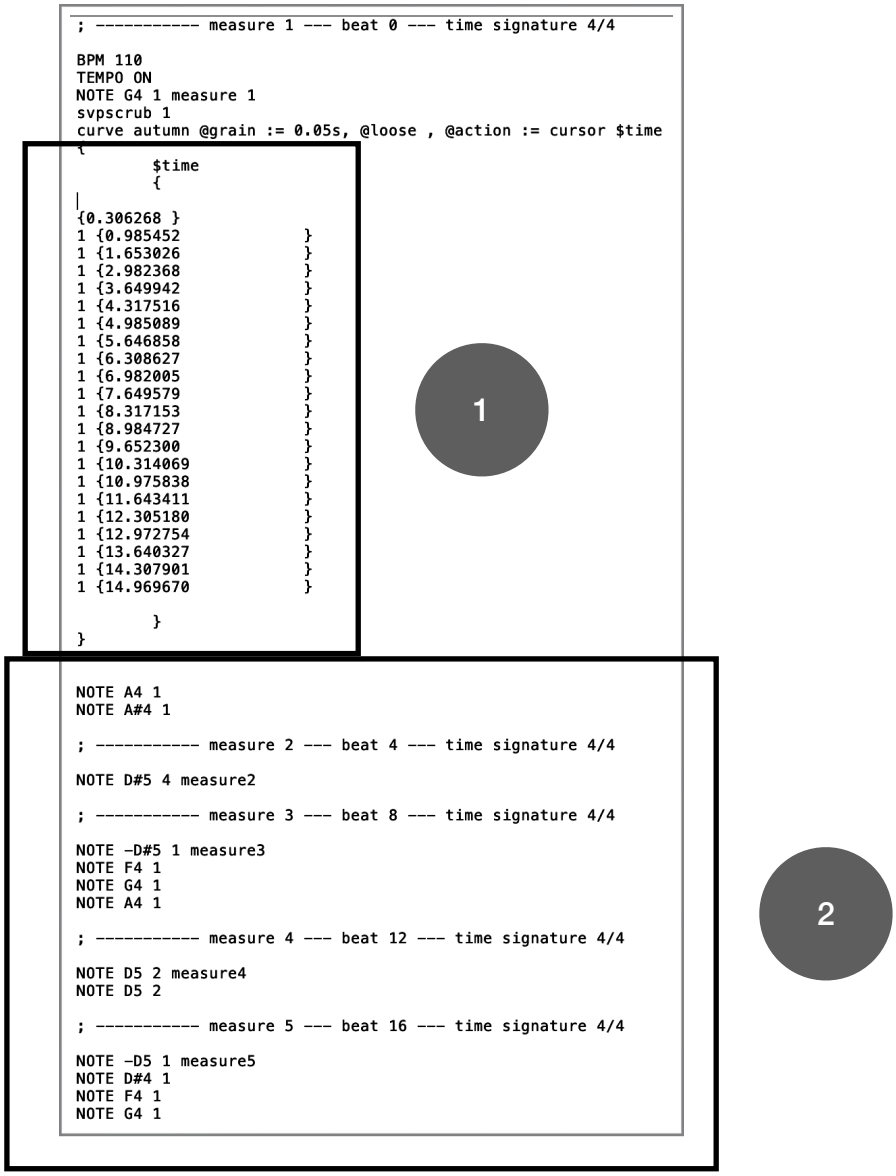
第２の問題点は、タイムストレッチ手法の適用性である。Antescofoは、「SuperVP for Max」の「supervp.scrub~」オブジェクト[18]で実装することを想定しており、このオブジェクトにおけるタイムストレッチ技術を使うためにシステムが提供するパラメータは、オーディオデータの再生時間（位置）の変数である。SuperVP は歴史的に有名な Phase Vocoder技術として捉えられ、特にFixed Mediaの作曲における複雑な音響を作成や、合成する際に広く用いられる。しかし、オーディオデータの音質を維持しながら再生時間を変更するという点におい

[17] Score Following技術を構成する最重要部分。あらかじめ入力されたスコアのデータと、実際の演奏音を照合するための技術である。

[18] プログラムのなかで、ある特定の機能を担うブロック。

ては、現時点では最良の選択とは言えない（詳細は後述（1.4 節で述べる））。特に、本研究のシステム構築では、Fixed Mediaの音質を最適化することが重要なポイントであるため、Antescofoが提供する再生位置パラメータは、より高品質なタイムストレッチ技術に実装することができないのが現状である。そのため、Fixed Mediaをより高音質で再生することが難しく、作曲者にとって最適な状態の中でFixed Mediaを再生させられないという問題がある。

第３の問題点は、システムセットアップの複雑さにある。Score Followingを利用するためには、システムが演奏者の演奏音と照合するためのスコアをあらかじめ用意しておく必要がある。Antescofoでこのようなスコアを作成するためには、演奏者の演奏譜の情報を反映させるだけでなく、オーディオデータの再生時間を演奏譜の情報と何らかの時間単位（１拍、２拍、特定の時間など）に提示し、それらの情報が正確に一致していなければならない。そのプロ



「図例1.3.3.4」Antescofoのシステム用スコアの一例。

セスは非常に複雑なものである。「図例1.3.3.4」は、Antescofoが用いるシステム用スコアのテキストファイルの抜粋である[19]。このスコアの中で表示されている①の部分は、楽譜に存在する音符に対するオーディオデータの再生時間を示している。②の部分は、演奏者の演奏音をMIDI情報として示したものである。このような「スコア」を用意するためには、２つの異なる情報を完全に一致させるために、多くの時間や労力を費やすことになる。特に長大な作品を作曲する場合、単純に演奏譜のリズムを１つ修正するだけでも、その都度、全体的に楽譜を作り直さなければならない。このため、システムの準備段階で、作曲家に大きなストレスを与えてしまう。

上記に挙げた３つの問題点から、筆者は本稿が提出された現時点（2020年12月）において、Antescofoが同期問題の改善・解決策として現実的な手法とは言い難いと考えている。また、Antescofoは非常に「不透明」であり、音楽家が直接その内容にアクセスする機会が非常に限られているため、専門的なエンジニアからのサポートなしには、安易に使用することができない。今後、AntescofoをはじめとするScore Following技術が、より多くの音楽スタイルに対応できるよう進化していくことが望まれるが、現時点では、そのような手法を直接利用して有効な結果を得ることは困難であると筆者は考えている。

1.3.4 更なる音楽的な演奏同期の達成へ

以上が、既存の演奏同期システムの振り返りである。先人の音楽家・開発者たちの功労により、Mixed musicは著しい発展を遂げることができた。しかし、キューの送信を用いる方法では、Fixed Mediaを自由に作曲することができる一方で、演奏者が自分の演奏時間を自由に表現することができないという問題点があった。マルチトラック即時再生システムを用いる方法では、演奏者は自分の演奏時間を自由に表現できるが、Fixed Mediaを断片化する必要があるため、Fixed Mediaの作曲に制限が課されるという問題点があった。Score Following技術は、この２つの制約を解消する可能性を持っているが、現状では有効に活用できるシステムが存在しないという課題があった。

このようにそれぞれの既存の手法には、解決に至っていない制約や問題が残されており、演奏者による自由なライブ演奏とFixed Mediaによる自由な作曲を可能にしながら、厳密なライブ同期を実現する方法は、今のところ存在していないのが現状である。だが、このことを換言するならば、多くの開発の余地、そして可能性が残されているとも言えよう。これからの演奏同期システムに求められる到達点とは、演奏者の演奏時間とFixed Mediaの再生時間の双方に制約がない、柔軟なライブ同期を実現することであると筆者は考えている。そのためには従来の技術をそのまま用いるだけでなく、新たな同期技術を創出させる必要があると考えている。

19　Joseph Kosma作曲《Autumn Leaves》冒頭５小節のみを用いた一例。

1.4 タイムストレッチ技術における新たな可能性

同期問題を解決する新たな可能性として、筆者は近年進化しているリアルタイム型のタイムストレッチ技術に注目している。この技術は、先に紹介したScore Following技術に用いられているタイムストレッチ技術と同様のアプローチに基づいて開発されているが、この新しいタイプのタイムストレッチ技術には、２つの大きな特徴がある。１つは、オーディオ素材の音質を維持したまま、再生時間をリアルタイムに制御できることである。もう１つは、音楽家たちが自分のアプローチに合わせて自由にプログラミングできることである。例えば、Abletonの『Ableton Live』[20]や、Cycling '74のMaxの「groove~」オブジェクトはこのようなことが実現できるタイムストレッチ技術が組み込まれている。例えば、ライブ演奏において、このようなタイムストレッチ技術を使って、Fixed Mediaの再生時間を何らかの方法で伸縮させ、演奏者の再生時間に同期させることができれば、両者のパートに制約のない同期したパフォーマンスが実現可能になると考えられる。前節では、Score Following技術を併用する可能性について述べた。Score Following技術における技術的な課題が残っているため、本研究における同期問題の改善に使用することは、期待できる同期演奏の結果を引き出すことは難しいと考える。そのため、新しいタイムストレッチ技術を使用する前提とした、新しいアプローチに基づいて考案する必要があると筆者は考えている。

1.5 新しいシステムの開発における歴史的展望

Mixed musicは、これまで半世紀以上の発展の歴史がある。作曲家たちは、この音楽スタイルで、複雑で表現力豊かな音楽作品を作り続けてきた。しかし、筆者の考えでは、このような複雑な電子音によるライブ演奏同期の問題は、厳密には大きな改善はされていないように思う。現時点では、演奏者が自由に演奏時間を表現でき、かつ厳密な時間同期を実現できる方法は、依然にして「1.3.2」節で紹介したマルチトラック即時再生システムのみである。双方のパートに対して制約を課されることなく、自由なライブ演奏同期を実現できる技術は現段階に存在していないと言える。この30年間のコンピュータ音楽分野の出版物の中には、このようなライブ同期の問題を論じたものが少なからずある。しかし、それらの議論の多くは、「音楽的時間」と「機械的時間」、「リアルタイム音楽」と「非リアルタイム音楽」（Mixed musicを指す）、「時間における人間の表現の可能性」と「作曲者の同期精度への期待」という問題が、音楽哲学、美学、心理学の分野の中で議論されており、いずれも、Fixed Mediaの再生時間が一定であることを前提としている（Emmerson 1994；Hagan 2016；Stroppa 1999）。現在のタイムストレッチ技術は、Fixed Mediaを固定的ではなく、可変的なものにすることができる。この利点を利用して、新たな「自由」を実現する演奏同期システムをどう構築するかが課題である。そのような新しい演奏同期システムは、必ずしも

[20] ライブ演奏に関する様々な機能に特化したDAW（デジタルオーディオワークステーション）ソフトウェアの一つ。詳細は以下を参照。https://www.ableton.com

従来の演奏同期システムの代替にはならないかもしれないが、Mixed music作品における演奏と作曲の新たな可能性をもたらしてくれるのではないかと確信している。本研究はここからの延長線上のものである。新しいタイムストレッチ技術を用いて、Mixed musicの同期演奏に対する自由な可能性を試みる。次の章の中では、まず、本研究の同期問題の改善、解決するに向けて、タイムストレッチ技術を使用して問題解決するための方法について論じる。

第２章 課題とその解決方法

2.1 概要

本章では、タイムストレッチ技術の応用を前提に、Mixed musicに対して効果の高い演奏同期システムの開発方法について論じる。タイムストレッチ技術（リアルタイム型）は、ライブ中のFixed Mediaの再生時間を伸縮させる可能性を持っている。このような可能性を具体的にどのように利用して、本研究における効果の高い同期演奏を実現することができるのか。筆者はMixed musicにおける音楽的な特殊性と、本研究の研究目標から、以下の３つの課題をより専門的に考察することが、効果の高い演奏同期システムを実現するための重要なポイントと繋がると考えている。(1）どのような同期時間間隔を用いるか、（2）どのようなインタラクティブ手法を用いるか、（3）どのような同期メソッドを用いるか、これら３つの課題である。本章では、これら３つの課題を、Mixed musicの発展の歴史や、筆者自身の創作・演奏体験と照らし合わせながら論じることによって、タイムストレッチ技術を使用した新しい演奏同期システムの開発方法を考察していく。

2.2 同期時間間隔について

まず、同期時間間隔の問題について説明する。同期時間間隔とは、演奏者の演奏時間とFixed Mediaの再生時間を「どの程度の頻度（細かさ）」で一致させるかという基準であり、同期演奏の音楽的結果に直結する重要な要素である。一般的な音楽ジャンル（クラシック、ポップス、ジャズ、ロックなど）では、この議論は不要であり、同期の必要性に応じて音楽のテンポを合わせることが前提である。前章で述べたScore Following技術は、基本的に演奏者のテンポレベル[21]で厳密な同期を実現することを目標に開発されたものである。しかし、現在のScore Following技術では、Mixed musicに対して効果的なライブ同期を実現することが困難である。そもそも、このレベルの同期を実現することは技術的に困難である（第１章を参照）。したがって、Mixed musicにおける同期の問題を改善するためには、テンポレベルでの同期を求めるのではなく、新たな時間間隔を設定・定義する必要がある。次に、どのような同期時間間隔を用いれば、有用性の高い同期を実現できるかという問題である。筆者は、Mixed musicの音楽的特性から、テンポレベルの同期時間間隔を用いるよりも、より長く同期時間間隔を設定することで、効果的なライブ同期を実現できると考えている。その理由は２つある。

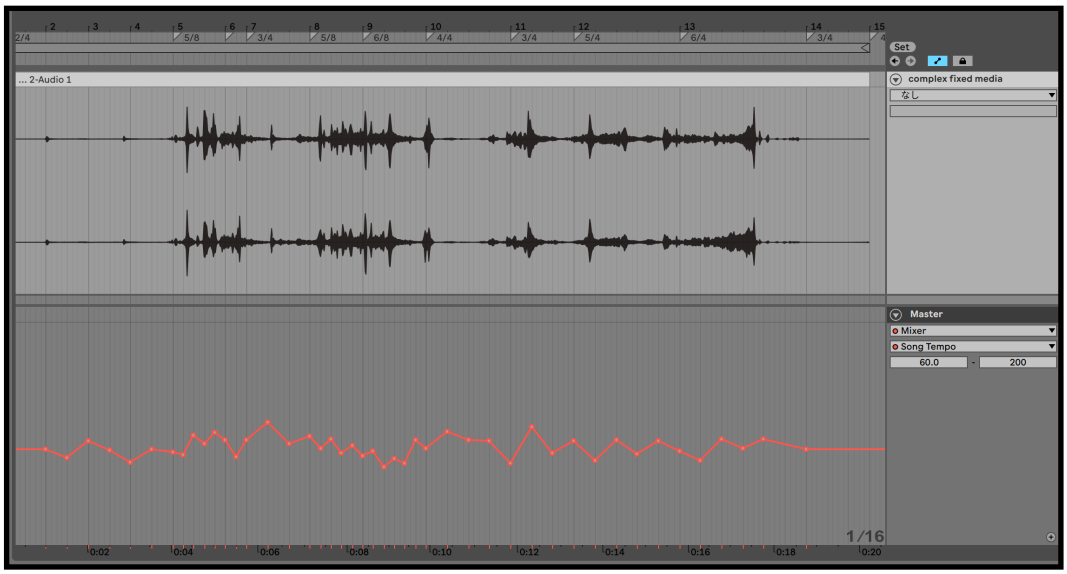
１つ目の理由は、Mixed musicの音楽性質にある。Mixed musicは、本質的に２つの音楽作品を組み合わせた音楽のスタイルと考えることができる。演奏者が演奏する楽器の音と、

[21] 人間が反応しうるテンポは、約30BPM（時間間隔：2秒）～240BPM（時間間隔：0.25秒）の間である（London 2004）。

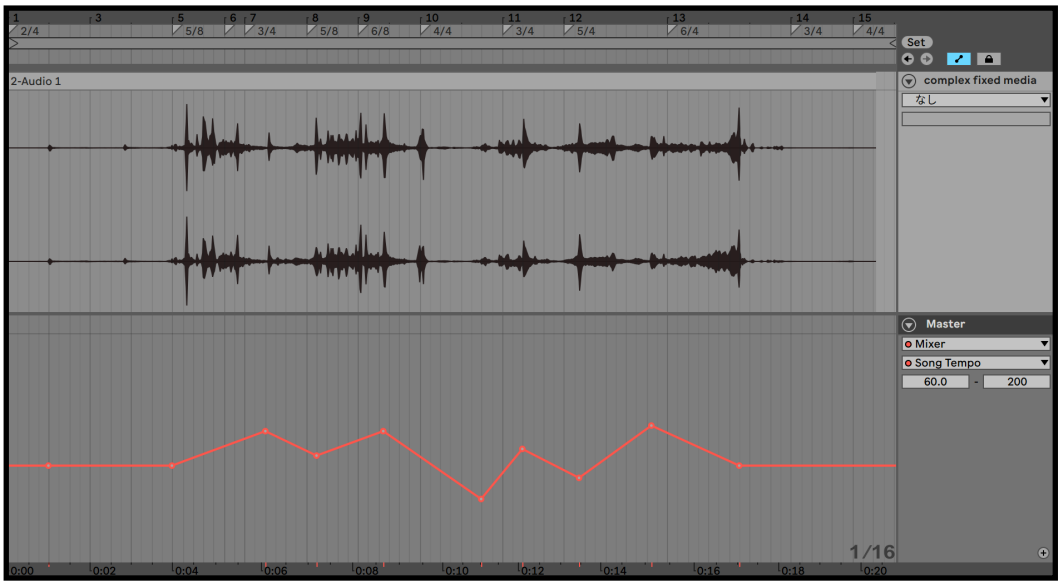
録音やコンピュータ技術によって記録・制作された音が、１つの作品の中で同時に演奏（再生）され、それぞれが全く異なるリズムと時間的変容の可能性を含んでいる（Road 2001；Road 2016；Smalley 1997；Stockhausen and Barikin 1962；Stroppa 1999；Toiviainen；2007）。双方の同期精度は、作曲者の意図によって決められる。例えば、K. Stockhausenの《Kontakte》（1960年）は極めて厳格な同期が要求され、歴史上でも、同期演奏の実現がもっとも難しい作品のひとつとされている。しかし、この作品で要求される厳格な同期は、２つの時間を高い頻度で関連付けるものではなく、指定された「時刻」の中でも厳密な一致性を求めるものである（楽譜上では0.1秒まで）。実際、二つの同期時刻の間の時間では、ほとんど厳密な同期を要求していない、演奏者は２つの同期時刻の幅の中で一定程度の自由な演奏時間の表現を認められている（「1.3.1節」を参照）。同様のアプローチは、マルチトラック即時再生システムにも見られる。このシステムは、再生されるFixed Media間の厳密な同期を要求するのではなく、作品の重要な「時刻」に新たなFixed Media断片を再生することで、音楽的な同期演奏を実現する。実際それらのFixed Media断片の中間部分に関してはほとんど厳密な同期を要求することはなく、演奏者が一定範囲内で自由に自らの演奏時間を伸縮することが可能である。クリックトラックを使えば、確かに２つの時間をテンポ領域で正確に合わせることができる。しかし、ここでのクリックトラックの使用は、演奏者側からFixed Mediaの時間に同期を取るための様々な「不可能性」を克服し、最終的に完成度の高いライブ同期演奏を実現するための補助的なものとして捉えられる。その場合、テンポ領域で２つの時間の同期をとることは必ずしも必要ではないと言える。このように、Mixed musicにおける同期時間間隔という概念の本質は極めて自由かつ芸術的なものであり、作曲者の意図によって自由に決めることができるものである。例えばタイムストレッチ技術を導入した演奏同期システムを開発しても、このような「伝統」を引き継ぐことができるのではないかと筆者は考えている。

２つ目の理由は、タイムストレッチ技術の応用における、Fixed Mediaの再生効果に対する影響にあると考えている。タイムストレッチ技術を用いて、Fixed Mediaの再生時間を変化させるということで、その変化させるための頻度（つまりここでの同期時間間隔）はそのFixed Mediaの最終的な音響結果に対して直接に影響を与える要素となる。しかし、筆者が何度も再生検証をおこなった結果、高い頻度（テンポレベル）で変化させるより、より低い頻度での再生時間を変化させる方が効果的であることが判明されている。

「図例2.2.1」と「図例2.2.2」は、この議論のために用意されたAbleton Liveのプロジェクトである。画面上部の波形は、筆者自身が作曲した20秒弱の短いFixed Media作品である。このFixed Media作品は、音響の変化に人間の演奏者によるライブ演奏が加わることが想定されており、演奏者が実行可能な演奏時間表記（テンポ、小節、拍子記号）などの情報が付加されている。この場合、ライブ演奏の際に演奏者は、計画時間に完全に従うことなく、一定程度の時間的な変動を生じることを想定する。そこで、Fixed Mediaの再生時間を伸縮させるこ



「図例2.2.1」テンポレベルでの同期時間間隔で変化させた例。



「図例2.2.2」 Fixed Mediaの時間構造に合わせて、
新たに設定した同期時間間隔で変化させた例。

とで、演奏者の演奏時間に合わせて、時間の同期を実現するようにプロジェクトを設定する[22]。「図例2.2.1」は、演奏者のテンポレベルに厳密な同期をとることを想定したものである。テンポレベル（120BPMを基準）で再生速度が高い頻度で変化するように設定したもの

[22] Ableton Liveでは、 指定されたオーディオデータの「Wrap」機能をオンにした状態でTempoパラメータ（画面の下部の赤い線）を操作することで、その再生速度を変化させることができる。

である。「図例2.2.2」は、演奏者のテンポレベルでの変動に影響をされることなく、Fixed Mediaの時間構造を考慮し、より長い時間幅で同期時間間隔（1.5〜４秒程度）を設定したものである。もちろん、理論的には「図例2.2.1」の方が、双方との細かい、厳密な同期を実現できると言えるが、タイムストレッチ技術を用いて、最終的に鳴らされたFixed Mediaの音響結果まで踏まえてみると、効果的と言えるのは、明らかに「図例2.2.2」の方であると言える。

もちろん、タイムストレッチ技術のアルゴリズムの違いも、そうした結果に影響される可能性がある。しかし、音楽的な観点からは、前者よりも後者の方が効果的な再生結果が得られると説明できると筆者は考えている。本研究では、人間の演奏者がリズムやテンポとして認識することが難しい複雑なリズムや時間変化を含むFixed Mediaを扱っている。このようなFixed Mediaのリズムや時間構造は、人間の表現のテンポレベルよりはるかに細かく、微細なレベルで構築されている。現在では、コンピュータ技術を駆使することで、オーディオのサンプリングレートレベルで極端な時間変化を実現することも不可能ではない。このようなFixed Mediaの再生時間を変化させる場合、その音楽的時間変化の特異な可能性の本質を考え、その時間構造を崩さずに再生時間を変化させる方法を考えることが必要である。「図例2.2.1」では、演奏者のテンポレベルに応じて再生時間を変化させている。これでは、高いレベルで行われたFixed Mediaのリズム表現が低いテンポレベルで再構築されてしまい、効果的な再生結果を得ることは難しい。一方、「図例2.2.2」では、Fixed Mediaのリズム変換に対する変化が同じ状態（加速と減速）で行われているため、従来の「個性」を失うことなく、時間的表情を加えた効果的な再生の結果に繋がると期待できる。このようなテストは、こういった筆者が作曲した短い音響断片だけでなく、K. Stockhausenの《Kontakte》の電子パートなど、筆者や他の作曲家のFixed Media作品でも行い、結果的に、Fixed Mediaのリズム・時間変容が複雑であればあるほど、そのような自由な時間幅を使用する同期時間間隔によって再生時間を変化させることが効果的であることがわかった。このような結果と上記の１つ目の理由と合わせて、長い時間に渡って、比較的自由に同期時間間隔を設けることが、本研究において非常に有効であると筆者は考える。

これらの理由から、筆者は本研究において、タイムストレッチ技術を用いた同期システムを開発するための一つの前提条件として、自由な同期時間間隔を用いるべきと考えている。新しい同期時間間隔は、双方の時間を音楽的に関連付ける可能性を提供し、Fixed Mediaを音楽的に再生することができると期待される。このことは、音楽同期において効果の高い演奏同期システムを実現するための重要な要素である。

2.3 インタラクティブ手法について

インタラクティブ手法は、ライブ演奏において、演奏者とコンピュータ（Fixed Media再生装置）がどのように連携するかを決定する要素である。つまり演奏同期システムを実行させるための仕組みである。第１章で述べた演奏同期システムの分類と同様に、オーディオデータと

ライブ演奏の同期を実現するインタラクティブ手法には、演奏者にキューの送信、トリガーの送信（マルチトラック即時再生システムと同じようなインタラクティブ手法）、Score Following技術、3つのタイプがある。この研究の目的では、制約のない演奏時間の実現のため、最初の方法である演奏者にキューの送信という手法は除外しなければならない。残る２つの方式は、２番目のキュー送信方式と３番目のScore Following技術である。もちろん、演奏者の演奏音が確実に認識できるようにシステムの設定が可能であれば、Score Following技術の応用は可能である。しかし、演奏者の演奏パートの構成に複雑さが要求される場合でも、自由に使えるという点では、トリガーの送信によるインタラクティブ手法が最も優れていると筆者は考える。

トリガーの送信によるインタラクティブ手法は、ライブの中、演奏者やエレクトロニクス・オペレーターがコンピュータにトリガーを送ることで、コンピュータ上に実行される電子的なプロセスを結びつけるものである。通常、トリガーを送るためには、MIDI機器（演奏者がトリガーを送るために使用するもの）やコンピュータのキーボードのキーを使用する。このインタラクティブ手法は、1980年代初頭にIRCAMで起源するインタラクティブ音楽[23]で初めて使用されて以来、30年以上にわたって使用されてきた。とりわけ複雑な表現力が求められる現代のコンピュータ音楽創作の分野では、今日でも、多くの音楽家に愛用されているインタラクティブ手法である。例えば、Pierre Boulez (1925-2016)、Cort Lippe (1953-)、莱孝之 (1954-)、Marco Stroppa (1959-)、Hans Tutschku、Jacopo Baboni Schilingi、Kaija Saariahoなどは、こういったインタラクティブ手法を用いて多くの名作を作曲した作曲家として挙げられる。

筆者は、この分野の学習を始めて以来、このインタラクティブ手法を使って多くの作品を作曲・演奏してきた。これらの経験を通して、このシンプルなインタラクティブ手法は、この研究にとって最も有効な手法であると確信している。例えば、筆者が作曲したヴィオラとエレクトロニクスのため《Au départ》（2017）[24]「譜例2.3.1」、スネアドラムとエレクトロニクスのための《To come into focus》（2017）[25]「図例2.3.2」は、演奏者の演奏パートに対して、音楽音響レベルで、複雑なリズム、ピッチなどの変化を多く必要とする極めて難易度の高い作品として作曲されている[26]。しかし、誰よりも作品を理解している演奏者や、作曲した筆

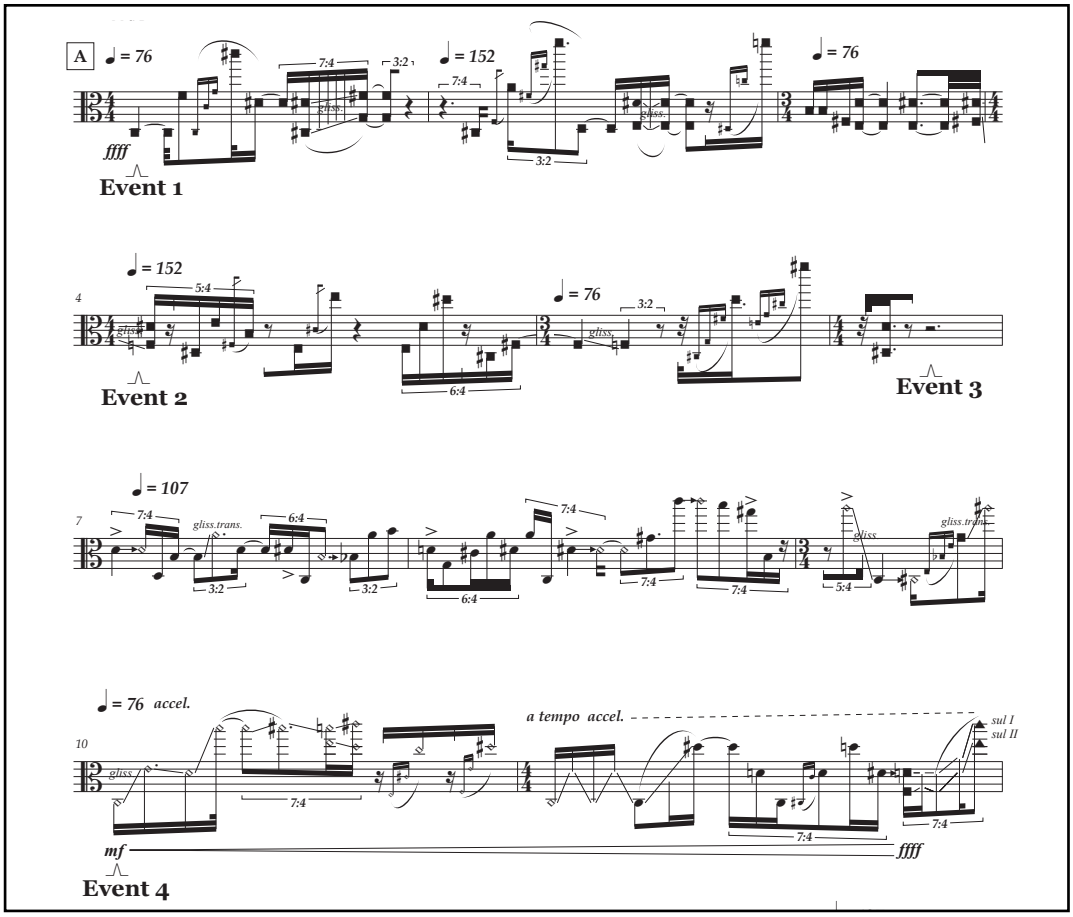
---

[23] 人間の演奏者（主にアコースティック楽器）によるライブ演奏と、コンピュータ上リアルタイムで実行される電子的な処理を組み合わせた音楽のスタイル。

[24] 《Au départ》演奏記録映像は以下リンクから視聴可能
https://youtu.be/gmw6d3uAQxs (accessed September 7, 2020).

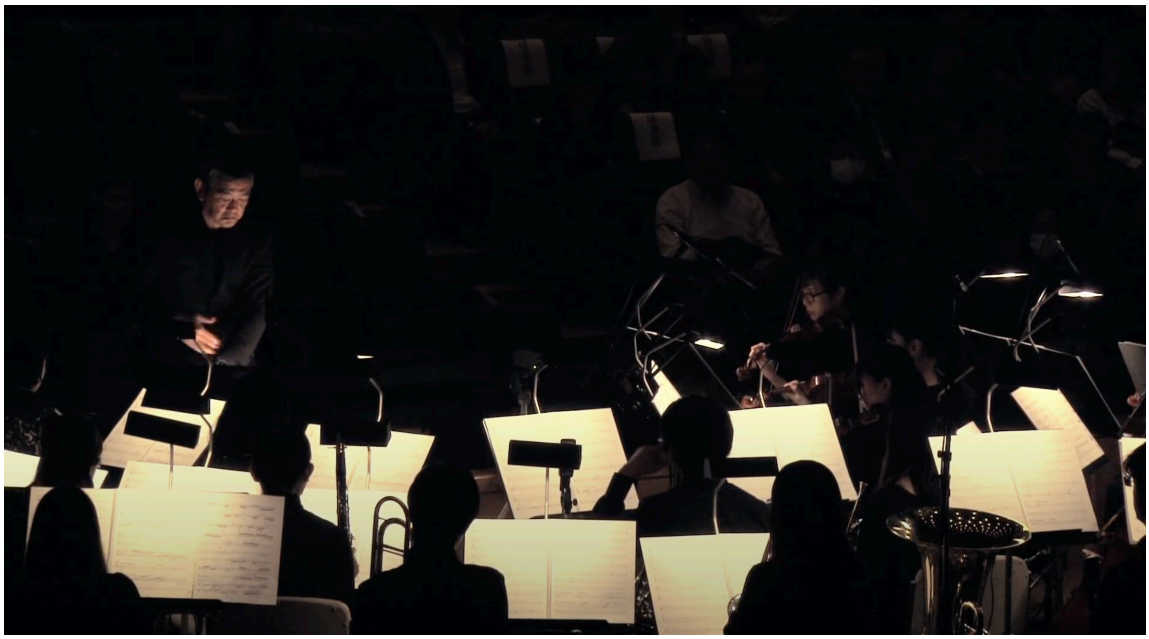
[25] 《To come into focus》演奏記録映像は以下リンクから視聴可能
https://youtu.be/1AJ4rCC6xlo (accessed September 7, 2020).

[26] どちらの作品も、Fixed Mediaを使わずに、演奏者のライブ演奏音をコンピュータ上に用意した様々なエフェクトによるリアルタイム処理のみとなる。異なるリアルタイム処理が行われるきっかけを作るために、トリガーを送るインタラクティブ手法が用いられている。



「譜例2.3.1」作品《Au départ》の冒頭部の楽譜。
五線の下にある「Event 1」「Event 2」は演奏中トリガーを送る時刻。

「図例2.3.2」パーカッション奏者・悪原至氏による
《To come into focus》の演奏の様子。



「図例2.3.3」《Répons》演奏中の様子。

者がトリガーを送ることができるため、これまで世界中で行われてきた数多くのライブ演奏では、全てのトリガーは正確なタイミングの中で確実に送ることができ、同期上での問題は一度も生じることがなかった。また、自作だけでなく、他の作曲家の作品のライブにも演奏者として参加した経験がある。最も記憶に残っているのは、2017年11月、オペレーター・アシスタントとして、国立音楽大学が主催するコンサート『聴き伝えるもの-20世紀音楽から未来に向けて-第12夜（ブーレーズとのレスポンソリウム）』の一環として、Pierre Boulezの６人のソリスト、アンサンブル、エレクトロニクスのための《Répons》(1981-84)のライブ演奏を体験したことである「図例2.3.3」[27]。その巨大な構成、極めて複雑で美しい作曲、特殊な空間配置などから、現在でもこの作品と匹敵できるものがない歴史的な「名作」と広く評価されている。ライブエレクトロニクスのパートでは、６人のソリストのライブ演奏の音をリアルタイムに処理しているため、オペレーターは６人の演奏者に合わせて、あらかじめ決められたタイミングでコンピューター側にトリガーを送る必要がある。40分弱の演奏時間の中で100回以上のトリガーを送る必要があるこの作品では、特殊な空間配置や常に複雑な音楽的変容が行われているため、実際のソリストの音を聞くだけではトリガーを送るタイミングを正確に判断することが難しい箇所も多くあった。しかし、指揮者の指示に従っていれば、自分では明確な判断ができなくても、最適なタイミングでトリガーを送ることができる。筆者は、リハーサルの段階からライブ演奏まで、すべての演奏に参加していた。同期やシステムの使用に関する心配は一切なく、演奏者たちは常に音楽的表現や解釈の面に集中して演奏を行うことがで

[27] 《Répons》演奏記録映像は以下リンクから視聴可能
https://www.youtube.com/watch?v=nYpjQsN4w1E (accessed September 7, 2020).
指揮・板倉康明先生　エレクトロニクス・今井慎太郎先生

きた。

筆者のこれまでの経験を踏まえると、トリガーを送信するインタラクティブ手法こそ、もっともフレキシブルかつ演奏者らしさを発揮できる、有用性の高い手段であると考えられる。設定の容易さ、高い安定性、そして演奏者とオペレーターによる臨機応変な調整が可能という点から、複雑な作曲をしても制約が課されることなく、ライブ演奏の際に双方の時間を確実に一致させる最大限の可能性を引き出すことが可能であると考えられる。

2.4 同期メソッドについて

新しい演奏同期システムを開発するための最後の課題は、どのように「同期メソッド」を構築するかと言うことである。同期メソッドとは、演奏同期システムの「頭脳」である、ライブ中に２つの時間をどのように一致させるかを決定する要素である。これまでに開発された「ライブ」同期システム（Score Following技術を主とする）は、主に演奏者の演奏時間を「予測」する手法を用いて、両者の同期を実現してきた。例えば、マルコフモデルや、隠れマルコフモデルなど、数学の統計学をベースにした計算手法が主に用いられている（ローズ 2001；Cont 2010；Robertson and Plumbley 2006）。これまでの研究報告や筆者自身の使用経験から、このような「予測型」同期メソッドを用いれば、特にテンポが一定とされている音楽分野（クラシック、ジャズ、ロック、ポップスなど）で、高い効果が実現できると考えている。例えば、筆者が行ったAntesocfoの演奏テストの結果によると、演奏中にテンポを自由に変えた場合でも、正確に予測して精度の高い同期を実現することができる。これは非常に強力だと筆者は考える。しかし、今回の研究での同期問題を改善・解決するためには、従来の「予測型」の同期メソッドをそのまま使っても、効果的なライブ同期を実現することは難しいと、以下２つの理由から筆者は考えている。

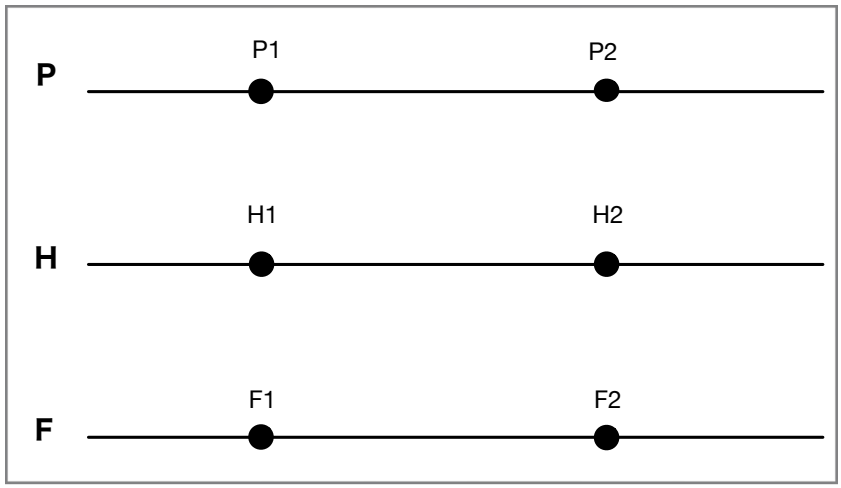
第１の理由は、同期時間間隔の性質の違いである。2.1節で述べた「新しい同期時間間隔」は、演奏者の演奏のテンポに従わない、時間幅の広い自由な同期時間間隔である。この時間間隔で得られる演奏者の時間に関する情報は、演奏者の演奏時間と一定時間内のFixed Mediaの再生時間との「時間差」のみである。この「時間差」だけでは、演奏者の演奏時間を正確に予測することはできず、効果の高いライブ同期を実現することは困難である。

第２の理由は、システムの開発アプローチの違いにある。本研究における演奏同期システムの開発目標は、複雑なFixed Mediaとの同時演奏において、演奏者が何の制約もなく自分の演奏時間に集中できる、表現力の高いライブ演奏を実現させるサポートを提供することである。演奏者が自由に演奏のテンポを変えられるシステムでは、演奏者の演奏時間だけでなく、Fixed Mediaの再生時間を大幅に変えることによる音質への影響などの問題があるため、作曲者にとって「望ましくない」演奏結果になってしまう可能性が高くなる恐れがある。

これら2つの理由から、従来の同期メソッドをそのまま本研究の演奏同期システムに用いても、効果的な同期の実現は期待できないため、代替案を検討する必要があると筆者は考えて

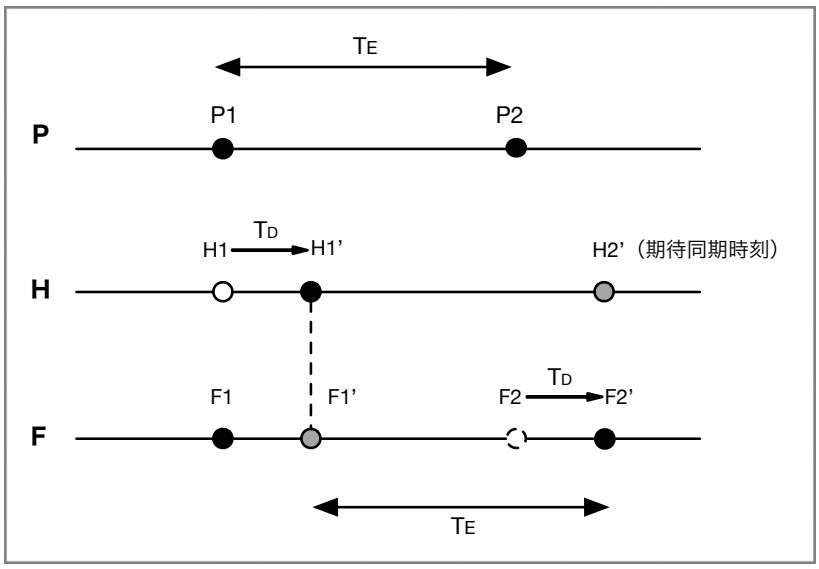
いる。実際、本研究では対象とする音楽が特殊的であり、研究目標も明確であるため、非常にシンプルな同期方法を用いても効果の高い同期演奏が実現できると考えられる。ここでは、私自身が考案した新しい同期メソッドを提案し、効果的かつ音楽的なライブ同期を実現できる方法について議論したい。

ここでは、「図例2.4.1」に示したグラフを用いて、新しい同期メソッドについて説明する。上部のP（Plan）軸は作曲時に決定された計画時間、真ん中のH（Human）軸はライブ演奏者の実際の演奏時間、そして一番下のF（Fixed Media）軸はFixed Mediaの再生時間を示している。それぞれの横軸の黒い点　（P1、H1、F1など）は求められる同期時刻である。「図例2.4.1」は、それぞれの同期時刻が縦軸で一致しており、同期が完璧に取れている例である。このような演奏結果は、例えば、クリックトラックなどの手法を用いれさえすれば達成できるが、度々繰り返しているように、この研究の当初の試みとして掲げているのは、演奏者が自分の演奏時間を解釈するための一定の時間を確保しつつ、Fixed Mediaとの高精度な同期を実現することである。そのためクリック以外の方法を用いて、新たなアプローチを生み出さなくてはならない。



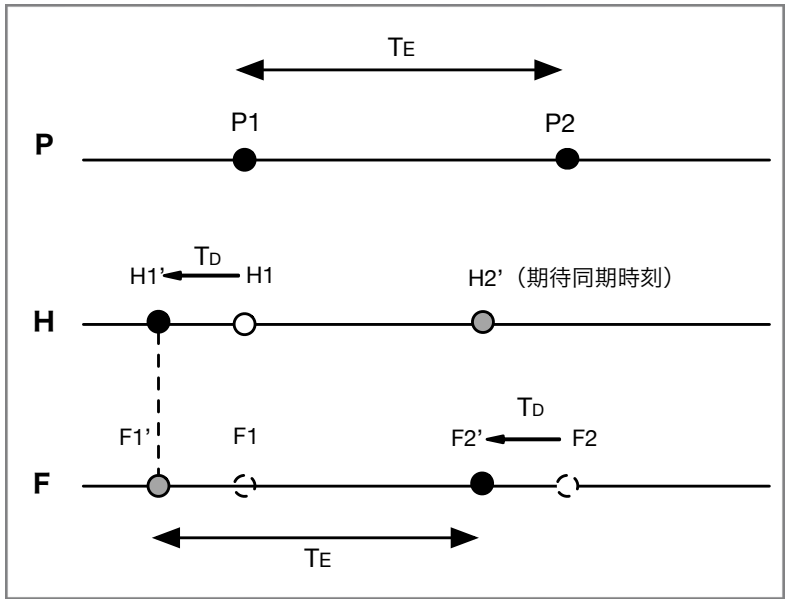
「図例2.4.1」
同期メソッドを論じるための図例（完全同期）。

「図例2.4.2」は、演奏者の演奏時間H1’が予定時間P1に対してT_Dの時間幅で遅れた場合を想定している。演奏者の演奏に遅延が発生しているため、Fixed Mediaを次の求められる同期のタイミングに合わせられるよう、その再生時間を減速させ、（調整する）必要がある。筆者の考案する新しい同期メソッドのアプローチというのは、Fixed Mediaの再生時間を、演奏者の遅延した時刻F1’からＦ２’まで（T_Eの時間幅を持って、その再生時間を遅らせていくというものである。ここでいうT_Eとは、演奏者の演奏時間を予測するような複雑なアルゴリズムに

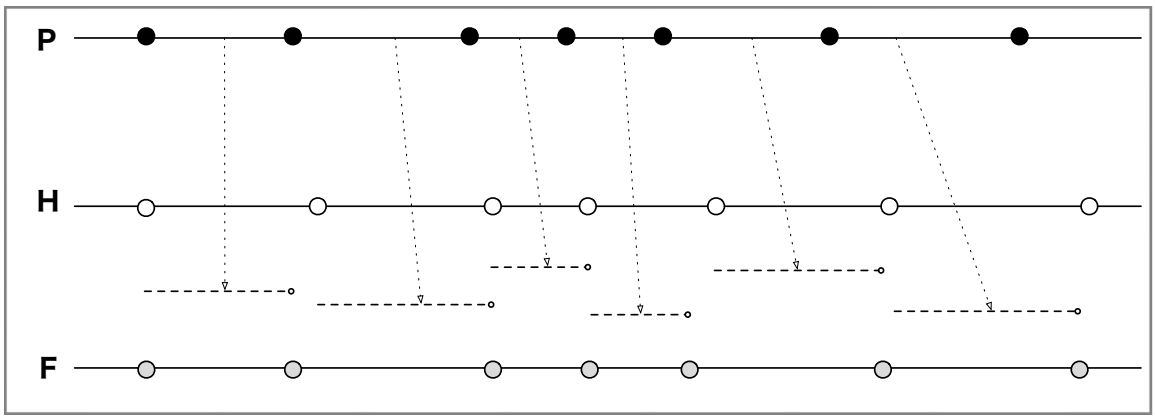


「図例2.4.2」筆者が提案するメソッド（演奏者が予定時刻より遅れた場合）。

よって算出したものではなく、作曲家が創作の過程で決定した計画の演奏（再生）時間そのものである。T_EのEはExpect（期待する）の略である。このようなT_Eを、H1'時刻から発生させることにより、演奏者は遅れた演奏時刻H1'から、作曲段階で決められた演奏時間に従うのであれば、理論的に次の同期時刻H2'との厳密な同期を取ることが可能である。「図例2.4.3」は、演奏者が予定時刻よりもT_Dの時間幅で早まった状況を示している。このような場合、同期を図る仕組みとしては、「図例2.4.2」と全く同じようなものであり、演奏者の早まった時刻H1'から、Fixed Mediaの再生時間をF1'からF2'までに加速させるようにする。最終的にT_Eでの時間幅を用いて、演奏者と伴い新しい期待する同期時刻H2'での完成度の高い同期を



「図例2.4.3」筆者が提案するメソッド（演奏者が予定時刻より早まった場合）。



「図例2.4.4」新しい同期メソッドを連続的なイベントの中に用いられる場合。

求めていくという仕組みである。

「図例2.4.4」は、長時間のMixed music作品に新しい同期方法を導入した場合に予想される同期演奏の結果を示したものである。演奏者側から長時間にわたってFixed Mediaと正確に同期することが困難な場合、一定の時間範囲内で演奏者の演奏時間が予定の時間よりも早くなったり遅くなったりすることがある。この場合、Fixed Mediaの再生は、常に制作段階で決定された予定の演奏（再生）時間を使用し、演奏者から与えられた新たな時間情報（H軸の白点）に応じて伸縮し続けるようにする。その結果、予定時刻と演奏者の演奏時刻の差が大きくなる一方で、新たな同期方法により再生時刻が変化しているFixed Mediaは演奏者の演奏時刻と厳密な関係を保つことができるようになる。例えば、複雑で長大なFixed Media作品との同期が必要であっても、双方の時間における高度な一致が実現できると理論的には考えられる。

筆者は、このような同期メソッドに関する議論は、音楽家がそれぞれのニーズに応じて、さまざまなアプローチで設計・導入することができると考えている。この節では、Mixed music作品の音楽家の立場から、音楽的で「効果の高い」ライブ同期を実現することが期待される新しい同期メソッドを提案してきた。新しい同期メソッドは、従来の演奏者の演奏時間から算出する「予測」方式ではなく、演奏者側から得られる最新の演奏時間情報を基に作曲段階で決定される期待演奏（再生）時間を用いることで、完成度の高い同期を実現するための支援となるものと考えている。このような同期メソッドを用いたライブ演奏では、演奏時間が作成段階で想定したものと大きく異なる場合において、高い精度で同期をとることは難しいが、ライブ演奏者は複雑なFixed Mediaとの同期を配慮する必要がなく、演奏中は自分の演奏に集中しながら完成度の高いライブ同期の実現することが期待できる。本研究の目指す、有効性の高い同期メソッドであると筆者は考える。

2.5 まとめ

本章では、Mixed music音楽におけるライブ同期の問題を改善・解決することを目的に、タイムストレッチ技術に基づく演奏同期システムの構築について考察した。同期時間間隔、インタラクティブ手法、同期メソッドの３つの課題について考察した結果、最終的に、新しいシステムを開発するために必要な視点が明確になった。その３つの視点とは、音楽的ニーズによって決まる自由な同期時間間隔、トリガーを送信するインタラクティブ手法、そして筆者が考案した新しい同期メソッドを用いることである。例えば、上記の視点を１つのシステムに共存させることができれば、本研究において効果の高い新しい演奏同期システムを実現することが理論的に可能であると考えている。次の章では、実際筆者が開発した新しいシステムを提案することによって、まずこのようなシステムをどのようにして実現させるかについて考察していく。

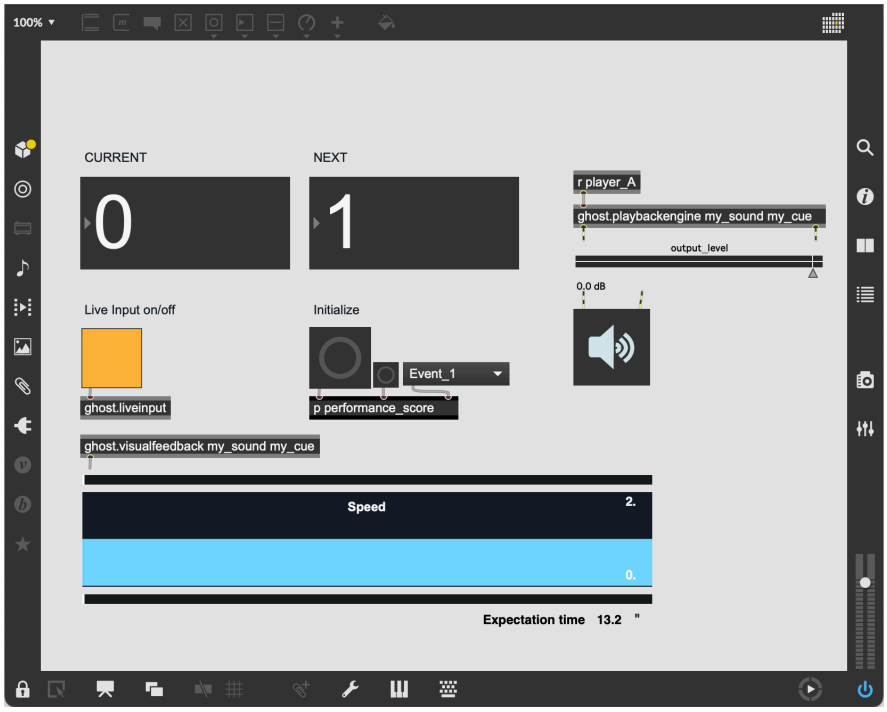
第３章 システムの提案

3.1 概要

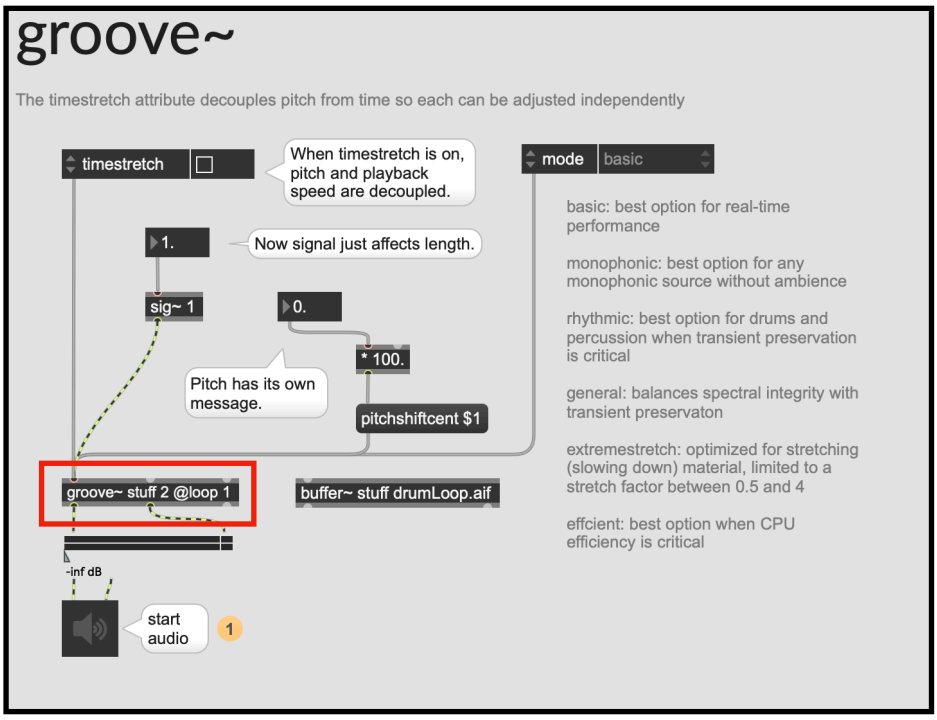
この章は筆者が開発した新しい演奏同期システムの提案を目的とする。このシステムは第２章の中で論じた開発方法に基づいてプログラミングしたものである。このシステムはFixed Mediaの再生と人間による演奏をライブの中で時間的に同期させることができ、目に見えない支援者のような存在である特徴に見立てて「Ghost」と名付けられた。本章では、Ghostの内容について詳しく論じることによって、システムの全体像を明らかにする。

3.2 プラットフォーム

Ghostは、Maxによってプログラミングされている。「図例3.2.1」はGhostのインタフェースを示している。Maxとは、リアルタイム・タスクを扱うためのプログラミング環境である。Miller PucketteがIRCAMで1985年に発明し、よく知られる「パッチ」のGUIを1988年に開発した。現在は、商用版Maxの開発者であるDavid Zicarelliが1997年に設立したCycling ’74社が、開発・販売している。オブジェクト（部品）と呼ばれるブロック状のものを線で繋ぎ合わせることによってプログラミングができる。その視覚的な分かりやすさとリアルタイムで実行が可能であること、これがMaxの最大の特徴である。Maxは昨今、特に現代のコンピュータ音楽の領域で多くのアーティストに使用されており、システムを共有するための最も



「図例3.2.1」Ghostシステムのインタフェース。



「図例3.2.2」「groove~」オブジェクトにおける
タイムストレッチ技術の使用に関するMaxのヘルプパッチ。

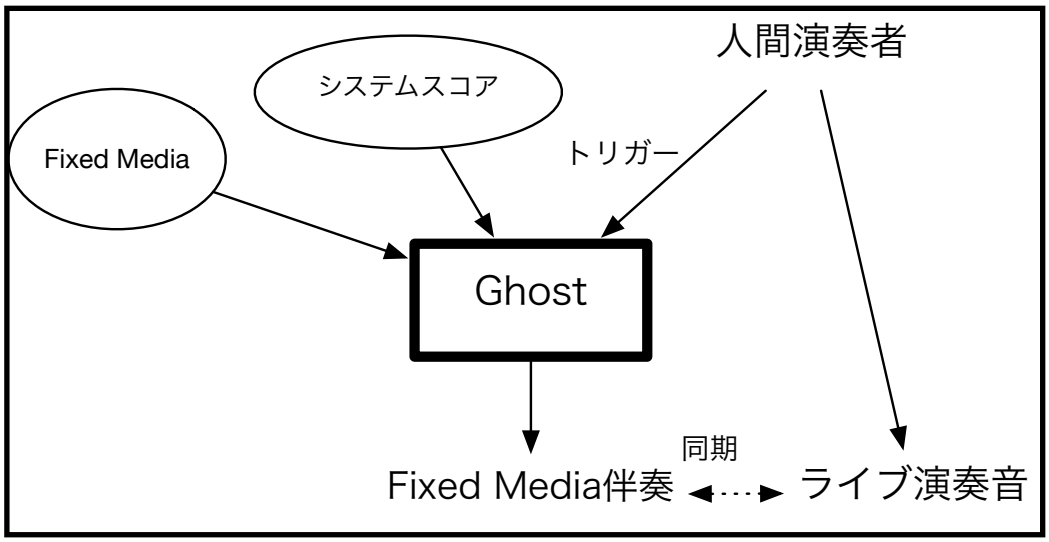
理想的なプログラミング言語と言える。

Ghostが使用するタイムストレッチ技術には、Maxに内蔵されている「groove~」オブジェクト（赤い線で囲まれた箇所）を用いる。「図例3.2.2」「groove~」オブジェクトは、オーディオデータの再生処理を行うために使用される。Maxのバージョン７（2014年11月リリース）以降、極めて高品質で、リアルタイムで実行可能なタイムストレッチ技術が搭載されている。「groove~」オブジェクトを使うと、オーディオ素材本来の音質を維持したまま、その再生速度を幅広い範囲の中で変化させることができる[28]。また、タイムストレッチ技術のアルゴリズムも数多く存在することから、本研究のシステム開発により、様々なFixed Media音楽スタイルに対応した高品質な再生の可能性を最大限に高めることができる。

3.3 システムの使用方法

「図例3.3.1」はGhostの使い方を示す模式図である。ライブの前には、あらかじめ作曲されたFixed Mediaと、このシステム専用のスコアを入力する必要がある（後述）。そしてライブ中には、演奏者や同時に演奏しているエレクトロニクス・パートのオペレーターからシステムにトリガーを送ることで操作していくものである。演奏者がトリガリング（トリガーを送ること）場合、通常、MIDIデバイス（MIDIペダルなど）を介して行う。一方、オペレーターが

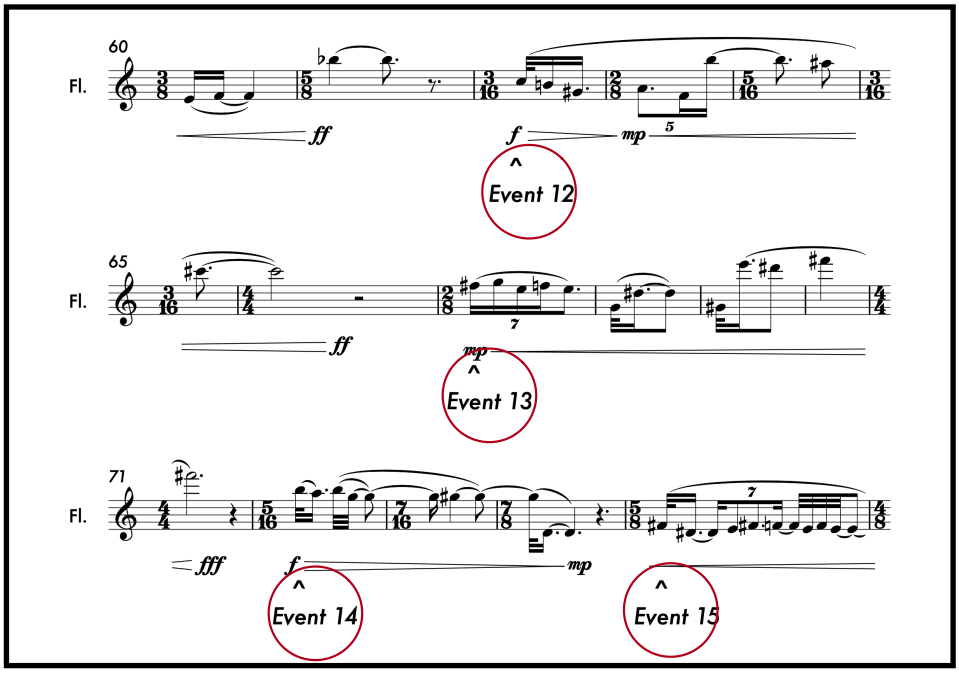
[28] 筆者の経験上、オーディオデータの再生速度を0.6倍速から1.7倍速の範囲で変化させても、音質に対する明確な影響は認められなかった。



「図例3.3.1」Ghostの使い方の模式図。

担う場合は、MIDIデバイスの他に、コンピューターキーボードのキーを操作することでもトリガリングすることができる。

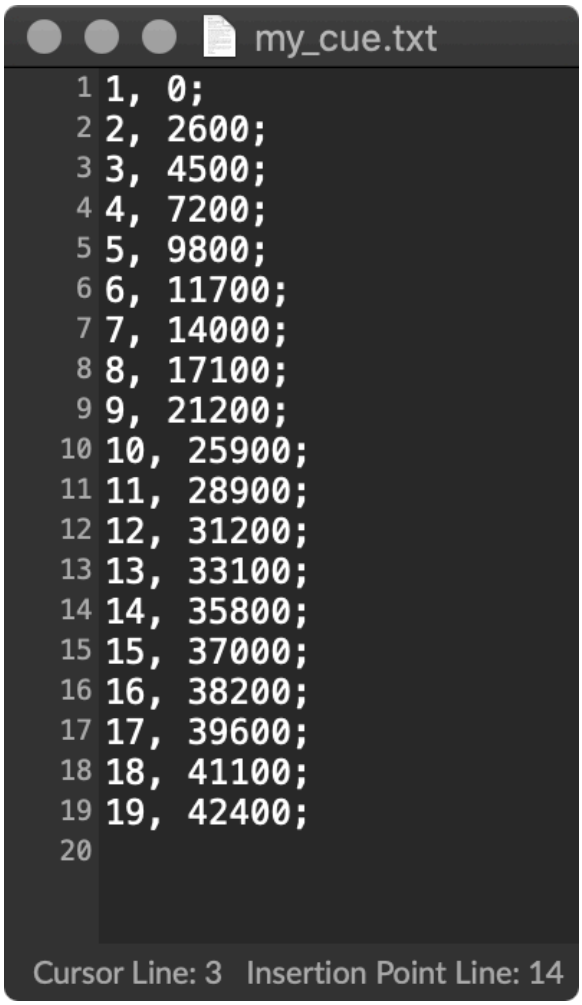
「図例3.3.2」は、Ghostの利用時に用いられる楽譜の 1 例である。五線譜の下にある記号「Event 12、Event 13……」（赤い丸に囲まれた箇所）は、トリガーを送信するべきタイミングを示している。演奏中、トリガリングの担い手は、演奏者の演奏位置が楽譜の中でマークされた位置に到達するごとに、順次、トリガーを送る必要がある。



「譜例3.3.2」Ghsotシステム使用時に用いる
ライブ演奏者用の楽譜。

3.4 システムスコアについて

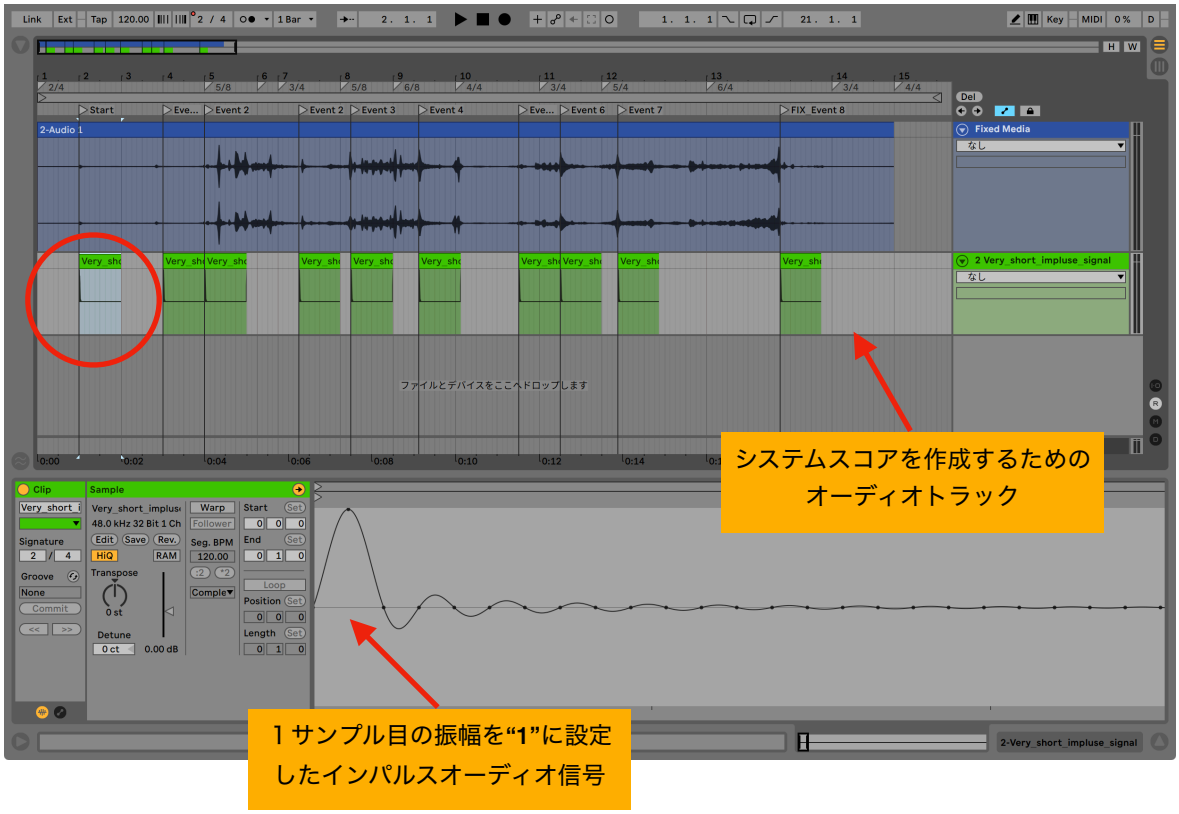
従来のScore Following技術と同様に、Fixed Mediaと演奏者の間の時間的位置関係を把握するために、システム用のスコアを用意する必要がある。しかし、Ghostでは、演奏者から送られてくる「トリガーの情報」を受け取るだけなので、そのスコアは非常にシンプルなものになっている。「図例3.4.1」にGhostのシステムスコアの１例を示している。左側の「１、２、３……」という数字はトリガー番号である。その順番は原則として、演奏者の楽譜に表示されているイベントの番号と一致させる必要がある。右側の“０、2600、4500”などの数値は、Fixed Mediaの時間情報をミリ秒単位で表したものである。トリガー番号と時間情報が対になることで、対応関係が示されている。このようなスコア（テキスト）の形式は、Maxのメッセージ情報を保存、読み込みする「coll」オブジェクトが使用するためのものである。システムを使用する前に、この形式のシステムスコアを作成する必要がある。



「図例3.4.1」Ghostのシステムスコア。

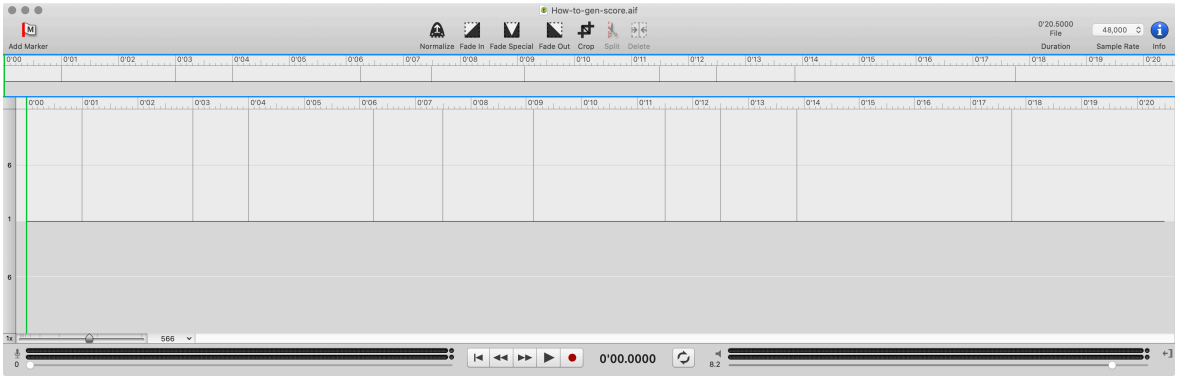
3.5 システムスコアの作成

ここでは、Ghostシステムのスコアの作成方法について説明する。前述の通り、スコアに含まれる情報量はそれほど多くないので、作成するのはそこまで複雑ではない。イベントの数が少なければ、手動で作成することも可能と考えられる。しかし、作品の時間が長い場合や、イベントの数が多い場合には、ある程度の効率的な手法を導入する必要が出てくる。ここでは、筆者が普段から活用している、効率的にシステムスコアを作成する方法を提案したい。それは、非常に短いインパルスオーディオ信号を使う方法である。

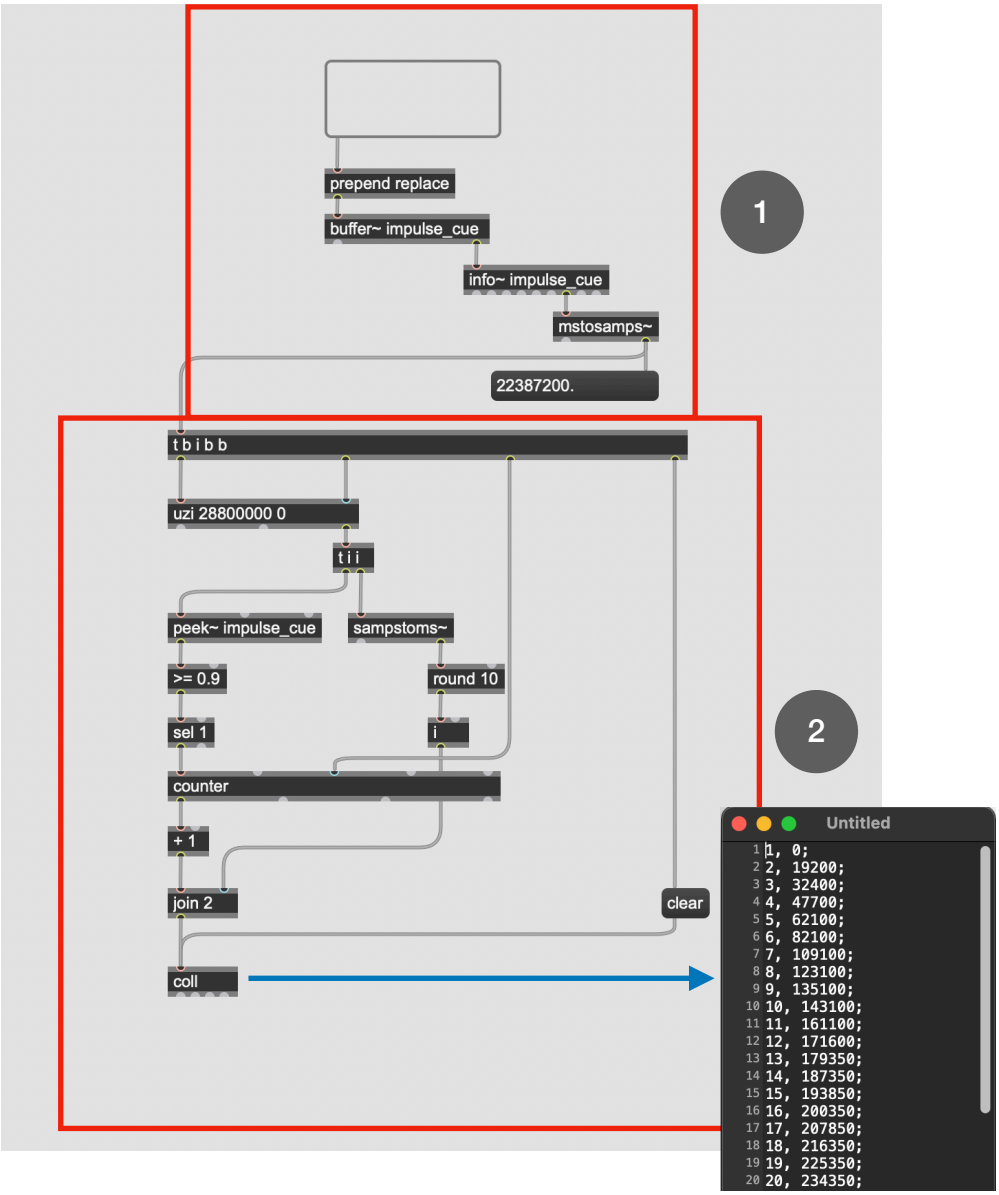


「図例3.5.1」システムスコアを作成するための
Ableton Liveプロジェクト。

「図例3.5.1」は、Ableton Liveで作曲したMixed musicのプロジェクトである。上のトラックが、この作品で使用されるFixed Media電子音響である。下のトラック（緑のトラック）はシステム用のスコアを作成するために用意されたトラックである。そのトラックの中に置かれている数多くのオーディオ素材は、システム用のスコアの作成に使用される数サンプルしかない非常に短いインパルスオーディオ信号である。この信号は、後に使用するMaxパッチの中で明確に識別する必要があるため、その中の最初のサンプルは、オーディオ信号における最大値である“１”に設定されている。これらのサンプルを、作品で想定される同期のタイミングに配置し、最終的に１つのオーディオファイルとして出力する。オーディオファイルの形



「図例3.5.2」短いオーディオインパルス信号で作られた
オーディオファイルの波形。



「図例3.5.3」Ghostのシステムスコアを素早く作成できる
Maxパッチ。

式は、必ず非圧縮オーディオ形式（wavまたはaiff）を用いる。「図例3.5.2」は出力されたオーディオファイルの波形を示している。

「図例3.5.3」は、国立音楽大学の准教授の今井慎太郎がプログラミングした、システムスコアを素早く出力するためのMaxパッチである。このパッチの一番上の長方形のブロックに、先ほど作成したオーディオファイルをドラッグ＆ドロップすると、パッチの一番下にある「coll」オブジェクトにシステムスコアとなるテキストが自動的に入力される。このシステムスコアをGhostシステムで扱うには、一番下の「coll」オブジェクトをダブルクリックし、表示された内容をテキストファイルとして、Ghostシステムのパッチが保存されているフォルダに保存する必要がある。

次に、このパッチの構造を説明する。最上部「dropfile」オブジェクト（長方形のブロック）から「mstosamps~」オブジェクトまでのエリアは、オーディオファイルの長さ（サンプル数）を出力する仕組みになっている（①のマークがついた部分エリア）。「dropfile」オブジェクトは、オーディオファイルのファイルパスを出力する。次に「prepend」オブジェクトにより、“replace”メッセージをそのパス情報の前に付加させ、オーディオファイルを「buffer~」オブジェクトに格納する。「info~」オブジェクトでは、「buffer~」オブジェクトからbangメッセージを受け取り、オーディオデータの長さ（ミリ秒単位）を出力する。次に「mstosamps~」オブジェクトにより、その数値をサンプル単位に変換する。

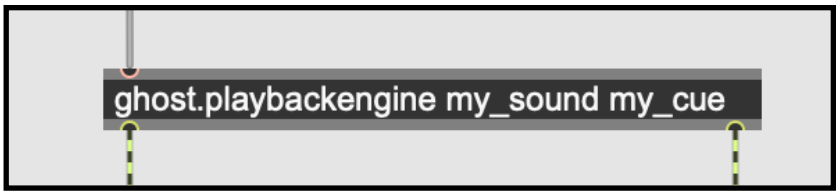
「ｔｂｉｂｂ」オブジェクトより以下の部分では、オーディオデータの長さを表す数値を、右から左の順に出力して、最終的にシステムスコアを生成する仕組みになっている（②のマークがついた部分エリア）。一番右のアウトレットは、“bang”メッセージを“clear”メッセージに送り、「coll」オブジェクトのコンテンツをクリアする。右から２番目のアウトレットでは、整数を小さい順に生成する「counter」オブジェクトを“０”からカウントするようにリセットする。右から３番目のアウトレットは、「uzi」オブジェクトが出力するデータの数を、オーディオデータの長さと同じ数（整数）に設定する。「uzi」オブジェクトの右アウトレットには、その数字に対応するインデックス情報[29]が出力される。「ｔｂｉｂｂ」オブジェクトの一番左側のアウトレットは、「uzi」オブジェクトの左インレットに“bang”メッセージを送り、システムスコア生成のプロセスを実行させる。「uzi」オブジェクトの右側のアウトレットに出力されるインデックス情報は、次の「ｔｉｉ」オブジェクトによって、右から左の順で２つのアウトレットに分けて送り出す。右アウトレットの先の部分では、その値を10ミリ秒単位とする整数値に変換する。左アウトレットの先の部分は、オーディオデータの中にあるインパルスオーディオ信号を検出し、“bang” メッセージを「counter」オブジェクトに送り出す仕組みになっている。「peek~」オブジェクトは、インデックス情報を受け取り、オーディオサンプルの振幅値を順次に出力する。「>= 0.9」オブジェクトは、“0.9”以上の数値を検出すると、

[29] 例えば、「uzi」オブジェクトの右インレットに "3 "を入力すると、1、2、3の数値順でインデックスデータが一番右のアウトレットに出力される。

整数の“１”を出力する。「sel」オブジェクトは“１”を受け取るたびに“bang”メッセージを「counter」オブジェクトに送信し、システムスコアのイベント番号を表す整数を順番に生成する。「counter」オブジェクトから出力される最初の数は“０”であるため、「+」オブジェクトによってそれに“１”を加える必要がある[30]。最後に、新しく生成されたイベント番号とそれに対する時間情報が「join」オブジェクトによって結合され、「coll」オブジェクトに入力される。

3.6 データの入力方法

Ghostシステムにデータを入力する方法について説明する。GhostシステムにFixed Mediaとシステムスコアを入力するには、システム用にプログラミングされたアブストラックション・オブジェクト[31]のアーギュメント[32]欄に、それぞれのファイル名を設定することで実現する。



「図例3.6.1」Ghostのアブストラクション・オブジェクト。

「図例3.6.1」に示している「ghost.playbackengine」というオブジェクトは、Ghostのアブストラクション・オブジェクトの１つである。オブジェクトのアーギュメント欄[33]に入力されている“my_sound”と“my_cue”というテキストは、今回の作品を演奏するために必要なFixed Mediaファイルとシステムスコアのファイル名である。それらのファイルは、「図例3.6.2」に示すように、必ずGhostのシステム用のパッチと同じフォルダに保存する必要がある。

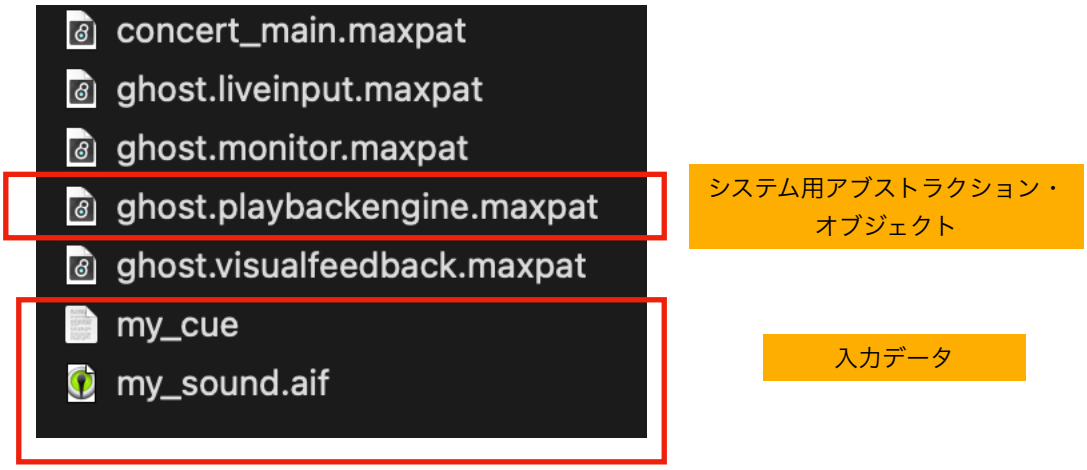
「図例3.6.3」は、「ghost.playbackengine」オブジェクトの中にある上記のファイルの入

[30] 原則的にはシステムイベントは１から始まる。

[31] Maxのサブ・パッチの一種。プログラムされたパッチはフォルダに保存され、同じフォルダ内の他のパッチはそのパッチを単独のオブジェクトとして使用することができる。

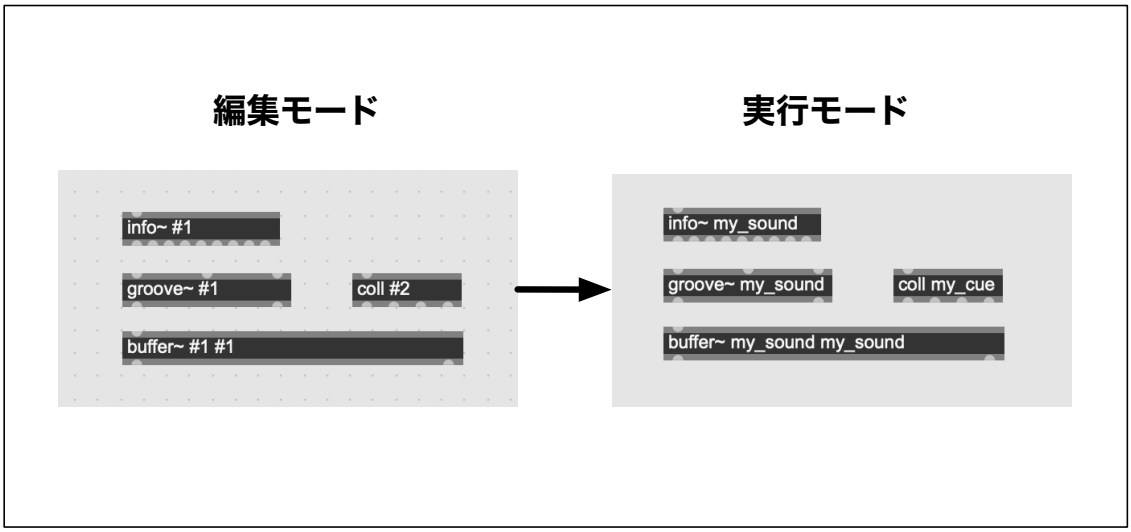
[32] Maxで作成されたオブジェクトの右側に入力される情報である。オブジェクトに様々な初期値を設定することができる。

[33] オブジェクト名の右側の領域。オブジェクトの初期値を設定する場所である。



「図例3.6.2」Ghostのシステムパッチが保存されているフォルダ。

力が必要とするオブジェクトの状態を示している。「編集モード」[34]で、各オブジェクトのアーギュメントとして設定された“#1”と“#2”が、「実行モード」に切り替えることによって、「ghost.playbackengine」オブジェクトのアーギュメント欄に入力した情報で置き換えられる。結果として、「info~」オブジェクト、「groove~」オブジェクト、「buffer~」オブジェクト、にはFixed Mediaオーディオファイル[35]が、「coll」オブジェクトにシステムスコアが自動的に入力されるようになる。



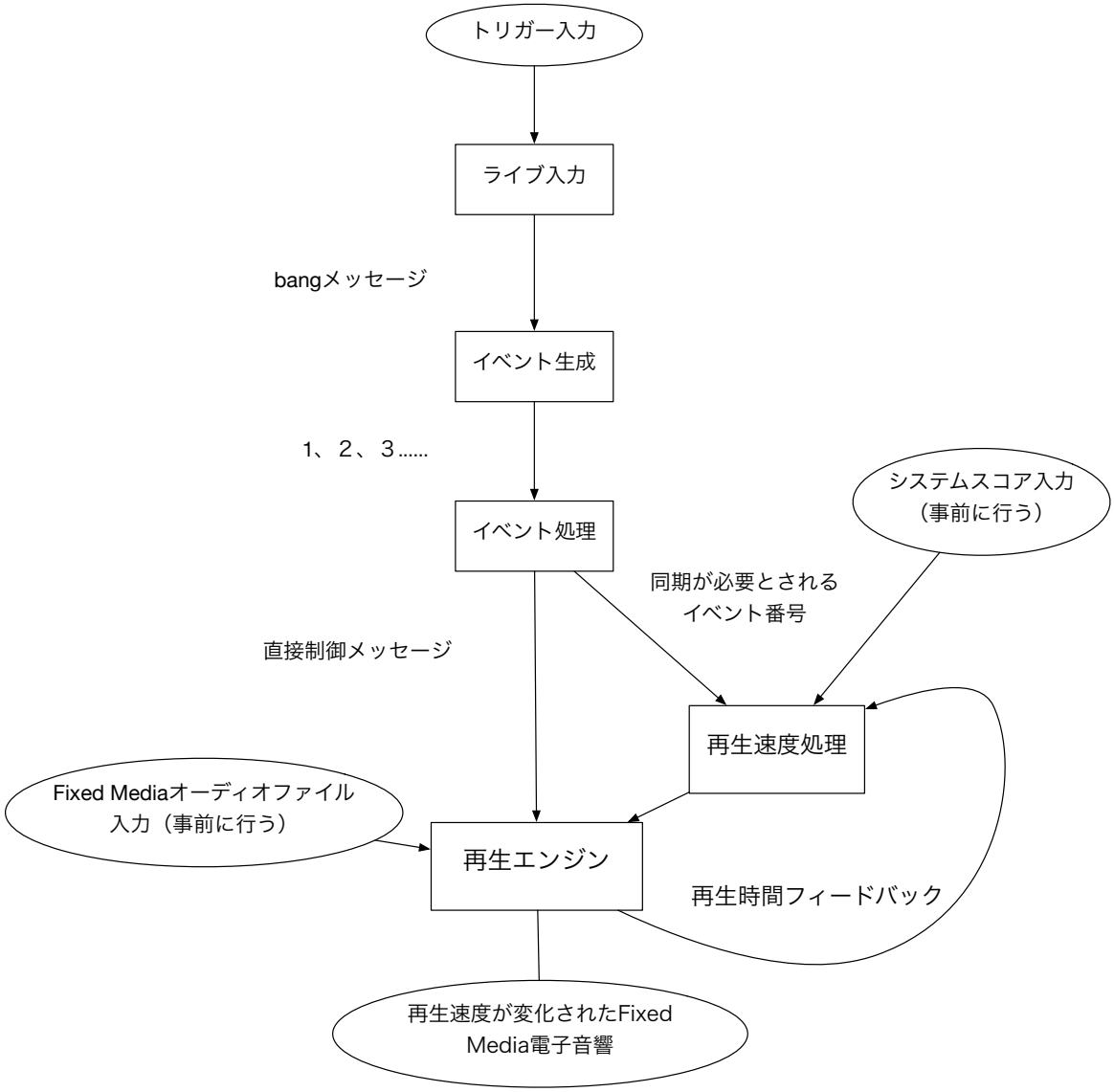
「図例3.6.3」アブストラクション・オブジェクトの中でデータ入力する必要があるオブジェクトの状態。

[34] 編集モードはプログラムの修正や変更が可能な状態、実行モードはパッチを実行する状態である。

[35] オーディオファイル自体は、「buffer~」オブジェクトにのみ格納される。「groove~」と「info~」オブジェクトは、「buffer~」オブジェクト内のオーディオデータを読み取るために、「buffer~」オブジェクトと１つ目のアーギュメントに同じ文字を設定する必要がある。

3.7 システムのデザイン

「図例3.7.1」はGhostシステムのプログラミングの概要図である。ライブ入力モジュールは、ライブの演奏中、渡されたトリガー情報をbangメッセージの形でイベント生成モジュールに送信する。イベント処理モジュールは、“bang”メッセージを受け取ると、イベント番号を表す整数を順番に出力する。それらの情報は、作品の表現意図に応じて２つのモジュールに対して送信する。再生エンジン・モジュールに送られる情報は、再生エンジンを直接制御するためのメッセージである。例えば、リセット、再生スタート・ストップ、再生位置などの情報である。再生速度処理に出力される情報は、同期が必要なイベント番号の情報である。例えば、イベント２〜20に対して同期が必要な作品の場合、その出力は２から20までの番号になる。再生速度処理モジュールは、演奏者の演奏者の演奏時間情報と、現在の再生時間とを

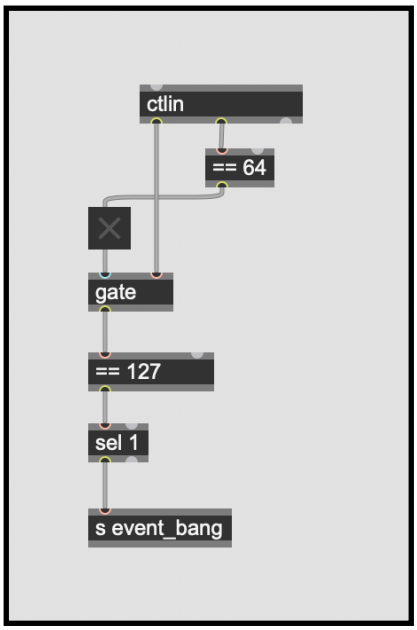


「図例3.7.1」システムの概要図。

照合させることで、Fixed Mediaが次のイベントに間に合うための必要な再生速度を計算し、再生エンジンに送信する。演奏に伴ってイベント処理を行うことで、再生速度を更新していく。最終的に、再生速度が変化されたFixed Media電子音響が再生エンジンから出力され、ライブでの同期演奏が実現する。

3.7.1 ライブ入力モジュール

ライブ入力は、演奏者が送るトリガーを“bang”メッセージに変換して出力するためのモジュールである。使用する入力機器によって設計が異なる。ここでは、入力装置として代表的な、MIDIペダルとコンピュータキーボードを用いた送信方法とそれらの入力に対する処理の仕組みを紹介する。



「図例3.7.1.1」MIDIペダルのトリガーによる“bang”メッセージを
送信する仕組み。

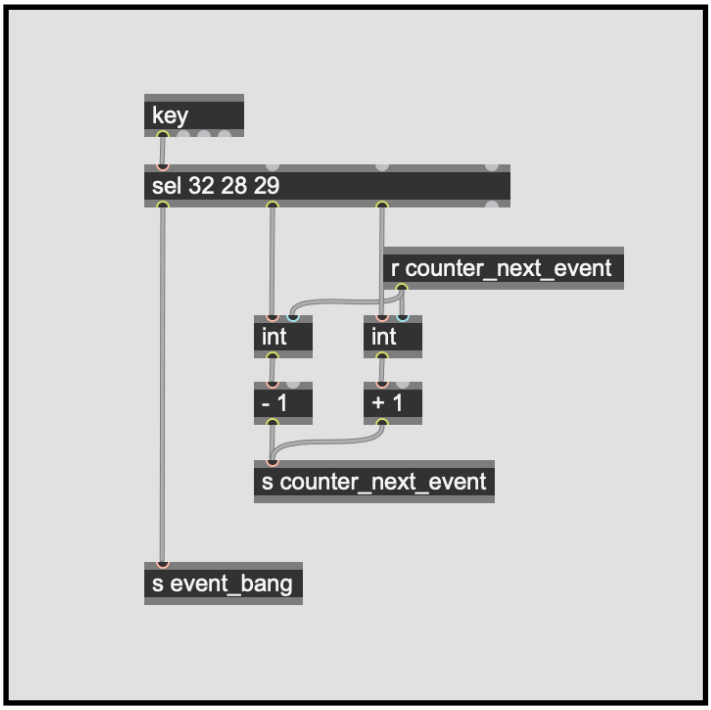
MIDIペダルでトリガーを送信する場合

「図例3.7.1.1」はMIDIペダルのトリガー（サステインペダルを踏む）による“bang”メッセージを送信する仕組みを示す。「ctlin」オブジェクトはMIDIペダルから渡されるMIDIコントロール情報を出力する。真ん中のアウトレットはコントロール番号を出力する。左から 1 番目のアウトレットはそれに対応するコントロールの値を出力する。出力されたコントロール

番号が“64”であれば[36]、「gate」オブジェクトの右インレットが開く、左側のアウトレットからそのコントロールの値を通過させる。そのコントロールの値が“127”であれば[37]、「sel」オブジェクトに“１”を送信し、「sel」オブジェクトのアウトレットから“bang”メッセージが出力される。無線送信機の「s event_bang」[38]オブジェクトによって、“bang”メッセージをイベント生成モジュールに送信する。

コンピュータのキーボードでトリガーを送信する場合

一般的なMIDIペダルではなく、コンピュータのキーボードでトリガーを送信する際には、“bang”メッセージを送る以外の操作も簡単に行うことができる。ここでは、スペースキー、右矢印キー、左矢印キーの３つのキーを使用してシステムを操作する仕組みを紹介する。スペースキーが押されると“bang”メッセージが送信される。右左矢印キーを押すと、次にスペースキーを押す時に送信されるイベント番号に対して加算・減算の操作を行うことができる。このようにして、必要に応じて送信されるイベント番号を調節することができる。この部分は、特に楽器演奏者のペダル操作ミスがあった場合、コンピュータ側のオペレーターが次の



「図例3.7.1.2」MIDIペダルのトリガーによる“bang”メッセージを
送信する仕組み。

---

[36] サステインペダルのMIDIコントロール番号。

[37] 通常の設定では、MIDIペダルを完全に踏んだ時、そのコントロールの値は127になる。

[38] 無線でメッセージを送信できるオブジェクト「s」(sendの略)である。アーギュメントに“event_bang”を指定することによって、同じアーギュメント“event_bang”で指定された「r」(receiveの略)オブジェクトに送信することができる。

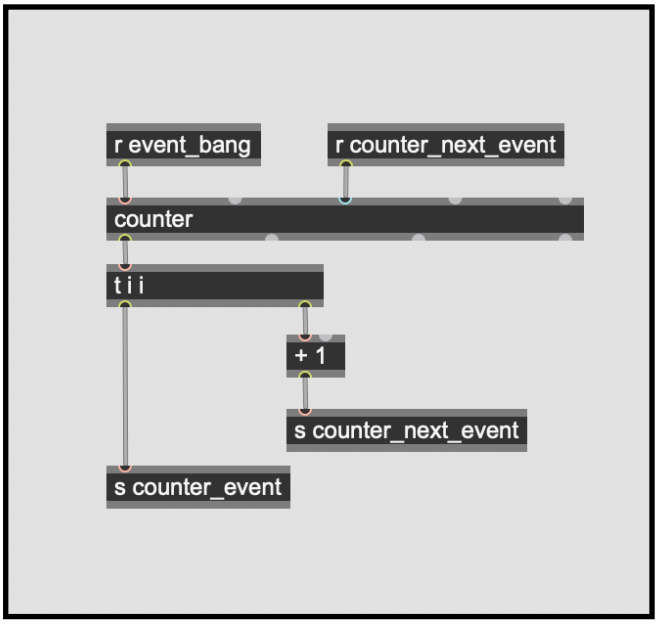
トリガーのイベント番号を正確に調整するために使用する。

「図例3.7.1.2」はこの仕組みを示している。「key」オブジェクトはコンピュータのキーボード上で押されたキーを数値として出力する。「sel」オブジェクトのアーギュメント欄の中に入力された32、28、29の３つの数値はそれぞれ、スペースキー、左矢印キーと右矢印キーに対応している。それぞれのキーを押すと、それに対応したアウトレットから“bang”メッセージが出力される。スペースキーを押すと直接に“bang”メッセージが「s event_bang」オブジェクトから出力されイベント生成モジュールへと直接的に入力される。左矢印キーや、右矢印キーを押すと、左側から２番と３番目のアウトレットから“bang”メッセージが出力される。“bang”メッセージは「int」オブジェクトに入力され、イベント生成モジュールから入力される「次のイベント番号の値」（右側のインレットの入力）を出力することになる。「int」によって出力された値を下の「+」や「-」オブジェクトに入力すると、その値にプラス１やマイナス１された計算結果の値が出力される。結果として、その値は「s counter_next_event」オブジェクトからイベント生成モジュールに送信され、次にトリガーが渡される際に送信する新しいイベント番号として設定される。

3.7.2 イベント生成モジュール

イベント生成モジュールは、渡される“bang”メッセージをイベント番号に変換して出力する仕組みである。その仕組みは、単純に１つの「counter」オブジェクトで実現されている「図例3.7.2.1」。

左上の「r event_bang」オブジェクトが「s event_bang」オブジェクトから送られる演奏者らのトリガーを受信し、「counter」オブジェクトに“bang”メッセージを送り出す。右上の



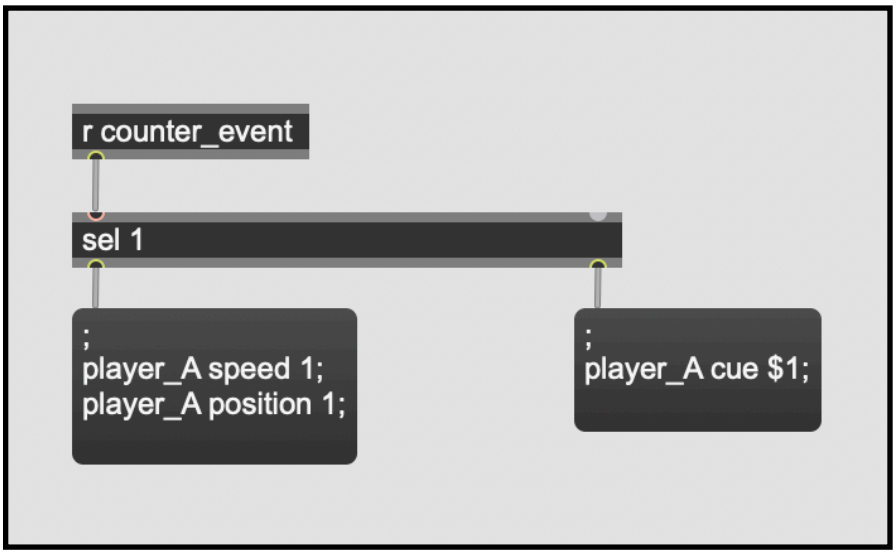
「図例3.7.2.1」イベント生成モジュール。

「r counter_next_event」オブジェクトは、「図例3.7.1.2」の「s counter_next_event」から送られてくる「調節された次のイベント番号」を受信し、「counter」オブジェクトが次に出力するためにカウントする数を設定する。「counter」オブジェクトの下にある「t i i」は出力された現在のイベント番号を２箇所に送信する。右アウトレットの先の部分では、「図例3.7.1.2」のパッチの中で「int」オブジェクトに「次のイベント番号」を格納する。左側のアウトレットでは、直接に「現在のイベント番号」がイベント処理モジュールに送信される。

3.7.3 イベント処理モジュール

イベント処理モジュールは、イベント生成モジュールから送られる「現在のイベント番号」を受信し、作品の意図に応じて再生エンジン・モジュールや、再生速度処理モジュールに対してコントロールメッセージを送信するための仕組みである。例えば、再生を開始するためには、イベント番号１を再生エンジンにメッセージを送る必要がある。イベント番号２以降は、演奏者の演奏時間に合わせて再生時間を変化させる必要があるため、再生速度処理モジュールにメッセージを送信する必要がある。Ghostシステムではこのような仕組みを「sel」オブジェクト１つ用いるだけで実現することができる。「図例3.7.3.1」は、このようなイベント処理の一例である。

「sel」オブジェクトはアーギュメント欄に設定された数値と同じ数値を左インレットに対して、入力された値に対応したアウトレットに“bang”メッセージを出力される。アーギュメント欄に設定されていない値が入力された場合には、その右側のアウトレットから入力された数値のままで出力される。「図例3.7.3.1」の場合、「sel」オブジェクトに“１”を入力すると、一番左側のアウトレットに繋いだメッセージ・ボックスに“bang”メッセージを入力される。それ以外の数値を入力されると、それらの数値は全て右側のアウトレットから出力し、

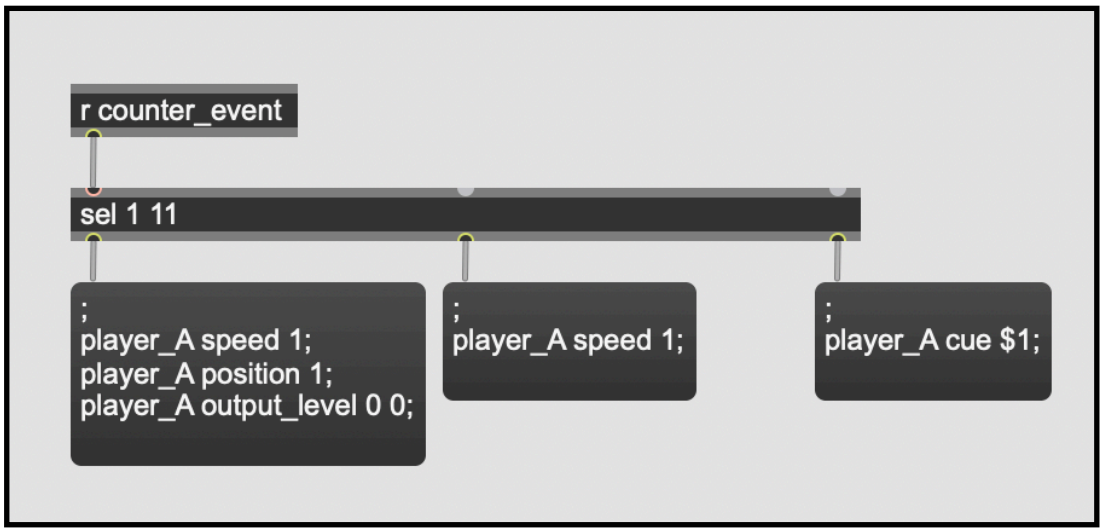


「図例3.7.3.1」イベント処理の例(１)。

そのアウトレットに繋いだメッセージ・ボックスにその数値のまま入力される。

左側のメッセージ・ボックスでは、“bang”メッセージを受信すると、セミコロンの後に続く文字“player_A”がアーギュメント欄に入力された「r」オブジェクトに、“speed 1”と、“position 1”という２つのメッセージを順番に送信する。それらのメッセージは再生エンジン・モジュールに送られる。“speed 1”ではオリジナル速度で再生する指示、“position 1”は「イベント１」の再生位置から再生するという指示である。右側のメッセージ・ボックスでは、一番右側のアウトレットに送られるイベント番号の数値が「$1」[39]という符号に代入されて、受信用の「r player_A」オブジェクトに送信される。例えば、数値“２”が入力されると、「r player_A」オブジェクトに“cue 2”メッセージが送信され、数値“７”を入力されると、「r player_A」オブジェクトに“cue 7”メッセージが送信される。この部分のメッセージでは、再生速度処理モジュールに送信する。メッセージの内容に関する詳しい紹介は次の節の中で行う。

この様な仕組みを使用するメリットとして、イベント毎にシステムの再生を具体的に管理することができる。例えば、「図例3.7.3.2」のパッチはその一例である。最初のイベントであるイベント１で再生を開始し、イベント２から以後のイベントは演奏者の演奏時間に合わせる必要がある。一方、間にあるイベント11では、演奏者の演奏が含まれないため、Fixed Mediaをオリジナルの再生速度で再生する必要がある。この場合では、「sel」オブジェクトのアーギュメント欄に“11”というアーギュメントを指定し、それに対応するアウトレットから、特別にオリジナル速度で再生するためのメッセージを設定することによって、簡単に実現することができる。

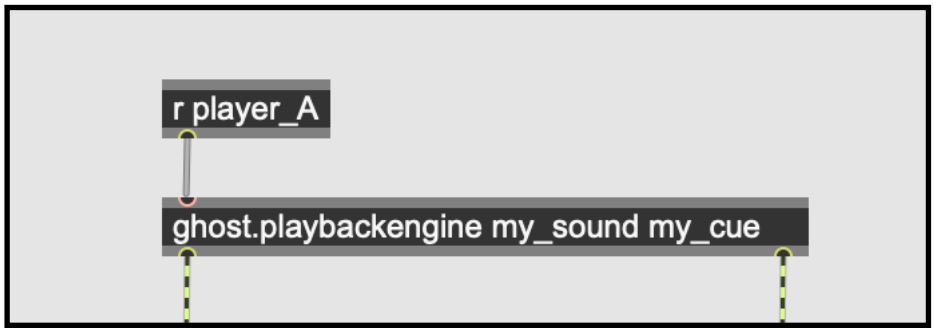


「図例3.7.3.2」イベント処理の例(２)。

[39] 「$」記号に数字の続く記号は、メッセージボックスの変数表現であり、受け取った数値をここに代入する。「$」記号の後の数字が“１”であれば、受け取ったメッセージの１つ目のデータをここに代入する。

3.7.4 再生エンジンと再生速度処理モジュール

再生エンジンと再生速度処理モジュールはGhostシステムの設計の最も中心的な部分である。この部分は、全てが「ghost.playbackengine」というアブストラクション・オブジェクトの中でプログラムされている。「図例3.7.4.1」「左上」の「r player_A」オブジェクトは、このオブジェクトに対するレシーバーである。先ほど紹介したイベント処理モジュールから送信されるメッセージ情報は全てここから入力される。オブジェクトのアーギュメント欄は、今回入力されるFixed Media作品とそのシステムスコアのファイル名である。



「図例3.7.4.1」再生エンジンと再生処理モジュールがプログラムされた
Maxアブストラックション・オブジェクト。

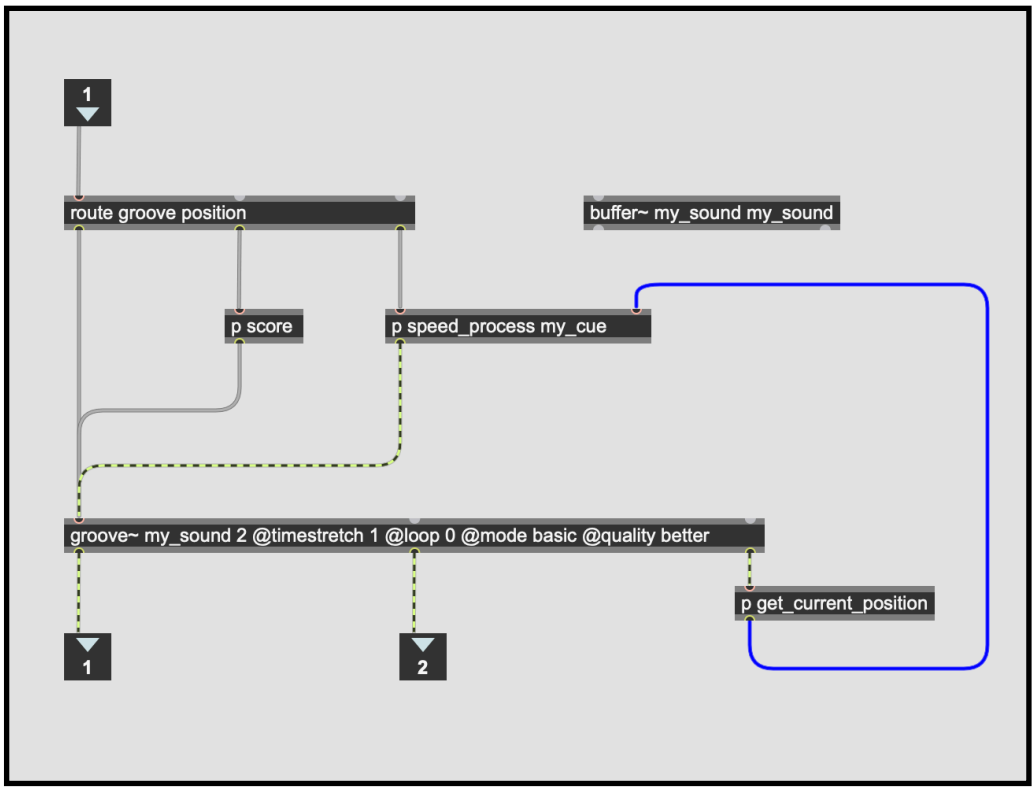
「ghost.playbackengine」オブジェクトの内部構造

「図例3.7.4.2」は「ghost.playbackengine」オブジェクトの内部構造を示している。この部分は「groove~」オブジェクトを制御するためにプログラミングされている。ここから論じるパッチの中身は全て実行モード状態なので、親オブジェクトのアーギュメント欄に入力されたFixed Mediaとシステムスコアのファイル名は既に必要な場所に入力された状態になっている[40]。ステレオ形式のFixed Mediaを使用する作品における演奏では、「groove~」オブジェクトの２つ目のアーギュメントに“２”を設定する[41]。その後に置かれた“@”[42]から始まる文字は、「groove~」オブジェクトの属性に関する設定を表している。“@timestretch 1”は、タイ

[40] パッチの中にある「buffer~」と「groove~」オブジェクトの１つ目アーギュメント欄は、編集モードの状態の中では“#1”と書き込んでいる。「groove~」オブジェクトが「buffer~」オブジェクトに格納されているオーディオデータを読み取るためには、その１つ目のアーギュメントを「buffer~」オブジェクトの１つ目のアーギュメントと一致させる必要がある。

[41] 「groove~」２つ目のアーギュメントは、再生のためのチャンネル数の設定である。2に設定すると、自動的にオーディオ用の2つの出力が用意される。

[42] オブジェクトの「アトリビュート（属性）」を設定するための記号。アトリビュートの書き方は「@アトリビュート名　その値」になる。



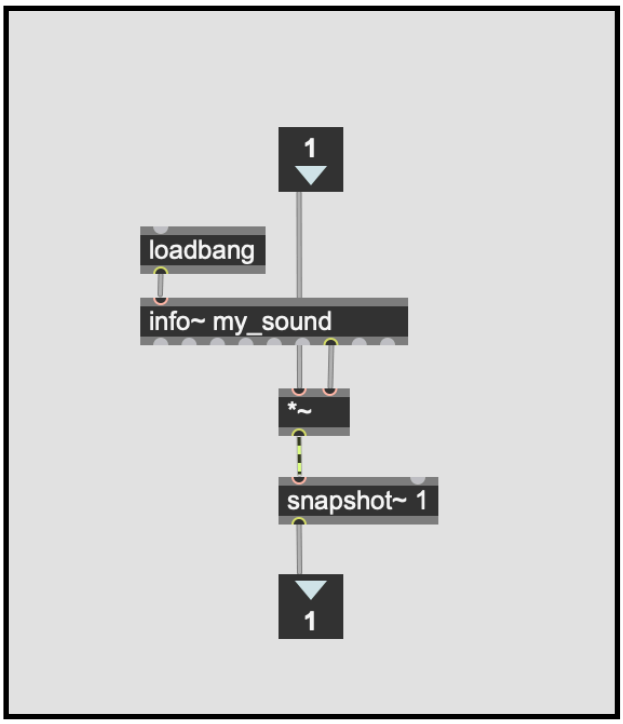
「図例3.7.4.2」「ghost.playbackengine」オブジェクトの内部構造。

ムストレッチ機能をオンにすることを意味する。“@loop 0”は、ループ再生をオフすることを意味する[43]。“@mode basic”と“@quality better”は、タイムストレッチのアルゴリズムを設定するためのものである。この部分は、使用するFixed Media音素材や、コンピュータの処理に負荷が掛からない限り、ここでの設定をそのまま利用することで、ほとんどの電子音の再生速度を聴覚上の劣化のない高音質のままリアルタイムに変化させることができる。

「図例3.7.4.3」には、「groove~」オブジェクトの右アウトレットから先に接続されているサブパッチの内部構造を示している。このサブパッチは、Fixed Mediaの再生時間をミリ秒単位に変換して、再生速度処理モジュール（「p speed_process my_cue」というサブパッチ）に送る役割を持っている。「groove~」の右側にあるアウトレットは、再生時間情報を0から1までによるオーディオ信号[44]として縮尺して出力する。例えば、再生開始時の最初のオーディオサンプルでは“0”を出力し、最後のオーディオサンプルでは“1”となる。この値を「p get_current_position」サブパッチの中にある「*~」オブジェクトの左インレットに送信する。「*~」オブジェクトでは、その値と「info~」オブジェクトの右から3番目のアウトレットより入力されたオーディオファイルの長さ（ミリ秒）の値が乗算され、ミリ秒単位の再生時

[43] Fixed Media伴奏を使用するライブ演奏は、伴奏音源をループさせる必要がない。

[44] Maxは、メッセージデータの処理に加えて、オーディオ信号の生成・処理も可能である。オーディオ信号の場合、緑と黒が絡み合っているようなパッチコードになる。オーディオ信号処理のためのオブジェクトは全て、オブジェクト名の末尾がチルダ記号（~）で終わる。



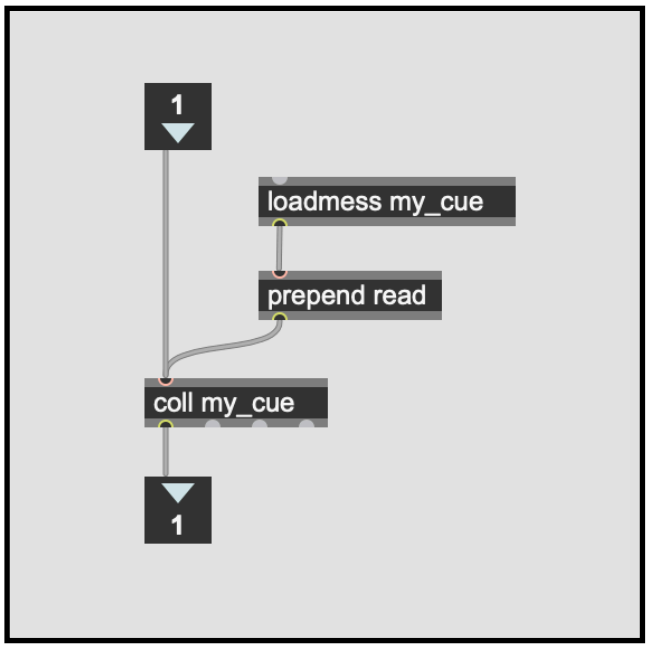
「図例3.7.4.3」サブパッチ「p get_current_position」
オブジェクトの内部構造。

間として出力される。最後に「snapshot~ 1」オブジェクトによって処理することで、1ミリ秒単位でサンプリングしたデータをメッセージデータの形に変換して出力する。

ここで、「図例3.7.4.2」の再生エンジンの主要構造を示すパッチの説明に戻る。パッチの左上にあるインレットは、外側にあるレシーバー（ここでは「図例3.7.4.1」の「r player_A」オブジェクトを示す）からのメッセージを受け取るために使われる。全ての受信メッセージは下部の「route」オブジェクトに送られ、メッセージ内のコンテンツが指定されたアウトレットから出力される。例えば、“groove”から始めるメッセージを受信した時に、grooveの後に続くコンテンツが「route」オブジェクトの左から１つ目のアウトレットから送られる。同様に“position”から始まるメッセージを受信した時に、その後のコンテンツは、「route」オブジェクトの左から２つ目のアウトレットから送られる。このシンプルな仕組みにより、送られてくるメッセージを必要な場所に送信することができる。

「route」オブジェクトの最初の“groove”メッセージで指定されたアウトレット（左から１番目）は、「groove~」オブジェクトを直接制御するためのメッセージを送信するために用意されている。例えば、再生を停止するための”stop”メッセージや、再生の開始位置を表す数値（ミリ秒）などを送ることができる。この出力は、特に「groove~」オブジェクトのリセットや初期化に使用される。

「route」オブジェクトに“position”メッセージで指定したアウトレット（左側から２番目）は、イベント番号に対応した再生時間情報を「groove~」オブジェクトに送信するために用意



「図例3.7.4.4」サブパッチ「p score」オブジェクトの内部構造。

されている。この出力は主にリハーサル時、システムに送信するイベント番に対応したFixed Mediaの時間から直接に再生するように使用する。そのアウトレットの先にある「p score」というサブパッチは、システムスコアを読み取るためのサブパッチである。「図例3.7.4.4」にサブパッチの内部を示す。中にある「coll」オブジェクトと「loadmess」オブジェクトのアーギュメント欄にはシステムスコアの名前が入力されている[45]。インレットから入力されたイベント番号は「coll」オブジェクトに送られ、「coll」オブジェクトではその入力番号に対応したFixed Mediaの時間を直接、出力することができる。例えば、“position”メッセージと一致するアウトレットから 1 が入力された場合、「coll」オブジェクトは“0”（オーディオデータの冒頭から再生する位置）を出力する。オブジェクトの上にある「loadmess」オブジェクトと「prepend」で構成された仕組みは、パッチの起動時のシステムスコアを自動入力する[46]。このサブパッチは、親アブストラクション・オブジェクトの中の様々な場所で使用され、各イベントの再生時間情報を出力するためにも使用されている。

「route」オブジェクトの一番右のアウトレットは、アーギュメントに指定されていないメッセージを受け取ると、そのメッセージをそのまま出力する。このアウトレットでは、再生速度処理モジュールに対する様々なコントロールメッセージを送信するように用意されている。

---

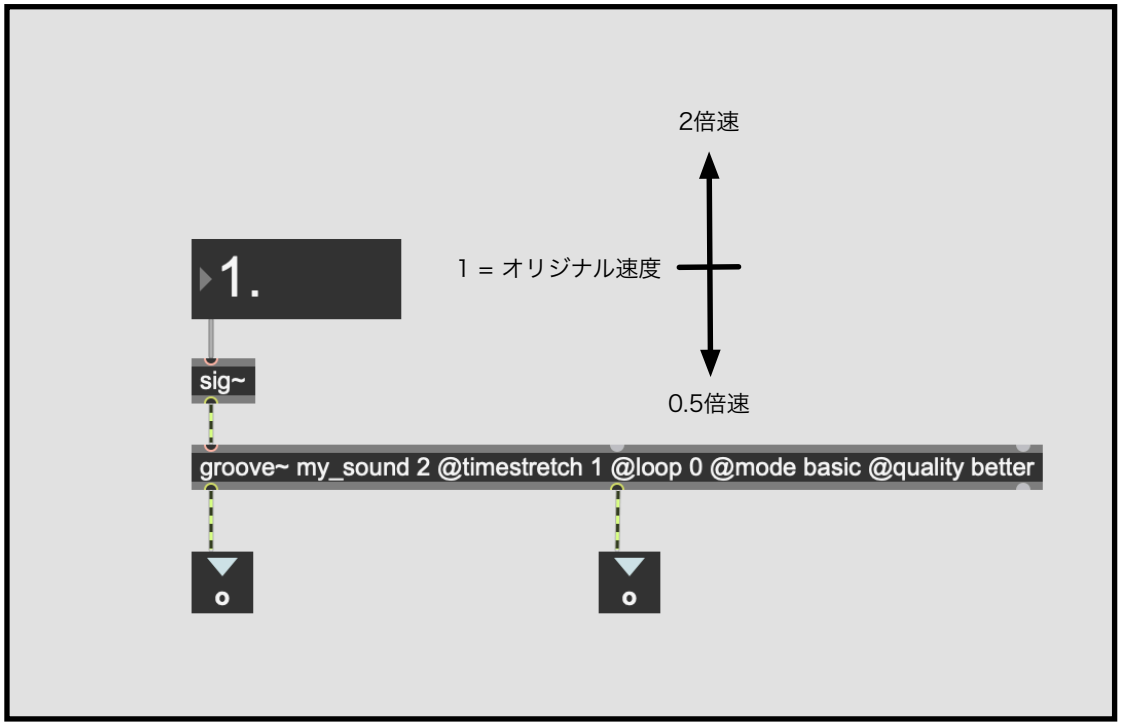
[45] それらのオブジェクトのアーギュメント欄は、編集モードの状態の中では“#2”と入力されている。

[46] 「loadmess」オブジェクトは、パッチを起動する際に、オブジェクトのアーギュメント欄に書き込んだメッセージを自動的に出力する。「prepend」オブジェクトに出力すると、書き込んだメッセージの前に指定された文字を付加する。この場合では、パッチを立ち上げる時に「coll」オブジェクトに“read my_cue”というメッセージを送信し、システムスコアを自動入力する仕組みを実現する。

再生速度処理モジュール

再生エンジンがシステムの心臓部だとすれば、再生速度処理モジュールはシステムの頭脳部と言える。このモジュールは、再生エンジンモジュールに再生速度情報を送信する役割を担っている。再生エンジンのコアは「groove~」オブジェクトなので、再生速度処理モジュールは、再生速度コントロール・パラメータを「groove~」オブジェクトに送信することに特化している。ここではまず、「groove~」オブジェクトが再生速度をコントロールする仕組みについて説明する。

「groove~」オブジェクトの左インレットでは、Maxのメッセージ・データを受信できる他に、オーディオ信号を受信することもできる。そのオーディオ信号の値は、「groove~」オブジェクトの再生速度を直接コントロールすることになる。「図例3.7.4.5」に、「groove~」オブジェクトにおける再生速度のコントロール方法を示している。

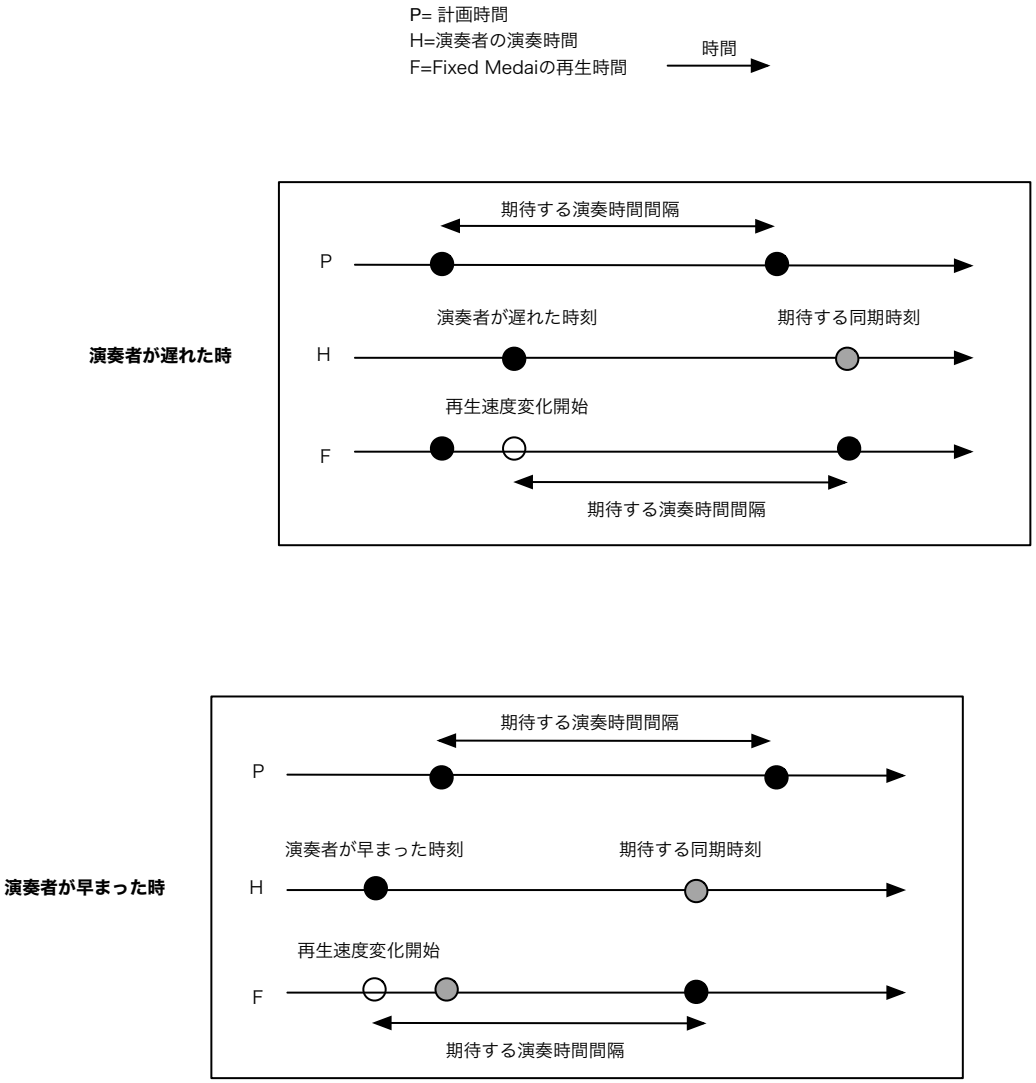


「図例3.7.4.5」「groove~」オブジェクトにおける再生速度コントロールの方法。

「groove~」オブジェクトの上部にある「sig~」オブジェクトは、メッセージデータをオーディオ信号に変換するために使用される。そのオーディオ信号の数値が“１”であれば「groove~」オブジェクトはオーディオデータをオリジナル速度で再生し、“２”であれば２倍速で再生し、“0.5”であれば0.5倍速で再生することになる。再生速度処理モジュールは、このパラメータをライブ演奏中に制御するために使用されるものである。この制御をどのように行うかは、Ghostシステムの同期メソッドによって決定される。

再生速度処理モジュールにおける同期メソッドの実装

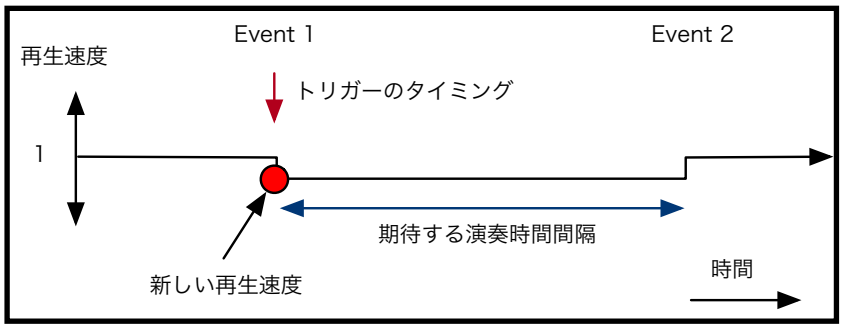
筆者の考えるGhostシステムは、新しい同期メソッドに基づき同期を図る（詳しい内容は2.4節に参照）。「図例3.7.4.6」は、この同期メソッドを図式化したものである。簡単に説明すると、ライブ演奏中に、演奏が予定同期時刻より遅れたり早まったりした時、その瞬間から新たな同期時刻を作り出し、演奏者、Fixed Mediaと一緒に新しい同期時刻に合わせて演奏（再生）することによって同期演奏を実現するという仕組みである。その新たな同期時刻は、演奏者の演奏時刻（遅れたり、早まったりした時）から、創作段階で想定した２つの同期イベント間の時間（つまり期待する演奏時間間隔）を加算したもので、いわば作曲者の「期待する同期時刻」と言えるようなものである。この同期メソッドを用いることで、一定の範囲内で時間的に自由な演奏表現が可能となるほか、制作段階で決定した演奏（再生）時間に最も近い同期演奏が期待できる。今回の研究においては、効果の高い演奏同期の結果が



「図例3.7.4.6」Ghostシステムが使用する同期メソッド。

期待できると言える。この新しい同期メソッドを実際のシステムで実現するためには、Fixed Mediaの再生は演奏者からトリガーを受け取った瞬間に、期待する同期時刻に同期させるために必要な再生速度の数値を算出する必要がある。その値は、変化の方法によってその算出方法が異なるため、事前に変化するためのパターンを決めておく必要がある。ここでは２種類の直線的な変化のパターンについて紹介し、比較的シンプルな計算方法について提供する。

再生速度パラメータの変化のパターン（１）



「図例3.7.4.7」再生速度パラメータの変化のパターン（１）。

「図例3.7.4.7」は１つ目の変化のパターンを示している。このパターンは、演奏者からのトリガーを受け取った瞬間に、新しい再生速度に変化し、次のイベントに達するまでその再生速度で再生を続ける変化のパターンである。このパターンを使用する場合、トリガーを受けた瞬間に、次のイベントに間に合うようなFixed Mediaの平均再生速度の値を求める必要がある。このため「図例3.7.4.8」に示す式（ラベル①）を適用して、平均再生速度 V_(n+1) の値を求める。

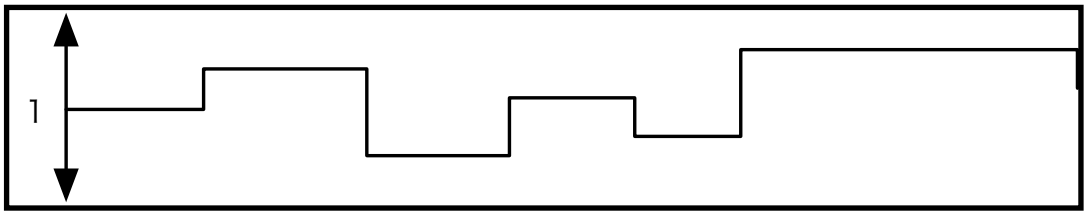
1

$$V_{n+1} = \frac{D_{diff} + D_{expc}}{T_{expc}}$$

「図例3.7.4.8」１つ目の変化のパターンにおける次のイベントに到達するまでの
平均再生速度の値を求める計算式。

D_{diff}は演奏者の演奏時間と予定再生時間の時間差、D_{expc} は作成段階で想定した２つのイベントの時間間隔である。ここでは単純に、D_{diff} と D_{expc} の和を予定した再生時間 T_{expc}で割ること

で必要な再生速度の値を求めることができる[47]。例えば、D_{diff} が正の数であれば、結果として得られる平均再生速度は 1 より大きくなり、D_{diff} が負の数であれば、結果として得られる平均再生速度は 1 より小さくなる。仮にこの変化のパターンをライブの中で連続的に使用すると、「図例3.7.4.9」のような再生速度の変化が想定される。実は、再生速度パラメータの変化は、全音、半音などの音程の変化や、４分音符と８分音符の変化などの譜面上の演奏情報の変化とは異なり、非常に繊細な感覚的な変化であるため、よほど極端な変化でない限り人間が明確に知覚することは困難である。その意味で、このようなシンプルな変化のパターンであっても高い実用性を持つと考えられる。しかし、Fixed Mediaのリズム・時間の変化の形によって、このような「瞬間的」な再生速度の変化では、不自然な表現になってしまう可能性もある。そのような場合には、よりスムーズな形の変化のパターンが求められる。

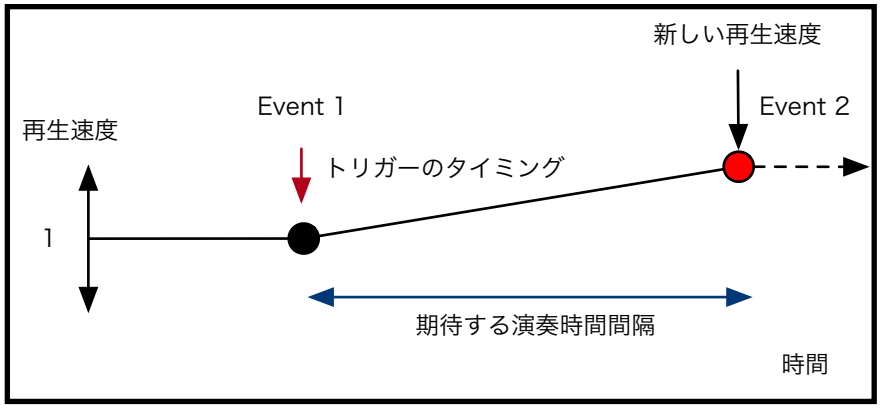


「図例3.7.4.9」再生速度パラメータ変化モデル（１）
（連続したイベントで使用する場合）。

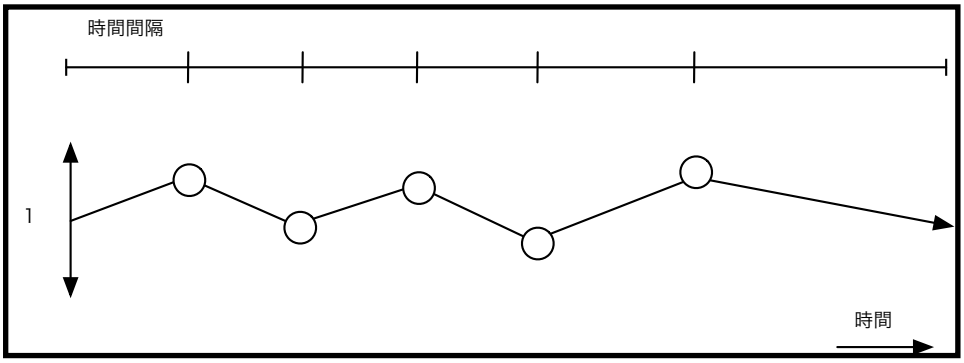
再生速度パラメータの変化のパターン（２）

「図例3.7.4.10」は２つ目の変化のパターンを示している。このパターンでは、演奏者からのトリガーを受けた瞬間に再生速度が急に変化するのではなく、次のイベントに間に合うような再生速度に直線的に補間しながら変化していくものである。仮にこのようなパターンをライブの中で連続的に使用すると、「図例3.7.4.11」のような変化図が想定される。再生速度の急激な変化は避けられ、各イベント間の再生速度をスムーズに連結させることができる。Maxでは、直線的な変化を作り出すために、「line」オブジェクトと 「line~」オブジェクトが用意されている。前述したように、「groove~」オブジェクトの再生速度の変化がオーディオ信号データによって制御されるため、ここではオーディオ信号データを出力する 「line~」オブジェクトを使用する。「図例3.7.4.12」に「line~」オブジェクトの動作の仕組みを示している。１つの数値を受け取ると、その数値をオーディオ信号データに変換して出力する。２つの数値からなるリストを受け取ると、それを“目標値　目標値への到達する時間”と解釈す

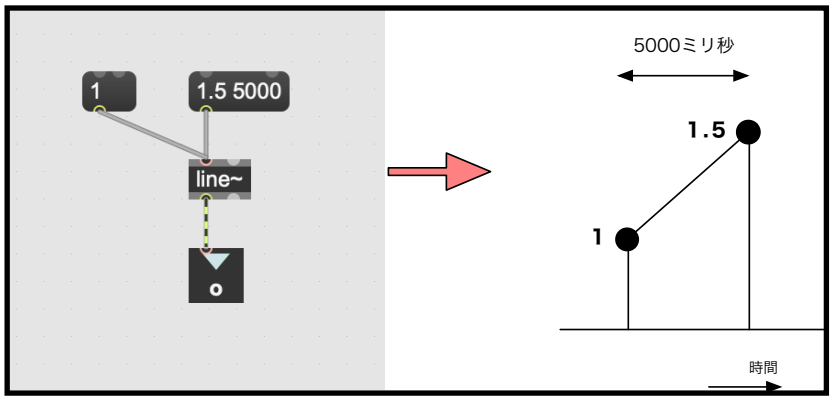
[47] 「図例3.7.4.7」を基にした例）演奏者が「Event 1」に到達したとき、その時点の Fixed Media 再生時間から700ミリ秒の時間差がある。「Event 1」から次の「Event ２」までの時間間隔を7000ミリ秒と仮定する。すると、「Event 1」に到達したときのFixed Mediaの再生速度は、(700＋7000)/7000と算出される。その結果、再生速度の値は1.1となる。



「図例3.7.4.10」再生速度のパラメータ変化モデル（２）
直線的かつ滑らかに変化するモデル。



「図例3.7.4.11」再生速度パラメータ変化モデル（２）
（連続したイベントで使用する場合）。



「図例3.7.4.12」「line~」オブジェクトの入力について。

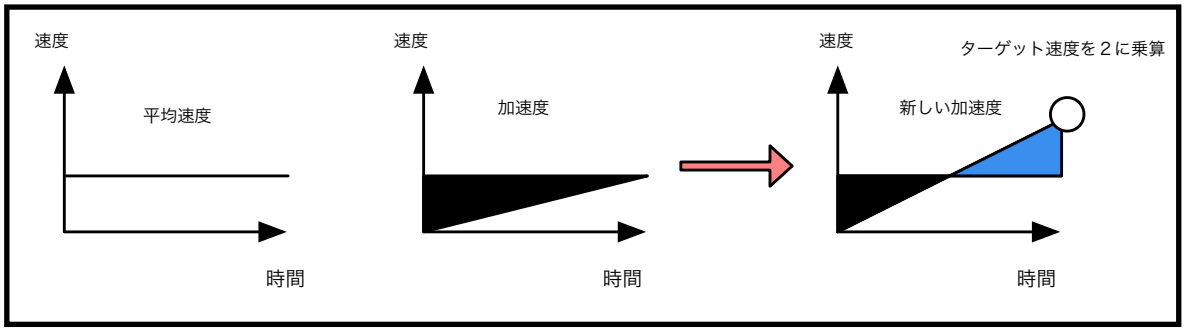
る。例えば、“1.5 5000”のリストを受信した場合、「line~」オブジェクトは、5000ミリ秒をかけて、出力信号値を現在の信号値（ここでは再生速度の値）から1.5へと漸次的に変化させる。1.5に到達した後、次のトリガーを受け取るまでオーディオ信号の数値は変化しない。1.5に到達する前、新しい目標値を受け取ると、その時点の値から目標値に到達するまでの新しい変化を作りだす。このような「line~」オブジェクトの動きを２つ目の変化のパターンに適用させるためには、単純に「目標再生速度」と「目標再生速度の値へ到達する時間」という２つの値を用意すればよい。「目標再生速度の値へ到達する時間」では「期待する演奏時間」の値をそのまま使用するので、特別な計算をする必要はない。残りの「目標再生速度」の値だけ求める必要がある。ここでは「図例3.7.4.13」に示す式（ラベル②）を適用して、目標再生速度V_(n+1) の値を求める。

2 $$V_{n+1} = \frac{2D_{diff} + D_{expc}}{T_{expc}} + (1 - V_{current})$$

「図例3.7.4.13」２つ目の変化のパターンにおける次のイベントに到達する
時の目標再生速度の値を求める計算式。

「図例3.7.4.13」の式は、１つ目の変化のパターンの式（ラベル１）と比べて、2つの項の計算過程が異なっている。１つ目は式の左側の分数の分子で、演奏者の演奏時間と実際の演奏時間の差である D_{diff} に２を掛け算するものである。２つ目の部分は、数式の右側の部分に加えられる(1-$V_{current}$) の項である。これらの項がないと、この変化のパターンを適用しても、Fixed Mediaを指定された時間に合わせて再生することはできない。ここでは、なぜそれぞれの項が必要なのかを説明する。

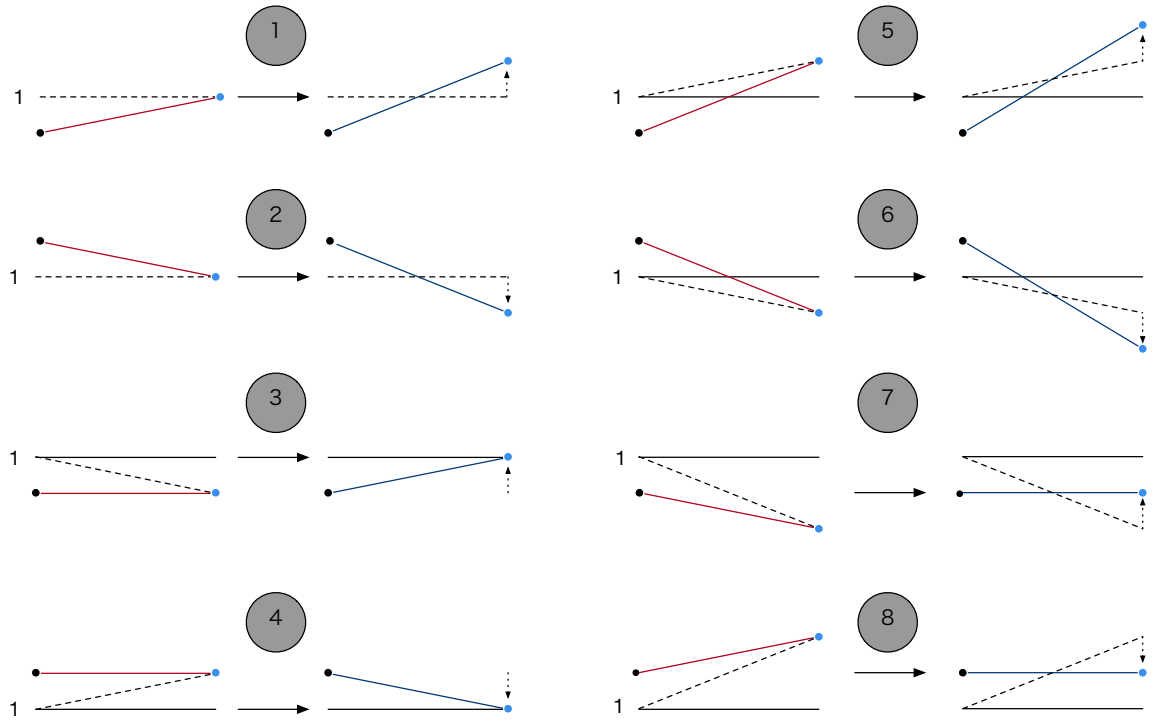
まず、１つ目の部分の違い（２D_{diff} という箇所）について理由を説明する。新しい変化パ



「図例3.7.4.14」D_{diff}を２に乗算する理由を示した図解図。

ターンを適用するFixed Mediaの再生速度は、目標再生速度へと漸次的に変化させるため、新しい変化パターンによる実際の再生時間の変化量は、新再生速度へと瞬時に変化した場合の変化量の半分となる。その変化量を補うためには、事前に行うべき変更量を２倍にする必要がある。「図例3.7.4.14」は、この計算で、不足している変化量を「補正」するプロセスを示している。

次に、式の右側の項 $(1\text{-}V_{current})$ に足す必要な理由を説明する。ここでいう $V_{current}$ は演奏者からトリガーを受け取ったときの再生速度の値である。この部分は、２番目以降のイベントに関する処理に必要となる。その理由は、最初のイベントの後、Fixed Mediaの再生は基本的にオリジナルの速度から変更されることはないが、ほとんどの場合、変更された状態から次のイベントに向けて変更していくからである。例えば、$+(1\text{-}V_{current})$ が式の中に存在しない場合、分子で使用する時間係数はオリジナルの速度でのデータなので、最終的にはオリジナルの速度である“１”からの変化による正確な再生速度にしかならない。しかし、イベント１以降のイベントではオリジナルの速度である１から変化するのではなく、“１”よりも速いか遅いかのいずれかの速度から次の速度への変化するため、正確な計算結果を得るためにその部分



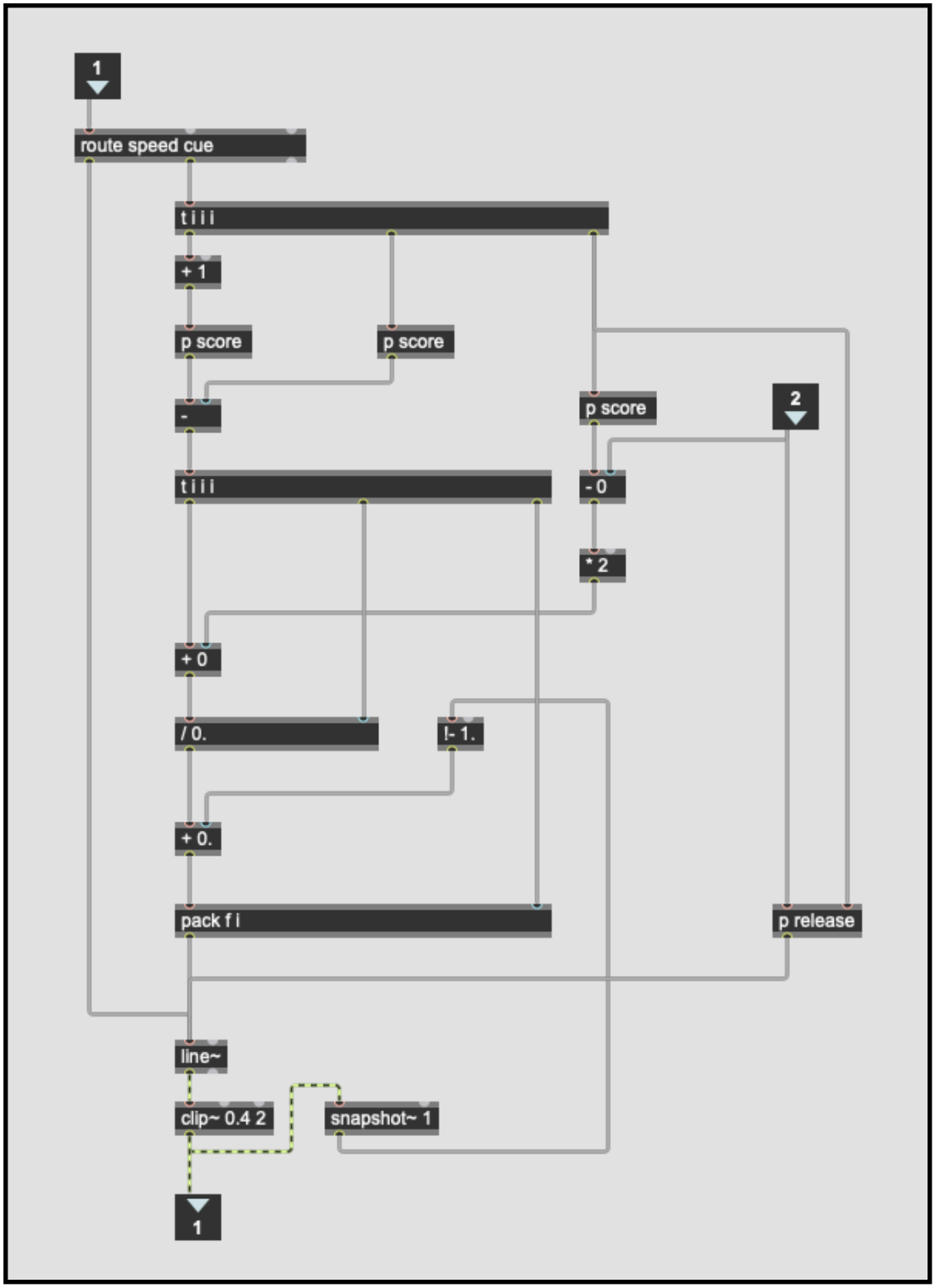
「図例3.7.4.15」式の右側に「$+(1\text{-}V_{current})$」が必要となる理由を示した図解図。
線の横にある数字の“1”はオリジナル速度の意味を示す。
赤線は、上述の計算処理を行わない場合のFixed Mediaの再生速度の変化を示す。
青線は、上述の計算処理を行った場合のFixed Mediaの再生速度の変化を示す。
点線は、オリジナルの速度から変化した場合の仮想的なケースを示す。

に対して「補正」しなければならない。具体的には、オリジナルの速度である“１”から、トリガーを受けたときの現在の再生速度値$V_{current}$を減算し、その結果を左側の分数の計算部分に加えるという計算になる。「図例3.7.4.15」はライブ中に演奏者らからのトリガーをシステムが受け取った際に発生しうる様々なパターンに対して、上述した計算処理により「補正」を行う過程を図示したものである。「ラベル１」と「ラベル２」は、演奏者の時間とFixed Mediaの再生時間（位置）が完全に同期しているが、Fixed Mediaの再生速度が本来の速度よりも遅くなったり速くなったりしている例である。「ラベル３」は、遅い再生速度で再生されるFixed Mediaの時間よりも、演奏者の演奏時間がそれと同じ値だけ遅れた例である。「ラベル４」は、速い再生速度で再生されるFixed Mediaの時間よりも、演奏者の演奏時間がそれと同じ値だけ早まった例である。「ラベル５」は、遅い再生速度で再生されるFixed Mediaの時間よりも、演奏者の演奏時間が早まった例である。「ラベル６」は、速い再生速度で再生されるFixed Mediaの時間よりも、演奏者の演奏時間が遅れた例である。「ラベル７」は、速い再生速度で再生されるFixed Mediaの時間よりも、演奏者の演奏時間がさらに早まった例である。「ラベル８」は、遅い再生速度で再生されるFixed Mediaの時間よりも、演奏者の演奏時間がさらに遅れた例である。各ラベルに読み取れるように、システムを演奏者らからのトリガーを受け取った瞬間にFixed Mediaの再生速度の値が“１”でない場合に、異なった再生速度により必ず余分な再生時間が生じてしまうことになる。しかし、オリジナルの速度値“１”引くトリガーを受け取った瞬間の再生速度の値を、分数の部分の計算結果に加えることによって、その余分な部分を「補正」させることはできる。このことにより、最終的に期待する同期時間に正確に合わせて再生することができる。よって、Ghostシステムの同期メソッドを用いることで、想定される全ての変化パターンに対して同期を完全に実現することができる。

サブパッチ「speed_process」の内部構造

「図例3.7.4.16」は、再生エンジン・モジュール（「図例3.7.4.2」）内のサブパッチ「p speed_process」の内部構造を示している。つまり再生速度処理モジュールの中身である。このパッチは、「line~」オブジェクトに対して再生速度を変化させるシグナルを生成するようにプログラミングされている。パターン②（「図例3.7.4.13」）を使用し、目標値まで滑らかに変化する。パターン①（「図例3.7.4.8」）の式とは異なる部分に該当する部分をパッチ内で省略できるため、ここでは技術的な説明は割愛する。

パッチの上部にあるインレット（ラベル１）は、再生エンジン内（「図例3.7.4.2」）の「route」オブジェクトの一番右のアウトレット（アーギュメント欄に一致しないメッセージを受け取った際に、こちらのアウトレットから出力される）からのデータを受け取る。その中で“speed”メッセージに一致するメッセージを受け取ると、「route」オブジェクトの左から１つ目のアウトレットから出力される。“cue”メッセージに一致するメッセージを受け取ると、「route」オブジェクトの左から２番目のアウトレットから出力される。左から１つ目の



「図例3.7.4.16」サブパッチ「p speed_process」の内部構造
（「line~」オブジェクトによる速度変化を前提とする）。

アウトレットは「line~」オブジェクトを直接制御するためのメッセージである。例えば、再生開始時に再生速度に“1”を送るメッセージや、再生速度の変更中に再生速度を一定の時間をかけてリセットするメッセージなどである。左から２つ目のアウトレットは、演奏者らからのトリガーのイベント番号を送信する。これより下の「pack f i」オブジェクトまでの部分は、再生速度を変化させる２つ目のパターンを生成するために、「line~」オブジェクトが読み取るメッセージ（目標再生速度の値とそれに到達するまでの時間の値のリスト・メッセージ）を生成するための仕組みである。

イベント番号をその先に繋がれた「t i i i」オブジェクトに送信する。「t i i i」はそのイベント番号の値を右から左に向かって３箇所に分けて順次に送信する。右から１番目のアウトレットでは、先にイベント番号をサブパッチ「p score」に送信し、演奏者の演奏時間（ミリ秒）の値に変換して出力する。そして、その値から“Fixed Mediaの現在の再生時間”を減算して、２を掛けることで、数式の $2D_{diff}$ の値を求める。ここではまず、次の処理を行うための準備として、 $2D_{diff}$ の値を「+ 0」オブジェクトの右インレットに入力し、「+ 0」オブジェクトの中に格納する。「t i i i」オブジェクトの左から２番目のアウトレットと左から１番目のアウトレットの出力では、まず次のイベントの時間から現在のイベントの時間を引いて、「期待する演奏時間間隔」の値 D_{expc} 及び T_{expc}[48]を求める（ミリ秒単位）。この値をその先に繋ぐ「t i i i」オブジェクトに送信し、右から左順で３箇所に分けて送り出す。右から１番目のアウトレットは、下にある「pack f i」オブジェクトの右インレットに目標再生速度に到達するまで必要な時間 T_{expc} の値を格納する。ここで「line~」オブジェクトに送り出すためのリストの２番目の数値の準備を完了する。「t i i i」オブジェクトの右から２番目のアウトレットでは、T_{expc} の値を割り算するための「/ 0.」[49]オブジェクトの右インレットに入力し、数式の分数部分の分母の値を設定する。左から１番目のアウトレットは D_{expc} の値を「+ 0」オブジェクトに送信し、先ほど算出した $2D_{diff}$ の値に足すことで、数式の$2D_{diff}+D_{expc}$ の値を算出する。その結果値を「/ 0.」オブジェクトに送り出し、先ほど用意してきた T_{expc} の値に割り算をすることで、数式の分数の部分の値（実数）を求める。「/ 0.」の先に繋ぐ「+ 0.」オブジェクトは、目標再生速度を「補正」（数式の $+(1-V_{current})$ の部分）するための仕組みである。「+ 0.」オブジェクトの右インレットに繋ぐ「!- 1.」オブジェクトは、オリジナル速度１から、現在の再生速度[50]を引く計算を実行する。ここで出力される値は目標再生速度の値である。その値を「pack f i 」オブジェクトに送信して、先ほど格納した T_{expc} の値とリストにして出力される。「line~」オブジェクトはリストを受け取り、再生速度の変化のオーディオ信号を作り出す。「clip~」オブジェクトは、再生速度の変化のシグナルに指定した範囲に限定し出力す

[48] Ghostの同期メソッドであるため、ここで２つの値は等しい。

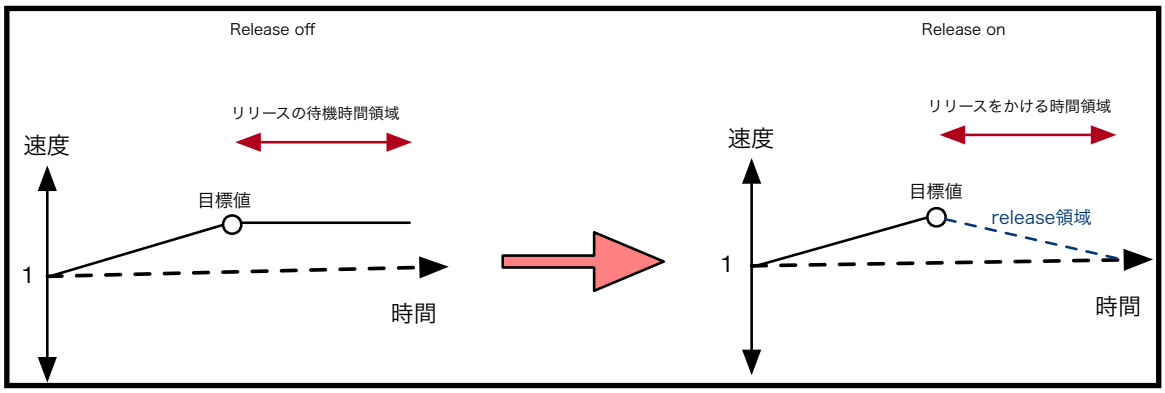
[49] アーギュメント欄の数値“０”に小数点を付加することにより、実数での処理が可能である。

[50] 「clip~」オブジェクトから出力されるシグナル（再生速度）を「snapshot~ 1」オブジェクトによって、1ミリ秒単位でサンプリングした現在の再生速度の値を出力する。

る（ここでは最小限0.5から最大限の２倍速までとする）。最終的に再生速度のオーディオ信号を、親パッチの中にある「groove~」の左インレットに入力し（「図例3.7.4.3」に参照）。Ghostの同期メソッドに適用して、Fixed Mediaの再生速度を変化させる。

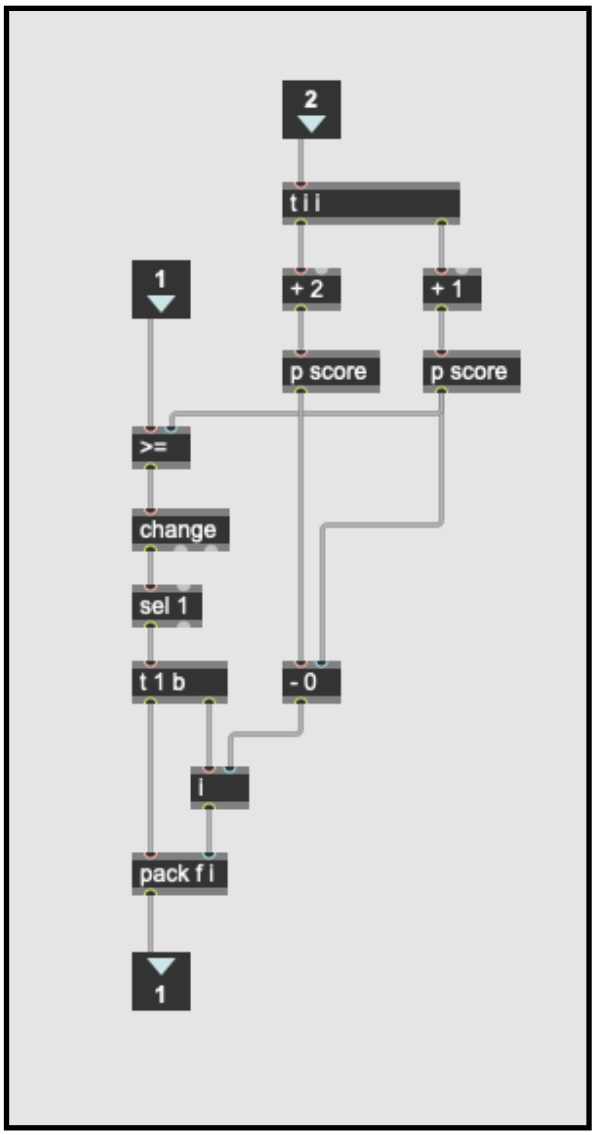
サブパッチ「p release」について

ここでサブパッチの右側にあるもう１つのサブパッチ「p release」について説明する。このサブパッチは、再生速度を自動的に「リリース」（オリジナル速度“１”に戻るための変化）するために設計した仕組みである。このモジュールは、Fixed Mediaをできるだけ元の速度に近い速度で（作曲者の意図した速度で）再生するためのものである。システムによって再生速度が変化させられた後に演奏者から次のイベントのトリガーを受け取っていない時、このモジュールは再生速度を自動的に１に戻し、これにより作曲者の意図した速度による再生が実現する。「図例3.7.4.17」はその仕組みの流れを可視化した図である。図の左は再生速度をリリースしない場合の再生速度を示す。目標再生速度に到達した後に演奏者らからのトリガーを受け取っていなければ、Fixed Mediaは、変化されてきた再生速度のままの中で継続して再生してしまうことになる。結果として作曲者が意図していないFixed Mediaの再生になってしまう可能性がある。このような時間を短縮させるために、再生速度の目標値に達成した後に、オリジナル速度へ「リリース」をさせる。このようにしてFixed Mediaはなるべくオリジナル速度の“１”に近い状態の中で再生することができ、より高い品質での再生することが期待できる。



「図例3.7.4.17」releaseの仕組みに関する図解図。

「図例3.7.4.18」はサブパッチの内部を表している。左側のインレット（ラベル１）にFixed Mediaの現在の再生速度の値が入力される。右側のインレット（ラベル２）に、演奏者らのトリガーを受け取った際のイベント番号が入力される。イベント番号が入力された時、「ｔｉｉｉ」オブジェクトによって、次の１つ後のイベントの時間から次のイベントの時間を減



「図例3.7.4.18」サブパッチ「p release」の内部構造。

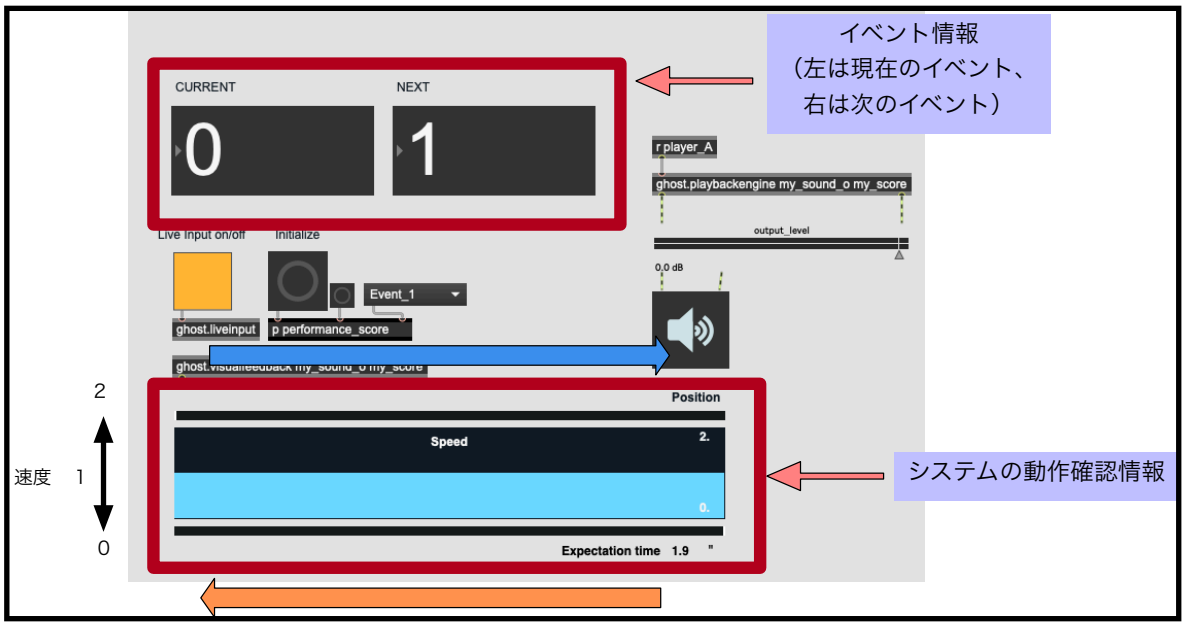
算して、その２つのイベントの間の期待する演奏時間の値を求める。その数値は、ここでオリジナル速度の”に戻るための「リリースをかけるための時間間隔」として使用する。まずはその数値を整数で出力するための「i」オブジェクトの右インレットに入力し、オブジェクトの中に格納する。「>= 」オブジェクトは、入力し続けるFixed Mediaの現在の再生時間と右インレットに入力された次のイベントの時間に比較し、Fixed Mediaの再生時間は次のイベントの時間に到達した時に”を出力する。「change」オブジェクトに渡ることで、変化しない時のデータをフィルタリングさせる。「sel 1」オブジェクトは、「change」オブジェクトから送られる１を受け取ると、その先に繋がれた「t 1 b」オブジェクトに“bang”メッセージを送信する。「t 1 b」オブジェクトは、“bang”メッセージを受け取ると、まず右側のアウトレットに“bang”メッセージを出力し、その先に繋がる一連の処理が完了すると、次に左側のアウトレットにオリジナル速度を意味する数値“１”を出力する。「i」オブジェクトは、「t 1 b」オ

ブジェクトからの“bang”メッセージを受け取ると、先ほど格納した「リリースをかけるための時間間隔」を出力する。「pack f i」オブジェクトは、右側のインレットに「i」オブジェクトからの「リリースをかけるための時間間隔」を受け取ると次に左側のインレットに「t l b」オブジェクトから数値“ 1 ”を受け取り、数値“ 1 ”を受け取ると「line~」オブジェクトに対して目標再生速度の“ 1 ”と“再生速度をリリースさせるための時間”のリスト・メッセージを送信する。これにより、再生速度を 1 に戻すように変化させることができる。

3.7.5 ビジュアル・フィードバック

システムのビジュアル・フィードバックの部分について説明する。Ghostシステムは主にコンピュータが演奏者の演奏時間に合わせるために設計されている。しかし、リハーサルの段階やシステム調節の段階の中では、システムの動きを詳しく知ることができるビジュアル・フィードバックのモジュールが必要とされている。そのモジュールのデザインは、基本的に自由なものであるが。ここで筆者が使用するものについて紹介する。

「図例3.7.5.1」にGhostシステムのメインパッチのインタフェースの中で、長方形のマークに囲まれている箇所は筆者が使用するビジュアル・フィードバックのモジュールである。左上の部分は、トリガーのイベント番号を確認する部分である。ここでは、現在のトリガーのイベント番号と次のイベント番号の２つの情報を提示している。この部分は基本演奏者が自らのトリガーが送信できたことを確認するために使用する。画面の下の部分ではFixed Mediaの再生の状態を示している。上と下の計器は、Fixed Mediaの再生時間位置を示すことができる。上の計器は左から右に向けて移動するカーソルの動きを表す。トリガーを受け取った後に、次のイベントに到達するまでの動きである。下の計器は、右から左に向けて移動するカーソル



「図例3.7.5.1」ビジュアル ・フィードバック。

の動きを表す。次のイベントに到達した後に演奏者らからの次のイベントのトリガーを受け取っていない場合における、次のイベントからその１つ後のイベントまでの動きである。システムの通常設定では、この時点から再生速度にリリースが適用されている。下のビジュアル・モジュールの真ん中の計器は再生速度の状態を表す。線が中心の位置になっていれば、再生速度はオリジナルの１である。上下に参考できる幅は再生停止の速度０から早い速度の２までに設定されている。この部分は、主にシステムを調節する時や、リハーサルの時に使用される。例えば、システムの動きを詳しく把握できるオペレーターがトリガーを送信する操作を行う場合に、この部分のビジュアル情報を参考しながら、Fixed Mediaの再生速度をさらなるバランスのよい状態の中で演奏者の演奏速度に合わせることが期待できる。上記に紹介したビジュアル・フィードバックに関するプログラミングは、容易に実現できるため、ここでの技術的な紹介は割愛する。

3.8 まとめ

以上が、Ghostシステムの全体像である。ここで紹介したプログラミング手法は、第２章で述べたようなMixed music同期問題を解決・改善することに対して有効性の高いライブ同期システムを実現するための方法を、高い完成度で実現したものである。実際、筆者がこのシステムを用いて実証したところ、このプログラムは、20秒までの２つのイベント時間間隔内での加速と減速の変化において、目標値に到達するまでの時間誤差をプラスマイナス５ミリ秒までに抑えることができ、第２章の中で論じたコンセプトを極めて高い精度で実現したと言える[51]。Ghostシステムが実際のライブ現場で有効かどうかを議論するためには、ライブで実施し、それに対して検証することが最も有効な方法だと考えている。次章では、さまざまな作品を用いて実証演奏を行い、その結果からシステムの有用性について考察を行う。

[51] システムのサンプリングレートの制限により、時間的な誤差を回避することはできないが、5ミリ秒以内であれば無視できる。

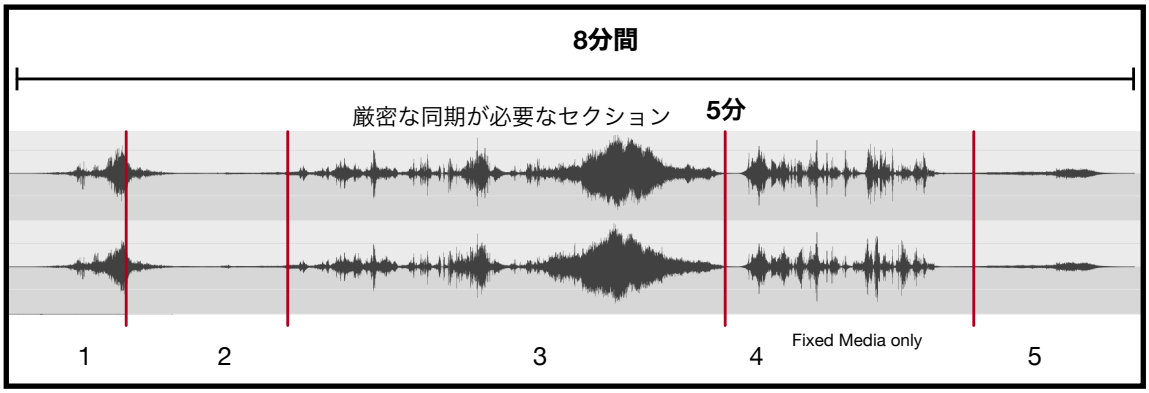
第４章 システムの検証

4.1 概要

本章では、第３章で提案した新しいシステム『Ghost』の有用性の検証について述べる。2018年１月頃にGhostシステムの最初のバージョンを完成させて以来、筆者が自作したMixed music作品や他の作曲家が作曲したMixed music作品など、様々なスタイルの作品の演奏の際にこのシステムを使用してきた。これらの作品の実際のライブ演奏の結果は、本システムの有効性を裏付ける客観性の高いデータとなっている。本章では、代表的な３作品の演奏結果を紹介しながら、このGhostシステムの有効性について論じてみたい。

4.2 《Equal-G》

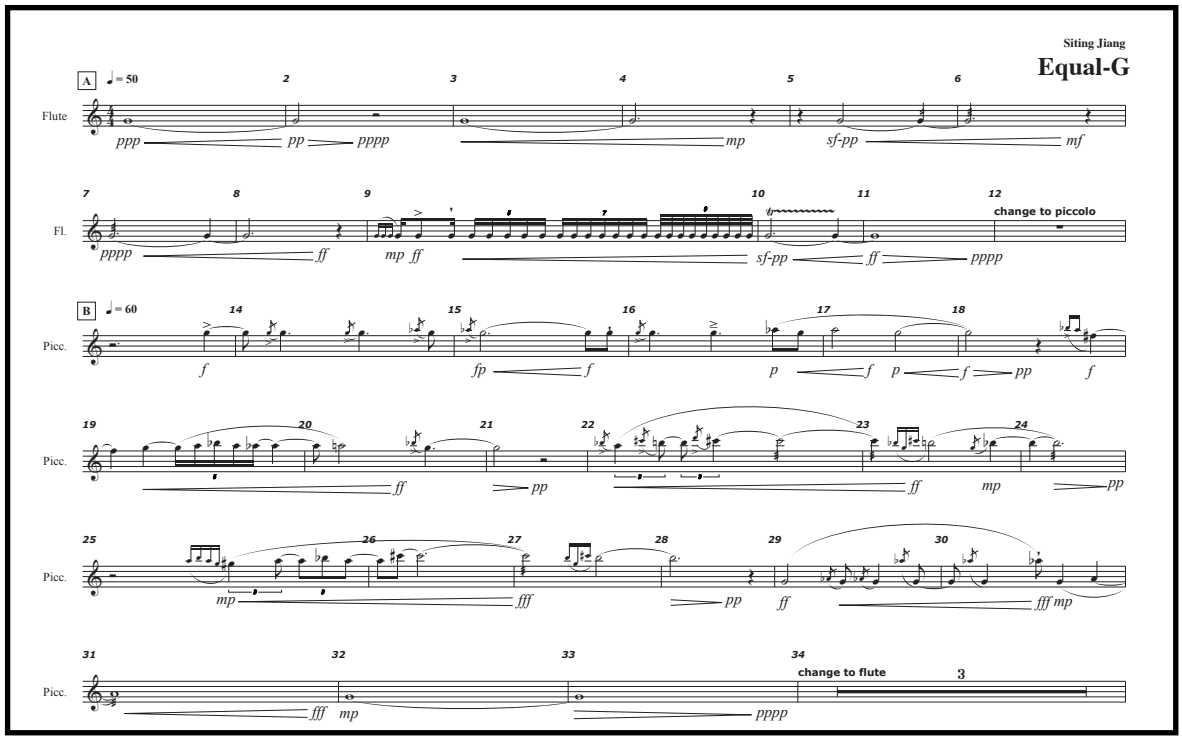
まずは筆者の自作、フルートとFixed Mediaのための作品《Equal-G》（2013）について論じる。この作品は、筆者が初めてMixed musicというスタイルを取り入れた作品である。創作スタイルとしてのMixed musicの魅力を体験すると同時に、ライブ演奏における時間同期の難しさを痛感した作品でもある。この作品は、８分にわたって再生されるFixed Mediaの電子音響に、フルートのライブ演奏を同時に行う必要があり、その中でも、楽曲のセクションによって、求められる同期の精度が異なることが特徴である。「図例4.2.1」は作品の時間構成を示している。セクション１、２、５はFixed Mediaの時間に対して厳密な同期は必要とされないが、セクション３のみ、厳密な同期を必要とする。セクション４はFixed Mediaの再生のみであるため、同期をとる必要はない。ここでは、2013年８月に国立音楽大学コンピュータ音楽研究室が開催した『Sonic Interaction 2013 Vol.2』で、フルーティスト・白柏幸恵氏によるシステムを使用していなかった初演の演奏結果[52]と、本システムを初めて使用したとき



「図例4.2.1」《Equal-G》のFixed Mediaの波形。

[52] 《Equal-G》初演時の記録映像は以下より視聴可能
https://youtu.be/rNuSb3fOBNo (accessed September 7, 2020).

「図例4.2.2」《Equal-G》の初演の様子。



「譜例4.2.3」《Equal-G》の演奏楽譜の 1 ページ目。

の演奏結果を比較することで、本作品を実演する上でのGhostの有用性について論じたい。

初演時の演奏について

初演では、視覚的なクリックトラックを使用することで、演奏の同期を実現した。「図例4.2.2」は初演時の演奏写真である。演奏者の前に演奏パートの楽譜が置かれ、譜面台の左側にあるディスプレイには、Fixed Meidaの再生時間情報が表示されている。クリックトラック形式の提示を行っているため、ディスプレイには演奏者による正確な演奏時間（小節と拍）が表示されている。「譜例4.2.3」は、初回演奏に使用した演奏者の演奏譜である。演奏者用楽譜はクリックトラックの使用を想定しているため、テンポは「50」や「60」に設定されており、拍子や時間的なフレージングの細かな指定はない。技術力の極めて高い演奏者に恵まれたこともあって、完成度の高い初演となったが、演奏者は常に画面左側のディスプレイに表示される「クリックトラック情報」を確認しながら演奏しなければならないため、演奏者自身の自発的な時間表現が大きく制限されてしまうと筆者は感じていた。初演時に感じたこのような制約への煩わしさが、本研究を始めた起因ともなっている。

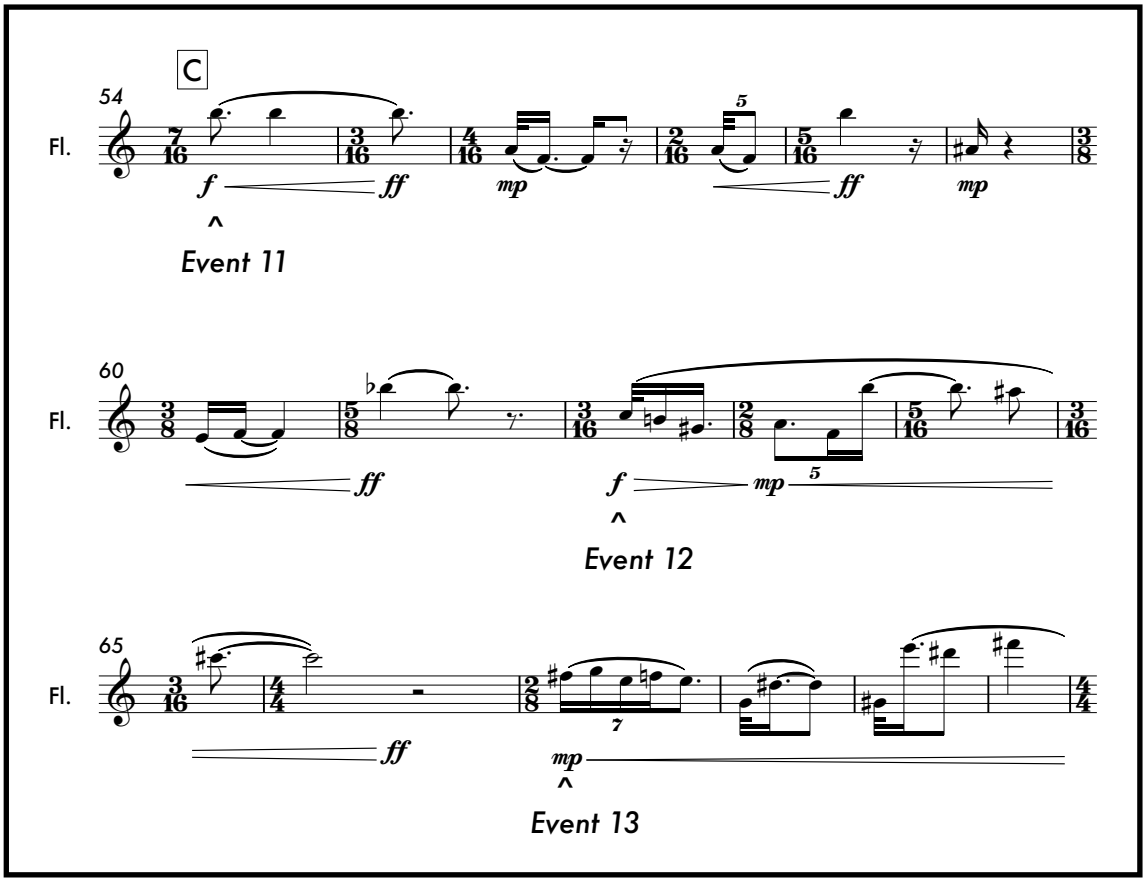
Ghostシステムを使用した演奏について

2018年５月に、宮本貴史氏主催のメディアアートイベント「Acousticlub Vol.3」において、本作は、Ghostシステム開発以来初となる演奏の機会を得た[53]。フルーティスト・渡邉玲子氏を迎え、さらにシステムの導入にあたり、まずは演奏譜の時間表記を大きく変更した。「譜例4.2.4」はGhostシステム使用時の演奏者パートの楽譜である。「譜例4.2.3」の楽譜の拍子表記とは対照的に、クリックトラックの提示を参照する必要がないため、より複雑なリズムや拍子を要求することが可能である。「譜例4.2.4」に示す楽譜は、筆者が意図した時間リズム表現に最も近い形になっている。このような楽譜を用いたライブ演奏からは、より音楽的、時間的に表現力のあるライブ演奏が期待できる。

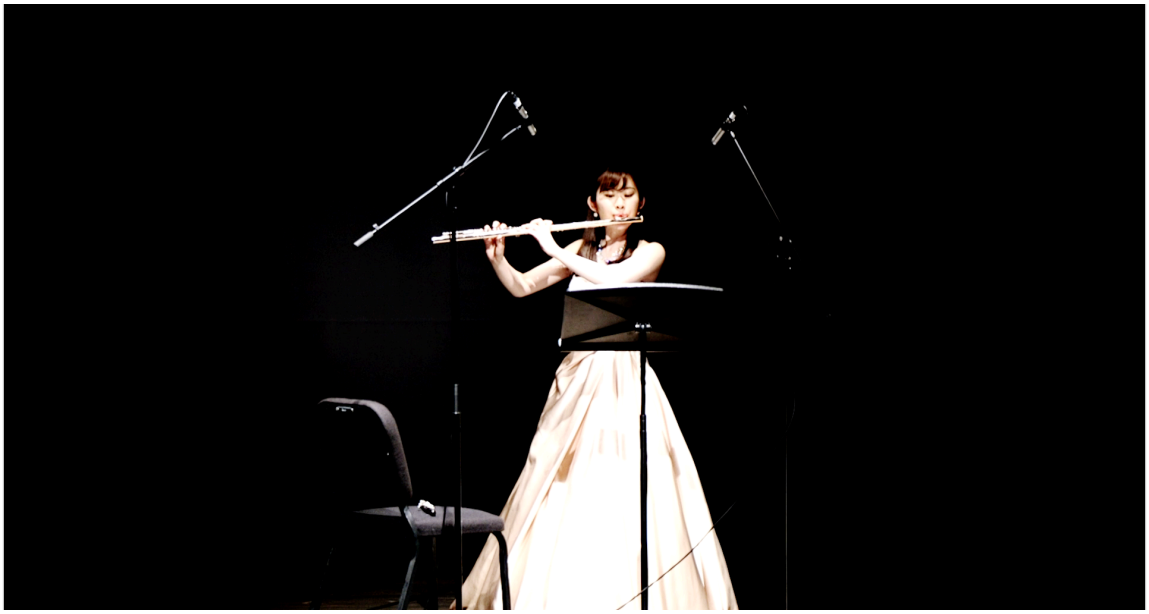
「図例4.2.5」はGhostシステムを実際に使用した際の演奏写真である。今回の演奏では、筆者が演奏と同時に操作するオペレーターの役割を担当し、演奏時に合わせてシステムにトリガーを送る。このような方法を採ることで、クリックトラックなどの従来のセットアップを省略し、演奏に必要なものを楽譜や楽器、マイクなどの最小限のもののみに絞り込むことができる。その結果、従来のセットアップ（クリックトラックなどを使用）よりもシンプルな、視覚的にも整った印象の演奏を実現できた。Ghostシステムを使うことで実現するこのようなシンプルな演奏環境による実演は、リハーサルから本番まで全ての演奏において、技術的な問題もなく、完成度の高い同期演奏を実現することができた。その上さらに、高性能のタイムストレッチ技術により、まるでオリジナルの音質を再生しているかのような聴取感覚と

---

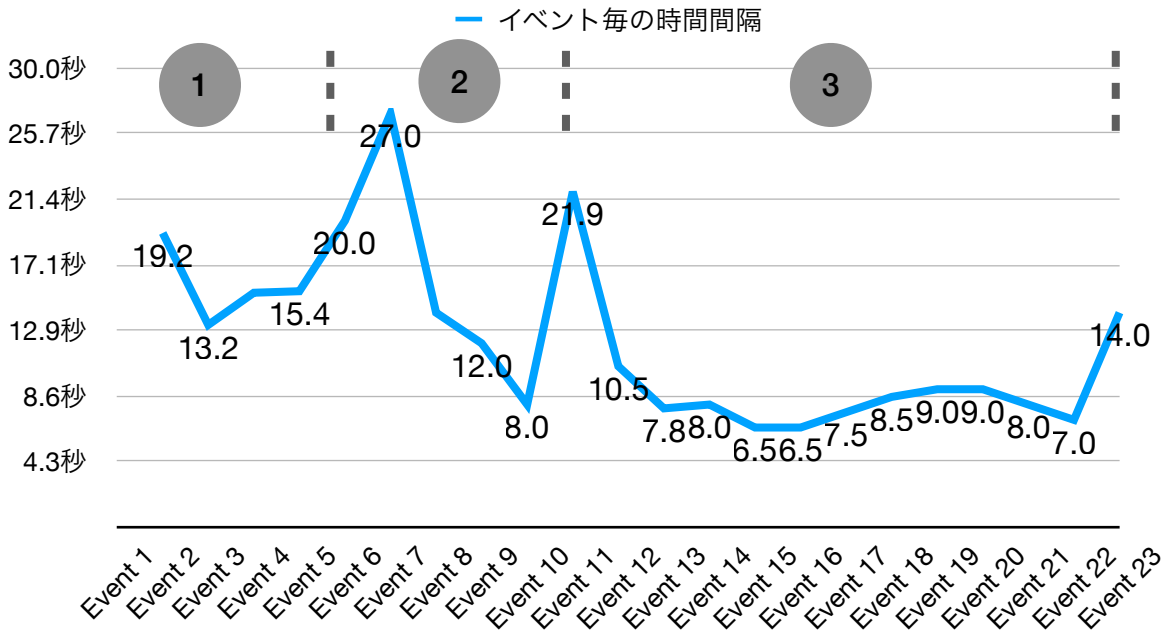
[53] Ghostシステムを使用した《Equal-G》の記録演奏映像は以下のリンクから視聴可能
https://youtu.be/5-nuUxzZuKE (accessed September 7, 2020).



「譜例4.2.4」《Equal-G》のGhostシステムを使用した演奏者パートの楽譜の一部分。



「図例4.2.5」Ghostシステムを用いた《Equal-G》演奏の様子。

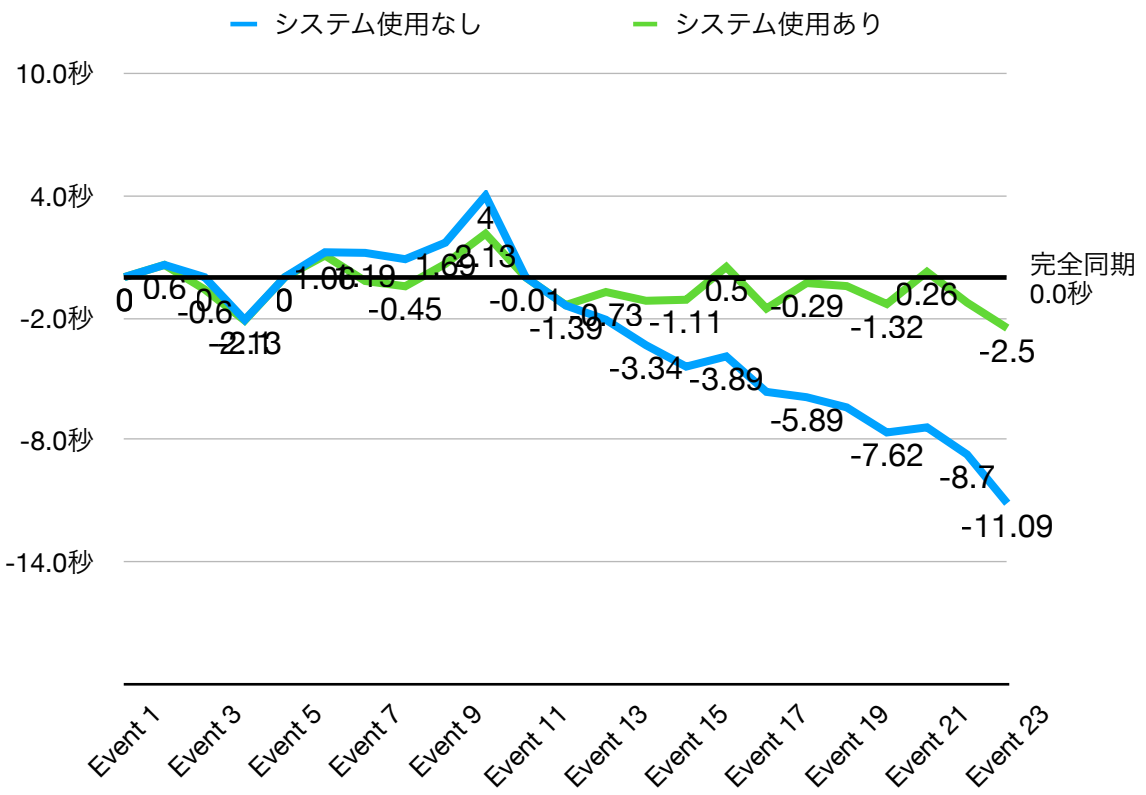


「図例4.2.6」
《Equal-G》のイベント毎の時間間隔の変動を示したグラフ。

なっている。これらの結果から、Ghostシステムの使用が非常に安定したライブ同期演奏に繋がったと言える。

次に、このライブ演奏とGhostシステムにおけるFixed Mediaの実際の再生時間との同期の正確さ（同期精度）について説明する。「図例4.2.6」は《Equal-G》の最初の３つのセクションにおける各イベントの時間間隔の関係を示している[54]。セクション1（ラベル１）は「イベント 1」から「イベント 4」まで、セクション2（ラベル２）は「イベント 5」から「イベント 10」まで、セクション3（ラベル３）は「イベント 11」から「イベント 23」までで構成されている。縦軸に示された数字は、そのイベントに対応する再生時間（秒）である。図からわかるように、最初の２つのセクションは、幅広い同期時間間隔を使用しているため、厳密な同期を必要とせず、より正確な時刻で次のセクションに入るための補助的な同期イベントとして設定されている。セクション３はFixed Mediaとの厳密な同期が必要なため、同期時間間隔を短くし、主に６秒前後の時間間隔を使用している。Ghostシステムの使用は、演奏者はスピーカーから鳴らされる電子音響を聴き、楽譜に記載されたテンポを参照しながら、演奏者自身が意図した時間的表現をもとに演奏することを前提とする。「図例4.2.7」は、ライブ本番での演奏における演奏者の演奏時間とFixed Mediaの再生時間の時間差を示すグラフである。青線はシステムを使用しない場合における、各イベントの演奏者の演奏時間とFixed

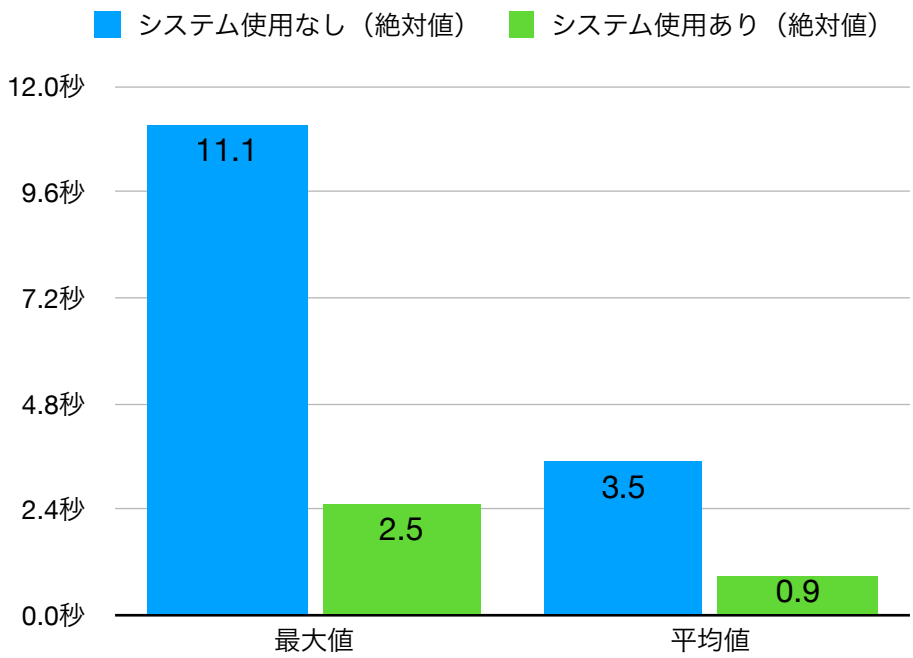
[54] 本作で同期が必要となるのは最初の３セクションのみ。



「図例4.2.7」演奏者の演奏時間とFixed Mediaの再生時間との時間差を示すグラフ。

Mediaの再生時間の時間差（秒単位）である。緑線は、実際のシステムを使用した場合の各イベントにおける演奏者の演奏時間と再生時間が伸縮されたFixed Mediaの再生時間との間の時間差を示している。太い水平の黒線は0の位置を示しており、数値は0に近いほど同期の精度が高いことを示している。この図から読み取れるように、イベント11までは、線と緑線の間の精度の差がきわめて小さいため、システムの効果を明確に確認することが困難である。イベント5からイベント11の間は、システムを導入していない場合と比較して、わずかながら精度が向上しているに過ぎない。これは、もともと演奏者の演奏時間が予定時間に対して高い精度で実現されていたことが一因であると考えられる。しかし、イベント11（セクション3）以降は、青線と緑線の間の同期精度の差が顕著になることがわかる。青線はイベントが進むにつれて完全同期である0秒から徐々に離れていくのに対し、緑線は最後まで完全同期に近い領域内で維持することができている。このことから、システムを使用することによって、使用しない場合と比べてFixed Mediaの再生時間と演奏とをより厳密な関係を維持しながら同期できたと言える。

次に、セクション3の中に行われるイベントの同期精度の具体的な値を比較する。「図例4.2.8」に、セクション3のイベントにおいて、システムを使用しない場合と使用した場合の



「図例4.2.8」全てのイベントに対する演奏者の演奏時間と
Fixed Mediaの再生時間との時間差の最大値と平均値を示したグラフ。

平均時間差と最大時間差を示している。青い棒はシステムを使用しない場合の数値を表す。緑の棒はシステムを使用した場合の数値を表す。この演奏では、演奏者が予定時刻より速く演奏しているパターンとなる。セクション３の最後のイベント、「イベント 23」に到達したとき、システムなしの場合は11.09秒（絶対値）であるのに対し、システムを使用することで2.5秒（絶対値）まで短縮することができる。また、平均時間差で見ても、システムなしの場合は3.5秒であるのに対して、システムを使用した場合は0.9秒までに短縮することができている。この楽曲の検証結果から、Ghostシステムを使用することによって、高い同期精度を引き出すことができたと考えられる。

《Equal-G》におけるGhostシステムの有効性について

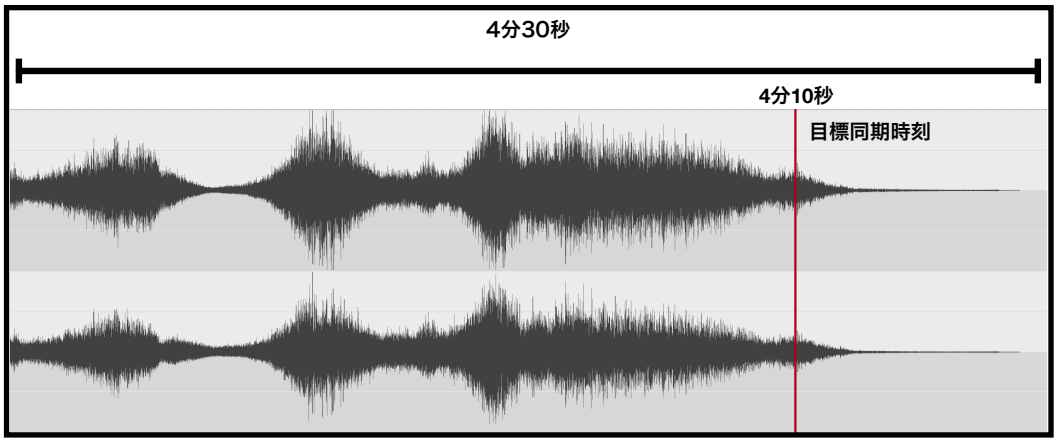
以上は、Ghostシステムを初めて実作品に導入した使用結果に関する考察である。コンサート会場では、約200人の聴衆がこのシステムを使ったライブ演奏を鑑賞した。聴衆の誰一人このシステムが使われていることを視覚・聴覚的に意識することなく、きわめて自然に再生されるFixed Mediaと、演奏者によるライブ演奏の有機的な融合という、これまでにない斬新な体験をすることができた。また演奏結果からは、このシステムが長時間のFixed Mediaに対して高い安定性をもって実行できることが確認できた。

4.3 《Tension》

今回取り上げる２作目、チューバ（F tuba）とエレクトロニクスのための《Tension》（2019）も筆者が作曲したものである。本作品はシステム開発後に作曲されたため、システムの利用を意識しながら試みている部分も含まれている。ここで取り上げる演奏データはシステムを検証するために、チューバ奏者の坂本光太氏が大学の演奏ホールで行った演奏を記録したものである[55]。

Ghostシステムの設定について

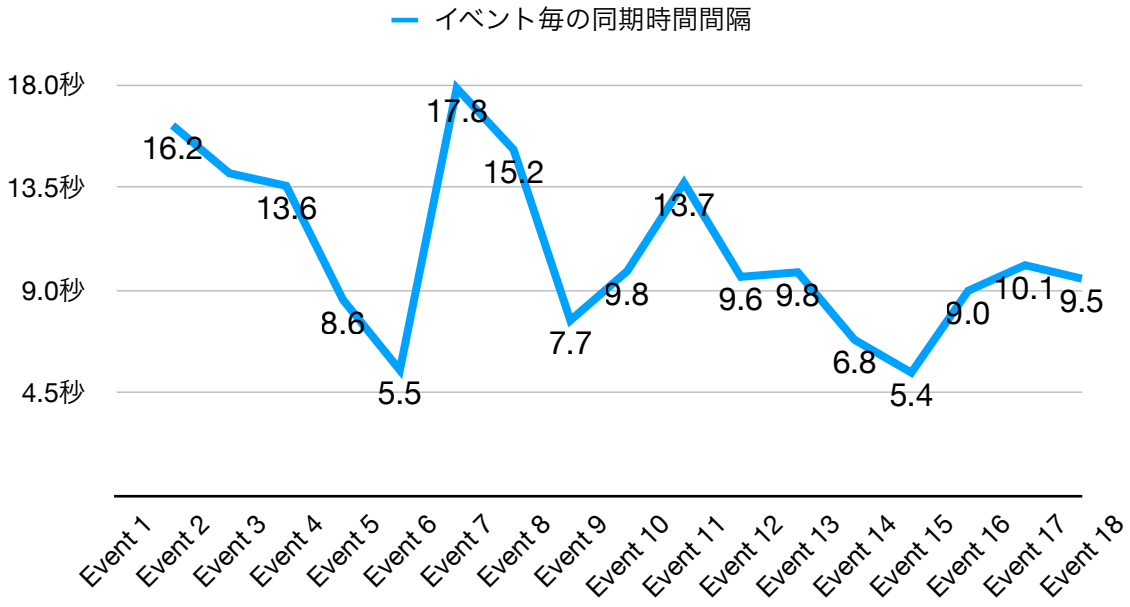
《Tension》も《Equal-G》と同様に曲の最初から最後までにFixed Mediaを再生し、演奏者とそのFixed Mediaとを様々な形で同期する必要がある。作曲上の関係で、全曲を１つのFixed Mediaのファイルを使用するのではなく、Fixed Mediaのファイルを断片化させ、同期が必要となる部分のみGhostシステムを使用する。「図例4.3.1」は《Tension》の中で同期演奏が必要となる一部のFixed Mediaの波形を示している。波形の形状から分かるように、この曲のFixed Mediaの変化は無秩序に進行し、演奏同期システムを使用しないかぎり、演奏者が確実に同期を図るのは不可能である。このファイルの総時間は４分30秒である。同期演奏の目標として、ファイルの再生が始まった箇所から４分10秒に達するまで、演奏者の時間とFixed Mediaの時間とをより高い精度で同期を実現することである。



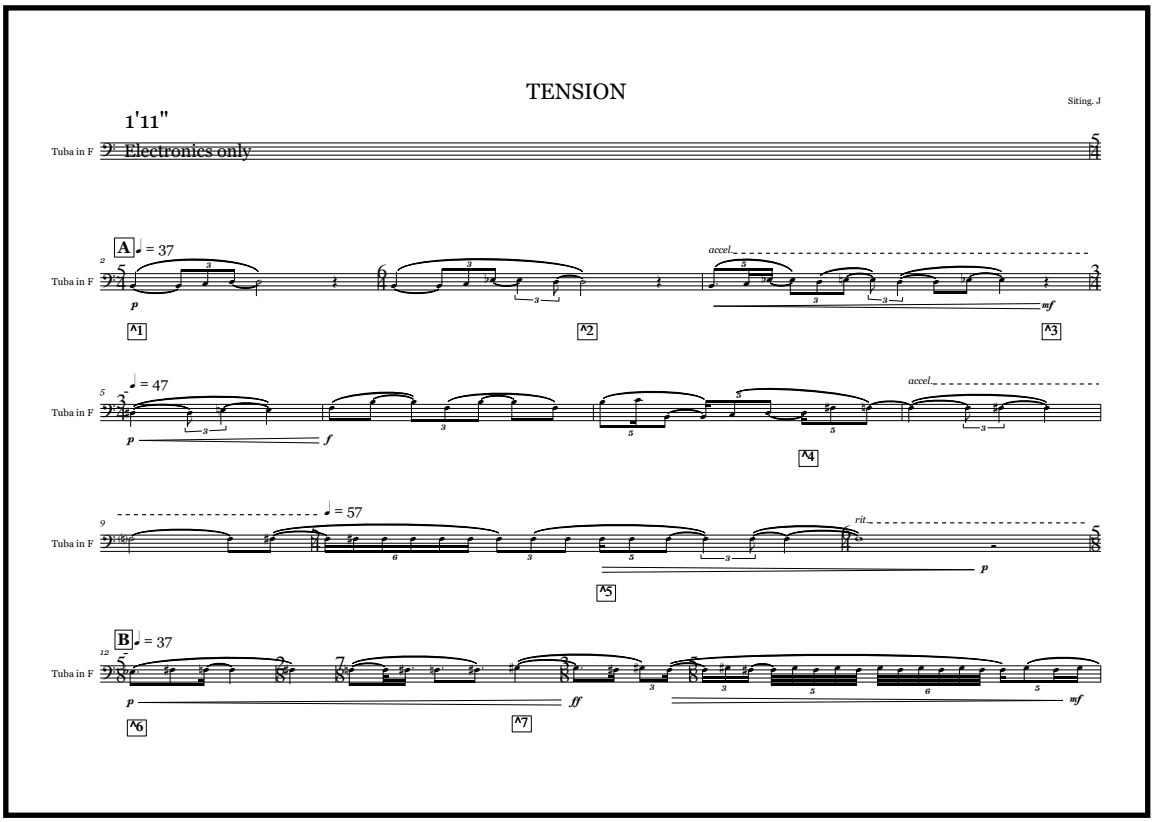
「図例4.3.1」《Tension》における同期演奏が必要な
Fixed Mediaの波形。

「図例4.3.2」は作品の中で設置される最初から18番までのイベントの時間間隔をグラフに表したものである。この作品はテンポが非常に遅いため、同期時間間隔の幅は《Equal-G》の幅よりもずっと長く、最小5.4秒から最大17.8秒までの数値になっている。ここで、システムの有効性をより明確に議論するために、同期が必要な最初の18イベントまでの演奏データを

[55] 検証演奏は2019年６月に行われた。



「図例4.3.2」《Tension》のイベント毎の時間間隔の変動を示したグラフ。



「図例4.3.3」《Tension》の演奏パートの楽譜の 1 ページ目。

取り上げることにする。

　このGhostシステムによって、作曲面に課される制約がなくなることを検証するため、《Tension》の演奏者の演奏パートの作曲には《Equal-G》よりも複雑な時間的表現を導入している。「譜例4.3.3」は、《Tension》の演奏パートの楽譜の最初のページである。演奏パートにおける時間的な表情を豊かにするために、演奏者は複雑なリズム表記だけでなく、曲中に頻繁に出てくるテンポの加速・減速の表記を読み取り、表現することが要求されている。通常、このような楽譜ではクリックトラックを正確に追うことは困難であり、演奏者が演奏時間の表現により集中できる環境を整える必要がある。今回は、このような複雑な時間的な表現を伴っていたとしても、GhostシステムによってFixed Mediaとライブ演奏を厳密に時間

「図例4.3.4」Ghostシステムを用いた
《Tension》演奏の様子（側面）。

「図例4.3.5」Ghostシステムを用いる《Tension》の
演奏の様子（背面から）。

同期させることができるかどうかを検証してみたい。

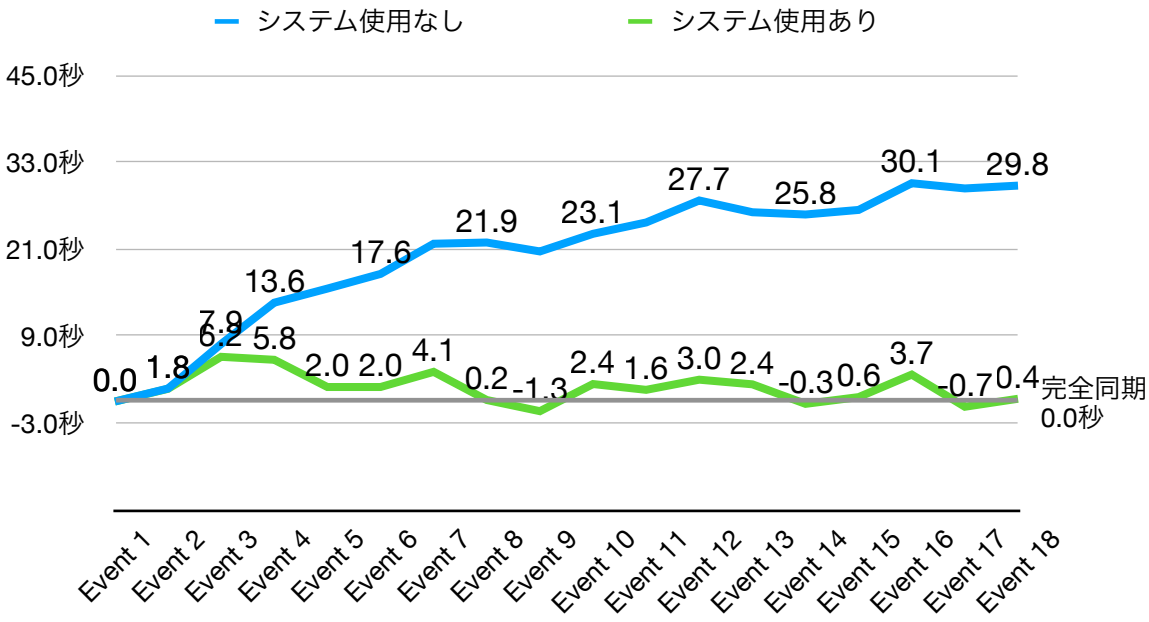
システムの使用の面について説明する。「図例4.3.4」は今回検証を行った際のセットアップの様子である。今回は、オペレーターがトリガーを送信する方式ではなく、演奏者が演奏中にトリガーを送信する方法を採用している。そのため、演奏者が楽譜を確認する譜面台のほかに、足元のペダルと、コンピュータが設置されている。コンピュータの画面には、ペダルを踏んだときに発生するイベントが映し出され、ペダルを踏んだ時点のイベントを確認できるようになっている。「図例4.3.5」は演奏者を後ろから撮影した写真であり、Ghostシステムによるライブ演奏の仕組みを演奏者の視点から写したものである。このように必ずしもオペレーターを設ける必要はなく、状況や、演奏者の意向によっては、演奏者本人がトリガリングすることも可能である。

Ghostシステムにおける同期精度の表現

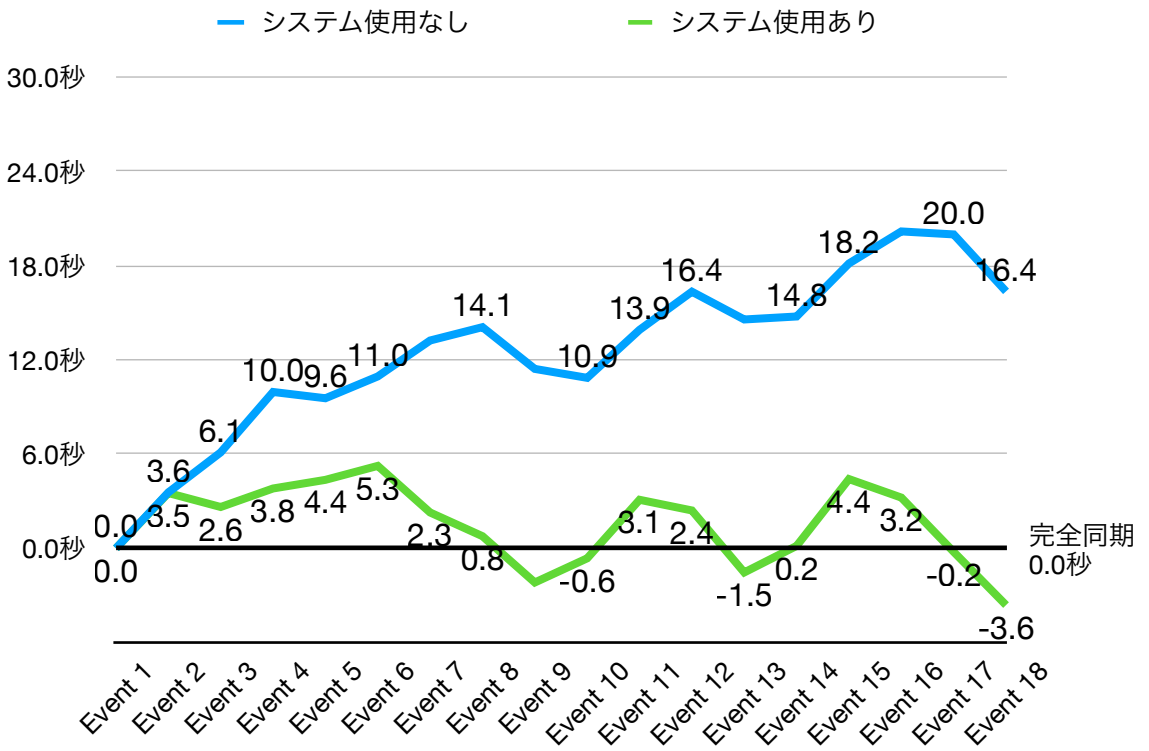
今回の演奏はシステムの検証が主な目的であるため、２度の演奏からデータを採取した。その際、演奏はすべて演奏者によるトリガリングによって行われたが、システム実行に関してはいかなるトラブルもなく、終えることができた。ここでは実際の演奏データから、本作におけるGhostシステムの同期性能について考察したい。「図例4.3.6」と「図例4.3.7」は、Ghostシステムを使用した場合と使用しなかった場合の、演奏者の演奏時間とFixed Mediaの再生時間との間の時間差を示すグラフである。「図例4.3.6」は１回目の演奏データを示し、「図例4.3.7」は２回目の演奏データを示している[56]。どちらの演奏結果からも演奏者の演奏時間はFixed Mediaの再生時間より遅れていることが分かる。それぞれのグラフで、青線（システム使用なし）がイベント２、３の位置から徐々に完全同期の線から離れて行くのに対して、緑線（システム使用時）は、完全同期に近い数値の範囲の中で収まっている。したがって、システムの使用により、使用しない場合に比べて、Fixed Mediaの時間との間に緊密な時間関係を構築できていることが示されたと言える。次に、具体的な数値について説明したい。１回目の演奏では、双方の誤差が0.2秒〜6.2秒（絶対値）の範囲の中で変動し、最終的には僅か0.4秒の誤差でイベント18に到達している。２回目の演奏では、双方の誤差が0.2秒〜5.3秒（絶対値）の範囲の中で変動し、最終的には3.6秒の時間差でイベント18へと到達した。すべてのイベントの誤差の平均値は、2.1秒（１回目の演奏）と2.5秒（２回目の演奏）となっている。全体的な誤差の平均値だけを見ると、《Tension》の演奏結果の方が《Equal-G》よりも高い数値となっており、精度が低まっていることがわかるが、これはGhostのイベント間の同期時間間隔が《Equal-G》よりも長いことが原因であると考えられる。しかし、どちらの作品にしても、作曲上の意図から見て、期待する同期演奏の完成度には到達できていると筆者は考える。

---

[56] ２回目の演奏の記録映像は、以下のリンクから視聴可能
https://youtu.be/ok5SVX_g_SE (accessed September 7, 2020).



「図例4.3.6」演奏者の演奏時間とFixed Mediaの再生時間の時間差を表すグラフ（１回目の演奏）。



「図例4.3.7」演奏者の演奏時間とFixed Mediaの再生時間の時間差を表すグラフ（２回目の演奏）。

《Tension》におけるGhostシステムの有効性について

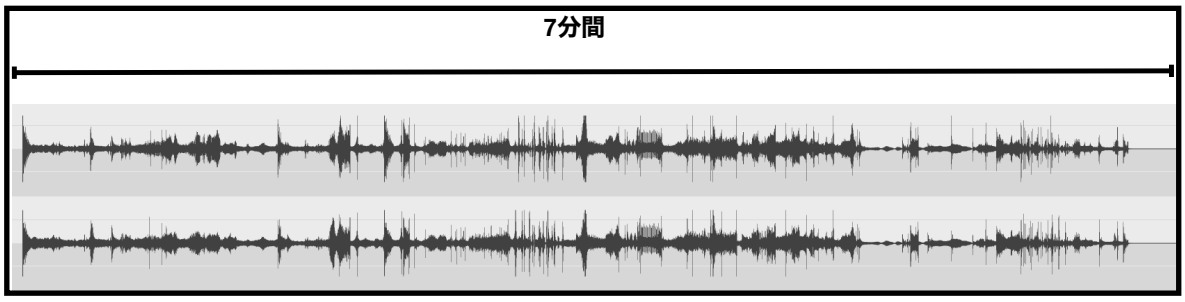
以上は、作品《Tension》におけるGhostシステムを使用した演奏結果に関する考察である。複雑な変化（テンポの変化を伴った時間変化）を伴った演奏パートの演奏であっても、問題なく使用することができたため、システムが、自由で複雑な楽曲であっても、自由に活用することができると言える。システムの同期精度では、使用する同期時間間隔に強く影響されることがわかったが、長時間のFixed Media作品を使用する場合に、演奏者の時間とのより厳密な時間関係を維持することは可能であると示されている。実際、検証演奏の後、演奏者の坂本氏から「このシステムなしには、この作品の同期演奏は不可能だった」というコメントをいただいた。演奏者の立場からも、システムの有用性の高さに確証が得られる結果となった。

4.4 《Danse d'atomes》

最後に検証する作品は、筆者の作品ではなく、日本の作曲家、北爪裕道氏によるソプラノまたはアルトのサックスとFixed Mediaのための《Danse d'atomes》（2015）である。北爪氏はこの研究のために、システムの有用性を検証に適した既成の自作品を提供してくれた。演奏者は、この作品に親しんでいるサックス奏者の本堂誠氏である[57]。

作品《Danse d'atomes》の演奏について

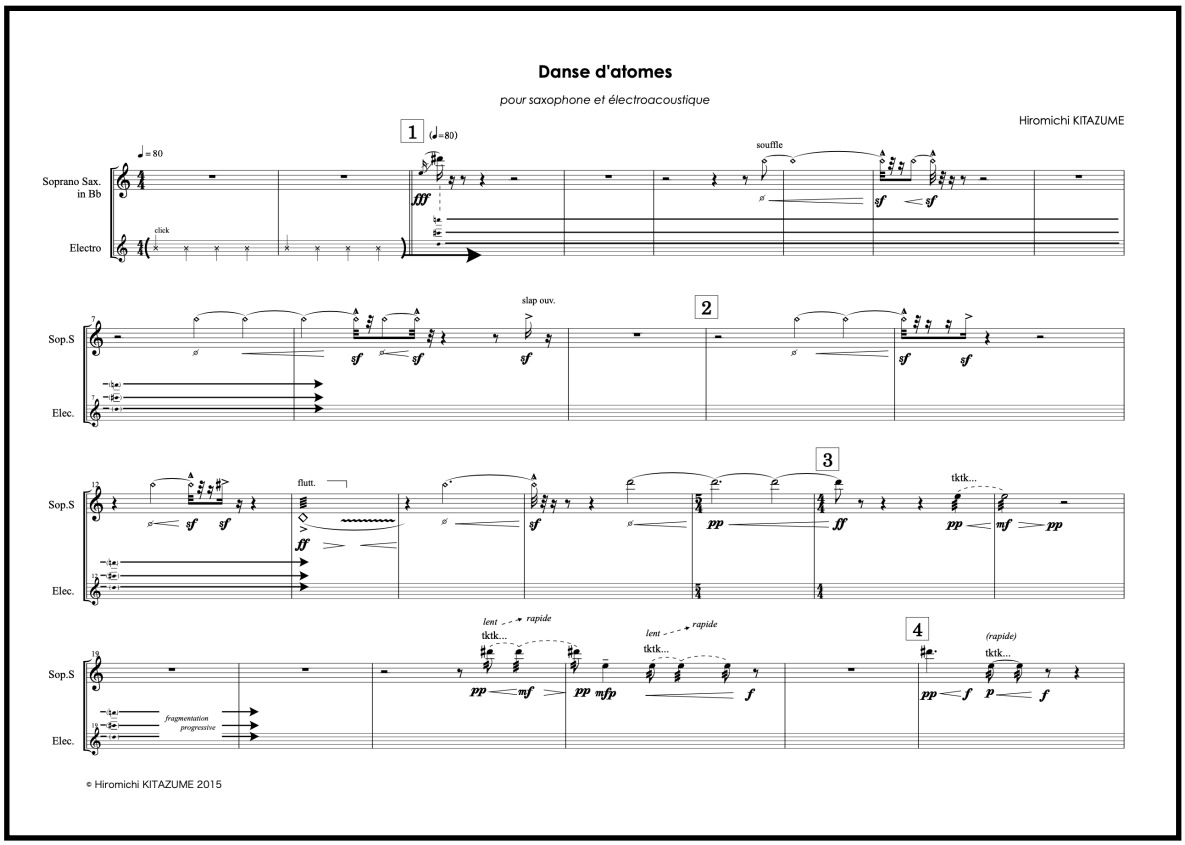
この作品は、特殊奏法を多く含むサックスの楽器演奏と、７分弱のFixed Mediaの再生を同時に行い、双方の時間同期（演奏と再生）に高い精度が要求される、複雑かつ表現力豊かな作品である[58]。「図例4.4.1」は、この作品で使用されているFixed Mediaの波形を示している。波形からわかるように、Fixed Mediaの音響には多くのアタック音が含まれており、これらのアタック音には、楽器の演奏パートにおける様々な箇所との厳密な同期が求められてい



「図例4.4.1」《Danse d'atomes》のFixed Mediaの波形。

[57] 検証演奏は2020年11月に行われた。

[58] 《Danse d'atomes》は以下のリンクから視聴可能。
https://soundcloud.com/hiromichikitazume/danse-datomes (accessed September 7, 2020).



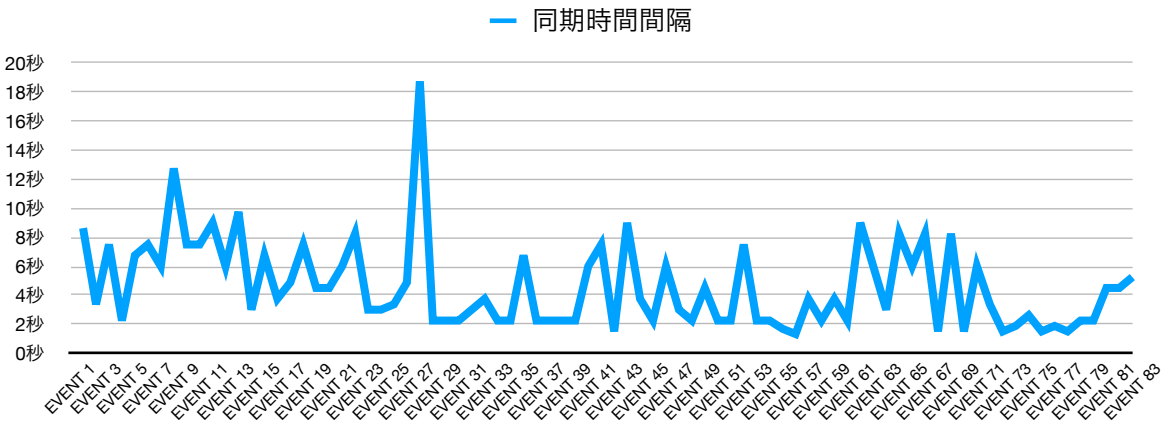
「譜例4.4.2」《Danse d'atomes》の演奏楽譜。

る。「譜例4.4.2」は演奏者用の楽譜を示しており、上段は演奏者が演奏する部分、下段はクリックトラックや電子音の変化の提示など、演奏者が電子音響の変化を参照できる記号が書かれている。この作品の過去のライブ演奏では、クリックトラックを使用することで非常に完成度の高い演奏が実現できたと聞いている。今回、クリックトラックの代わりにGhostシステムを用いることで、どのような影響と効果が表れるのかを検証する。

Ghostシステムの設定について

《Tension》同様、《Danse d'atomes》も、演奏者がMIDIペダルを踏むことでトリガリングする手法を採用した。「図例4.4.3」は、システムのセットアップの様子である。譜面台が演奏者の前に置かれ、MIDIペダルは演奏者の右下にセットされている。このように演奏者は自分のパートを演奏しながら、指定されたタイミングでペダルを踏み、Fixed Mediaと同時演奏を行っていく。作品の同期イベントの設定について説明する。この楽曲の場合、厳密な同期を求める作曲家の意図を鑑み、７分弱の演奏時間の中に83個ものイベントが設定されている。「図例4.4.4」は、各イベントの時間間隔の関係を示したものである。ほとんどの時間間隔が２秒〜６秒の範囲で設定されており、イベントの最小値は1.3秒で、平均値は4.6秒である。このことから、これまでGhostシステムに求められていた以上の同期精度を実現すること

「図例4.4.3」Ghostシステムを用いた
《Danse d’atomes》の演奏の様子。

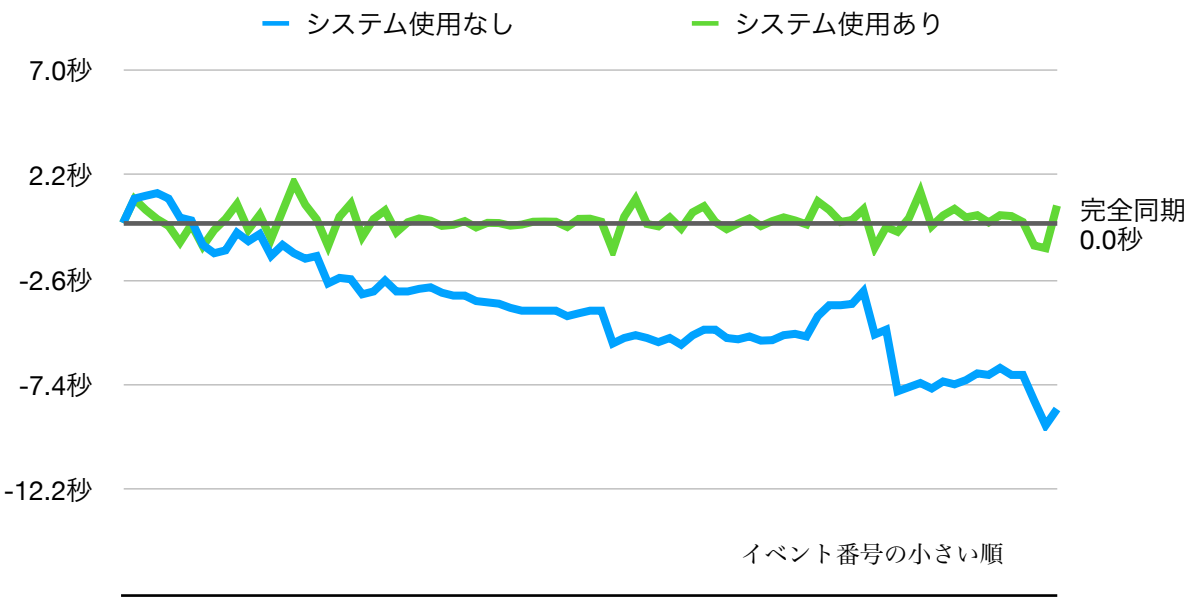


「図例4.4.4」《Danse d’atomes》の
イベント毎の時間間隔の変動を示したグラフ。

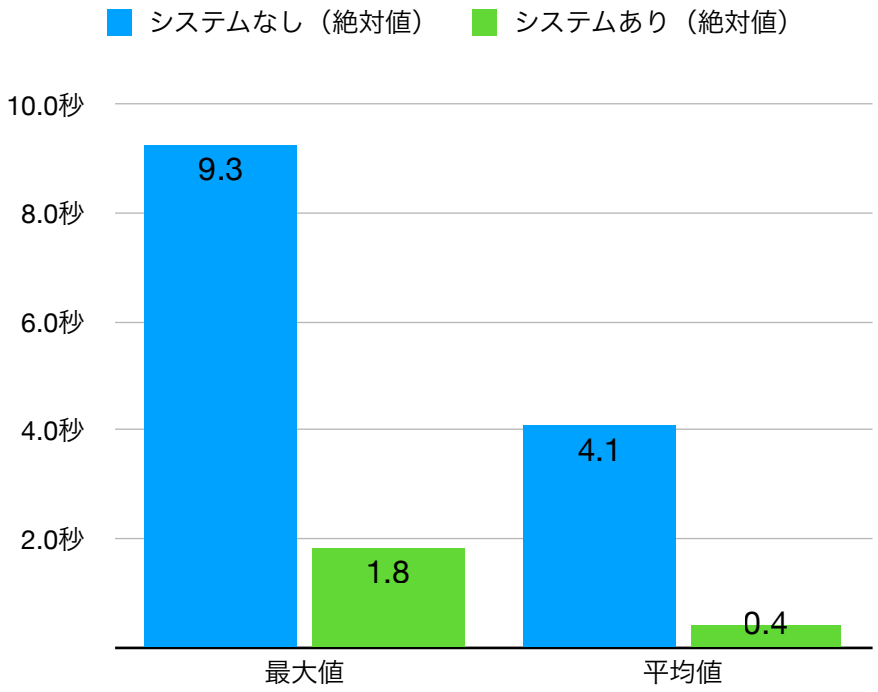
ができるかどうかが試されることになる。

Ghostシステムを用いた演奏とGhostシステムの検証結果

興味深いのは、演奏者はこのFixed Mediaをクリックトラックで再生したライブ演奏を何度も行っており、Fixed Mediaの音が変化するタイミングをすでに細かく把握している。実は、演奏者はクリックトラックを使わずに、ライブでFixed Mediaの音を聴きながらこの曲を演奏しても、同じように高い同期精度を得ることができたのである。一方、Ghostシステムを使ったライブ演奏では、演奏中に頻繁にトリガリングをかける必要があり、実際にシステムを使った演奏ができるようになるまでには、かなりの練習が必要であった。このような背景から、Ghostシステムを使用した同期演奏の目標として、演奏者をより自らの演奏時間の表現することに重点を置いている。システムの使用を慣れるために多くのリハーサルを行い、システムを使用して、完成度の高い同期演奏を実現することができた。「図例4.4.5」は、同期精度を表す、1テイクの図例である。今回の演奏は、演奏者の演奏時間が計画時間より早まったパターンとなっている。グラフから読み取れるように、青線（システム使用なし）の動きに対して、緑線は、完全同期を示す0秒に極めて近い領域の中に維持されている。このことから、システムを用いることで、演奏と再生とを極めて高い精度で同期させる事ができることが分かった。次に双方の時間差に関する具体的なデータについて分析する（「図例4.4.6」を参照）。システムを使用しない場合の最大誤差が9.3秒（最後のイベント）であるのに対して、システムを使用した場合の最大誤差は1.8秒であり、システムを使用することで誤差を大幅に短縮することができた。また、誤差の平均値については、システムを使用しなかった場合の4.1秒に



「図例4.4.5」演奏者の演奏時間とFixed Mediaの再生時間との時間差を表すグラフ。



「図例4.4.6」全てのイベントに対する演奏者の演奏時間とFixed Mediaの再生時間との時間差の最大値と平均値。

対して、使用することによって、0.4秒までに短縮することができている。このことから、システムを使用することで、両方の時間を高い精度のレベルの中で同期させることができたと言える。もちろん演奏者がシステムに慣れるまでに、ある程度の練習が必要だった点は配慮すべきであるが、システムを使用することで、演奏者が演奏表現の域を広げられると確信している。実際システムの検証演奏を行った後に、演奏者の本堂氏から「システムの使用を慣れるまでに時間を要したが、一度慣れてしまうと、システムが自分の演奏についてきてくれるため、自分の望むような演奏時間を表現することができた」とコメントした。作曲者の北爪氏にもシステムの有用性を高く評価していただき、今後、システムの利用を前提とした創作活動を展開していきたいと言及している。こうした評価からも、Ghostシステムが、ライブ中において、双方の時間を同期させるだけでなく、演奏者らの表現力を十二分に発揮させることにもつながると確信することができた。

4.5 まとめ

以上、Ghostシステムの有用性について考察する内容である。上記の作品から得られた演奏結果が、すべての作品に適用できるとは言い難いが、基本的には、Ghostシステムは安定性が高く、本研究の冒頭で提示した同期問題の大幅な改善・解決が期待できると筆者は考えている。もちろん、システムの設計面や使用する同期方法の性質上、従来のシステムに比べて実現

が難しい様々な「限界」もある。それらの限界を明確に知ることは、システムの効果的な性能を発揮するための重要なポイントにもなる。次章では、システムのメリットとデメリットを明確にした上で、今後の開発の展望を述べ、本論の総括としたい。

第５章 終章

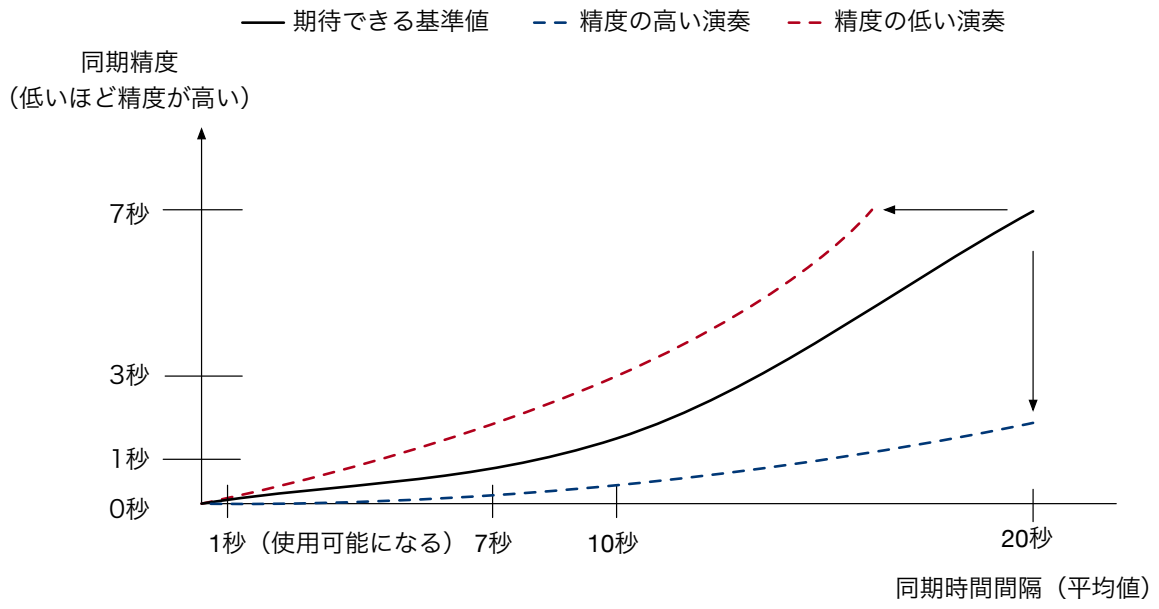
5.1 概要

本研究では、現在のタイムストレッチ技術の応用による、Mixed musicの演奏時に使用される演奏同期システムの開発について論じてきた。最終的には『Ghost』という、筆者自身が開発した新しいシステムを提案し、Mixed musicにおける同期の問題を解決するための新しい可能性のひとつを提示することができた。Ghostは、シンプルな仕組みと、既存の演奏同期システムとは異なるアプローチに基づいて開発されている（詳しくは第２、３章を参照）。いくつかの楽曲を通じて検証した演奏結果からは、GhostがMixed musicのライブ同期演奏において、これまで実現困難とされてきた自由で柔軟な演奏同期を実現する高い可能性があることがわかった。本章では、Ghostのメリットとデメリットを明確にし、課題と今後の展望について述べ、本研究の総括を行う。

5.2 メリット

Ghostシステムを用いてMixed musicのライブ同期演奏を行う場合、以下の効果を期待できる。

１）既存の演奏同期システムを使用することなく、比較的に長い時間（例えば３分以上）にわたってFixed Mediaと比較的に高い精度な同期演奏を達成できること。精度のレベルは「図例5.2.1」のように図式で表すことができる。



「図例5.2.1」Ghostにおける同期精度の可能性を示したグラフ。

横軸は使用した同期時間間隔の平均値を示している。縦軸は、演奏者の演奏時間とFixed Mediaの時間差、つまり同期精度を示している。縦軸の時間差の値が小さいほど、精度が高いことを意味する。赤い点線は、演奏時間の精度が低い演奏者が（計画時間に対して）演奏した場合に想定されるシステムの同期の精度である。青い点線は、演奏時間の精度の高い演奏者が（計画時間に対して）演奏した場合に想定されるシステムの同期の精度である。システムが実現できる同期精度は、演奏者の演奏時間の精度と、使用する同期時間間隔の長さに大きく影響される。図が示すように、同期時間間隔を１～７秒とした場合、Fixed Mediaとの同期の誤差を１秒程度に抑えることができる。同期時間間隔が７秒を超えると、同期精度が低くなり、おおよそ誤差が１～３秒程度になる。しかし、それ以上の時間幅の同期間隔（７秒以上20秒未満）になると、双方で厳密な同期を取ることは難しいが、より長いFixed Media作品（例えば３分以上）では、より厳密な時間関係を持った同期を実現することが可能となる。

２）幅ひろい自由な創作を行うことが可能。Ghostを使う場合、トリガーさえ送ることができれば、基本的に、演奏パートの作曲に対して制約が課されることがない。音響的な変化を中心とした楽器の特殊奏法で演奏したり、複雑なリズムの変化を利用したりしても、自由に作曲することが可能である。

３）Fixed Mediaによる音楽的な再生。システムを使用する際に自由に同期時間間隔を設定できるため、特にFixed Mediaの時間構成に合わせて設定するような場合においては、Fixed Mediaをより表現力豊かに、音楽的に再生することが可能である。

４）ビジュアル面での向上。Ghostを使用するのに必要なセットアップに関して、演奏者側必要なのは、譜面台や楽譜、マイクといった最低限のセットアップに限られる。オペレーター側のセットアップは必要だが、舞台袖など、客席から見えない位置に配置できるため、ステージ上が乱雑になる状況が避けられ、洗練された演出が可能になる。

５）リハーサルの効率の向上。Ghostシステムを使用する場合、演奏者はシステムの存在を意識する必要がなく、自分の演奏パートの解釈に集中することができる。これにより、演奏者はFixed Mediaの音との時間的相互関係を短時間で理解・把握することができ、リハーサル時間を大幅に短縮することが可能である。

5.3 デメリット

Ghostの設計の仕方やシステムの使い方によって、以下のような「限界」が生じる。

１）高い精度でのライブ同期の実現。Ghostが使用する同期時間間隔とその同期方法から、従来の演奏同期システムのキュー送信と比較して高い同期精度を得ることは困難である。例えば、クリックトラックやスクリーンスコアなどの技術を利用した場合では、テンポレベル以上の高精度な同期を実現することは可能であるが、Ghostではそのレベルでの同期を実現することは難しい。したがって、高い同期精度が求められる場合には、Ghostの使用は避けるべき

である。

2）Fixed Mediaの音質に対する高い要求をする再生。Ghostシステムは、Fixed Mediaの音をタイムストレッチさせることで、ライブ演奏の同期を図っている。使用されているタイムストレッチ再生エンジンは、Maxの「groove~」オブジェクトで、現在最も進んだタイムストレッチ再生エンジンといってもよい。この技術により、高品質かつ柔軟なオーディオデータをリアルタイムでタイムストレッチさせることが可能になった。しかし、Fixed Mediaの音響変化の形や、その再生時間（速度）の変化幅によって、再生音質に好ましくない影響が生じることがある。このような現象については、タイムストレッチ技術のさらなる進歩により、将来的に改善されることが期待される。現時点では、オリジナルと同程度の音質が求められる音楽作品に対して、Ghostを使用することは難しいと言わざるを得ない。

5.4 システムにおける既存の課題と開発の展望

Ghostシステムの課題、及び、今後の展望について次の２つの点が挙げられる。

１つ目の課題は、短い同期時間間隔（例えば３秒以下の時間領域）を頻繁に利用する場合、システムを正確に実行することが困難であることが挙げられる。例えば、４章で取り上げた北爪氏の作品《Danse d’atomes》でGhostを使用した同期演奏の際、トリガーを正確に送り、同時演奏を実現するまでに多くの努力と時間を費やしたのは事実である。そのため、双方の時間を高い精度で合わせる必要がある場合（短い同期時間間隔を使用して、高い頻度で同期させる必要がある場合）には、システムの使い勝手が悪くなってしまうことがある。この問題を改善するためには、トリガーの送信をある程度「予測」したり、自動的に「補正」したりできるような「自動化」されている仕組みを考案する必要がある。そのような仕組みの具体的な開発案は、今後さらに性能テストを行った上で開発する予定である。

２つ目の課題は、Fixed Mediaの再生速度（時間）を変化させるパターンが少ないことである。第３章の中では、Fixed Mediaの再生速度（時間）を変化させる２種類のパターンと、目標再生速度を正確に算出する計算式を明記している。この２つのパターンを用いることで、ほとんどのFixed Media作品において、音楽的に再生速度（時間）を変化させることが期待できる。しかし、Fixed Mediaのリズム形態によってさらに効果的な再生効果を生み出すためには、より多くのパターンを提供することが有効であると考えている。ただし、今回提供した変化のパターンよりも複雑な変化（曲線の変化など）を実現するためには、より複雑な計算式が必要となる。今後もこの分野の研究を続け、さらに発展させていきたいと考えている。

5.5 終わりに

Mixed musicの同期問題に関して、直接的に参考し得る先行研究が非常に少なかったため、筆者自身による創作物や演奏体験に基づくことで、この研究の成果であるGhostの構築を行った。実際に、このシステムを使った演奏結果から判断すると、GhostシステムはMixed

musicの同期問題を改善・解決するための有力な可能性を持っていることは間違いない。この論文の発表に先立ち、筆者はこのシステムの中身、およびその有用性について、この分野の第一人者である作曲家や・研究者たち、例えば、Barry Truax氏、Cort Lippe氏、Mikhail Malt氏、Jacopo Baboni Schilingi氏、今井慎太郎氏と議論してきた。これらの音楽家たちからは、このシステムの有用性について肯定的なフィードバックを受けることができた。例えば、Barry Truax氏は、「システムが作曲上に制約を与えず、器楽のライブ演奏だけでなく、器楽を含まない（例えば演劇）ライブ演奏の同期にも応用し得る」とコメントしている。Cort Lippe氏では、「演奏者の自由な演奏時間の表現は、演奏者の表現力の最も重要な部分であり、Ghostシステムはこの部分に対する新しい可能性を与えてくれるだろう」と述べている。Mikhail Malt氏は、「Ghostは、Mixed musicの同期問題をタイムストレッチ技術によって直接的に解決するものである」と評価した。Jacopo Baboni Schilingi氏と今井慎太郎氏からは、Ghostを今後自らの創作の中に応用することも考えているという話が上がっている。これらのことから、Ghostが、今後Mixed musicの分野において、多くの音楽家のあいだで広がっていくと筆者は確信している。Ghostが従来のシステムの代用として使えるか、という問いに対しては、筆者の答えは「ノー」である。例えば演奏者が自由に演奏しながらも、オーディオ伴奏音源と厳密な同期（テンポレベル）を実現したい場合、Antescofoに代表されるScore Following技術が現段階で最も効果的な手段であると考えられる。あるいは作曲者の意図したような演奏時間を正確に表現したい場合、クリックトラックやスクリーンスコアなどのキューイング的手法が最も効果的である。断片化することが可能なFixed Mediaを用いて同期演奏する場合、Fixed Mediaがオリジナル音質のまま再生できるという利点があるため、マルチトラック再生システムが現段階では最も適切な手法である。しかし、演奏者の演奏パートやFixed Mediaの作曲に対し、なんの制約も課されることなく、自由かつ柔軟なライブ同期演奏を行いたい場合、筆者は、Ghostこそが最も相応しい手法であると考えている。Ghostは単なる同期問題の解決・改善するばかりでなく、演奏の機会の増加や、Mixed musicの作曲の可能性を押し広げ、ひいてはMixed musicの発展に繋がっていくと筆者は考えている。今後、Ghostの開発が、Mixed musicの歴史において、新たな演奏同期システムの有力な候補になることを切に願っている。

謝辞

本論の執筆にあたり、多くのご支援を賜りました。終始熱心に論文指導を頂きました三浦雅展先生、久保田慶一先生、Cathy L. Cox先生、そしてシステムの開発や検証実験に際して、多くの貴重なご意見およびご指導くださった今井慎太郎先生に深く感謝いたします。

また、開発したシステムの検証にあたっては、Ieva Sruogyté氏、悪原至氏、渡邉玲子氏、坂本光太氏、北爪裕道氏、本堂誠氏から素晴らしい演奏のご協力を賜りました。心からの謝辞を申し上げます。ありがとうございました。

参考文献

日本語文献

コープ, デイヴィッド．2011．『現代音楽キーワード事典』石田一志, 三橋圭介, 瀬尾史穂訳. 東京：春秋社.

シュトックハウゼン, カールハインツ．1999．『シュトックハウゼン音楽論集』清水穣訳. 東京：現代思潮社.

ノイマンピアノ（赤松正行, 左近田展康）．2006.『2061：Max オデッセイ 音楽と映像をダイナミックに創造する！最高の開発環境を徹底解説』東京：リットーミュージック.

ノイマンピアノ（赤松正行, 左近田展康）．2009.『Maxの教科書』東京：リットーミュージック.

ボスール, ジャン＝イブ. 2008.『現代音楽を読み解く８８のキーワード 12音技法からミクスと作品まで』栗原詩子訳．東京：音楽之友社.

松平敬．2019．『シュトックハウゼンの全て』東京：アルテスパブリッシング.

ローズ, カーティス. 2001.『コンピュータ音楽歴史・テクノロジー・アート』青柳龍也, 小坂直敏, 平田圭二, 堀内靖雄訳・研修．東京電機大学出版局.

英語文献

Ableton AG. n.d. “The latest from Ableton.” Accessed September 7, 2020. https://www.ableton.com.

——. 2015. “Introducing BeatSeeker by Andrew Robertson.” Press, August 27, 2015. Accessed September 7, 2020. https://www.ableton.com/en/press/press-archive/introducing-beatseeker-andrew-robertson/.

Baboni-Schilingi, Jacopo. n.d. “Music for concerts.” Accessed September 7, 2020. http://www.baboni-schilingi.com/index.php/music/music-for-concerts.

Barrett, Natasha. 1997. “Structuring processes in electroacoustic composition.” PhD diss., City University London. Accessed September 7, 2020. https://openaccess.city.ac.uk/id/eprint/7468/.

Bell, Jonathan. 2016. “Audio-score, a resource for composition and computer-aided performance.” PhD diss., Guildhall School of Music and Drama. Accessed September 7, 2020. https://openaccess.city.ac.uk/17285/.

Bernsee, Stephan (Neuronaut). 1999. “Time Stretching And Pitch Shifting of Audio Signals - An Overview.” Stephen Bernsee’s Blog: Gedanken About Digital Signal Processing, August 18, 1999. Accessed September 7, 2020. http://blogs.zynaptiq.com/bernsee/time-pitch-overview/.

Berio, Luciano. 2007. *Différences : per cinque strumenti e registrazione stereofonica : 1958-1959.* Vienna: Universal Edition UE31387.

Bowers, Roger. n.d. “Proportional notation.” *Grove Music Online. Oxford Music Online.* Accessed September 7, 2020. https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000022424?rskey=eHNPGq&result=1.

Boulez, Pierre. 2002. *Répons : für 6 Solisten, Ensemble, Computerklänge und Live-Elektronik (1981)*. London: Universal, UE17487.

Brown, Andrew R, and Toby Gifford. 2013. “Prediction and Proactivity in Real-Time Interactive Music Systems.” In *Musical Metacreation: Papers from the 2013 AIIDE Workshop*, edited by Phillippe Pasquier et al. AAAI Workshop - Technical Report (WS-13-22), 35-39.

Chadabe, Joel. 1997. *Electric Sound: The Past and Promise of Electronic Music.* New Jersey: Prentice Hall.

Collins, Nick, Margaret Schedel, and Scott Wilson. 2013. *Electronic Music*. Cambridge Introductions to Music. Cambridge: Cambridge University Press.

Cont, Arshia. 2010. “A coupled duration-focused architecture for realtime music to score alignment.” *IEEE Transaction on Pattern Analysis and Machine Intelligence* 32: 974-987.

——. 2008. “ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music.” *Proceedings of the International Computer Music Conference (ICMC 2008)*, 33-40.

Cont, Arshia, José Echeveste, Jean-Louis Giavitto and Florent Jacquemard. 2012. “Correct Automatic Accompaniment Despite Machine Listening or Human Errors in Antescofo.” *Proceedings of the International Computer Music Conference (ICMC 2012)*, 194-199.

Cycling ’74. n.d. “The lastest from Cycling ’74.” Accessed September 7, 2020. https://cycling74.com.

Dannenberg, Roger B. 1984. “An On-line Algorithm for Real-Time Accompaniment.” *Proceedings of*

the International Computer Music Conference (ICMC 1984), 193-198.

Davidovsky, Mario. 1963. *Synchronisms No.1: for flute and electronic sound*. New York: McGinnis & Marx Music Publishers.

——. 1964. *Synchronisms No.3: for cello and electronic sound*. New York: McGinnis & Marx Music Publishers.

——. 1966. *Three Synchronisms*. Composers Recordings Inc., CRI: SD204.

Dannenberg, Roger B. and Christopher Raphael. 2006. "Music Score Alignment and Computer Accompaniment." *Communications of the Association for Computing Machinery* 49 (August 2006). http://doi.acm.org/10.1145/1145287.1145311.

Deutsch, Diana. n.d. "Psychology of music." *Grove Music Online. Oxford Music Online.* Accessed September 7, 2020. https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000042574.

Ding, Shiau-uen. 2006. "Developing a Rhythmic Performance Practice in Music for Piano and Tape." *Organised Sound* 11, no. 3: 255–72.

Doornbusch, Paul. 2011. Early Hardware and Early Ideas in Computer Music: Their Development and Their Current Forms. In Roger T. Dean (ed.) *The Oxford Handbook of Computer Music*, New York: Oxford University Press: 44-84.

Emmerson, Simon. 1994. "'Live'versus 'real-time'." *Contemporary Music Review* 10, no. 2: 95-101.

——. 2011. Combining the acoustic and the digital: music for instruments and computer or pre-recorded sound. In Roger T. Dean (ed.) *The Oxford Handbook of Computer Music*, New York: Oxford University Press: 167-190.

Garnett, Guy E. 2001. "The Aesthetics of Interactive Computer Music." *Computer Music Journal* 25, no. 1: 21–33.

Ghent, Emmanuel. 1967. "Programmed Signals to Performers: A New Compositional Resource." *Perspectives of New Music* 6, no. 1: 96-106.

Giavitto, Jean-Louis, Arshia Cont, José Echeveste and MuTAnt Team Members. 2016. "Antescofo: A not-so-short introduction to version 0.x." Ircam Support. Accessed September 7, 2020. http://support.ircam.fr/docs/Antescofo/AntescofoReference.pdf.

Hagan, Kerry L. 2016. “The Intersection of ‘Live’ and ‘Real-Time.’” *Organised Sound* 21, no. 2: 138–46.

London, Justin. n.d. “Rhythm.” *Grove Music Online. Oxford Music Online.* Accessed September 7, 2020. https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000045963.

Keislar, Douglas. 2011. “A Historical View of Computer Music Technology.” In *The Oxford Handbook of Computer Music* edited by Roger T. Dean. New York: Oxford University Press, 11-43.

Kimura, Mari. 2003. “Creative Process and Performance Practice of Interactive Computer Music: a Performer's Tale.” *Organised Sound* 8, no. 3: 289–96.

Lippe, Cort. 1994. “Real-time Granular Sampling Using the IRCAM Signal Processing Workstation.” *Contemporary Music Review* 10, no.2: 149-155.

——. 1996a. “Real-Time Interactive Digital Signal Processing: A View of Computer Music.” *Computer Music Journal* 20, no. 4: 21–24.

——. 1996b. “A Look at Performer/Machine Interaction Using Real-time Systems.” *In Proceedings of the International Computer Music Conference (ICMC1996)*, 116-117.

——. 2002. “Real-Time Interaction Among Composers, Performers, and Computer Systems.” Proceedings, Information Processing Society of Japan SIG Notes.

——. 2020. n.d. “Compositions.” https://www.cortlippe.com/compositions.html.

London, Justin. 2004. *Hearing in time: psychological aspects of musical meter.* Oxford; New York: Oxford University Press.

Loy, Gareth. 2011. *Musimathics*, Volume 1: The Mathematical Foundations of Music. Cambridge, MA: The MIT Press.

Maderna, Bruno. 1960. *Musica su due dimensioni*. Scores. Milano: Edizioni Suvini Zerboni, S.5573Z.

Manning, Peter. 2013. *Electronics and Computer Music*. Oxford: Oxford University Press.

McNutt, Elizabeth. 2003. “Performing Electroacoustic Music: a Wider View of Interactivity.” *Organised Sound* 8, no. 3: 297–304.

Nono, Luigi. 1964. *La Fabbrica Illuminata*. Milan: Ricordi, 2007, NR13973800.

Orio, Nicola and François Déchelle. 2001. “Score Following Using Spectral Analysis and Hidden Markov Models.” *in Proceedings of the International Computer Music Conference (ICMC 2001)*. Accessed September 7, 2020. http://hdl.handle.net/2027/spo.bbp2372.2001.038.

Puckette, Miller. 1991. “Combining event and signal processing in the max graphical programming environment.” *Computer Music Journal* 15, no. 3: 68-77.

——. 2002. “Max at Seventeen.” *Computer Music Journal* 26, no. 4: 31-43.

Puckette, Miller and Cort Lippe. 1992. “Score Following in Practice.” i*n Proceedings of the International Computer Music Conference (ICMC 1992)*, 182-185.

Puckette, Miller and Zack Settel. 1993. “Nonobvious roles for electronic in performance enhancement.” *in Proceedings of the International Computer Music Conference (ICMC 1993)*, 134-137.

Raphael, Christopher. n.d. “Christopher Raphael's Music Plus One.” Accessed September 7, 2020. https://music.informatics.indiana.edu/~craphael/music_plus_one/.

——. 1999. “Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models.” IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(4): 360-370.

——. 2003. “Orchestral Musical Accompaniment from Synthesized Audio.” *Proceedings of the International Computer Music Conference (ICMC 2003)*. Accessed September 7, 2020. http://hdl.handle.net/2027/spo.bbp2372.2003.029.

Raphael, Christopher and Gu Yupeng. 2009. “Orchestral Accompaniment for a Reproducing Piano.” *Proceedings of the International Computer Music Conference (ICMC 2009)*, 501-504. Accessed September 7, 2020. http://hdl.handle.net/2027/spo.bbp2372.2009.113.

Risset, Jean-Claude. 1996. *Passages.* Scores. Milan: Ricordi, SLB 00216400.

——. 1999. “Composing in real-time?” *Contemporary Music Review* 18(3): 31-39.

Roads, Curtis. 2001. *Microsound*. Cambridge, MA: The MIT Press.

——. 2014. “Rhythmic Processes in Electronics Music.” *Proceedings of the International Computer Music Conference (ICMC 2014),* 27-31. Accessed September 7, 2020. http://hdl.handle.net/2027/spo.bbp2372.2014.005.

——. 2016. *Composing Electronic Music: A New Aesthetic*. New York: Oxford University Press.

Robertson, Andrew and Mark Plumbley. 2006. “Real-time Interactive Music Systems: An Overview.” *Proceeding of Digital Music Research Network Doctoral Researchers Conference (DMRN-06)*.

——. 2007. “B-Keeper: A Beat-Tracker for Live Performance.” *Proceedings of the International Conference on New Interfaces for Musical Expression*, 234–237. Accessed September 7, 2020. http://doi.org/10.5281/zenodo.1177231.

Rowe, Robert. 1993. *Interactive music system: machine listening and composing*. Cambridge, Mass: MIT Press.

——. 1999. “The Aesthetics of Interactive Music Systems.” *Contemporary Music Review* 18, no. 3: 83-87.

Saariaho, Kaija. 1994. *Six Japanese Gardens*. Brighton: Chester Music Ltd.

——. n.d. “Works.” Accessed September 7, 2020. https://saariaho.org/works/.

Schwarz, Diemo, Arshia Cont, and Nicola Orio. 2006. “Score Following at Ircam.” *7th International Conference on Music Information Retrieval (ISMIR)*. http://articles.ircam.fr/textes/Schwarz06d/index.pdf.

Smalley, Denis. 1997. “Spectromorphology: Explaining Sound-Shapes.” *Organised Sound* 2, no. 2: 107–26.

Smith, Brandon. n.d. “Ableton.” *Grove Music Online. Oxford Music Online.* Accessed September 7, 2020. https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-4002261064.

Stockhausen, Karlheinz. 1966. *Kontakte*. Performance score. Vienna: Universal Edition UE14246.

——. 1991. *Elektronische Musik 1952-1960*. Stockhausen Verlag: CD 3.

——. 1993. *Kontakte*. CD. Stockhausen Verlag: CD 6.

Stockhausen, Karlheinz and Elaine Barkin. 1962. “The concept of unity in electronic music.” *Perspectives of New Music* 1(1): 39-48.

Stockhausen, Karlheinz and Jerome Kohl. 1996. “Electroacoustic Performance Practice.” *Perspectives of New Music* 34(1): 74-105.

Stroppa, Marco. 1999. “Live Electronics or … Live Music?” Towards a Critique of Interaction. *Contemporary Music Review* 18(3): 41-77.

Toiviainen, Petri. 2007. The psychology of electronic music. In *The Cambridge Companion to Electronic Music* edited by Nick Collins & Julio d'Escrivan. Cambridge University Press:

218-231.

Truax, Barry. 1994. “Discovering Inner Complexity: Time Shifting and Transposition with a Real-Time Granulation Technique.” *Computer Music Journal* 18, no. 2: 38–48.

Tutschku, Hans. n.d. “instrumental works (with and without electronics).” Accessed September 7, 2020. https://tutschku.com/works/instrumental/.

——. 2016. “Combining Performer with Soundtracks: Some Personal Experiences.” *Ideas Sonicas/ Sonic Ideas* 17, 10-18. Accessed September 7, 2020. https://en.cmmas.com/ideassonicas-16/ideassonicas%2Fsonicideas-23.

——. n.d. “Recording & Fixed Media.” Accessed September 7, 2020. https://www.sfu.ca/~truax/EMS14/FixedMedia.html.

Varèse, Edgard. 1959. *Déserts*. New York: G. Ricordi.

Vercoe, Barry. 1984. “The Synthetic Performer in the Context of Live Performance.” *Proceedings of the International Computer Music Conference (ICMC 1984),* 199-200. Accessed September 7, 2020. http://hdl.handle.net/2027/spo.bbp2372.1984.026.

Winkler, Todd. 1998. *Composing Interactive Music - Techniques and Ideas Using Max*, Cambridge: MIT Press, 1998.

Wishart, Trevor. 1992. *Red Bird / Anticredos*. CD. UK: October Music - Oct001.

——. 1994. *Audible Design*. York, UK: Orpheus the Pantomime. https://www.trevorwishart.co.uk/AUDIBLE_DESIGN.pdf.