

# リズムを考慮した旋律分析のための NumPy 配列による楽譜表現

山谷 歌穂 202003536

## 1 はじめに

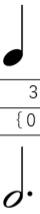
音楽において、リズムとは、一定の拍子の中で音の長さのパターンを形成することである。ここで拍子とは、一定間隔の拍の連なりのことである。例えば4/4拍子の場合、4分音符4つ分の長さで1小節を形成することを示している。このように、拍子とは音楽を形成する心臓のようなものであり、リズムはその上にたって楽曲に個性を与えるものである。さらに、リズムを持った音の高さの連なりを旋律、またはメロディーと呼び、旋律は音楽の中で最も重要な要素である。楽譜や音響情報から旋律を見きわめることは「旋律分析」と呼ばれ、音楽分析の中で重要な作業の一つである。ここで、音楽分析の入力は一般的に、楽譜または音響信号であるが、本研究では楽譜を入力する場合を扱う。楽譜のデータ表現として主要なものに、MIDIとMusicXMLがある。MIDIによる楽譜表現では、音の高さが数値表現されているので、計算機で扱いやすく、先行研究においてもMIDIによる楽譜表現が、用いられている[1]。しかしMIDIでは、楽譜に含まれているすべての情報を表すことができない。特に、MIDIでは音の長さを開始時刻と終了時刻によって表すため、これは、必ずしも楽譜に書かれた正確なリズム情報ではない。例えば、タイなどの演奏記号情報が消えてしまったり、演奏表現によっては、書かれた音価との差異が生まれたりする。一方MusicXMLは、楽譜に書かれたすべての情報を、表現可能である。しかし、XML形式のデータであるため、例えばある時刻に同時になっている情報を取り出したい場合は、すべての楽器及びパートに対応する時刻を含むXMLタグを、見つけ出さなくてはならない。そのためMIDIに比べて検索に手間がかかってしまう。そこで本研究では、MusicXMLを配列形式に変換した新たなデータ表現を提案し、MusicXML形式からの正確なデータ表現の活用と、MIDI形式の利便性を両立する。このデータ表現は、将来的にMIDI形式の楽譜よりも正確な情報に基づく分析などに応用できることを期待する。

## 2 設計

本研究では、MusicXML形式の楽譜データから必要なデータを取り出し、配列形式の表現に変換する。配列の1マスは、対象とする楽譜の最小音価である。例として、図1の最小音価が8分音符の楽譜の、赤い矢印がさしている5小節目の2個目の4分音符と、青い矢印がさしている6小節目の最初の付点2分音符の場合を考える。図2の表の下段の1マスは、Numpy配列の要素の1つとする。上の4分音符は8分音符2個分の長さなので2マス、下の付点2分音符は8分音符6個分の長さなので6マス必要である。さらに、リズム情報を配列の中に表現するために、音符の開始時刻と終了時刻を、復元可能な2次元のインデックス表現  $\{x,y\}$  を作る。 $\{0,x\}$  は音符の開始地点を、 $\{-x,0\}$  は音符の終了地点を示している。この時の  $x$  は音符によって数が変わる。4分音符の場合は1になり付点2分音符は5となる。また、開始地点と終了地点以外では、その地点のデータが音符のどの位置であるかを表すために、 $\{ \}$  の左側は開始地点からの距離、右側は終了地点からの距離を示している。例の付点2分音符では、2マス目の8分音符単位のデータは、開始地点から8分音符2個分、終了地点から3個分の位置にあることを示している。上段は時間表現のインデックスなので、図1の赤い矢印の4分音符は、楽譜の最初から数えて8分音符単位で34~35、付点2分音符は40~45となる。



図1 図2の4分音符と付点2分音符の楽譜上での位置



3 4	3 5
{0,1}	{-1,0}

4 0	4 1	4 2	4 2	4 4	4 5
{0,5}	{-1,4}	{-2,3}	{-3,2}	{-4,1}	{-5,0}

図2 最小音価が8分音符の時の4分音符と付点2分音符の例

して、楽譜上の音符の開始位置が、`measure.offset` に格納されている。`measure.offset` を `min_duration` をで割ることで、その音符配列表現上でのデータ挿入開始時を指定する。これを `mo` とする。`note.offset` は4分音符の初めを 0.0 とするものである。これを `min_duration` で割ったものを `no` とする。`ln` は、音符が最小音価で何個分の長さがあるかを計算するものである。しかしこのままでは、MIDI 番号の情報しか格納されないで、配列に挿入すべき必要な情報をまとめるためのクラス `Onpu` を定義し、下記のコードを `ln` の下に挿入する。

### 3 実装

#### 3.1 music21

`music21[2]` は、Python の音楽情報処理ライブラリの一つで楽譜の出力や編集、楽譜や音声の解析などの多くの処理をすることができる。本研究では、XML 形式で書かれたデータから必要な表現を取り出すのに便利な機能を備えたライブラリとして、`Music21` ライブラリを使用している。

#### 3.2 Numpy 配列

`Numpy` は Python の拡張モジュールである。これを使用することによって、数値計算をより早く、効率的に行うことができる。本研究では処理した情報を入れる場所として `Numpy` 配列を使用している。また、使用している楽譜によるが、`Numpy` は特定の範囲のデータを一括して出したり、代入したりできるスライシングなど、多次元配列表現を扱うのに便利な機能を備えている。

#### 3.3 コード

```
mo=int(measure.offset/min_duration)
for note in measure.notes:
    no=int(note.offset/min_duration)
    if isinstance(note,music21.note.Note):
        ln= int(note.duration.quarterLength
        /min_duration)
        s[part_index, mo+no:mo+no+ln]
        =note.pitch.midi
```

上記のコードは、楽譜の MIDI 番号配列を挿入するプログラムである。MIDI 番号とは、音の高を低い音から順に 0~127 の数値で表現するものである。音符を示す `note` オブジェクトは、通常 `part` の中の `measure` の中にある。上記のコードは、`note` オブジェクトにたどり着く少し前からのコードである。4分音符1つ分を 1.0 と

```
class Onpu:
    def __init__(self, midi, start, end):
        self.midi = midi
        self.start = start
        self.end = end

    for i in range(ln):
        s[part_index, mo+no+i]
        = Onpu(note.pitch.midi, -i, ln-i-1)
```

`Onpu` クラスによって、MIDI 番号に加え、図2示したような音符の開始時刻と終了時刻を、復元可能な2次元のインデックスを埋め込んだ配列表現が可能となる。

### 4 まとめ

上記のコードを使うことによって、音符が持つリズムの情報を失うことなく、データを取得できた。しかし、楽譜には、最小音価で表すことのできない三連符や、一つのパートが二つの音を鳴らす重音や、スラーなどまだまだ取得できていないものが多数ある。これらのデータも出力されるようにするのが、今後の課題である。

### 謝辞

神奈川大学 上原 由衣特別助教に多大なご助言を頂きました。心より感謝申し上げます。

### 参考文献

- [1] 芹澤ら, 複数のパートに分散したメロディを抽出するための一手法, 岡山県立大学大学院情報系工学研究科.
- [2] `music21` Documentation, <https://web.mit.edu/music21/doc/index.html>.