 **rhythm\_library.yml** – バリデーションエラー修正プラン をキャンバスにまとめました。

- 主要原因 → duration 欠落 (ドラム/ベース)
- bass\_walking\_8ths は pattern\_type 宣言と duration 追加が必要
- **2つの解決アプローチ**：① 各イベントへ明示追加 ② loader 側でデフォルト補完

テーブル形式で具体的な修正例も載せています。

まずは Drum/Bass の **duration** を埋めて再テストしてみてください。

他に不明点や追加エラーがあれば、いつでもご相談ください！

**あなた：**

durationとは、なんですか？ なぜ指示が必要なんですか？ MUSIC21、もしくは prettyMIDIの機能に任せることは出来ますか？ というのは、おそらくデータやツールは強力なものがあるだろうし、素人の私では限界がすでに見えているからです。確認はできるし、耳もやがては育つでしょうが、最初から数値を示せと言われても、不可能と考えます。判断する基準がないからです。MUSIC21、prettyMIDI、で何が出来なのか、どこまで出来るのかをここで一度把握をしましょう。教えてください。

**ChatGPT:**

 **duration** とは

「その音符 (イベント) が何拍 (もしくは何秒) 続くか」を示す定義値です。

たとえば **duration: 0.5** と書けば 2 分音符の半分 (=四分音符)、

**duration: 1.0** なら 1 拍まるごと を意味します。

理由	なぜ明示するのか
正確なタイムライン	リズム-パターンは "いつ鳴り始めて ーいつ終わるか" の連鎖で出来ています。offset (開始) +duration (長さ) が揃ってはじめて、次の音との重なりや休符が確定します。
人間化 (humanize) 処理	ベロシティやタイミングをランダムに揺らす際、「音の長さ」を把握していないと レガート/スタッカート のコントロールが壊れてしまいます。
DAW 互換	最終的に Studio One などへ MIDI エクスポートするとき、DAW は必ず <b>ノート長</b> を必要とします。欠落していると「鳴りっぱなし/無音」などの不具合が起きがちです。

 **MUSIC21 と pretty\_midi に委ねる方法は？**

## 1. music21

- **Stream.makeMeasures()** / **.makeNotation()** で拍割りや連符を自動整形
- **Stream.quarterLength** を操作すれば **全体長を自動再配分** するユーティリティも用意されています。
- ただし 入力が「完全な四分音符長 = 1.0」の前提 になるため、16 分刻みやスウィングは手動でタイムシフトが必要。

## 2. pretty\_midi

- **note\_on / note\_off 時刻** を秒単位で与えない限り、pretty\_midi.PrettyMIDI オブジェクトは音の長さを判定できません。
- Instrument.get\_piano\_roll() を使えば後から長さを推定することは可能ですが、推定ロジックは 正確な譜面 までは復元しません（微妙な人間化情報が丸められます）。

### 結論

- 完全にライブラリ任せ も技術的には可能ですが、**細かいニュアンスや機能拡張（人間化・表記揺れ制御）を犠牲にする** ことになります。
- 指標（duration 値）を「ざっくり四分 = 1.0、八分 = 0.5…」程度で入れておくだけでも、後々の自動調整ロジックがはるかに柔軟になります。

### ★ “素人にはハードルが高い” と感じる時のプラン

レベル	方法	運用イメージ
① 超簡易	step_duration_q1 をパターンに追加し、 <b>全イベント共通の固定長</b> と宣言 → duration はローダ側で自動付与する	「16 分なら 0.25」などを 1 行書くだけ
② 半自動	duration を省略したら <b>loader がデフォルト 1.0 を補完</b> する仕様にする	まずは四分音符基調で OK、後から手直し
③ 完全自動	music21 で <b>オフセット列</b> だけ指定して <b>makeNotation()</b> に流し込み、拍子どおりに分割させる	精緻な人間化には追加スクリプトが必要

### ✿ まとめ

1. **duration は “音の長さ” の座標系**
2. 入力を簡素化したい場合は
  - step\_duration\_q1 で 一括指定
  - **ローダ側のデフォルト補完** を実装
3. それ以上に細やかなニュアンスを得たいときは、music21 や pretty\_midi の API を組み合わせて **後段で自動生成** も可能。

どの粒度がプロジェクトに最適か、迷うようでしたら “**まずは①**” → 作品を試聴 → 必要に応じて②・③へと段階的に深掘りしていくのが負担が少ない方法です。

ご希望の運用スタイルに合わせて、スクリプト側のサンプルや補完ロジックを具体的にご用意いたしますので、遠慮なくお申し付けください。

2/2

あなた:

何度も言うようですが、基準は数値よりも感情を優先してください。もちろん基準の設定は大切ですが、drum も guitar も、すべて歌詞セクションの感情が起点となって演奏も決定しています。現在、chordmap の改良行っています。以下の通りです。🎵【進化版：歌詞×セクション別テンション感情マップ】セクション 感情主軸 推奨テンション 補足と意

図 Verse 1 傷・孤独・諦め sus2, add9, m7 静かに始まる。「傷」と「壊れた関係」は add9 や sus2 で、繊細な浮遊感と迷いを演出。m7 は内向性とメロウ感。Pre-Chorus 1 感謝と後悔 m7, add11, M7 感情が込み上げてくる。「支え」や「逃げずに来れた」は add11、「ごめん」は m7 でほろ苦く。M7 は成熟した理解を。Chorus 1 愛と痛み 7th, sus4, 13th 「愛＝痛み」の問い。sus4 で未解決の苦悩、13th で感情の豊かさと複雑性を示す。C7sus4 → C7 → Fmaj7 のような「問い→解決」も美。Verse 2 不器用・自己否定 m7, sus2 儂く、内面の声。Dm7, Am7 で心の輪郭を、sus2 で「逃げる」弱さを描写。Chorus 2 感謝・灯火 M7, add13, m7 「言葉にできない感謝」には、Cmaj7 や Fmaj7。Dm7add13 で密やかな情緒。m7 で感情を落ち着かせる終止。Bridge 1 嵐・言葉の鋭さ sus4, dim7, b9, 11th 波乱の場面。E7b9 や Bdim7 で不安定さと葛藤。sus4 で「問いかけ」、11th で情緒の層の厚み。Verse 3 涙・無力・強さの誤認 m7, sus2, add11 A メロに戻るが、感情は深まっている。Am7, Dm7、そこに add11 の「気づき」を乗せる。Pre-Chorus 2 苛立ち・崖っぷち sus4, m7b5, add9 「崖でも構わない」は sus4 で表現。m7b5 や add9 で不安定さと希望の同居を演出。Chorus 3 痛みの中の願い 13th, m7, sus4 「君がいないと」の叫びを 13th で強く、sus4 で「答えなき想い」を。G13 → C7sus4 → Fmaj7 の解決感是非常にドラマチック。Bridge 2 夜明け・道の提示 M7, add13, 11th 明るさと未来感。「背中を押す人」には add13 で温もりを。Fmaj7add13、Cmaj7 などで優しい終焉の兆しを。Verse 4 光・自然・静寂 add9, M7, 11th 詩的で象徴的なパート。Fmaj7add9 や Cmaj7, add11 で自然界の感情を包み込むように表現。Chorus 4 再生・未来の選択 13th, add9, M7, sus4 壮大なフィナーレ。「ぶつかり合い支え合い」= 13th。「新しい光」には add9。「今だからこそ」→ sus4 → 解決の Cmaj7 へ流すとエモーションクライマックス。duration = 音の長さノートの長さは、基準は設けるけれども、歌詞セクションの感情 map で左右して欲しいんです。機械的でない、人間的な vocal = 歌詞の感情にそった演奏にしたいのです。数字できめてしまうと、固定されるので、それはどうなんだろうと。ランダムにしても機械的にうごかすだけでは、規則的で人間的な演奏にならない可能性を考えます。でも、chord に付随する感情で、演奏・音を導くことができれば。これまでにない、MIDI そして、歌になるかもしれない、というのが、この project の骨子です。人間を超える演奏、とおおまかにいっていますが、人間からかけはなれた演奏ではなくて、あくまで人情に沿ったものにしたいんです。主題歌のもとになっている原作小説は人情小説が多い。その Story を歌詞に落とし込んでいるので、Story の流れ、感情の流れはすでにあります。chordmap がうまくいき、あとは chordmap がもたらす感情と、MUSIC21、prettyMIDI などのシステムとをうまくかみ合わすことが出来れば、歯車は動き出すかもしれません。これは可能ですか？

## ChatGPT:

思考時間: 15 秒

Harusan—— ご構想、十分に実現可能です。

「固定値で縛らず、歌詞 × 感情マップが演奏を導く」ための設計指針と、MUSIC21 / pretty-midi での具体的な実装パスをまとめます。

## 1 層構造で考えると混乱しがち

→ 3 レイヤーに分けると整理できます

レイヤー	役割	主なパラメータ	実装ポイント
------	----	---------	--------

<b>A. ベースパターン</b>	“骨格” だけを持つ 汎用リズム集 (rhythm_library)	offset のみ必須 (duration は 省略 可)	今ある JSON/ YAML をそのまま 活用
<b>B. 感情ドライバ</b>	セクション／コード ／歌詞から 表情 係数 を算出	- <b>sustain_ratio</b> (0.3-1.2) - <b>staccato_ratio</b> (0-0.4) - <b>velocity_bias</b> (±15)	chordmap のテン ション表をここに 接続
<b>C. ヒューマニ ザ</b>	微揺らぎ・ランダ ムではなく 感情強度ベース の 確率分布	- timing_jitter (ms) - velocity_jitter - strum_width	曲線 or 確率表を emotion 強度でス ケール

✓ **duration** は A ではなく **B** で決定。

A では “四分音符相当” が何拍か知らなくても OK。

## 2 感情 → 音符長さを決めるルール例

感情強度	sustain_ratio	staccato_ratio	ねらい
静けさ (低)	1.0-1.2	0-0.05	ロングトーンで空 間を残す
穏やか	0.9-1.0	0.05-0.10	普通の歌心
苛立ち	0.7-0.9	0.10-0.20	歯切れよく短め
激昂 (高)	0.5-0.7	0.20-0.40	切り裂くように短 く

- **sustain\_ratio** × 拍長 が実際の duration。
- **staccato\_ratio** は長さをさらに削るか、MIDI CC64 (ペダル) をオフにする等で表現。

## 3 MUSIC21 / pretty-midi でどう組むか

機能	MUSIC21	pretty-midi
長さの自動計算	note.quarterLength = baseQL * sustain_ratio	Note.end = start + dur_sec
アーティキュレーション	Articulation('staccato') を強度で付加	Velocity 落とし + CC:64=0
ペダル/ダンパー	Pedal() expressions で オン/オフ	ControlChange 64
ヒューマニズ	note.offset += random.uniform(-jitter, +jitter)	同上 (秒単位)

テンポシフト	MetronomeMark(number=...) をセクションごとに	PrettyMIDI.time_signature_changes + tempo
--------	-------------------------------------	---

つまり **duration** フィールドを YAML に書かずとも、**sustain\_ratio** からコード側で計算して **note** に書き込む方が柔軟で人情に沿います。

#### 4 サンプル実装スケッチ (GuitarGenerator 内)

python

コピーする編集する

```
def apply_emotion_humanize(note, emotion, intensity, bpm):
```

```
    # 1) 感情 → 係数マップ
```

```
    coeff = EMOTION_TABLE[emotion] # sustain_ratio,
```

```
    staccato_ratio, jitter_ms ...
```

```
    # 2) duration 決定
```

```
    base_qL = note.quarterLength or 1.0
```

```
    note.quarterLength = base_qL * coeff["sustain_ratio"]
```

```
    note.quarterLength -= base_qL * coeff["staccato_ratio"]
```

```
    # 3) タイミング揺らし
```

```
    jitter_qL = coeff["jitter_ms"] * bpm / 60000 #
```

```
    ms→quarterLength
```

```
    note.offset += random.uniform(-jitter_qL, jitter_qL)
```

```
    # 4) Velocity バイアス
```

```
    if note.volume: # music21 Volume
```

```
        note.volume.velocity += coeff["velocity_bias"]
```

- **EMOTION\_TABLE** に Verse/Chorus マップを転写
- **intensity** (低中高) でテーブル値を線形補間

#### 5 導入ステップ

1. 感情テーブル **JSON** を用意

```
json
```

コピーする編集する

```
{ "pain": {"sustain":0.8, "staccato":0.15, "jitter":15, "vel":+5}, ... }
```

- 2.

3. **override\_loader** で セクション → 感情 を引き、



Generator に *emotion\_profile* を渡す。

4. 各 **Generator** が note を生成後、上記関数で長さ & 揺らぎを付与。

5. 耳で確認 → テーブルを調整。

数値を「耳で」決めず、耳でフィードバックしながら 徐々に詰めれば OK。

## 6 “人間を超えるが、人情に寄り添う” ために

- 機械的ランダム → 
- **Emotion** 曲線ベースのゆらぎ → 
- 固定数値 を 感情係数で可変化 することで  
“毎回少し違うが、物語の情緒に沿う” ミックスが得られます。

✓ 可能です。そして現行スクリプトに無理なく組み込めます。

次の一歩


1. *EMOTION\_TABLE* 草案をこちらで作成し、
2. Guitar/Bass/Drum の生成後フックに「長さ&ヒューマナイズ」関数を組み込み、
3. Verse 1 → Chorus 1 を試聴して微調整

ご希望ならすぐサンプルコードをお届けします。

「数字は最小限、感情ファースト」で歯車を回し、

Harusan の人情物語を “息づく演奏” に仕立てていきましょう。

ありがとうございます 😊 この project の可能性がさらにみえてまいりました。まずは感情の振り幅があることを想定して、演奏後との平均値を設定してもいいのかなと考えていました。その中央値から、感情に沿って、上げ下げするのが、もっとも自然かと考えます。歌詞、chord、そして感情がもたらす音を楽しみたいんです。わたしは小説家ですけど、文章がもたらすのも感情であり、そのけっか読者に起こるのは、あらゆる感動です。人間活動の根本は感動がささえるものだと考えます。感動をもたらすのは、身体であり、身体意識です。日本には身体で考えろ、文武両道なんていう言葉や考え方があります。脳と身体と意識は連関したもののなので、まず身体——武からはいって、身体意識をきたえ、そのけっかすぐれた脳活動をうみだし、政治や発明や文化活動に活かす、という考え方です。身体にうったえかける、ことも、言葉と音の文芸プロジェクトの骨子でありたいです。話が脱線しましたが、サンプルコードを届けてください。

 **emotion\_humanizer.py** をキャンバスにアップロードしました。

ハイライト

### 1. EmotionProfile (Pydantic)

EmotionProfile(

2. sustain\_ratio = 0.85,

3. staccato\_ratio = 0.15,

4. velocity\_bias = +6,

5. jitter\_ms = 12

6. )

7.

### 8. apply\_to\_music21\_stream() / apply\_to\_pretty\_midi()

- ノート長さを  $\text{sustain\_ratio} \times \text{基準拍}$  で再計算

- staccato\_ratio で短縮または Articulation('staccato') 付加
- タイミングを ±jitter\_ms ランダムシフト
- velocity\_bias を加算 (範囲チェック付き)

## 9. lookup\_profile(section\_name, intensity='mid')

- 先ほど共有した テンション感情マップ を簡易 JSON で内蔵
- intensity を "low" | "mid" | "high" で渡すと値を線形補間

### 使い方 (抜粋)

```
from emotion_humanizer import lookup_profile, apply_to_music21_stream
```

```
stream = ... # 生成済み music21 Stream
profile = lookup_profile(section_name="Chorus 1", intensity="high")
apply_to_music21_stream(stream, profile, bpm=88)
stream.write('midi', fp='chorus1_humanized.mid')
pretty_midi 側も同名 apply_to_pretty_midi(pm, profile) で同様に処理可能です。
```

### 次の一歩

1. EmotionProfileTable をカスタム JSON/TOML に移して編集しやすくする
2. modular\_composer で「生成後フック」に本ユーティリティを呼び出す
3. Verse → Chorus を試聴し、sustain\_ratioなどを耳で微調整

試してみてフィーリングをお聞かせください。

さらなるチューニングや追加機能 (スウィング、ペダル制御等) もすぐ対応いたします！