# A Perceptual Study of Amplitude Modulation Vibrato

*Max Henry*

Music Technology Area
Department of Music Research
Schulich School of Music

McGill University
Montréal, Québec, Canada
August 15, 2021

# Abstract

Sometimes misconstrued as simple frequency modulation, vibrato is actually a complex combination of frequency, amplitude, and spectral-envelope modulation. It is indeed possible to hear vibrato in the absence of any frequency modulation, a fact that has been observed by many researchers, though this phenomenon has not yet been thoroughly investigated. We conduct a listening experiment in which participants rate synthetic vibrato notes for their perceived realism and perceptual fusion. Eleven stimulus conditions are designed to remove possible necessary conditions for the perception of vibrato in the absence of frequency modulation. Conditions that destroy plausible resonant structure result in realism and fusion ratings that are not statistically different from unadulterated vibrato signals, suggesting that a plausible resonant structure is not necessary for the perception of amplitude modulation vibrato. Spectral flatness is shown to have strong predictive power in fusion and realism ratings. Perceived realism and fusion ratings are also shown to vary in opposite directions. These findings are discussed in terms of the theory of resonant enhancement, as quantified by the spectral flatness metric.

# Abrégé

Parfois interprété à tort comme une simple modulation de fréquence, le vibrato est une combinaison complexe de modulation de fréquence, d'amplitude et d'enveloppe spectrale. Il est possible d'entendre le vibrato en l'absence de toute modulation de fréquence, bien que ce phénomène n'ait pas encore été étudié en profondeur. Nous menons une expérience d'écoute dans laquelle les participants évaluent le réalisme perçu et la fusion perceptive de notes de vibrato synthétique. Onze conditions de stimulus sont conçues pour supprimer d'éventuelles conditions nécessaires à la perception du vibrato en l'absence de modulation de fréquence. Les conditions qui détruisent la structure résonante plausible produisent des notes dont le réalisme et la fusion ne sont pas statistiquement différentes des signaux de vibrato purs, suggérant qu'une structure résonante plausible n'est pas nécessaire pour la perception du vibrato à modulation d'amplitude. Il est démontré que la planéité spectrale a un fort pouvoir prédictif dans les évaluations de fusion et de réalisme. Le réalisme perçu et la fusion des notes varient également dans des sens opposés. Ces résultats sont discutés en termes de théorie de l'amélioration de la résonance.

# Acknowledgements

To Stephen: for evoking Robert Duncan's *Often I Am Permitted to Return to a Meadow* in a scientific context, and everything that represents. To Philippe: for teaching math so beautifully. To Arielle: for our panicked conversations about statistical methods. To my family: for our panicked conversations about statistical methods that you probably didn't understand.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Occasionally misconstrued as simply frequency modulation (FM), vibrato is in fact a complex combination of frequency, amplitude, and spectral-envelope modulation (Verfaille, Guastavino, & Depalle, 2005). In many cases, vibrato results in a relatively complex partial-specific amplitude modulation (AM) that is a direct consequence of simple FM interacting with the resonant structure of an instrument (McAdams, 1989). The significant contribution of this partial-specific amplitude modulation to the vibrato percept has been observed by many researchers (Gough, 2005; Maher & Beauchamp, 1990; McAdams & Rodet, 1988; Fletcher & Sanders, 1967).

Mellody and Wakefield (2000) find that synthetically “flattened” vibrato notes, i.e., those processed to remove all FM, maintain a strong vibrato percept. Analysis of individual partials reveals little coordination among partial amplitudes, save for the fact that their amplitude modulations share a common period at the fundamental frequency (the vibrato rate). The observed phenomenon of AM-only vibrato (AMV) raises two key questions: (1) what are the necessary conditions for the vibrato percept in the absence of any true FM, and (2) what prevents the individual partials from being perceived as an uncoordinated cloud of harmonically related sound, i.e., what are the conditions that fuse the partials into the percept of a single vibrato note?

In this document, we motivate and conduct an experiment to explore the perception of AMV. Specifically, we test the following assumptions: that AMV (1) requires a common fundamental AM rate that is phase-agnostic; and (2) partial modulations do not need to follow a plausible resonant structure. The first assumption is based on a large body of evidence for internal, across-channel AM rate selectivity (Bregman, Levitan, & Liao, 1990; Dau, Kollmeier, & Kohlrausch, 1997; Moore, Glasberg, Gaunt, & Child, 1991), and would provide evidence for a kind of modulation-rate-based perceptual grouping process. The second assumption aims to sharpen the first.

This thesis takes the following structure. First, we highlight relevant literature in music psychology and acoustics focusing on vibrato, modulation and resonant structure. From this review we suggest a number of possible necessary conditions for the perception of AMV. Next, we consider a synthesis engine and eleven stimulus conditions designed to remove possible necessary conditions for AMV. We then outline the design and results of a listening experiment in which participants rate these synthetic stimuli for their perceived realism and fusion. The thesis includes two appendices: a signal processing appendix covering the theoretical considerations for spectral envelopes and their practical extraction for analysis; and an appendix with the synthesis engine code, written in Python.

# Chapter 2

# Background

In this chapter, we summarize some of the key observations about frequency modulation and amplitude modulation in musical vibrato, and their link through resonant structure.

## 2.1 Contribution of amplitude modulation to spectral envelope identification

McAdams and Rodet (1988) sought to expand on prior research suggesting that the auditory system extracts the spectral envelope as a cue for perceptual grouping. They hypothesized that listener awareness of the spectral envelope would be stronger in modulated, rather than static, signals. This reasoning may be illustrated with a simple example. Consider a spectral envelope that attenuates frequencies at 1 kHz and boosts them at 1.1 kHz. Now consider a static harmonic excitation signal with a fundamental frequency of 350 Hz, and a harmonic at 1050 Hz. Under this spectral envelope, the harmonic would assume an intermediate gain between the valley at 1 kHz and the peak at 1.1 kHz. If, however, this excitation signal were modulated in frequency, its gain would increase with increasing frequency (as the partial approached the peak at 1.1 kHz), and decrease with decreasing frequency (as it approached the valley at 1 kHz), thereby tracing out the spectral slope of the envelope, and betraying a good deal more information about the underlying spectral structure. Subjects were presented with one-second stimuli in one of two varieties: either as a complex harmonic tone, or a single sine tone corresponding to the second harmonic of the complex signal. All signals were frequency modulated at depths up to 10% of the fundamental frequency (675 Hz). As in the example above, their synthesis engine was designed such that frequency modulation would impart a concurrent amplitude modulation, here reading frequency-specific gains from one of two fixed spectral envelopes. The envelopes were cleverly designed such that the second harmonic would have the same time-varying amplitude in either condition; however, depending on the envelope, this modulation would either be positively or negatively correlated with the fundamental frequency modulation. All the other harmonics, however, were affected in the identical way by both spectral envelopes. As a result, the behaviour of the second harmonic would provide the only clue as to difference in spectral envelope from one condition to another. The experimenters found a sensitivity for these minimally different envelopes from as little as 1.2% vibrato width. In a follow-up experiment, one subject was able to identify the spectral envelopes with as little as 0.6% vibrato width. This finding, that just one partial can have an impact on the perceived spectral envelope, is striking because of the spectral fusion imparted by harmonicity (Bregman, 1990). It is exceedingly difficult to hear out the amplitude trajectory of an individual partial in a complex harmonic tone, and so it is rather unlikely that the participants were reasoning on the behaviour of one partial explicitly. These results suggest an internal, more holistic mechanism for tracing a spectral envelope, based on the collective behaviour of all partials.

In a later experiment (McAdams, 1989), McAdams demonstrated an effect of spectral envelope on the perceived segregation of modulated vowel sounds. Notes were synthesized at one of three pitches equally spaced in pitch by a perfect fourth, each having one of three spectral envelopes corresponding to three vowels. The notes were played simultaneously in one of four conditions: either none, the first, second or third pitch was independently modulated with sinusoidal vibrato (the “target” signal). In each case, the other two “ground” notes were either stationary, or modulated in unison, though with a different rate and depth than the target signal. The experiment found vowels that were sinusoidally modulated were judged to be more prominent than the others, and this prominence is unaffected by concurrent modulation of the other two vowels. Importantly, the experiment also found that for the two coherently modulated ground notes, modulation did not reduce their perceived separation from one another, suggesting a strong role for their respective spectral envelopes in segregation.

A follow-up experiment (Marin & McAdams, 1991) was conducted in order to isolate the effect of spectral-envelope tracing in the relative prominence of vowels in a mixture. Frequency-modulated stimuli were synthesized either with or without a constant spectral envelope. Spectral envelope tracing in this case was not found to have a significant effect on the relative prominence of syllables. The authors attributed this negative result in part to an insufficient AM depth imparted by the spectral envelopes used in their experiment.

## 2.2 Interactions of amplitude modulation with resonant structure

Maher and Beauchamp (1990) set out to explicitly investigate the interconnected nature of amplitude and frequency modulation of partials in natural vocal vibrato signals, a phenomenon they referred to as "spectrum modulation."

The experimenters recorded four singers (soprano, alto, tenor and bass) singing the same vowel sound (/a/). The vibrato observed tended to be nearly sinusoidal in frequency modulation, and consistent in rate among singers (between 5 and 5.7 Hz). Despite the relatively wide partial-specific amplitude modulation depths (of up to 11 dB), the global amplitude modulation depth was observed to be relatively shallow (between 0.5 – 3 dB of RMS amplitude). This effect was attributed to the relative phase of the amplitude modulations: the observed amplitude maxima and minima of the various partials were frequently out of phase with one another. We note that a similar effect has been observed in violin vibrato by Curtin and Rossing (2010), what the authors refer to as “sizzle.”

Using a sinusoidal model (McAulay & Quatieri, 1986), the vibrato tones were resynthesized in various simplified conditions. Among the simplifications, a condition was constructed in which the partial-specific amplitudes were systematically averaged and applied globally to the signal. In a similar fashion, a condition was constructed in which the partial-specific frequency modulation was averaged and applied globally to the harmonic signal. The averaging of partial-specific amplitude was found to degrade the quality of the resynthesized signal “substantially,” much more so than simplifying the frequency modulations across partials, an effect that has been observed in other experiments (Mellody & Wakefield, 2000; McAdams, Beauchamp, & Meneguzzi, 1999).

**Patterns of modulation.** Describing the complex relationship between the time-varying frequency and amplitude of the partials, Maher and Beauchamp suggest three patterns that betray an underlying resonant structure: (1) a partial whose amplitude modulation is in phase with its frequency modulation (i.e., while the partial increases in frequency, it also increases in amplitude) suggests that its frequency sits on the low side of a resonant peak; (2) a partial whose amplitude is in anti-phase with the frequency modulation suggests that it is on the upper side of a resonant peak; and (3) a partial whose amplitude modulation contains two peaks suggests that it is swept across a resonant peak, or trough between two adjacent peaks. Similar patterns have been observed in other investigations (Mathews & Kohut, 1973; Gough, 2005). These patterns call to mind the finding by McAdams and Rodet (1988) that even single partials may betray the spectral envelope of its signal.

**Simplifications for a vibrato synthesis engine.** With the aforementioned modulation patterns in mind, the authors constructed a simple synthesis engine in which partial amplitudes were read from a time-varying array of spectral envelopes; the envelopes were linearly interpolated over time between two spectral frames, corresponding to the maximum and minimum pitch of one cycle of observed vocal vibrato. While such an engine only accounts for the first two observed patterns of partial modulation (i.e., it cannot generate a partial that oscillates twice in one cycle of vibrato), the simplification seemed to have little effect on the resulting quality of the synthesis, yielding reportedly “good-quality” synthetic sung vowels.

The human voice, however, has relatively few resonances. If one were looking to investigate the interactions of frequency modulation with resonant structure, string instruments provide a more resonant-rich alternative.

## 2.3 Resonances in string instruments

In order to produce their own synthesis system for a bodiless violin, Mathews and Kohut (1973) endeavoured to simulate the resonant structure of a violin from an electronic signal.

They recorded a dry signal of a bodiless violin through a magnetic pickup applied to its bridge. A taped performance was used to excite a bank of resonant circuits. This setup permitted the authors to add or remove resonators, and to tune the filter frequencies and bandwidths to different arrangements, while controlling for the violin performance. Informal listening tests suggested that there is a "sweet spot" for the relative prominence of resonant peaks, the ratio in gain from peaks to valleys: too little prominence produced a dead sound that was unresponsive to vibrato; too much prominence produced a hollow sound that was unevenly responsive to notes played across the instrument range.

Assuming that an ideal peak prominence was maintained, violin tone quality appeared surprisingly robust to other changes in the filter-bank setup. Roughly doubling the density of resonators, from 20 to 37, resulted in a slightly brighter tone, though the perceived difference in sound quality was reportedly small. Tuning the resonant frequencies alternately to those measured from a Stradivarius violin, or to entirely arbitrary values (adjusting so that the overall resonance was sufficiently even) had little effect on sound quality.

From their observations, Mathews and Kohut suggested a theory of "resonant enhancement of tone quality." They propose the following rules for a satisfactorily resonant timbre: (1) that resonance frequencies must be sufficiently narrow such that the spectral envelope permits steep slopes for any conceivable partial—allowing for substantial amplitude variation for any given partial given a slight frequency modulation (as in vibrato); and (2) that the resonances must be sufficiently dense such that the valleys between spectral peaks do not drop below at most 15 dB—this rule is to avoid generating a “hollow” sound.

## 2.4 Vibrato analysis by synthesis

Mellody and Wakefield (2000) conducted an analysis of violin vibrato using an expressly designed time-frequency distribution they referred to as “the modal distribution,” to analyze the frequency and amplitude modulations in violin vibrato.

Like the spectrogram, the modal distribution is a member of Cohen’s class of bilinear time-frequency distributions and can therefore be expressed as a transformation of the Wigner-Ville distribution (WVD) with a filtering kernel (for a theoretical discussion of the WVD please see Section A.2.1). In the case of the modal distribution, the filter is specifically tuned to the pitch of the signal under consideration. It is an ideal lowpass filter operating on the time axis, set to a cutoff of half of the anticipated minimum frequency difference between adjacent partials (for harmonic signals, this difference is the fundamental frequency), therefore smoothing out between-partial cross-terms without greatly affecting the relatively slow-varying amplitude of each partial. Using this distribution, the instantaneous power can be extracted by integrating over the region of a local maximum in time; likewise, the instantaneous frequency can be measured as the first spectral moment (or spectral centroid), integrating over the region of a local maximum in frequency.

In their analysis using the modal distribution, Mellody and Wakefield observe a maximum partial-specific fluctuation of 15 dB, which is consistent with the theory of resonant enhancement proposed by Mathews and Kohut (1973).

They also make a striking observation about the pattern of amplitude modulation by analyzing the *spectrum* of the amplitude modulation signal of each partial. Each partial-specific spectrum had peaks at integer multiples of the vibrato modulation rate; in other words, all amplitude modulations were "harmonic signals" of the vibrato rate. They also note that there is close to zero correlation between the initial phase of the amplitude modulation signals of the partials.

To demonstrate the usefulness of the modal distribution, a direct comparison between the spectrogram and the modal distribution is reported. The spectrogram is shown to greatly underestimate the partial-specific fluctuations in amplitude, deviating from the ground truth by over 30% of linear amplitude depth.

### 2.4.1 Necessary conditions for amplitude modulation induced vibrato

These observations taken together suggest that partial-specific AM has a rather important role in vibrato perception—as suggested by the AM-only reconstruction condition in Maher and Beauchamp (1990). Given the apparently random behaviour of the partial-specific amplitudes as observed by Mellody and Wakefield (2000) and Mathews and Kohut (1973), the conditions leading to this percept are not yet completely determined. Therefore, we seek in this study to determine the necessary conditions for the perception of amplitude-modulation-induced vibrato (AMV). Specifically, we wish to know the following:

1. What is the role of amplitude-modulation phase in AMV? Can the relative phase be randomly distributed between partials, or is a fixed phase relationship required?

2. Given the highly independent nature of the amplitude modulations, what binds them together perceptually? Specifically, are they bound perceptually by a common "fundamental" modulation rate?

3. What is the role of resonant structure and partial-specific amplitude depths in AMV? Must the partials collectively trace a plausible resonant structure, such that a holistic image of a spectral envelope emerges?

In order to answer these questions, we designed a listening experiment in which participants rated synthetic stimuli, each having a targeted degradation of the spectral envelope of a cello note with vibrato.

# Chapter 3

# Experiment: Perceived fusion and realism of degraded vibrato sounds

## 3.1 Selecting the model vibrato

The stimuli in this experiment are based on measurements of a sustained cello note from the Vienna Symphonic Library sample collection, performed with medium vibrato at a forte dynamic on a C3 (“VC_mV_sus_f_C3.wav”). This note was determined to carry a particularly strong amplitude-modulation-only vibrato percept. The choice of this excerpt over other candidate notes is result of a series of informal listening tests, described in this section.

### 3.1.1 Pilot tests

We chose four candidate instruments for initial analysis, all traditionally capable of performing vibrato and all playing in a comparable range: trombone, cello, tenor voice, bassoon. The note C3 was determined as a common pitch, sitting in a comfortable performing range for each instrument. The bassoon, cello and trombone samples were taken from the Vienna Symphonic Library collection. The voice sample was excerpted from the Vocalset dataset (Wilkins, Seetharaman, Wahl, & Pardo, 2018) from an extended passage singing on the vowel /a/. This passage had a similar dynamic and pitch compared to the other samples auditioned.

**Excerpting steady vibrato.** All files were summed to mono and normalized to their largest absolute value in amplitude. To establish a common note onset time, the beginning of each file was trimmed for initial silence or ambient noise. Ambient noise level was determined to be anything below $-5$ dB of the peak level of the amplitude envelope of the sustained note. This extremely conservative threshold was chosen to accommodate the high noise floor in the Vocalset recordings. We take this new value to be the *de facto* note onset time, and excerpt each signal starting after an additional delay of 750 ms, and ending at 2.5 seconds. The delay at the start of the excerpt is meant to avoid capturing any potentially slower warm-up vibrato (Maher & Beauchamp, 1990). The duration of 1.75 seconds accommodates at least six full cycles of vibrato for analysis, allowing for a conservatively slow vibrato rate of 4 Hz, and 12.5-ms audio fades at the beginning and end of the excerpt. Each of the resulting excerpts was observed to have a steady vibrato rate and pitch percept.

**Analysis with the sinusoidal model.** The excerpts were then analyzed and resynthesized using SMT, a Matlab implementation of the sinusoidal model[1] (McAulay & Quatieri, 1986). This model considers a signal as a mixture of amplitude- and frequency-varying sinusoids, tracking spectral peaks from frame to frame to trace plausible sinusoidal components. Such a model can accommodate varying degrees of synthesis detail; improvements of the basic model include cubic interpolation to retain original partial-specific phase and modeling the transfer function of the residual noise (Serra, 1997). In the case of these particular steady-state tones, reconstruction without phase interpolation or residual noise yielded results that were indistinguishable from the original excerpts and was therefore sufficient for our purposes.

SMT uses a pitch-synchronous analysis, which determines the frame size and effective frame rate of the analysis. Each frame captures three full cycles of the signal. The analysis algorithm hops by default at 50% of this frame size. Assuming a fundamental pitch of C3 = 130.81 Hz, we chose to use a window size of 22.9 ms (1012 samples at a 44100 Hz sample rate) and a frame rate of 87.2 Hz. Such an analysis can safely capture modulations up to a rate of 43.6 Hz, or roughly, the first seven harmonics of a vibrato at 6 Hz.

[1]https://github.com/marcelo-caetano/sinusoidal-model

**Removing frequency modulation.** All excerpts were resynthesized in an amplitude-modulation-only condition. To remove frequency modulation, the time-varying frequency of each partial was replaced with a static geometric mean. We use the geometric mean to accommodate the logarithmic scaling of pitch with frequency (Stevens, Volkmann, & Newman, 1937). The resulting resyntheses were auditioned and assessed informally. Of the four excerpts, the cello appeared to have the strongest vibrato percept without frequency modulation, perhaps due to the rich resonant structure of string instruments (Curtin & Rossing, 2010). The bassoon vibrato excerpt was largely unchanged by the FM flattening, though its vibrato percept was more subtle than that of the cello. The flattened trombone excerpt sounded natural, but completely lacking in vibrato. The voice excerpt appeared to have lost all vibrato and also sounded decidedly unnatural.

**Characterization of partial-specific amplitudes.** Figure 3.1 shows the sinusoidal analysis of the cello vibrato excerpt with its FM removed. The signal has a roughly $-10$ dB/octave lowpass shape. The partials range in modulation depth from 4.89 dB to 27.18 dB from peak to trough; the mean partial-specific dynamic range is 13.73 dB. These values are largely consistent with previous observations of string instrument vibrato (Mathews & Kohut, 1973). We note the inclusion of two partials below the fundamental frequency: a near-DC signal having relatively little energy (approximately -40 dB), and a sub-harmonic having a modulation depth close to that of partial at the fundamental frequency. Figure 3.2 shows the waveforms of selected partials. Gough (2005) observed

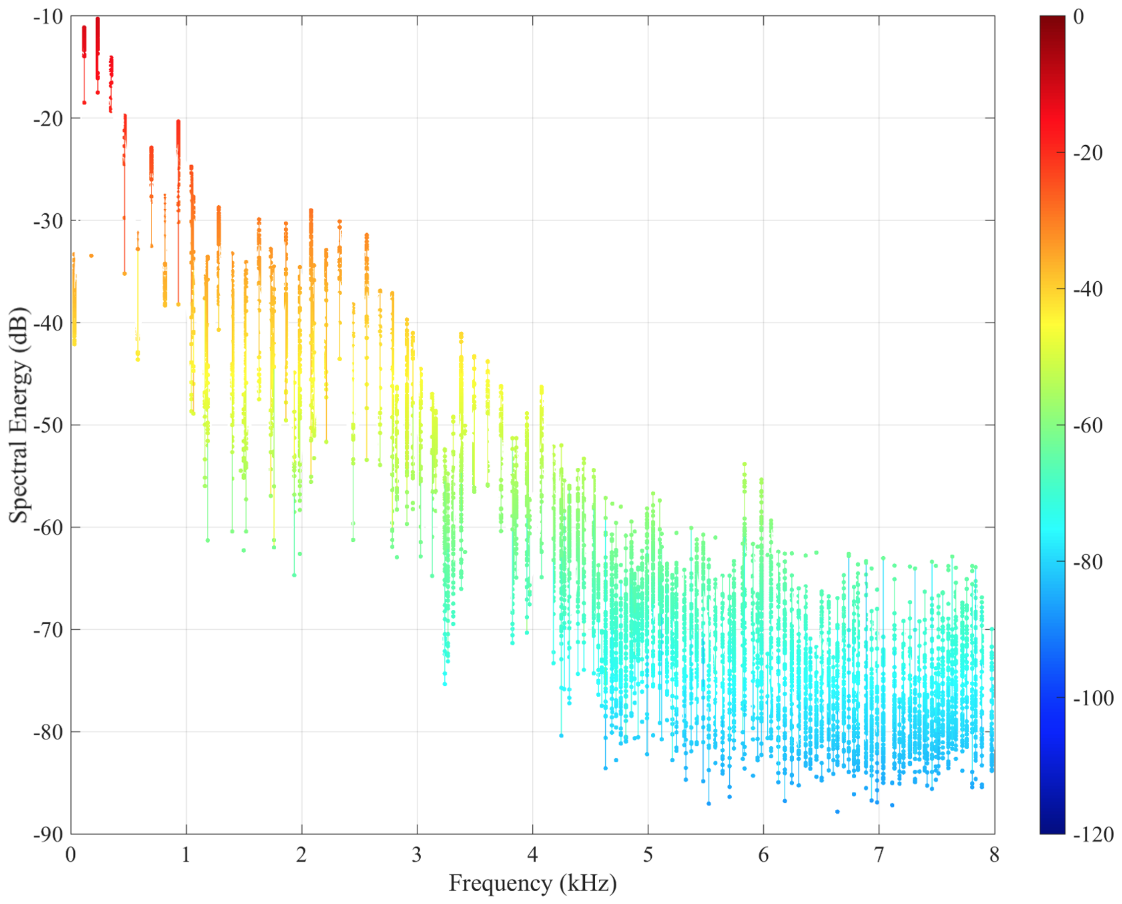

**Figure 3.1:** Cello vibrato excerpt analysis by the sinusoidal model. Each vertical stripe indicates the dynamic range and average frequency of one partial.

that violin partials can have an asymmetric amplitude envelope, i.e., an envelope whose periodic rise is shorter than its periodic decay, due to the warmup and ringing characteristics of the resonant modes in a violin body. However, in our excerpt the partials appear to be relatively symmetric on a broad time scale.

From visual inspection alone, it is hard to determine a pattern that binds the partial amplitudes together, leading to an emergent vibrato percept. While their modulation

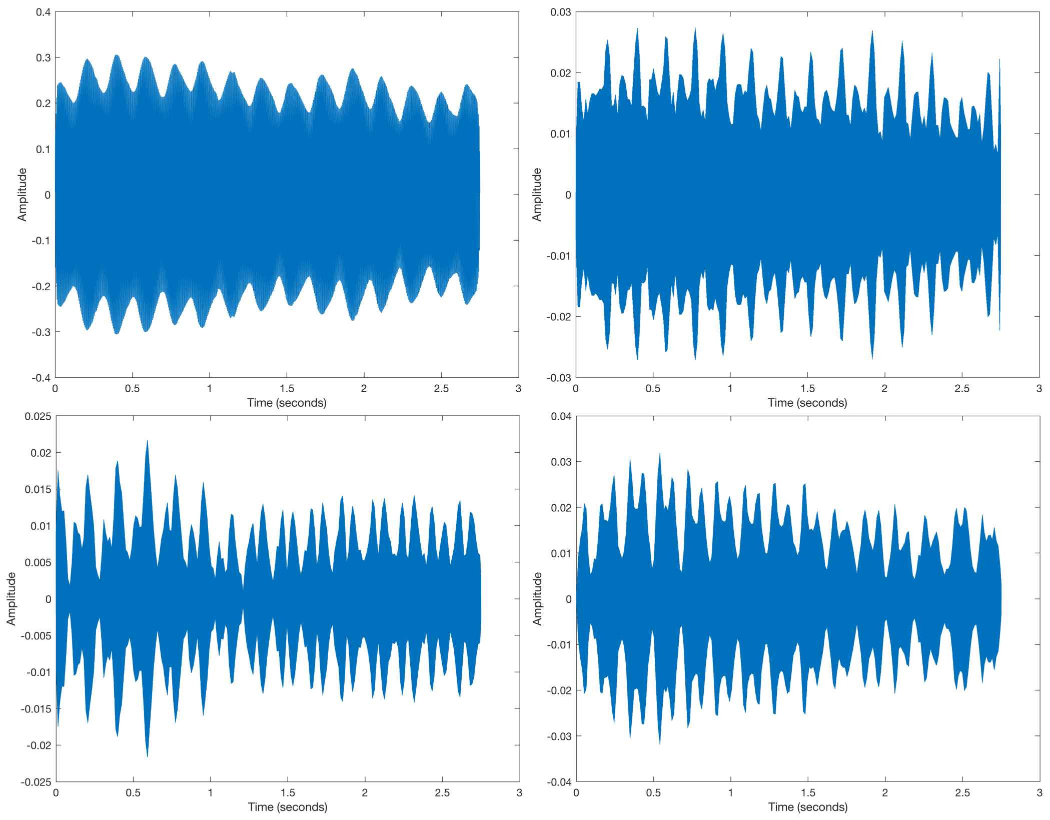

**Figure 3.2:** Extracted partials from a cello vibrato excerpt. Top left, fourth partial. Top right, seventh partial. Bottom left, sixteenth partial. Bottom right, twentieth partial.

patterns can be rather complex, in isolation each partial sounds like a simple periodic tremolo; that is to say, the detail of the amplitude modulation does not appear to translate to the ear. Instead, these modulations may implicitly contribute a great deal to the perceived spectral envelope of the instrument (McAdams & Rodet, 1988).

### 3.1.2 Predictions

From these observations, we hypothesize that the AMV percept is predominantly a function of a common partial-specific modulation rate, as observed by Mellody and Wakefield (2000). Following Mathews and Kohut (1973), we further hypothesize that a plausible resonant structure is not necessary to establish the percept. We propose an experiment in which participants rate systematically degraded vibrato signals, in which each degradation is aimed at a presumed necessary condition for AMV perception.

Degraded stimuli will be rated for their perceived realism. When considering the role of the spectral envelope in perceptual grouping (McAdams & Rodet, 1988), it is also possible that by decoupling amplitude and frequency modulations that agree on a resonant structure, some distortions will result in a signal that is no longer perceived as coming from a single source. McAdams (1984) found that the depth of frequency modulation, whether tracing a plausible resonant structure or not, had a marked effect on perceived fusion. Consequently, we conduct a parallel experiment in which stimuli are rated for their perceptual fusion.

## 3.2 Method

We design a listening experiment in which participants rate synthetic signals that are based on cello vibrato. Each synthesis condition (described in Section 3.2.2) aims to remove a possible necessary condition of the AMV percept. The experiment has two phases, drawing from two separate participant pools, each aimed at a different aspect of the synthetic stimuli: phase one is concerned with the perceived fusion of vibrato partials, and phase two is concerned with apparent vibrato realism.

### 3.2.1 Participants

Participants were gathered from the Prolific[2] online participant pool (Palan & Schitter, 2018). Prolific is a popular crowdsourcing platform that collects and pays participants for online studies. All participants were pre-screened to have self-reported normal hearing. To ensure that they would understand the rating task, we also screened by language. In the first phase, participants were asked to be fluent in English; in the second phase, we requested participants who spoke English specifically as a first language. All participants agreed to an informed consent form and were compensated for their participation. The study was certified for ethical compliance by the McGill University Research Ethics Board II.

For the first phase of the study, concerned with perceived fusion, we ran 54 participants, four of whom did not complete the experiment due to technical difficulties (see below), leaving a total of N = 50 participants (19 female) having a median age of 23.5. Though we screened for fluency in English, participants reported a variety of first languages, including Polish, Portuguese, Catalan, Italian, Greek, Spanish and Turkish.

For the second phase of the experiment, concerned with perceived realism, we ran 50

[2]https://www.prolific.co/

participants, four of whom did not complete the experiment, again due to technical difficulties, leaving a total of N = 46 participants (27 female) having a median age of 25. All participants reported speaking English as a first language.

**Technical issues.** Participants reported having to wait for extended periods at a blank screen in order to progress to the next trial, or to be returned to the Prolific platform after the final trial. Prolific can direct a large volume of participants to a server at once, and we suspect this slow behaviour was due to a sudden influx of traffic directed at an under-powered academic server. In the first phase of the experiment, an additional 35 participants left early and forfeited their results; they were not paid for their time, and their data were not collected. No participants reported a delay between hearing and then being able to rate a sound, nor did any participant report a distortion in the playback audio. Comparing our trial time-stamps to the Prolific data, it appears that most participants accrued a significant amount of extra time on the final screen after having completed the experiment, waiting to be returned to Prolific to report their completion. Four participants completed the first phase of experiment but were unable to submit their results due to significant delays; they were paid for their time. For the second phase of the experiment, we limited server access to ten participants at a time; this phase was run largely without incident. Four participants were unable to submit their results after having completed the experiment. They were paid for their time.

**Language.** While technical difficulties necessitated the bulk of our private correspondence with participants, some first-phase participants revealed themselves to be less than fluent in English, casting doubt on the efficacy of the pre-screening procedure. A poor command of English could lead to a misunderstanding of the experimental task. For our second experiment, we pre-screened for participants speaking English exclusively as a first language.

### 3.2.2 Stimuli

**Spectral envelope extraction.** Rather than attempt to reconstruct a prolonged vibrato excerpt, which is subject to broader trends in pitch and amplitude (Maher & Beauchamp, 1990), we choose to build stimuli based on an isolated cycle of vibrato. While this decision results in a slightly artificial sounding synthesis, it allows for greater control in the synthesis process. Counterintuitively, choosing a stimulus that is at its base apparently artificial allows for “apparent realism” to be removed as a possible confounding factor in the experiment. In this way, all stimuli present with an equal *fidelity handicap*, and therefore the perceived differences in each condition can be safely attributed to the specific manipulations of each synthesis condition.

We use fundamental pitch to determine and extract one period of the cello vibrato. We extract the pitch trajectory and time-varying spectral envelope from the cello recording at a framerate of 200 Hz, using the WORLD suite of algorithms (Morise, Yokomori, & Ozawa,

2016) as implemented in the Python library PyWorld[3].

From the pitch trajectory, we extract the time indices of local maxima, each demarcating the beginning and end of one cycle of vibrato modulation. It should be noted that isolating frequency modulation cycles by local maxima is not a particularly robust method, especially in the potential of more complex pitch trajectories. However, given the largely sinusoidal character of vibrato pitch modulation (Mellody & Wakefield, 2000), we determine this method to be sufficient for our purposes. From the six full vibrato cycles present in the excerpt, we choose the fourth cycle because it begins and ends at a nearly identical fundamental frequency, suggesting that it would make for the most successful loop. We store the spectral envelope of the fourth full vibrato modulation cycle as the basis of our stimuli.

**Synthesis engine.** The synthesis engine has two components: (1) a synthesis module that reads partial amplitudes from an array of time-varying spectral envelopes, and (2) a module for making specific modifications to a time-varying spectral envelope.

The synthesis engine assumes a simple additive model:

$$x(t) = \sum_{p=1}^{P} A_p(t) \cos\left(2\pi p \int_{\tau=0}^{\tau=t} \nu_0(\tau)\mathrm{d}\tau + \varphi_p\right), \tag{3.1}$$

where $x(t)$ is a harmonic signal made of $P$ partials, $A_p(t)$ is the partial-specific time-varying amplitude, $\nu_0$ is the instantaneous fundamental frequency trajectory, and $\varphi_p$ is the partial-

[3]https://github.com/JeremyCCHsu/Python-Wrapper-for-World-Vocoder

specific initial phase.

In our stimuli, $\varphi_p$ is randomly selected for every partial from a uniform distribution between $\pi$ and $-\pi$. The time-varying frequency $\nu_0(t)$ takes on one of two conditions. For stimuli with no pitch vibrato, it is an array of a constant value fixed at 130.81 Hz (C3). In the conditions with pitch vibrato, it takes the following form:

$$\nu_0(t) = {\nu_0}' \left(1 + \Delta\nu \cos(2\pi\nu_m t)\right), \tag{3.2}$$

where ${\nu_0}'$ is the central fundamental frequency (C3), $\nu_m$ is the vibrato rate, and $\Delta\nu$ is the pitch vibrato extent, determined by:

$$\Delta\nu = 2^{s/12} - 1, \tag{3.3}$$

where $s$ is the pitch vibrato extent in semitones. In our experiment, we use the value extracted from the cello excerpt: 0.1314 semitones measured from center to peak.

True vibrato signals contain transient inharmonicities due to frequency dependent group delay in the partials (Maher & Beauchamp, 1990). In a study with spectrotemporally simplified stimuli, McAdams et al. (1999) found that frequency incoherence was discriminable from stimuli having artificially homogenized frequency trajectories 70% of the time in violin sounds. In our study, all of the frequency-flattened stimuli will be necessarily harmonic at all times. Therefore, as a control measure, we chose to ignore the temporary inharmonicity in frequency-modulated vibrato signals to avoid introducing a potential confounding factor.

As the spectral envelopes are sampled at a rate of 200 Hz, the engine must interpolate audio rate amplitude values between spectral frames. We use linear interpolation. Linear interpolation yields some spectral imaging distortion; however we found that in this application, more computationally expensive techniques like *sinc* interpolation yielded qualitatively similar results. Given the large volume of stimuli needed for this experiment, and the considerably longer computational time for little apparent gain imparted by sinc interpolation, we opted to use linear interpolation.

The engine reads from spectral envelopes having 1025 non-negative frequency bins. Our experiment is conducted at an audio rate of 44100 Hz, which means that each spectral bin spans 21.53 Hz. For partials whose frequency corresponds to a fractional bin value, its framewise amplitudes are linearly interpolated between adjacent spectral bins; these spectrally interpolated values are then interpolated over time as above.

To generate audio, the synthesis engine loops over a single cycle of spectral envelopes. At each loop point, the final frame values are linearly interpolated into the first frame values. We note that while this approach can lead to aliasing in one or more partials, should a partial have to modulate suddenly at a rate greater than 100 Hz, the time-varying spectral envelope used in this experiment was sufficiently circular across all frequencies to ensure that each partial made a perfect loop.

**Spectral envelope modification.** In order to create the desired vibrato distortions, many conditions require significant modifications to the spectral envelope. We developed a parallel engine that, given a fundamental frequency and an array of time-varying spectral envelopes, modifies the amplitudes of the harmonic partials. In the case where the engine modifies partials whose frequencies correspond to fractional bin values, it modifies both adjacent bins in parallel. For example, the fundamental frequency 130.81 Hz lies at the bin value 6.07; in this case, the engine modifies both bin 6 and bin 7 in parallel. At this spectral resolution and fundamental frequency, the partials are well enough separated that no two share a common spectral bin.

### Stimulus conditions

For each synthesis condition, the engine generates 70 partials spaced at integer multiples of the fundamental frequency (130.81 Hz), whose amplitude trajectories are read from the time-varying spectral envelopes of one cycle of cello vibrato. All sounds are equalized for duration (2 s) and adjusted for loudness after synthesis. The signal is faded in and out with 200-ms linear ramps. Stimuli can take on one of eleven conditions, modifying either the frequency modulation (FM) characteristics, or the amplitude modulation (AM) characteristics of the original cello excerpt. Each condition is designed to remove a possible necessary condition for the perception of AMV. Parameters for all random stimuli are logged.

**Simplified reconstruction (BASIC).** The BASIC stimulus has both FM and the complex AM derived from scanning the resonant structure of the instrument, as described by Mathews and others (Mathews & Kohut, 1973), and serves as a positive control. The FM is implemented with Equation 3.2 using the measured pitch excursion of 0.1314 semitones. To the extent that this reconstruction is perceived as “fused,” or “realistic,” it is purely a function of the synthesis engine.

**Partial-specific amplitude modulation (FROZEN).** The FROZEN condition keeps the complex AM of a plausible resonant structure, but contains no FM. It is otherwise identical to the BASIC condition. The FROZEN condition corresponds to the FM-flattened signals in our listening experiments and represents the greatest unadulterated potential of the cello signal to carry the AMV percept.

**Frequency modulation only (FM-ONLY).** The FM-ONLY condition applies FM to a static spectral envelope. The engine determines the gain of each partial by averaging the spectral envelopes over time. We note that this average does not necessarily correspond to the spectral envelope of the instrument “at rest,” as partials are emphasized differentially throughout the vibrato cycle (Maher & Beauchamp, 1990; Gough, 2005). We note here that only the BASIC and FM-ONLY conditions have frequency modulation. All of the following stimulus conditions are AM only.

**Average global amplitude modulation (PAM).** The pure amplitude modulation (PAM) condition takes the amplitude envelope of the FROZEN condition (i.e., the sum of partial-specific amplitude trajectories) but applied globally to signal, rather than differentially to each partial. This kind of modulation is sometimes called tremolo. To built the amplitude envelope, the engine traces each partial trajectory from the spectral envelope array, then sums and averages them into one global envelope. This envelope is then applied to all partials. Each partial is given a relative gain based on the time-averaged spectral envelope, as in the FM-ONLY condition. In practice, each PAM resynthesis is slightly different due to randomization of the initial phase of the partial oscillators. As has been previously observed in violin and vocal vibrato (Maher & Beauchamp, 1990; Curtin & Rossing, 2010), the relative amplitude modulations of each partial tend to counterbalance one another. As a result, PAM signals have only a very slight amplitude modulation that is often imperceptible.

**Amplitude modulation with shuffled phase (SHUFFLE).** The SHUFFLE condition reconstructs the AM trajectory of each partial, but shuffles the initial phase of each modulation signal by a random quarter of the cycle: $\{0, \pi/2, \pi, 3\pi/2\}$. While the synthesis engine can accommodate arbitrary divisions of the vibrato cycle, here we chose four possible randomizations to reflect four possible idealized amplitude trajectories across a resonant mode: ascending a resonant peak, descending a resonant peak, and two intermediate modulations across a spectral peak or valley, respectively. This condition aims to scramble the plausible resonant structure of the instrument while maintaining the fine structure of the individual amplitude modulations.

**Simplified amplitude modulation (SIMPLE).** The SIMPLE condition is built from the partial-wise modulation *depths* of the original vibrato; however, the modulator signals are replaced with simplified, one-cycle sinusoids. As in the FM-only and PAM conditions, each partial takes the time-averaged spectral envelope as its center amplitude. Partials are amplitude-modulated such that their peak and minimum gains match the cello vibrato time-varying spectral envelope. We measure modulation depth as:

$$\Delta G_p = 20 \log_{10} \left( \frac{\max A_p(t)}{\min A_p(t)} \right) \tag{3.4}$$

This condition, when compared to the SHUFFLE condition, aims to determine the effect of the fine structure of the modulation signals on the AMV percept. Like the SHUFFLE condition, the initial phase of each modulator is randomized.

**Randomized resonant structure (RAG).** The random gain (RAG) condition assigns a random modulation depth, within $0 - 10$ dB, to each partial. It is otherwise constructed like the SIMPLE condition. This dynamic range is a conservative compromise between the observed mean modulation depth of the analyzed cello vibrato cycle (13.73 dB), and the 10-dB gain observed to be effective in arbitrarily distributed resonators for recreating violin sounds (Mathews & Kohut, 1973). The lower limit of 0 dB accommodates the possibility that some partials do not modulate, or modulate only slightly.

**Randomized amplitude modulation frequency (RAF).** The SHUFFLE, SIMPLE, and RAG conditions are also alternately synthesized in a random AM frequency (RAF) condition. Here each modulation signal is randomly stretched or compressed in time, as selected from a random exponential distribution between 4 Hz and 12 Hz. This range spans the commonly performed vibrato rates (Verfaille et al., 2005). The RAF conditions aim to determine the effect of a common amplitude-modulation rate, or "fundamental frequency," as observed in the analysis of vibrato excerpts in Section 3.1.1.

**Non-modulating (CONTROL).** We include a CONTROL stimulus that has no AM or FM. In the two listening experiments, this stimulus is meant to anchor what is a maximally fused and minimally realistic vibrato.

#### Effect of randomization on stimulus generation

In order to avoid any systematic effect of the randomization in the stimulus conditions, we generate a new batch of stimuli for each participant. Each of the 96 participants is to rate all stimuli eight times; we generate 8448 stimuli in total. It is reasonable to wonder to what extent each variety of randomization produces a particular, consistent effect. Some statistical analyses depend on averaging over condition-specific ratings; it is therefore worth investigating the acoustical variety induced by randomization in each condition.

We use the Timbre Toolbox software package for Matlab (Peeters, Giordano, Susini, Misdariis, & McAdams, 2011) to extract a suite of acoustical descriptors from the stimuli. The toolbox allows for the calculation of over 30 time-varying descriptors, each of which can be summarized by its median or interquartile range (IQR) as measures of central tendency and variability, respectively. Quantile-based metrics are used in lieu of average and variance because they are robust to the noise imparted by acoustical analysis. Given this abundance of values, the authors provide a correlation-based clustering analysis of the summary descriptors, grouping them into ten principal clusters. For our analyses, we use the median and interquartile range of five descriptors, chosen to maximize coverage of the ten principal clusters while maintaining matched median and IQR figures. We cover 7 of the 10 principal clusters identified by the authors. They are: spectral centroid, spectral crest, harmonic odd-to-even ratio, harmonic energy, and spectral flatness. Possible interpretations of these descriptors, in the context of these stimuli, are outlined in Section 3.3.1.

The toolbox calculates descriptors based on two signal representations: the power spectrum and the harmonic model. Descriptors calculated on the power spectrum are based on a STFT representation with 23.2-ms Hamming-windowed frames, hopping at 5.8 ms between frames. Features calculated on a harmonic model are extrapolated from a sinusoidal-model decomposition, built on estimates from 100-ms Blackman-windowed frames, hopping at 25 ms. The model assumes harmonicity, and tracks the amplitudes and

**Table 3.1:** Means for spectral descriptors across all stimulus conditions. **SC** = spectral centroid, **SCr** = spectral crest, **OE** = odd-to-even ratio, **HE** = harmonic energy, **SF** = spectral flatness.

| condition | SC Med | SC IQR | SCr Med | SCr IQR | OE Med | OE IQR | HE Med | HE IQR | SF Med | SF IQR |
|---|---|---|---|---|---|---|---|---|---|---|
| BASIC | 464.713 | 40.833 | 0.230 | 0.012 | 0.826 | 0.390 | 0.054 | 0.006 | 0.022 | 0.039 |
| CONTROL | 438.997 | 0.003 | 0.228 | 0.000 | 0.736 | 0.000 | 0.066 | 0.000 | 0.406 | 0.000 |
| FM_ONLY | 453.321 | 4.676 | 0.217 | 0.003 | 0.895 | 0.007 | 0.065 | 0.001 | 0.136 | 0.119 |
| FROZEN | 466.528 | 43.469 | 0.229 | 0.017 | 0.823 | 0.381 | 0.054 | 0.006 | 0.051 | 0.086 |
| PAM | 452.617 | 0.130 | 0.217 | 0.001 | 0.892 | 0.000 | 0.060 | 0.003 | 0.398 | 0.007 |
| RAG | 454.375 | 94.638 | 0.255 | 0.049 | 0.891 | 0.548 | 0.050 | 0.016 | 0.204 | 0.171 |
| RAG_RAF | 453.826 | 78.631 | 0.252 | 0.050 | 0.892 | 0.428 | 0.044 | 0.012 | 0.222 | 0.140 |
| SHUFFLE | 456.397 | 32.152 | 0.227 | 0.017 | 0.889 | 0.180 | 0.054 | 0.007 | 0.157 | 0.176 |
| SHUFFLE_RAF | 456.700 | 30.664 | 0.226 | 0.016 | 0.889 | 0.151 | 0.050 | 0.006 | 0.108 | 0.104 |
| SIMPLE | 460.135 | 49.372 | 0.230 | 0.022 | 0.891 | 0.252 | 0.054 | 0.008 | 0.145 | 0.168 |
| SIMPLE_RAF | 460.287 | 39.057 | 0.228 | 0.022 | 0.891 | 0.186 | 0.049 | 0.007 | 0.179 | 0.154 |

frequencies of 20 harmonics. The harmonic assumption permits analyses unique to harmonic signals, such as the odd-to-even harmonic ratio. The model correctly estimated the fundamental pitch of each stimulus, suggesting that its results can be interpreted reliably.

To assess the variety of stimuli generated within each stimulus condition, we calculate the mean and standard deviation of each condition-specific median and IQR. These values can be seen in Table 3.1 and Table 3.2. Note that the deterministic synthesis conditions (CONTROL, FM ONLY, BASIC and FROZEN) have standard deviations that are necessarily close to zero. To help understand the meaning of these values, we will focus on the spectral centroid (SC), which correlates strongly to perceived brightness (Peeters et al., 2011).

The SC median is relatively consistent across all conditions, having a range of approximately 439 to 467 Hz (for the CONTROL and FROZEN conditions, respectively). The SHUFFLE condition produces the greatest variety of SC medians, varying on average

**Table 3.2:** Standard deviations for spectral descriptors.

| condition | SC Med | SC IQR | SCr Med | SCr IQR | OE Med | OE IQR | HE Med | HE IQR | SF Med | SF IQR |
|---|---|---|---|---|---|---|---|---|---|---|
| BASIC | 0.206 | 0.112 | 0.000 | 0.000 | 0.000 | 0.001 | 0.012 | 0.001 | 0.014 | 0.020 |
| CONTROL | 0.003 | 0.001 | 0.000 | 0.000 | 0.000 | 0.000 | 0.015 | 0.000 | 0.016 | 0.000 |
| FM_ONLY | 0.006 | 0.005 | 0.000 | 0.000 | 0.000 | 0.000 | 0.016 | 0.000 | 0.067 | 0.091 |
| FROZEN | 0.195 | 0.184 | 0.000 | 0.000 | 0.000 | 0.000 | 0.012 | 0.001 | 0.030 | 0.051 |
| PAM | 0.005 | 0.040 | 0.000 | 0.000 | 0.000 | 0.000 | 0.015 | 0.001 | 0.037 | 0.039 |
| RAG | 4.417 | 48.707 | 0.027 | 0.019 | 0.024 | 0.297 | 0.012 | 0.008 | 0.104 | 0.098 |
| RAG_RAF | 4.747 | 28.475 | 0.015 | 0.017 | 0.022 | 0.164 | 0.010 | 0.005 | 0.117 | 0.096 |
| SHUFFLE | 5.553 | 12.541 | 0.005 | 0.006 | 0.026 | 0.076 | 0.012 | 0.003 | 0.081 | 0.084 |
| SHUFFLE_RAF | 1.765 | 5.106 | 0.002 | 0.002 | 0.010 | 0.034 | 0.011 | 0.002 | 0.081 | 0.072 |
| SIMPLE | 2.128 | 24.343 | 0.008 | 0.008 | 0.011 | 0.117 | 0.013 | 0.004 | 0.073 | 0.088 |
| SIMPLE_RAF | 1.967 | 8.599 | 0.003 | 0.003 | 0.010 | 0.041 | 0.010 | 0.002 | 0.099 | 0.090 |

by 5.57 Hz from the mean. These figures suggest a randomization that is relatively contained. However the IQR, which captures more of the temporal character of the stimulus, tells a very different story.

The RAG condition has the widest standard deviation of SC IQR, at 48.72 Hz about a mean IQR of 94.45 Hz. This statement requires some elaboration. Towards its narrow end, a RAG stimulus can have an SC IQR of $94.45 - 48.72 = 45.73$ Hz, which is close to the mean SC IQR of the BASIC stimulus. Such a stimulus therefore resembles a plausible cello vibrato by this measure. However on average, the RAG stimuli SCs vary almost twice as broadly as this value and can have a within-stimulus IQR of up to $94.45 + 48.72 = 143.17$ Hz and beyond. This indicates that the RAG condition creates a signal whose brightness varies greatly over time, and that there is a great deal of variability in the stimuli generated under this condition.

Other highly variable conditions are SIMPLE, RAG RAF and SHUFFLE conditions, having SC IQR standard deviations of 24.42, 12.60 and 28.39 Hz, respectively. This variability is rather surprising in the SHUFFLE condition, which differs only from plausible vibrato by the relative phase of its amplitude modulations. It may be that in natural cello vibrato, the distribution of instrument resonances allows for a more balanced time-varying brightness.

Auditioning the stimuli at random confirms that some oscillate wildly in apparent brightness, whereas others are relatively stationary. We suspect that the wild oscillations are due to a chance aligning between shuffled partials. Should multiple partials having large modulation gains be aligned by phase, that would certainly contribute to a broad oscillation of the spectral centroid, though it is equally conceivable that a chance contrary situation should flatten a spectral centroid trajectory.

We wish to reiterate that, because we are targeting the necessary conditions for AMV, we are interested in the effect of *type* of randomization rather than the particular realizations of individual randomized stimuli. However, the observed variability of stimuli within the same synthesis condition suggests that one should proceed with caution when reasoning on acoustical descriptors averaged within conditions.

### 3.2.3 Procedure

The two experiment phases follow an identical structure, differing only in their instructions, rating prompt and scale labels. Participants are directed from the Prolific website to our academic server.

The experiment begins with a standard text asking that participants be on a desktop computer, using headphones, and situated in a quiet environment. Participants are then presented an informed consent form. Consenting participants proceed to a sound level check, where they are played six representative stimuli selected by the experimenter. Participants are asked to adjust their audio output to a comfortable listening level, and afterward to leave the level unchanged for the rest of the experiment. They are then presented with rating instructions and brought to a two-trial practice block.

In each trial, participants hear one stimulus, which is played automatically. They are concurrently presented with a prompt and horizontal slider (see below for specific prompts and slider labels). Participants indicate a response by clicking and dragging the slider. They can then click a button marked “Continue,” bringing them to the next trial where the next stimulus plays immediately.

The main experiment takes place over two blocks, allowing for an optional two-minute break. Participants rate all eleven stimulus conditions four times in each block, making for a total of 44 trials per block and 88 ratings total, with eight repetitions per stimulus condition. Stimuli are presented in random order within each block. After the final trial, participants are presented a debriefing message that describes the main goals of the experiment. They are then redirected to the Prolific website. The experiment software collects data only from participants who are successfully returned to Prolific. Participants are paid through the Prolific platform.

In the first phase of the experiment, participants are asked to indicate the *perceived fusion* of the stimulus, using a slider labeled “greater multiplicity (more sources)” and “greater unity (less sources)” on the left and right sides, respectively. The trial prompt reads “How fused is this sound?”. Before the practice block, participants are shown the following instructions:

> **What does fused mean?** All sound is made up of many simple components. Oftentimes they seem to be ‘fused’ together. Components that are fused make a sound that we hear as coming from one source or having a sense of **unity**. For example, the sound of a violin is made up of many fused components, but we hear it as one source. Components that are not fused make a sound that seems cloudy or coming from multiple sources at once. Such sounds have a sense of **multiplicity**. To help remind you of their meaning, the scale has additional labels in brackets: **greater multiplicity (more sources)** and **greater unity (less sources)**.

In the second phase of the experiment, a new set of participants is asked to rate the *perceived realism* of the stimulus using a slider labeled “not realistic at all” and “very realisticm” on the left and right sides, respectively. The trial prompt reads “How realistic is this vibrato?”. Before the practice block, participants are shown the following instructions:

> **What is vibrato?** Vibrato is a smooth, cyclic variation of sound that performers use to add expression to musical notes. Opera singers are well-known for their wide pitch vibrato. However, vibrato does not have to be a variation of pitch.

> Some kinds of instrument vibrato have very little change in pitch at all. You are about to hear sounds that have varying kinds of 'cyclic change.' We want to know which kinds of change sound like vibrato to you. To help remind you of the task, the scale has the following labels: **not realistic at all** to **very realistic**.

Participant ratings are coded on a scale from 0 to 1. In practice, the slider is set to range from zero to the maximum allowable integer in JavaScript ($2^{53} - 1$) and later normalized in analysis.

We ran both phases over two days. Data collection for the first experiment took place over one evening. The average participant run time, from the welcome screen until the rating of the final stimulus, was about 11 minutes, with a median run of 9.5 minutes. The slowest participant took 29.5 minutes. In the second phase of the experiment, the median elapsed time per participant was about 9 minutes.

## 3.3 Results

All statistical analyses were run in Python using the Pingouin statistics library (Vallat, 2018). Interactive notebooks with data analysis and visualizations are available on the project Github page[4]. All ratings were range-normalized within participant. Figure 3.3 shows box-whisker plots of the distributions of ratings for each condition, averaged within-subject over eight repetitions, for the first and second experiment phases. The box edges

[4]https://github.com/maxsolomonhenry/amp_mod

**Table 3.3:** A short summary description of stimulus conditions.

| Condition | Description |
|---|---|
| BASIC | Reconstruction of signal with simplified FM and original AM. |
| FROZEN | Reconstruction of signal with no FM, but having original AM. |
| FM-ONLY | Reconstruction based on average cello spectrum and no AM, but having simplified FM. |
| PAM | Amplitude modulation applied globally (similar to tremolo). No FM. |
| SHUFFLE | Partial reconstruction of the original spectral envelope, where initial phase of partial modulators are randomly shuffled. No FM. |
| SIMPLE | Simplified AM. Partial modulation depths are matched, but replaced with sinusoids. Initial phase of modulators are randomly shuffled. No FM. |
| RAG | Simplified AM where modulation depths are randomly sampled for each partial. No FM. |
| RAF | Additional randomization condition where partial-specific modulations are randomly compressed or stretched in time. Applied to SHUFFLE, SIMPLE and RAG conditions. |
| CONTROL | Non-modulating control signal that has no FM or AM. |

indicate the rating interquartile range, and the whiskers indicate the full range of the response values up to 1.5 times the interquartile range. Dots indicate outliers, determined as those data sitting outside of 1.5 times the interquartile range.

For convenience, we also include a brief summary of all the stimulus types in Table 3.3.

### 3.3.1 Part 1: Fusion

Table 3.4 shows the means and standard deviations of ratings for each stimulus condition. The CONTROL condition has the highest mean fusion rating at 0.83, close to the PAM and FM ONLY condition. The lowest mean rating is the BASIC condition, closely rated to the FROZEN and SIMPLE conditions, having means of 0.28, 0.29, and 0.30, respectively. The

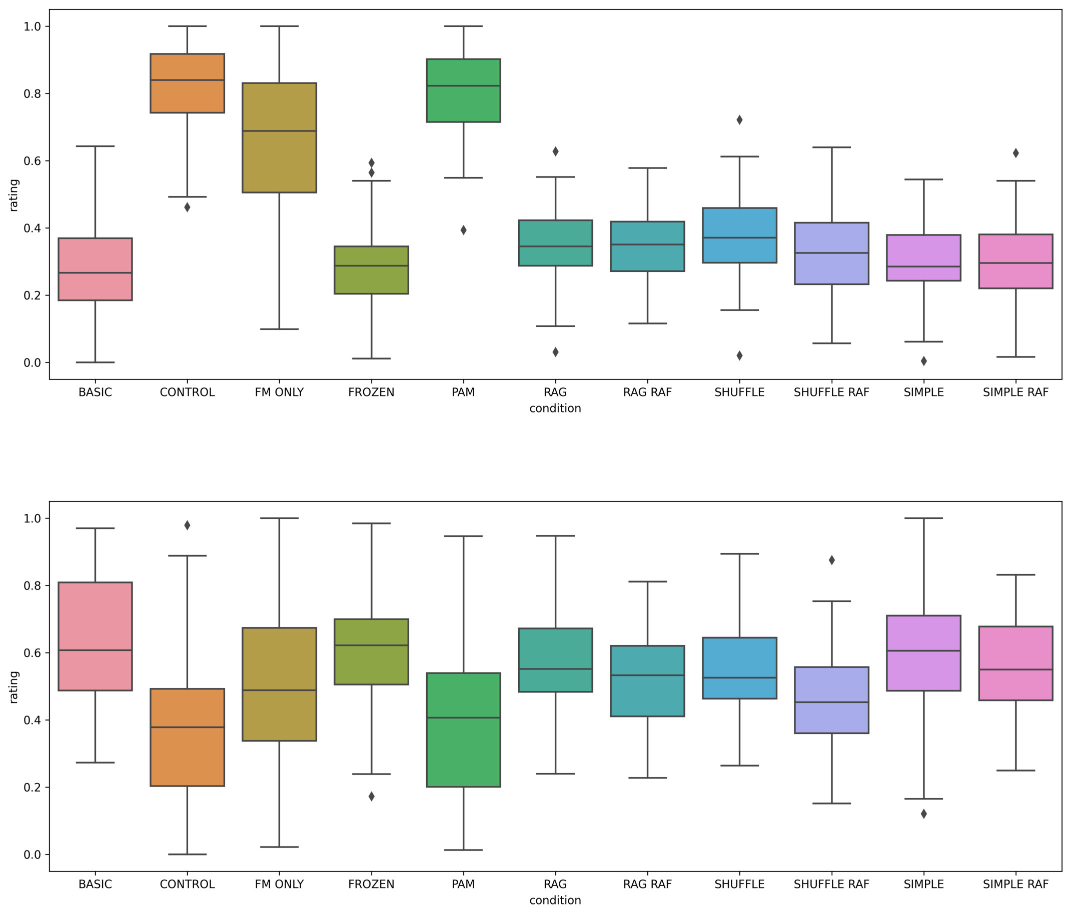

**Figure 3.3:** Box whisker plots for fusion ratings, above, and realism ratings, below. Box edges indicate interquartile range, and whiskers capture up to 1.5 interquartile range. Dots indicate outliers.

**Table 3.4:** Fusion rating means and standard deviations for each stimulus condition.

| condition | mean | std |
|---|---|---|
| BASIC | 0.284 | 0.229 |
| CONTROL | 0.827 | 0.210 |
| FM ONLY | 0.654 | 0.302 |
| FROZEN | 0.289 | 0.213 |
| PAM | 0.802 | 0.226 |
| RAG | 0.351 | 0.232 |
| RAG RAF | 0.339 | 0.239 |
| SHUFFLE | 0.376 | 0.236 |
| SHUFFLE RAF | 0.335 | 0.242 |
| SIMPLE | 0.300 | 0.215 |
| SIMPLE RAF | 0.302 | 0.226 |

other conditions (RAG, both SHUFFLE conditions, and SIMPLE RAF) have generally low mean fusion ratings, all below 0.4. The standard deviations for all conditions are fairly wide, ranging from 0.21 for the CONTROL condition to 0.30 for the FM ONLY condition.

Given the within-condition acoustical variability of the stimuli (see Table 3.2), and the considerable variability of participant fusion ratings, we decided to investigate the rating consistency of each participant individually: while the global picture of response ratings is fairly diffuse, it might be that each participant had consistent, though idiosyncratic, rating strategies.

For each participant, we calculated the standard deviation of the eight responses within each synthesis condition, yielding eleven values that were then averaged across all participants. These values are presented in Table 3.5. The values reveal that ratings were fairly variable within-subject, though less so than across participants. The CONTROL

**Table 3.5:** Standard deviations of fusion ratings within-subject, within-condition, averaged across all participants.

| condition | response |
|---|---|
| CONTROL | 0.147 |
| BASIC | 0.157 |
| PAM | 0.158 |
| FROZEN | 0.159 |
| SIMPLE | 0.179 |
| SIMPLE RAF | 0.180 |
| FM ONLY | 0.193 |
| SHUFFLE RAF | 0.195 |
| SHUFFLE | 0.198 |
| RAG | 0.199 |
| RAG RAF | 0.216 |

condition, which is the most deterministic of the synthesis conditions, has the most focused within-subject response. In order of increasing standard deviation, CONTROL is followed closely by the other deterministic conditions: BASIC, PAM, and FROZEN. More variable ratings come from the randomized stimuli and their RAF counterparts, SIMPLE, SIMPLE RAF, SHUFFLE RAF and SHUFFLE, RAG and RAG RAF. It stands to reason that the most variable rating comes from the RAG RAF condition, which has both randomized gains and partial-specific modulation frequencies. A notable exception to this order is the deterministic FM ONLY condition, which is the fifth most variable stimulus family for fusion ratings.

**Analysis of variance.** We performed a one-way repeated measures ANOVA on stimulus condition with 11 levels and rating means as dependent variable. A Mauchly test indicates that the data violate sphericity ($W = 0.002$, $p < 0.001$), and so we include a Greenhouse-Geisser correction factor ($\varepsilon = 0.348$). The corrected ANOVA indicates a strong statistical difference between condition means ($F(3.48, 170.47) = 133.09, p < 0.001$).

We also conducted a Shapiro-Wilk test for normality on each of the condition's ratings. Of the eleven conditions, three were found to have non-normal ratings: CONTROL ($W = 0.94, p < 0.01$), FM ONLY ($W = 0.94, p < 0.01$) and PAM ($W = 0.94, p < 0.05$). Figure 3.4 shows quantile-quantile plots comparing the CONTROL, FM ONLY and PAM ratings to exemplary normal distributions (data from a normal distribution would sit along the line indicated in red). Broadly speaking, ANOVA assumes normally distributed data. However, given the relative consistency of the variance between conditions and the large sample size, the ANOVA is most likely to be valid (Howell, 2010).

We calculated $\binom{11}{2} = 55$ Wilcoxon signed-rank tests between group means, using Bonferroni-Holm correction to compensate for familywise error rate. Given the relatively large sample size with the violation of normality, we opted for the non-parametric Wilcoxon tests across all pairs. In practice, this approach yields the same statistically different pairs as two-sided t-tests. The results of the Wilcoxon tests can be seen Table 3.6.

The majority of pairs was found to be significantly different (34 out of 55). Among these pairs are what one might expect a priori: for example, CONTROL and PAM, which are both deterministic conditions having no spectral modulation, are significantly more fused than all other conditions, and are not significantly different from one another ($p = 1$). The FM-

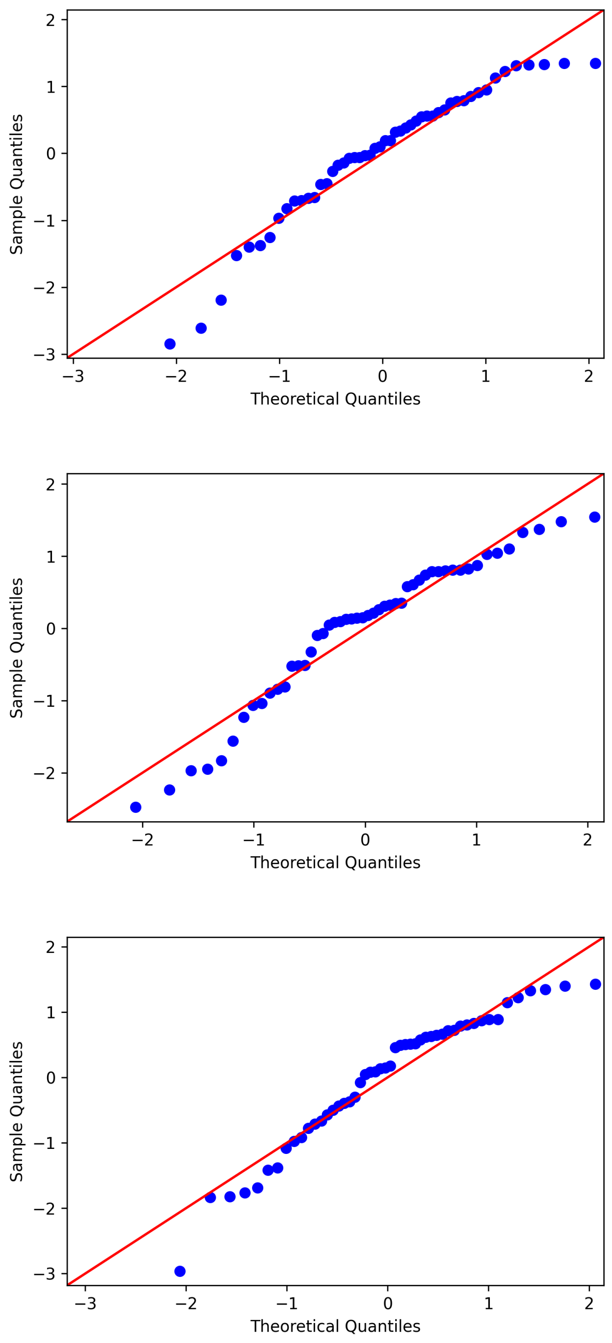

**Figure 3.4:** Quantile-quantile plots for three non-normally distributed stimulus conditions. From top to bottom: CONTROL, FM ONLY, and PAM.

**Table 3.6:** Results of Wilcoxon tests for fusion ratings across all pairs of stimulus conditions.

| A | B | W-val | p-corr |
|---|---|---|---|
| BASIC | CONTROL | 5.0 | <0.001 |
| BASIC | FM_ONLY | 13.0 | <0.001 |
| BASIC | FROZEN | 597.0 | 1.000 |
| BASIC | PAM | 0.0 | <0.001 |
| BASIC | RAG | 272.0 | 0.010 |
| BASIC | RAG_RAF | 403.0 | 0.454 |
| BASIC | SHUFFLE | 210.0 | 0.001 |
| BASIC | SHUFFLE_RAF | 405.0 | 0.454 |
| BASIC | SIMPLE | 524.0 | 1.000 |
| BASIC | SIMPLE_RAF | 505.0 | 1.000 |
| CONTROL | FM_ONLY | 142.0 | <0.001 |
| CONTROL | FROZEN | 1.0 | <0.001 |
| CONTROL | PAM | 511.0 | 1.000 |
| CONTROL | RAG | 6.0 | <0.001 |
| CONTROL | RAG_RAF | 0.0 | <0.001 |
| CONTROL | SHUFFLE | 6.0 | <0.001 |
| CONTROL | SHUFFLE_RAF | 4.0 | <0.001 |
| CONTROL | SIMPLE | 3.0 | <0.001 |
| CONTROL | SIMPLE_RAF | 4.0 | <0.001 |
| FM_ONLY | FROZEN | 45.0 | <0.001 |
| FM_ONLY | PAM | 164.0 | <0.001 |
| FM_ONLY | RAG | 77.0 | <0.001 |
| FM_ONLY | RAG_RAF | 90.0 | <0.001 |
| FM_ONLY | SHUFFLE | 104.0 | <0.001 |
| FM_ONLY | SHUFFLE_RAF | 65.0 | <0.001 |
| FM_ONLY | SIMPLE | 38.0 | <0.001 |
| FM_ONLY | SIMPLE_RAF | 55.0 | <0.001 |
| FROZEN | PAM | 1.0 | <0.001 |
| FROZEN | RAG | 184.0 | <0.001 |
| FROZEN | RAG_RAF | 315.0 | 0.041 |
| FROZEN | SHUFFLE | 136.0 | <0.001 |
| FROZEN | SHUFFLE_RAF | 323.0 | 0.051 |
| FROZEN | SIMPLE | 510.0 | 1.000 |
| FROZEN | SIMPLE_RAF | 541.0 | 1.000 |
| PAM | RAG | 5.0 | <0.001 |
| PAM | RAG_RAF | 1.0 | <0.001 |
| PAM | SHUFFLE | 4.0 | <0.001 |
| PAM | SHUFFLE_RAF | 1.0 | <0.001 |
| PAM | SIMPLE | 0.0 | <0.001 |
| PAM | SIMPLE_RAF | 1.0 | <0.001 |
| RAG | RAG_RAF | 570.0 | 1.000 |
| RAG | SHUFFLE | 421.0 | 0.593 |
| RAG | SHUFFLE_RAF | 552.0 | 1.000 |
| RAG | SIMPLE | 272.0 | 0.010 |
| RAG | SIMPLE_RAF | 367.0 | 0.183 |
| RAG_RAF | SHUFFLE | 437.0 | 0.995 |
| RAG_RAF | SHUFFLE_RAF | 597.0 | 1.000 |
| RAG_RAF | SIMPLE | 403.0 | 0.454 |
| RAG_RAF | SIMPLE_RAF | 453.0 | 0.995 |
| SHUFFLE | SHUFFLE_RAF | 426.0 | 0.625 |
| SHUFFLE | SIMPLE | 181.0 | <0.001 |
| SHUFFLE | SIMPLE_RAF | 188.0 | <0.001 |
| SHUFFLE_RAF | SIMPLE | 459.0 | 0.995 |
| SHUFFLE_RAF | SIMPLE_RAF | 450.0 | 0.995 |
| SIMPLE | SIMPLE_RAF | 595.0 | 1.000 |

ONLY condition occupies a middle ground; it is significantly less fused than the CONTROL and PAM block, and significantly more fused than all of the random conditions and BASIC. The SHUFFLE, RAG, RAG RAF and SHUFFLE RAF conditions form a moderate cluster below FM-ONLY. A slightly lower, overlapping cluster contains all of the RAF conditions plus RAG. The BASIC condition, representing the unadulterated vibrato as rendered by the synthesis engine, forms a cluster with the FROZEN and SIMPLE conditions at the very bottom of the fusion ratings, along with the three RAF conditions.

We note that the BASIC condition is not significantly less fused than two of the milder distortions: FROZEN ($p = 1$), and SIMPLE ($p = 1$). The FROZEN and SIMPLE conditions are also not statistically different ($p = 1$). However, the SHUFFLE condition, which maintains the original modulation fine structure, and the RAG condition, which takes on random modulation gains, were both found to be *more* fused than this lowest cluster.

### Correlation with acoustic descriptors

We sought to explore to what degree ratings of fusion could be explained by acoustic features. We use the same ten features which, as outlined in Section 3.2.2, were determined to best span the acoustic variability described by the set of common spectral and temporal features available in the Timbre Toolbox.

The task of acoustic analysis is particularly challenging in this case, where all the stimuli

**Table 3.7:** Mean, standard deviation, and coefficient of variation for acoustical descriptor values of all stimulus conditions.

| | mean | std | CV |
|---|---|---|---|
| Spectral Centroid Med | 456.173 | 7.520 | 0.016 |
| Spectral Centroid IQR | 37.602 | 34.551 | 0.919 |
| Spectral Crest Med | 0.231 | 0.015 | 0.066 |
| Spectral Crest IQR | 0.019 | 0.018 | 0.952 |
| Odd to Even Ratio Med | 0.865 | 0.050 | 0.058 |
| Odd to Even Ratio IQR | 0.229 | 0.212 | 0.922 |
| Harmonic Energy Med | 0.055 | 0.014 | 0.262 |
| Harmonic Energy IQR | 0.006 | 0.005 | 0.846 |
| Spectral Flatness Med | 0.184 | 0.139 | 0.751 |
| Spectral Flatness IQR | 0.106 | 0.096 | 0.910 |

abide by rules strongly promoting perceptual fusion: they are all harmonic signals, whose partials share common frequency modulation and amplitude onset behaviour (Bregman, 1990). Nevertheless, we suspect that each acoustical descriptor may provide valuable insight. As before, we include both medians and IQRs to contrast the explanatory power of general trend (medians), and change in time (IQRs).

Table 3.7 shows the descriptor means, standard deviations, and coefficients of variation (CV) across all generated stimuli used in the experiment. The CV is the ratio of standard deviation to mean and provides a standardized measure of dispersion to compare variability across the descriptors, which differ considerably in their relative magnitude.

**Spectral centroid.** The spectral centroid (SC) is best known as a strong correlate of perceived brightness, and an often-used explanatory variable for timbre spaces (McAdams, Caclin, & Smith, 2002). It is a measure of central tendency of the spectrum, and therefore captures the necessary context for the interpretation of other spectral descriptors; for example, a particularly extreme SC value (e.g., 50 Hz, or 8000 Hz) would preclude any relevant interpretation of descriptors assuming a harmonic signal (like odd-to-even ratio). More importantly, the SC IQR provides a broad sense of the perceived change in timbral character over time. McAdams (1984) found that the depth of frequency modulation, whether tracing a plausible resonant structure or not, had a marked effect on perceived fusion. Such a modulation would be captured by SC IQR. The SC is the least variable of the time-varying medians in our stimulus pool, having a CV of 0.02 corresponding to a standard deviation of 7.5 Hz. Given that the average stimulus SC IQR is 40.8 Hz, it is doubtful that the relatively small shift in median SC would correlate to any meaningful rating behaviour.

**Spectral crest and spectral flatness.** Spectral crest and spectral flatness are unitless values that range from 0 to 1, both designed as measures of the likely noisiness of a signal. Here they may be interpreted as (reciprocal) measures of spectral peak prominence. A random harmonic stimulus may be modulated such that the relative contributions of its spectral components are fairly even over time, as in the "sizzle" observed by Curtin and Rossing (2010) in violin vibrato, leading to a high spectral flatness and low spectral crest. It is also conceivable that randomization leads to a configuration in which some partials "peak" out above the rest, potentially breaking the illusion of vibrato. It should be noted that while both spectral crest and spectral flatness describe a similar trend in the signal, they are measured on two different underlying representations (power spectrum and harmonic model, respectively), therefore we include both in our analyses as a measure of agreement between representations. In our stimulus pool, spectral flatness median is the most variable median descriptor, having a CV of 0.75 and a standard deviation of 0.14. Spectral flatness IQR is highly disperse with a CV of 0.91; likewise spectral crest IQR has a CV of 0.95.

**Odd-to-even harmonic ratio.** Following a similar logic to spectral crest, it may be that a randomly generated stimulus has an equal prominence of odd-to-even harmonics over time; or it may be that the stimulus, by chance, "lilts" heavily between odd and even harmonics. It is conceivable that such a sound would break the illusion of vibrato, and potentially diminish the perceived fusion of the sound, much like the aforementioned effect of frequency modulation. In our stimuli, odd-to-even ratio IQR is highly dispersed with a CV of 0.92.

**Harmonic energy.** Harmonic energy (here, the total energy as measure by the harmonic model) captures the signal's dynamic behaviour over time. The aforementioned "shimmer" effect (Curtin & Rossing, 2010) is dependent on a fairly even influence of partials on the global amplitude envelope; therefore, one would expect a signal having a low IQR of this metric to be more likely perceived as a plausible vibrato sound. Among the generated stimuli, harmonic energy IQR has a fairly high CV of 0.85.

**Table 3.8:** Results of stepwise linear regression of acoustical descriptors on fusion ratings. Estimate weights are standardized, and displayed with standard error and p-value.

| | estimate | SE | p |
|---|---|---|---|
| Intercept | 0.000 | 0.081 | 0.999 |
| Harmonic Energy Med | 0.595 | 0.092 | <0.001 |
| Spectral Flatness Med | 0.575 | 0.092 | <0.001 |

**Stepwise linear regression.** To identify the possible acoustic characteristics associated with fusion rating choices, we calculated a stepwise linear regression, taking the condition-averaged acoustic descriptors as independent variables, and condition-averaged fusion rating as the dependent variable. The model made an excellent fit to the data (adjusted $r^2 = 0.93$), and selected two descriptors: median harmonic energy ($p < 0.001$) and median spectral flatness ($p < 0.001$). Both values are positively correlated with fusion ratings and make roughly equal contributions (see Table 3.8).

### 3.3.2 Part 2: Vibrato Realism

Table 3.9 shows the means and standard deviations of *realism* ratings by stimulus condition. The rating standard deviations are wide here, even wider than for the fusion ratings; however, the rating means tell a logical story. The BASIC condition, which by design has the highest fidelity to the original cello vibrato, has the highest mean realism rating (which is itself rather moderate, at 0.636), followed by FROZEN and SIMPLE conditions. The lowest rating is the PAM condition, followed closely by the CONTROL condition.

**Table 3.9:** Realism rating means and standard deviations for each stimulus condition.

| condition | mean | std |
|---|---|---|
| BASIC | 0.636 | 0.281 |
| CONTROL | 0.383 | 0.320 |
| FM_ONLY | 0.495 | 0.313 |
| FROZEN | 0.594 | 0.266 |
| PAM | 0.380 | 0.313 |
| RAG | 0.572 | 0.258 |
| RAG_RAF | 0.514 | 0.274 |
| SHUFFLE | 0.555 | 0.264 |
| SHUFFLE_RAF | 0.462 | 0.282 |
| SIMPLE | 0.594 | 0.274 |
| SIMPLE_RAF | 0.560 | 0.263 |

The within-subject standard deviations of responses are presented in Table 3.10. While the realism ratings vary greatly across conditions, the values stand to reason. The least variable ratings come from the deterministic CONTROL condition. The most variable ratings come from the RAG RAF condition, which is the most random stimulus condition. We note that within-subject variation is more focused than the variation across participants.

We computed a one-way repeated measures ANOVA on stimulus condition with 11 levels and realism-rating means as a dependent variable. Once again, a Mauchly test reveals that the data violate sphericity, and so we include a Greenhouse-Geisser correction factor ($\varepsilon = 0.221$). The corrected ANOVA indicates a statistical difference between condition means ($F(2.21, 108.29) = 10.90, p < 0.001$). Multiple Shapiro-Wilk tests determine that, in this case, the ratings for each condition are normally distributed. The means are compared

**Table 3.10:** Standard deviations of realism ratings within-subject, within-condition, averaged across all participants.

| condition | response |
|---|---|
| CONTROL | 0.185 |
| FROZEN | 0.192 |
| PAM | 0.202 |
| BASIC | 0.204 |
| SIMPLE_RAF | 0.208 |
| FM_ONLY | 0.208 |
| SIMPLE | 0.211 |
| RAG | 0.212 |
| SHUFFLE | 0.220 |
| SHUFFLE_RAF | 0.224 |
| RAG_RAF | 0.226 |

with 55 t-tests whose p-values are adjusted with Bonferroni-Holm correction. The results of the t-tests can be found in Table 3.11.

In contrast to the fusion ratings, the minority of pairs was found to be statistically different in ratings of vibrato realism (23 out of 55). Furthermore, the t-tests exhibit some rather confounding irregularities, which are discussed below. As expected, the BASIC condition was found to be more realistic than the CONTROL ($p < 0.05$) and PAM ($p < 0.05$) conditions. CONTROL and PAM were themselves confused ($p = 1$), forming a cluster at the bottom of the realism ratings. Encouragingly, CONTROL, which contains no modulation and should therefore be rated low for realism, was not found to be statistically different from the highly randomized RAG RAF ($p = 0.26$) and SHUFFLE RAF ($p = 1$) conditions. Likewise PAM, which also contains no spectral modulation and should be rated low, is not statistically different from the highly scrambled RAG RAF ($p = 0.18$) and

**Table 3.11:** Results of t-tests for realism ratings across all pairs of stimulus conditions.

| A | B | T | p-corr |
|---|---|---|---|
| BASIC | CONTROL | 4.703 | 0.001 |
| BASIC | FM_ONLY | 3.119 | 0.095 |
| BASIC | FROZEN | 2.197 | 0.714 |
| BASIC | PAM | 4.728 | 0.001 |
| BASIC | RAG | 2.921 | 0.158 |
| BASIC | RAG_RAF | 4.388 | 0.003 |
| BASIC | SHUFFLE | 3.604 | 0.030 |
| BASIC | SHUFFLE_RAF | 5.366 | <0.001 |
| BASIC | SIMPLE | 1.802 | 1.000 |
| BASIC | SIMPLE_RAF | 3.262 | 0.068 |
| CONTROL | FM_ONLY | -3.679 | 0.025 |
| CONTROL | FROZEN | -3.768 | 0.020 |
| CONTROL | PAM | 0.194 | 1.000 |
| CONTROL | RAG | -4.089 | 0.008 |
| CONTROL | RAG_RAF | -2.708 | 0.258 |
| CONTROL | SHUFFLE | -3.514 | 0.037 |
| CONTROL | SHUFFLE_RAF | -1.555 | 1.000 |
| CONTROL | SIMPLE | -4.238 | 0.005 |
| CONTROL | SIMPLE_RAF | -3.417 | 0.046 |
| FM_ONLY | FROZEN | -2.097 | 0.788 |
| FM_ONLY | PAM | 3.554 | 0.033 |
| FM_ONLY | RAG | -1.863 | 1.000 |
| FM_ONLY | RAG_RAF | -0.436 | 1.000 |
| FM_ONLY | SHUFFLE | -1.315 | 1.000 |
| FM_ONLY | SHUFFLE_RAF | 0.729 | 1.000 |
| FM_ONLY | SIMPLE | -2.207 | 0.714 |
| FM_ONLY | SIMPLE_RAF | -1.347 | 1.000 |
| FROZEN | PAM | 3.842 | 0.016 |
| FROZEN | RAG | 1.166 | 1.000 |
| FROZEN | RAG_RAF | 3.404 | 0.046 |
| FROZEN | SHUFFLE | 2.122 | 0.788 |
| FROZEN | SHUFFLE_RAF | 5.251 | <0.001 |
| FROZEN | SIMPLE | 0.007 | 1.000 |
| FROZEN | SIMPLE_RAF | 2.106 | 0.788 |
| PAM | RAG | -4.181 | 0.006 |
| PAM | RAG_RAF | -2.860 | 0.179 |
| PAM | SHUFFLE | -3.630 | 0.028 |
| PAM | SHUFFLE_RAF | -1.658 | 1.000 |
| PAM | SIMPLE | -4.270 | 0.005 |
| PAM | SIMPLE_RAF | -3.501 | 0.037 |
| RAG | RAG_RAF | 2.326 | 0.590 |
| RAG | SHUFFLE | 0.817 | 1.000 |
| RAG | SHUFFLE_RAF | 3.934 | 0.012 |
| RAG | SIMPLE | -1.448 | 1.000 |
| RAG | SIMPLE_RAF | 0.573 | 1.000 |
| RAG_RAF | SHUFFLE | -2.298 | 0.604 |
| RAG_RAF | SHUFFLE_RAF | 2.537 | 0.383 |
| RAG_RAF | SIMPLE | -3.244 | 0.069 |
| RAG_RAF | SIMPLE_RAF | -2.451 | 0.455 |
| SHUFFLE | SHUFFLE_RAF | 4.641 | 0.002 |
| SHUFFLE | SIMPLE | -1.834 | 1.000 |
| SHUFFLE | SIMPLE_RAF | -0.367 | 1.000 |
| SHUFFLE_RAF | SIMPLE | -4.606 | 0.002 |
| SHUFFLE_RAF | SIMPLE_RAF | -5.350 | <0.001 |
| SIMPLE | SIMPLE_RAF | 1.717 | 1.000 |

**Table 3.12:** Results of stepwise linear regression of acoustical descriptors on realism ratings.

| | estimate | SE | p |
|---|---|---|---|
| Intercept | 0.000 | 0.142 | 0.999 |
| Odd to Even Ratio IQR | 0.413 | 0.171 | 0.042 |
| Spectral Flatness Med | -0.628 | 0.171 | 0.006 |

SHUFFLE RAF($p = 1$) conditions.

The BASIC condition was not considered more realistic than the FM-ONLY condition. Tellingly, it was also not considered more realistic than three mild distortions, FROZEN, SIMPLE, RAG, though it was considered more realistic than the SHUFFLE condition ($p < 0.05$). Rated below these, the randomized conditions (excluding SHUFFLE RAF) form an intermediate cluster of realism ratings, this cluster sitting above the CONTROL/PAM block.

**Irregularities.** A number of confusing irregularities are present in the t-test pairs. PAM is not different from SIMPLE RAF, but it is different from its immediate neighbours in order of mean rating: SIMPLE RAF and RAG. Likewise, PAM is not different from RAG RAF, but it is different from its surrounding conditions, SHUFFLE and FM-ONLY. The BASIC condition is not significantly different from FM-ONLY, but it is different from its own neighbours, RAG RAF and SHUFFLE RAF. CONTROL is also not rated differently from RAG RAF, though it is different from the surrounding FM-ONLY and SHUFFLE conditions. We note among these poorly behaved pairs the overwhelming presence of RAF stimuli, which are highly acoustically variable.

### Correlation with acoustical descriptors

As with the fusion ratings, we performed a regression analysis using the ten selected acoustic descriptors as potential explanatory variables.

**Stepwise linear regression.** Once more, we fit a stepwise linear regression to the condition-averaged acoustic features and response data. The model has moderately less explanatory power (adjusted $r^2 = 0.78$). The algorithm selected two features: the median spectral flatness ($p < 0.001$), which is negatively emphasized, and the novel odd-to-even ratio IQR ($p < 0.05$), which is positively emphasized (see Table 3.12). The relative contribution of median spectral flatness is about 50% greater than that of odd-to-even ratio IQR.

## 3.4 Discussion

Over two separate listening experiments, participants rated the perceived fusion and vibrato realism of 11 stimulus conditions. Each condition, through targeted distortions of the time-varying spectral envelope of a true cello signal, is designed to remove a possible necessary condition for hearing amplitude-modulation vibrato (AMV). Broadly speaking, we challenged the role of a plausible resonant structure in the phenomenon, speculating that a number of distortions (e.g., scrambling relative modulator phase, assigning arbitrary modulation depths) would have little impact on the strength of percept. We also hypothesized that a common modulation rate, as observed by Mellody and Wakefield (2000) in violin vibrato, would lead to a greater sense of perceived realism and fusion than a randomized modulation rate.

In both the fusion and realism ratings, the BASIC stimulus, in essence a reconstruction of the cello excerpt, was undifferentiated from many of the targeted distortions. By some measure, this can be considered a confirmation of our first hypothesis. We first discuss this result, and its implications for the role of resonant structure in AMV.

Next we discuss the fusion ratings, which paint a counterintuitive picture, as the BASIC stimulus was rated as minimally fused. We consider this result in terms of the polysemous nature of fusion as a terminology, pivoting on recent parallel work in musical texture perception.

We next discuss the realism ratings which, taken as a whole, are almost entirely reciprocal to the fusion ratings. We discuss this relationship in terms of median spectral flatness, which was given considerable positive emphasis on the fusion ratings, and negative emphasis on the realism ratings, by two linear regression models.

Finally, we turn our attention to the RAF stimuli, which exhibited anomalous, at times paradoxical, statistical behaviour in our analyses. We discuss this result in terms of the differential role of phase in modulation-rate-based masking and modulation-based grouping principles.

### 3.4.1 Assumptions about resonant structure

Both the fusion and realism ratings cast doubt on the role of plausible resonant structure in the AMV percept. Each of the random stimulus conditions scrambles the spectral envelope, and thereby the apparent resonant structure, of the vibrato signal in some way. Each confusion of a randomized stimulus with a faithful one (i.e., the BASIC condition) is telling.

BASIC and FROZEN stimuli were regularly confused with the phase-shuffled SIMPLE condition. The SIMPLE stimulus scrambles the phase and removes all fine structure of the original time-varying spectral envelope, casting into doubt the necessity of modulation relative phase and fine structure in the AMV percept.

The RAG condition was also not significantly differentiated from the BASIC condition in either fusion or realism ratings. The RAG condition was designed as an incremental distortion of the SIMPLE condition, in which the partials modulate at a random depth within 0 – 10 dBs, behaving according to the simple peak prominence rules implied by Mathews and Kohut (1973). These criteria may be well captured by the spectral flatness descriptor, a point we discuss below. This result provides further evidence that plausible resonant structure is unnecessary for the AMV percept; however, the partials may have to abide by a certain peak prominence.

The SHUFFLE condition, which maintains the fine modulation structure but shuffles their relative initial phase, was rated as more fused than the SIMPLE condition. The mechanism for this finding is uncertain. If AMV is a function of a common modulation rate, shuffling the relative phase of complex modulation signals may obscure this apparent correlation between partials. In this sense, the SHUFFLE condition may be heard as closer to the RAF conditions, whose effect is discussed below in terms of perceived homogeneity.

### 3.4.2 Polysemous nature of perceptual fusion

The fusion ratings present the rather unintuitive result that the BASIC condition, designed to be the most similar to the original cello excerpt, is considered as the least fused of all stimulus conditions. It may be, as McAdams (1984) found, that a modulated signal is perceived as having greater "multiplicity," despite abiding by the constellation of perceptual grouping principles (e.g., harmonicity, common onset and offset) that would suggest otherwise. This interpretation is slightly unsatisfying, however, considering the key role of spectral fusion in forming an auditory image. A spectral component should logically either be ascribed to a given source or not; this is the notion described by Bregman as belongingness (Bregman, 1990). Seen in this light, tasking a participant to rate a sound along a continuous (i.e., not binary) scale of fusion may be somewhat ill-posed.

Let us consider the striking inverse relationship between fusion and vibrato realism ratings (see Figure 3.3). At first glance, one would expect vibrato realism to be commensurate with fused sounds; after all, realistic vibrato tends to issue from a singular, fused musical source. It may be that participants, being unfamiliar with the synthetic nature of the stimuli, did not understand how to interpret the questions of fusion and realism, effectively scrambling their results; however this explanation does not account for the inverse relationship between ratings. We suggest that in the case of a rating task—wherein the participant indicates a response on a continuous scale from 0 to 1—the notion of fusion takes on another, parallel meaning.

Noble, Thoret, Henry, and McAdams (2020) conducted a listening experiment on music textures, in which participants were asked to rate excerpts of sound mass music on scales of perceived fusion, complexity and homogeneity. The authors found a strong negative correlation between perceived fusion and complexity, as well as a strong positive correlation between fusion and homogeneity. This experiment provides an interesting comparison to our work. The pattern that emerged from these three, presumedly orthogonal, scales highlights the polysemous nature of semantic descriptors. When presented with a word, a scale, and a sound, a lay participant may first search the word for all of its possible meanings, and rate according to the specific meaning that describes, to them, the greatest variation in the corpus of sounds. A similar effect has been shown in timbre dissimilarity ratings, where ratings are sensitive to the corpus of sounds being presented (Thoret, Caramiaux, Depalle, & McAdams, 2021).

Auditory texture presents an interesting challenge to fusion ratings; the sound of rainfall, for example, is made up of multiple independent acoustic events, individual rain drops, each of which is “fused” within itself, though necessarily separate from other rain drops. Taken as a whole, such a texture could not be considered fused in the strict sense of the word; but it could certainly be considered more coherent than a collection of arbitrary, disparate events. Such textures may certainly be considered homogeneous, for example, a notion that correlates demonstrably with fusion. One can even imagine more extreme cases: for example, a sparse drizzle in which the raindrops are perceived collectively as a rhythmic pattern of sorts, or a particularly strong rainfall that resembles white noise, which may indeed be fused in the strictest sense. Regardless, auditory texture presents a continuity of scenarios in which sound imparts a vague sense of cohesion. This effect of textural cohesion has been compellingly argued as, at least in part, a function of amplitude modulation coherence (McDermott & Simoncelli, 2011; McWalter & Dau, 2017).

We propose that our participants, not finding any substantial variation in the perceived unity or multiplicity of sounds, rated them instead by perceived homogeneity. This line of reasoning, while speculative, is supported by discussions with colleagues who had piloted the experiment. Those colleagues expressed both identifying the BASIC condition as coming from a single source (i.e., as fused), and at the same time wanting to indicate a meaningful difference between the BASIC condition and the less-modulating, more homogeneous CONTROL and PAM conditions. Seen in this light, the middling fusion ratings of the RAF stimuli, which sound both uncoordinated—by their partial-specific modulation rates—and yet coherent—by their harmonic structure and common onset and offset—make perfect sense. Compared to the BASIC, FROZEN, and SIMPLE conditions they are much more homogeneous, making an inscrutable mass of sound. By the same logic, they are less homogeneous than the PAM and CONTROL conditions, whose unmodulating partials hold them strongly together.

### 3.4.3 Acoustical analysis

Perhaps the most striking feature in our acoustical analyses is the selection, by two respective linear regression models, of median spectral flatness for both fusion and realism ratings. More striking still is their reciprocal weighting. The fusion model positively weights spectral flatness; the realism model negatively weights spectral flatness, and both in a fairly even measure: their standardized beta values are 0.58 and −0.63, respectively. This pattern echoes the curious reciprocal relation of mean fusion and realism ratings, which holds true almost condition-for-condition (see Figure 3.3).

Multiple authors describe the globally steady amplitude envelope in vibrato, in spite of the comparatively wild amplitude-behaviour of its partials (Curtin & Rossing, 2010; Gough, 2005; Maher & Beauchamp, 1990; Mathews & Kohut, 1973). Of particular note is the work by Mathews and Kohut (1973), who endeavoured to construct a plausible violin resonance on top of a dry bowed-excitation signal. The authors found that, within certain very permissive limits, it was possible to create the illusion of a true violin body.

The authors suggest that even a random or exponential distribution would suffice for the illusion, supposing the following conditions were met: (1) the resonances must be sufficiently steep almost everywhere across the spectrum, such that the magnitude of the derivative of the response curve is generally large, and (2) that the spectral peaks must be close enough together that the intervening valleys do not dip below about -15 dB from the maxima. These conditions, when taken together, also forgo an envelope having any strong anti-resonances, i.e., zeroes.

The former condition ensures the desired “liveliness” quality of a violin, which is otherwise reported by players to be unresponsive. The second condition puts an effective upper limit on the peakiness of the resonances. In their listening tests, the authors found that too much of a resonant peak produces a hollow and uneven sound.

At its maximum value of 1, spectral flatness describes a perfectly flat spectrum, such as in white noise. A value close to 0 is maximally peaky and correlates to a signal having very strong resonances. This descriptor provides for a strikingly meaningful acoustical interpretation of the experimental results. We will proceed by describing the stimulus conditions at the extrema of both scales, and then move to the more nuanced cases in the middle.

We refer the reader to the summary values in Table 3.1. On average, the median spectral flatness is maximal for the CONTROL (0.41) and PAM (0.40) conditions, and minimal for the BASIC (0.01) and FROZEN (0.03) conditions. These values can be interpreted in terms of Mathews’ and Kohut’s criteria. A maximally flat spectrum violates the first condition for realistic vibrato: it is not sufficiently peaky to be considered lively. Consequently, such sounds are rated as low for vibrato realism.

By the same token, a sound in which individual partials “stick out” over the others may, however mildly, violate perceptual grouping cues that favour coherently modulated spectral components. As a result, signals having a low spectral flatness may be considered as less fused than other sounds. As in the aforementioned McAdams (1984) investigation regarding the effect of frequency modulation on perceived fusion, such a modulation may, however unexpectedly, imply a sense of multiplicity in a sound that would otherwise be heard as coming from one source. Taken alongside the discussion of polysemy above, this account makes for an alternative or complementary explanation of fusion ratings. The latter argument being anchored in an acoustical correlation, rather than a speculative line of reasoning, makes it relatively more appealing.

### 3.4.4 Effect of amplitude modulation rate

The RAF stimuli were designed to target the hypothesis that coherent amplitude modulation rate binds the (otherwise disparate) partials together, contributing a sense of fusion and realism to the AMV percept. Unfortunately, the results concerning these stimuli are inconclusive. The RAF mean realism ratings are, in a number of cases, statistically different from neighbouring conditions (i.e., conditions having a similar mean rating), though not statistically different from conditions having a more distant mean rating. We attribute this pattern to two underlying causes: one of experimental design and one more theoretical in nature.

In order to disrupt the effect of a common fundamental modulation rate, the RAF stimuli scramble partial-specific rates by selecting from a random distribution of rate values (between 4 and 12 Hz). This process results in the most acoustically varied stimuli of all the synthesis conditions (see Table 3.2 in Method).

**Accidental harmonicity.** In practice, the random selection of modulation rates may result in many accidental harmonic relationships between modulators. For example, the rates 6, 9 and 12 Hz are all harmonics of 3 Hz; the rates, 4, 6, 8, 10, 12, and 14 Hz are harmonics of 2 Hz. In pitched harmonic sounds, the ear is fairly permissive to the mistuning of individual partials (Darwin & Gardner, 1986). The frequency of an overtone may deviate substantially from an integer multiple of a fundamental frequency, and still be perceptually fused into the harmonic sound. If the same permissiveness holds true for *modulation rate* sensitivity, e.g., if a 10.1-Hz modulation could be perceptually folded into a 5 Hz vibrato fundamental, then many of the chance combinations of the RAF modulation rates would be perceived as having a common fundamental rate.

This scenario annuls the desired effect of the RAF condition. If the RAG RAF and SIMPLE RAF stimuli were indeed heard as having a common fundamental rate, their ratings would resemble those of the RAG and SIMPLE conditions; which is indeed the case, statistically so, in both fusion and realism ratings. In future studies, care must be taken to properly constrain this randomization, such that accidental modulation "harmonicity" is minimized.

**Modulation rate sensitivity.** Moore et al. (1991) describe an across-channel masking effect of amplitude modulation frequency. The experiment found that when target and distractor signals were amplitude-modulated at similar rates, it became comparatively more difficult to detect small changes of modulation depth in the target signal. Adjusting the modulation rate of the distractor signal, to either much slower or much faster than the target modulation rate, attenuated this masking effect. The pattern of interference, drawn as a function of difference in modulator rate, resembles a bandpass characteristic in a presumed internal "modulation-rate filter." Adjusting the relative phases of the modulators had no impact on this effect. Moore et al. refer to this phenomenon as across-channel masking. However, they make a point to differentiate this finding from the notion of modulation-based fusion, which is undeniably phase-dependent. Indeed, the bandwidth of phase-agnostic modulation sensitivity is far too wide to accommodate any meaningful grouping: the 10 Hz modulated target signal was susceptible to interference by distractor modulation rates ranging from 5 to 15 Hz (though the effect was maximal with a 10 Hz distractor). This wide modulation rate tuning has been corroborated by other models (Dau et al., 1997).

# Chapter 4

# Conclusion

In the current study, we assessed the plausible necessary conditions for amplitude modulation induced vibrato (AMV). These conditions were gleaned from a corpus of literature dedicated to vibrato, the acoustical characterisation of resonant instruments, and the perceptual influence of the spectral envelope. We fashioned a synthesis engine that generates modulating signals read from an array of time-varying spectral envelopes, and a parallel engine for applying targeted distortions to the spectral envelope. Based on an exemplary cello vibrato signal having a strong AMV, eleven distortion conditions were devised, each targeting a possible necessary condition for the perception of vibrato. Stimuli generated in these conditions were rated in two online listening experiments for their perceived fusion and realism.

Two of the distortion conditions, having randomized initial modulation phase and partial-specific modulation depths, were regularly confused with an unadulterated vibrato signal for both fusion and realism ratings. This finding suggests that a plausible resonant structure is not necessary to generate a sense of vibrato without frequency modulation. Stimuli having individually randomized partial modulation rates were rated as intermediate for both fusion and realism. We suggested that such stimuli are rated according to the limited context of the experimental task, and considered in terms of auditory texture, evoking a parallel interpretation of the term fusion.

Fusion and realism ratings were found to vary in opposite directions across nearly all stimulus conditions. This phenomenon was well-captured by the spectral-flatness metric, which correlates positively to perceived fusion, and negatively to vibrato realism. We suggest that spectral flatness embodies the conditions outlined by Mathews and Kohut (1973) in the theory of resonant enhancement of tone quality (for a rich timbral modulation, an instrument must be sufficiently resonant, though without having severe spectral peaks).

From casual listening and discussion with other experimenters, it is clear that the AMV percept can be easily mistaken for a mild frequency modulation. To this end, many questions remain: to what extent do listeners hear a change in frequency when there is none; what is the magnitude of the perceived change in frequency; and what are the perceptual mechanisms that lead to it; to what extent is illusory pitch modulation limited to oscillatory modulation?

Broadly speaking, modulation-rate-based phenomena necessitate an across-channel communication that hints at higher-order perceptual processes. It is known that modulation rates play a role in the perception of auditory texture (McDermott & Simoncelli, 2011; McWalter & Dau, 2017). This line of investigation, however, has not seen much attention in the musical domain. To what extent are musical sounds perceived as textural, as mediated by amplitude modulation rate? For example, is a cello *section* perceived as a texture; i.e., is it stored in the brain as statistical summary information, including band-specific AM rates? At what point is a sound source considered to have multiplicity, and how is this process mediated by amplitude modulation phase, as in the "chorus effect" (Kahlin & Ternstrom, 1999)? Some listeners noted that the RAF stimuli resemble string ensembles. To what extent are ensemble sounds characterized more broadly by their modulation characteristics in the brain? And to what extent are such sounds resilient to the distortions explored here?

One can certainly imagine reproducing the preceding experiment under more ideal conditions. Slight adjustments may be made to the RAF synthesis to avoid accidental harmonicity among modulators. Taking the experiment offline, having access to a common, controlled listening environment, and being able to communicate in real-time with participants, may well lead to qualitatively superior results; particularly in the realism ratings, where the diffuse data suggest participants many have been unsure about the directions. Would music psychology students rate fusion as perceived homogeneity, as our pilot studies suggest? To prevent this polysemy, a more targeted experimental paradigm may be used to collect fusion ratings; for example, a forced-choice experiment.

Taken as a whole, our study sheds light on the complex inner workings of a musical note, and the sensitivity of the ear to the complex independent behaviour of partials that conspire to create the internal structure of a beautiful tone.

# References

Bogert, B. P. (1963). The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking. *Time Series Analysis*, 209–243.

Bregman, A. S. (1990). *Auditory Scene Analysis: The Perceptual Organization of Sound.* Cambridge, Mass: MIT Press.

Bregman, A. S., Levitan, R., & Liao, C. (1990). Fusion of auditory components: Effects of the frequency of amplitude modulation. *Perception & Psychophysics*, *47*(1), 68–73.

Cohen, L. (1966). Generalized phase-space distribution functions. *Journal of Mathematical Physics*, *7*(5), 781–786. doi: 10.1063/1.1931206

Curtin, J., & Rossing, T. D. (2010). Violin. In T. D. Rossing (Ed.), *The Science of String Instruments* (pp. 209–244). New York, NY: Springer New York. doi: 10.1007/978-1-4419-7110-4_13

Darwin, C. J., & Gardner, R. B. (1986). Mistuning a harmonic of a vowel: Grouping and phase effects on vowel quality. *Journal of the Acoustical Society of America*, *79*(3), 838–845. doi: 10.1121/1.393474

Dau, T., Kollmeier, B., & Kohlrausch, A. (1997). Modeling auditory processing of amplitude modulation. II. Spectral and temporal integration. *Journal of the Acoustical Society of America*, *102*(5), 2906–2919. doi: 10.1121/1.420345

Flandrin, P. (1999). The time-frequency problem. In G. Longo & B. Picinbono (Eds.), *Time-Frequency/Time-Scale Analysis* (p. 39). San Diego: Academic Press.

Fletcher, H., & Sanders, L. C. (1967). Quality of violin vibrato tones. *Journal of the Acoustical Society of America*, *41*(6), 1534–1544. doi: 10.1121/1.1910516

Gabor, D. (1947). Theory of communication. *Journal of the Institution of Electrical Engineers - Part I: General*, *94*(73), 429–457. doi: 10.1049/ji-1.1947.0015

Gough, C. E. (2005). Measurement, modelling and synthesis of violin vibrato sounds. *Acta Acustica United With Acustica*, *91*, 229–240.

Griffin, D. W., & Lim, J. S. (1985). A new model-based speech analysis/synthesis system. In *ICASSP '85. IEEE International Conference on Acoustics, Speech, and Signal Processing* (Vol. 10, pp. 513–516). Tampa, FL, USA: Institute of Electrical and Electronics Engineers. doi: 10.1109/ICASSP.1985.1168385

Howell, D. C. (2010). *Statistical methods for psychology* (7th ed.). Belmont, CA: Thomson Wadsworth.

Kahlin, D., & Ternstrom, S. (1999). The chorus effect revisited-experiments in frequency-domain analysis and simulation of ensemble sounds. In *Proceedings 25th EUROMICRO Conference. Informatics: Theory and Practice for the New Millennium* (Vol. 2, p. 75-

80). doi: 10.1109/EURMIC.1999.794765

Kawahara, H., Morise, M., Takahashi, T., Nisimura, R., Irino, T., & Banno, H. (2008). Tandem-STRAIGHT: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, F0, and aperiodicity estimation. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 3933–3936. (ISSN: 1520-6149) doi: 10.1109/ICASSP.2008.4518514

Maher, R., & Beauchamp, J. (1990). An investigation of vocal vibrato for synthesis. *Applied Acoustics*, *30*(2-3), 219–245.

Makhoul, J. (1975). Linear prediction: A tutorial review. *Proceedings of the IEEE*, *63*, 561–580.

Marin, C. M. H., & McAdams, S. (1991). Segregation of concurrent sounds. II: Effects of spectral envelope tracing, frequency modulation coherence, and frequency modulation width. *Journal of the Acoustical Society of America*, *89*(1), 341–351. doi: 10.1121/1.400469

Mathews, M. V., & Kohut, J. (1973). Electronic simulation of violin resonances. *Journal of the Acoustical Society of America*, *53*(6), 1620–1626. doi: 10.1121/1.1913511

McAdams, S. (1984). *Spectral fusion, spectral parsing and the formation of auditory images* (Doctoral dissertation). Stanford University.

McAdams, S. (1989). Segregation of concurrent sounds. I: Effects of frequency modulation coherence. *Journal of the Acoustical Society of America*, *86*(6), 2148–2159. doi:

10.1121/1.398475

McAdams, S., Beauchamp, J. W., & Meneguzzi, S. (1999). Discrimination of musical instrument sounds resynthesized with simplified spectrotemporal parameters. *Journal of the Acoustical Society of America*, *105*(2), 882-897.

McAdams, S., Caclin, A., & Smith, B. K. (2002). A confirmatory analysis of four acoustic correlates of timbre space. *Journal of the Acoustical Society of America*, *112*(5), 2239. (2239) doi: 10.1121/1.4778883

McAdams, S., & Rodet, X. (1988). The role of FM-induced AM in dynamic spectral profile analysis. In H. Duifhuis, J. W. Horst, & H. Wit (Eds.), *Basic Issues in Hearing* (pp. 359–369). London: Academic Press.

McAulay, R., & Quatieri, T. (1986). Speech analysis/synthesis based on a sinusoidal representation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *34*(4), 744–754. doi: 10.1109/TASSP.1986.1164910

McDermott, J., & Simoncelli, E. (2011). Sound texture perception via statistics of the auditory periphery: Evidence from sound synthesis. *Neuron*, *71*(5), 926–940. doi: 10.1016/j.neuron.2011.06.032

McWalter, R., & Dau, T. (2017). Cascaded amplitude modulations in sound texture perception. *Frontiers in Neuroscience*, *11*, 12.

Mellody, M., & Wakefield, G. H. (2000). The time-frequency characteristics of violin vibrato: Modal distribution analysis and synthesis. *Journal of the Acoustical Society of America*, *107*(1), 598–611.

Moore, B. C., Glasberg, B. R., Gaunt, T., & Child, T. (1991). Across-channel masking of changes in modulation depth for amplitude- and frequency-modulated signals. *The Quarterly Journal of Experimental Psychology*, *43*(3), 327–347. doi: 10.1080/14640749108400976

Morise, M. (2015). CheapTrick, a spectral envelope estimator for high-quality speech synthesis. *Speech Communication*, *67*, 1–7. doi: 10.1016/j.specom.2014.09.003

Morise, M., Yokomori, F., & Ozawa, K. (2016). WORLD: A vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, *E99-D*(7), 1877–1884. doi: 10.1587/transinf.2015EDP7457

Noble, J., Thoret, E., Henry, M., & McAdams, S. (2020). Semantic dimensions of sound mass music: An exploration of mappings between perceptual and acoustic domains. *Music Perception*, *38*(2), 211–238.

Palan, S., & Schitter, C. (2018). Prolific.ac—A subject pool for online experiments. *Journal of Behavioral and Experimental Finance*, *17*, 22–27. doi: 10.1016/j.jbef.2017.12.004

Peeters, G., Giordano, B. L., Susini, P., Misdariis, N., & McAdams, S. (2011). The Timbre Toolbox: Extracting audio descriptors from musical signals. *Journal of the Acoustical Society of America*, *130*(5), 2902–2916. doi: 10.1121/1.3642604

Plazak, J., & McAdams, S. (2017). Perceiving changes of sound-source size within musical tone pairs. *Psychomusicology: Music, Mind, and Brain*, *27*(1), 1–13. doi:

10.1037/pmu0000172

Serra, X. (1997). Musical sound modeling with sinusoids plus noise. In C. Roads, S. Pope, A. Piccialli, & G. De Poli (Eds.), *Musical Signal Processing* (pp. 91–122). New York: Routledge.

Stevens, S. S., Volkmann, J., & Newman, E. B. (1937). A scale for the measurement of the psychological magnitude pitch. *Journal of the Acoustical Society of America*, *8*(3), 185–190.

Thoret, E., Caramiaux, B., Depalle, P., & McAdams, S. (2021). Learning metrics on spectrotemporal modulations reveals the perception of musical instrument timbre. *Nature: Human Behaviour*, *5*(3), 369–377. doi: 10.1038/s41562-020-00987-5

Unser, M. (2000). Sampling—50 years after Shannon. *Proceedings of the IEEE*, *88*(4), 569–587. doi: 10.1109/5.843002

Vallat, R. (2018). Pingouin: Statistics in Python. *Journal of Open Source Software*, *3*(31), p. 1026. doi: 10.21105/joss.01026

Verfaille, V., Guastavino, C., & Depalle, P. (2005). Perceptual evaluation of vibrato models. In *Proceedings of the Conference on Interdisciplinary Musicology* (pp. 1–19). Montreal, Canada.

Wilkins, J., Seetharaman, P., Wahl, A., & Pardo, B. (2018). Vocalset: A singing voice dataset. In *International Society for Music Information Retrival Converence.* Paris.

# Appendices

# Appendix A

# Time and frequency

In this experiment, we are concerned with the spectral envelope and its effect on the perception of vibrato. This appendix considers some of the theoretical implications of measuring time-varying spectral information, and briefly discusses CheapTrick (Morise, 2015), the algorithm used for extracting spectral envelopes in this experiment.

## A.1 Time-frequency trade-off and uncertainty

A musical signal may be considered for its temporal or spectral information; both can provide useful insights. For example, temporal information is vital for localizing note onset and offset times, while spectral information is useful for determining note pitch and timbre. However, there is a fundamental limit to the level of detail of spectral and temporal information that can be accessed at once. This limitation, known as the time-frequency trade-off, is the principle challenge in extracting time-varying spectral information (such as in a time-varying spectral envelope). This section establishes the theoretical foundation for the time-frequency trade-off, beginning with the translation of a time-domain signal into the frequency domain, as is typically accomplished with the Fourier transform. The Fourier transform is expressed:

$$X(\nu) = \int_{-\infty}^{\infty} x(t)\exp(-j2\pi\nu t)\mathrm{d}t, \tag{A.1}$$

where $x(t)$ is the time-domain signal under consideration, $t$ is time and $\nu$ is frequency; $X(\nu)$ is the spectrum, and $j = \sqrt{-1}$ is the imaginary number.

The spectrum is a complex-valued function, and is usually considered in its polar form—magnitude and angle—rather than its real and imaginary components. Its magnitude measures the amplitudes of the spectral components of the signal; its angle indicates their frequency-specific phase.

The inverse-Fourier transform returns a spectrum to the time-domain:

$$x(t) = \int_{-\infty}^{\infty} X(\nu)\exp(j2\pi\nu t)\mathrm{d}\nu. \tag{A.2}$$

There exists a curious, yet fundamental reciprocal relationship between time and frequency information. Let us consider a simple signal having a limited bandwidth, i.e., a limited amount of spectral content:

$$B(\nu) = \begin{cases} 1, & |\nu| < \nu_0 \\ 0, & \text{otherwise} \end{cases} \tag{A.3}$$

where $\nu_0$ is an arbitrary positive frequency. We can determine the time-domain equivalent of this signal with the inverse-Fourier transform:

$$\begin{aligned}
b(t) &= \int_{-\infty}^{\infty} B(\nu)\exp(j2\pi\nu t)\mathrm{d}\nu && \text{(A.4)}\\
&= \int_{-\nu_0}^{\nu_0} \exp(j2\pi\nu t)\mathrm{d}\nu && \text{(A.5)}\\
&= \frac{1}{j2\pi t}\exp(j2\pi\nu t)\bigg|_{-\nu_0}^{\nu_0} && \text{(A.6)}\\
&= \frac{1}{j2\pi t}\left[\exp(j2\pi\nu_0 t) - \exp(-j2\pi\nu_0 t)\right] && \text{(A.7)}\\
&= \frac{\cancel{j2}}{\cancel{j2}\pi t}\sin(2\pi\nu_0 t) && \text{(A.8)}\\
&= 2\nu_0\,\mathrm{sinc}(2\pi\nu_0 t). && \text{(A.9)}
\end{aligned}$$

When $\nu_0$ is small, $B(\nu)$ has a narrow bandwidth and $b(t)$ oscillates slowly. A narrow bandwidth means that $B(\nu)$ takes up relatively little spectral extent, while a slower oscillation means that $b(t)$ requires proportionately *more time* to fully oscillate. A signal that is sharply defined in the spectrum is relatively more ambiguously defined in time. The same is also true in other direction: a signal that is sharply defined in time is relatively more ambiguous in frequency.

We will now proceed to a more rigorous analysis of the problem. The time-frequency trade-off was famously described by Gabor (1947) in his Theory of Communication, itself a reformulation of the uncertainty principle originally stated by Heisenberg in 1927. The following proof follows closely the logic of Flandrin (1999). We will mathematically define an extent in time, and an extent in frequency; then we will prove that the time-frequency area defined by these extents cannot be arbitrarily small. Consider a signal $x(t)$ having a finite quantity of energy, i.e.,

$$E_x = \int_{-\infty}^{\infty} |x(t)|^2 \mathrm{d}t < +\infty. \tag{A.10}$$

For the sake of simplicity, we consider a signal with zero centroids in time and frequency, which would otherwise have to be systematically subtracted to ensure generality:

$$\int_{-\infty}^{\infty} t|x(t)|^2 \mathrm{d}t = 0, \tag{A.11}$$

$$\int_{-\infty}^{\infty} \nu|X(\nu)|^2 \mathrm{d}\nu = 0. \tag{A.12}$$

To represent extent in time $\Delta t^2$ and frequency $\Delta\nu^2$, we use the energy-normalized second moments which are equivalent, in some sense, to variance in probability theory:

$$\Delta t^2 = \frac{1}{E_x} \int_{-\infty}^{\infty} t^2 |x(t)|^2 \mathrm{d}t, \tag{A.13}$$

$$\Delta \nu^2 = \frac{1}{E_x} \int_{-\infty}^{\infty} \nu^2 |X(\nu)|^2 \mathrm{d}\nu. \tag{A.14}$$

We want to find the area described by $\Delta t \Delta \nu$. In order to do so, we define an interaction term that captures something of the cross energy between a signal in time and in frequency, for reasons that will soon be clear:

$$I \equiv \int_{-\infty}^{\infty} t x^*(t) \frac{\mathrm{d}x(t)}{\mathrm{d}t} \mathrm{d}t. \tag{A.15}$$

Here, $*$ indicates the complex conjugate. Taking the squared magnitude of this term leads to the following Cauchy-Schwartz inequality, which serves as the basis for our ultimate proof:

$$|I|^2 = \left| \int_{-\infty}^{\infty} t x^*(t) \frac{\mathrm{d}x(t)}{\mathrm{d}t} \mathrm{d}t \right|^2 \leq \int_{-\infty}^{\infty} \left| t^2 x^*(t)^2 \right| \mathrm{d}t \int_{-\infty}^{\infty} \left| \left( \frac{\mathrm{d}x(t)}{\mathrm{d}t} \right)^2 \right| \mathrm{d}t \tag{A.16}$$

Of the two integrals on the right-hand side of the inequality, the first can be seen as non-normalized time extent as defined in Equation (A.13), $E_x \Delta t^2$. The second integral can be connected to the non-normalized frequency extent $E_x \Delta \nu^2$ as follows. Note that the time derivative of $x(t)$ may be expressed in terms of an inverse Fourier transform:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t} \int_{-\infty}^{\infty} X(\nu) \exp(j 2\pi \nu t) \mathrm{d}\nu = \int_{-\infty}^{\infty} X(\nu)(j 2\pi\nu) \exp(j 2\pi \nu t) \mathrm{d}\nu. \tag{A.17}$$

Its squared magnitude is then,

$$\left| \left( \frac{\mathrm{d}x(t)}{\mathrm{d}t} \right) \right|^2 \leq \int_{-\infty}^{\infty} |X(\nu)|^2 |j 2\pi\nu|^2 |\exp(j 2\pi\nu t)|^2 \mathrm{d}\nu \tag{A.18}$$

$$= 4\pi^2 \int_{-\infty}^{\infty} \nu^2 |X(\nu)|^2 \mathrm{d}\nu \tag{A.19}$$

$$= 4\pi^2 E_x \Delta \nu^2, \tag{A.20}$$

where we have substituted in Equation (A.14). Substituting Equation (A.20) into Equation

(A.16) gives a first hint at the interaction between time and frequency:

$$|I|^2 \leq 4\pi^2 E_x^2 \Delta t^2 \Delta\nu^2. \tag{A.21}$$

We will now manipulate $|I|^2$ into a more comprehensive form. Integrating by parts, we get:

$$\begin{aligned} I &= \int_{-\infty}^{\infty} t x^*(t) \frac{\mathrm{d}x(t)}{\mathrm{d}t} \mathrm{d}t && \text{(A.22)} \\ &= -\int_{-\infty}^{\infty} x^*(t)x(t)\mathrm{d}t - \int_{-\infty}^{\infty} t x(t) \frac{\mathrm{d}x^*(t)}{\mathrm{d}t} \mathrm{d}t && \text{(A.23)} \\ &= -E_x - I^*, && \text{(A.24)} \end{aligned}$$

and therefore, that:

$$\begin{aligned} I + I^* = 2\Re\{I\} &= -E_x && \text{(A.25)} \\ \Re\{I\} &= -\frac{E_x}{2}. && \text{(A.26)} \end{aligned}$$

We now return to Equation (A.21). Rather than $|I|^2$, we may consider the square of its real component, which is guaranteed to be less than or equal to its magnitude as a complex number (i.e., $|\Re\{z\}|^2 \leq |z|^2, z \in \mathbb{C}$). Replacing this value on the left-hand of the inequality gives:

$$\frac{{E_x}^2}{4} \leq 4\pi^2 {E_x}^2 \Delta t^2 \Delta\nu^2, \tag{A.27}$$

leading to the famous result:

$$\Delta t \, \Delta\nu \geq \frac{1}{4\pi}. \tag{A.28}$$

Here we have the product of time and frequency. For our purposes, the value $\frac{1}{4\pi}$ is not particularly meaningful in and of itself, except for the fact that it is greater than zero. This result indicates that the "area," defined by time and frequency cannot be arbitrarily small. In other words, there is an effective *maximum resolution* to the joint decomposition in time and frequency, due to the fact that the two are inextricably bound by definition.

## A.2 Time-frequency representations

Despite this theoretical limitation, listeners tend to consider what they hear—certainly music—in terms of both time and frequency simultaneously. Just like a musical score, it is natural to seek out a mathematical representation of sound that offers a reliable presentation of both time and frequency information. The most well known of these representations is perhaps the spectrogram, which is the square of the magnitude of the short time Fourier transform (STFT). However as demonstrated by Mellody and Wakefield (2000), the spectrogram is not the most reliable representation for measuring precise amplitude fluctuations in the spectrum over time.

We begin our discussion of time-frequency distributions with the Wigner-Ville distribution, which despite being relatively less commonly used, has certain desirable theoretical properties which demonstrate the potential for precision in time-frequency distributions, and will hopefully shed light on why some distributions are better suited than others for particular kinds of analysis.

### A.2.1 The Wigner-Ville distribution

The WVD is written as:

$$W_{xx}(t,\nu) = \int_{-\infty}^{\infty} x\left(t+\frac{\tau}{2}\right) x^*\left(t-\frac{\tau}{2}\right) \exp(-j2\pi\nu\tau)\mathrm{d}\tau, \tag{A.29}$$

which may be seen as the Fourier transform of a local auto-covariance of $x(t)$, centered at time $t$. The WVD has a number of useful properties; for example, under certain circumstances, it permits perfect localization in time and frequency. Let us now consider the example of an *analytic* signal, which is a complex-valued signal having only non-negative frequencies. A real-valued signal $x(t)$, with a spectrum $X(\nu)$, has the following relation to its analytic equivalent $Z(\nu)$:

$$Z(\nu) = X(\nu)(1+\mathrm{sign}(\nu)). \tag{A.30}$$

Using an analytic signal permits many conveniences; for example, one may calculate a meaningful average frequency for an analytical signal, as any real signal, whose spectrum is therefore hermitian symmetric, has an average frequency of zero.

The experiment in this thesis is largely concerned with partials having a fixed frequency (i.e., no FM) but having an amplitude that varies in time (AM). Such a signal may be expressed analytically as:

$$z(t) = A(t)\exp(j2\pi\nu_0 t), \tag{A.31}$$

where $\nu_0$ is the signal frequency in Hz, and $A(t)$ is its time-varying amplitude. Through the WVD, this signal takes on a very clean form:

$$W_{zz}(t,\nu) = \int_{-\infty}^{\infty} z\left(t+\frac{\tau}{2}\right) z^*\left(t-\frac{\tau}{2}\right) \exp(-j2\pi\nu\tau)\mathrm{d}\tau \tag{A.32}$$

$$= \int_{-\infty}^{\infty} A\left(t+\frac{\tau}{2}\right) A^*\left(t-\frac{\tau}{2}\right) \exp\left(j2\pi\nu_0\left[\left(t+\frac{\tau}{2}\right)-\left(t-\frac{\tau}{2}\right)\right]\right) \exp(-j2\pi\nu\tau)\mathrm{d}\tau \tag{A.33}$$

$$= \int_{-\infty}^{\infty} A\left(t+\frac{\tau}{2}\right) A^*\left(t-\frac{\tau}{2}\right) \exp\left(-j2\pi\tau(\nu-\nu_0)\right)\mathrm{d}\tau \tag{A.34}$$

$$= W_{AA}(t,\nu-\nu_0). \tag{A.35}$$

From Equation (A.35) it is clear that the WVD of a time-varying partial is simply the frequency-shifted WVD of its amplitude modulation signal. From this result, we may say that the WVD is *translationally invariant* in frequency, a desirable property of the distribution. Indeed, the WVD is translationally invariant in time as well, which can be demonstrated by following a similar line of reasoning.

Another useful property of the WVD is its ability to collapse meaningfully across the time or frequency axes into valid a marginal distribution of either:

$$\int_{-\infty}^{\infty} W_{xx}(t,\nu)\mathrm{d}\nu = |x(t)|^2, \tag{A.36}$$

$$\int_{-\infty}^{\infty} W_{xx}(t,\nu)\mathrm{d}t = |X(\nu)|^2. \tag{A.37}$$

In this way, through the WVD it is always possible to measure the time-varying energy of a signal, by Equation (A.36); or its power spectral density at any given moment in time, by

Equation (A.37). However, these convenient properties do not come without a price. The principle drawback of the WVD is its introduction of cross-terms. Consider the combination of two signals, $x(t)$ and $y(t)$. Its WVD would take the form:

$$W_{x+y} = W_{xx} + W_{yy} + 2\Re\{W_{xy}\}, \tag{A.38}$$

where here the final term is the real component of the cross-WVD of $x(t)$ and $y(t)$.

By simply following Equation (A.29), the introduction of cross-terms is evident from the outset:

$$W_{x+y} = \int_{-\infty}^{\infty} \left[x\left(t+\frac{\tau}{2}\right) + y\left(t+\frac{\tau}{2}\right)\right]\left[x^*\left(t-\frac{\tau}{2}\right) + y^*\left(t-\frac{\tau}{2}\right)\right]\exp(-j2\pi\nu\tau)\mathrm{d}\tau \tag{A.39}$$

$$= W_{xx} + W_{yy} + \int_{-\infty}^{\infty}\left[\underbrace{x\left(t+\frac{\tau}{2}\right)y^*\left(t-\frac{\tau}{2}\right)}_{\text{first term}} + \underbrace{y\left(t+\frac{\tau}{2}\right)x^*\left(t-\frac{\tau}{2}\right)}_{\text{second term}}\right]\exp(-j2\pi\nu\tau)\mathrm{d}\tau. \tag{A.40}$$

For the sake of illustration, let us consider the case of two stationary, analytical partials, having distinct amplitudes and frequencies:

$$x(t) = A_x \exp(j2\pi\nu_x t), \tag{A.41}$$

$$y(t) = A_y \exp(j2\pi\nu_y t). \tag{A.42}$$

We will focus on the integrand of the final term of (A.40), substituting in (A.41) and (A.42).

Its first term becomes:

$$x\left(t+\frac{\tau}{2}\right)y^*\left(t-\frac{\tau}{2}\right) = A_x \exp(j2\pi\nu_x t)\exp\left(j2\pi\nu_x\frac{\tau}{2}\right)A_y^*\exp(-j2\pi\nu_y t)\exp\left(j2\pi\nu_y\frac{\tau}{2}\right). \tag{A.43}$$

Factoring the complex exponentials by $\tau$ and $t$ gives:

$$\exp\left(j2\pi\frac{\tau}{2}(\nu_x+\nu_y)\right)A_xA_y^*\exp(j2\pi t(\nu_x-\nu_y)). \tag{A.44}$$

We now combine this result with the second, similarly factored term in the integrand, to yield:

$$\exp\left(j2\pi\frac{\tau}{2}(\nu_x+\nu_y)\right)\underbrace{\left[A_xA_y^*\exp(j2\pi t(\nu_x-\nu_y))+A_x^*A_y\exp(-j2\pi t(\nu_x-\nu_y))\right]}_{\text{complex conjugates}}. \tag{A.45}$$

Note that the terms in square brackets are complex conjugates of one another. Once again, we make use of the fact that $z+z^*=2\Re\{z\}, z\in\mathbb{C}$. Therefore, the final term of (A.38) becomes:

$$2\Re\{W_{xy}\} = 2\Re\left\{A_xA_y^*\exp(j2\pi t(\nu_x-\nu_y))\right\}\int_{-\infty}^{\infty}\exp\left(j2\pi\frac{\tau}{2}(\nu_x+\nu_y)\right)\exp(-j2\pi\tau\nu)\mathrm{d}\tau \tag{A.46}$$

$$= A_xA_y\cos(2\pi t(\nu_x-\nu_y))\,\delta\left(\nu-\frac{1}{2}(\nu_x+\nu_y)\right). \tag{A.47}$$

Equation (A.47) is a clean expression of the Wigner-Ville cross-term; it is an oscillating signal in time (as indicated by the cosine term), and it is perfectly localized in frequency (as indicated by the dirac delta term). It takes the average frequency of the two partials, and oscillates in time at their difference in frequency. Note that because the cross-term oscillates as a cosine above and below zero, the WVD is not strictly non-negative. As a result, its value cannot be read literally as a localized measure of time-frequency energy.

This oscillation, however, also points to a possible improvement to the WVD. Because the interaction terms are oscillatory, they may be effectively eliminated with the appropriate choice of averaging filter. Filtering removes cross-terms, thought at the expense of some of the desirable precision of the WVD. As demonstrated by Cohen (1966), many time-frequency representations can be commonly characterized by their relation to the WVD via a distribution-specific compromise, represented by a smoothing kernel in time and frequency. For example, convolving the WVD with a two-dimensional Gaussian kernel results in a spectrogram (assuming its window is a Gaussian), whose value is always non-negative.

## A.3 Methods to measure the spectral envelope

What we can learn from the preceding is that no time-frequency plane can purvey the unadulterated truth of time or frequency. A more flexible concept is the spectral envelope, which is discussed in this section.

### A.3.1 Source-filter models

One approach to extracting spectral information is to consider the sound as an interaction of two components: a spectrally rich generator or excitation signal, that is filtered by a system that carries the *spectral envelope* of the sound. For a stationary sound, the magnitude frequency response of a filter to an excitation signal can be written as:

$$|S(\omega)| = |H(\omega)| \cdot |E(\omega)|, \tag{A.48}$$

where $S(\omega)$, $H(\omega)$ and $E(\omega)$ are the frequency-domain representations of the filtered signal, the filter itself and the excitation signal, respectively. This action can be expressed in the time domain as a convolution:

$$s(t) = (h * e)(t) = \int_{-\infty}^{\infty} h(\tau)e(t-\tau)\mathrm{d}\tau, \tag{A.49}$$

where $s(t)$ is the filtered sound, $h(t)$ is the impulse response of the filter and $e(t)$ is the excitation signal. As with many such models, it has its roots in speech analysis and synthesis, and may be illustrated by an example in this domain.

The excitation signal in speech is a combination of a harmonically rich pitched sound produced by the periodic opening and closing of the vocal-folds, and noise generated from turbulent air. Classically, excitation signals are considered to be either "voiced" or "unvoiced." The former is periodic and carries pitch and prosody; the latter is necessary for articulating stopped consonants and fricatives. Later models developed a more nuanced notion of aperiodicity, wherein different parts of the excitation spectrum can be either voiced or unvoiced simultaneously (Griffin & Lim, 1985).

In speech synthesis, the spectral envelope reflects the multi-resonant structure of the vocal tract, namely the pharynx and the oral cavity (Maher & Beauchamp, 1990), which is responsible for shaping vowel sounds. Likewise, the spectral envelope of a string instrument can be seen as tracing the contributions of many narrow bandwidth resonators (Mathews & Kohut, 1973).

A fixed spectral envelope, when coupled with a time-varying excitation signal, provides a strong cue for the resonant structure of the sound source. The spectral envelope has been shown, for example, to provide clues about the source size (Plazak & McAdams, 2017) and can be essential for sound segregation in a complex mixture when coupled with frequency modulation (McAdams, 1989). In practice the spectral envelope of a sound is often time-varying, as in speech that shifts from one vowel sound to another, or in the primarily spectral vibrato of some brass and reed instruments (Verfaille et al., 2005).

Assuming that a sound is generated by an underlying source-filter structure, there are a number of techniques available to decouple an excitation source from its spectral envelope. For this experiment, we investigated two such techniques: linear predictive coding, and the CheapTrick algorithm (Morise, 2015).

### A.3.2 Linear predictive coding

Linear predictive coding (LPC) attempts to predict a signal as a linear combination of a fixed number of its preceding samples (Makhoul, 1975). The premise of LPC can be articulated mathematically as follows (note here that we switch from continuous to discrete time, as indicated by the use of square brackets):

$$s[n] = -\sum_{k=1}^{p} a_k s[n-k] + Ge[n]. \tag{A.50}$$

Here $s[n]$ is the output signal, $e[n]$ is an excitation signal with a gain coefficient $G$, and the coefficients $a_k$ represent weights on the past $p$ samples of $s[n]$. The values $a_k$ can be seen as the coefficients of an autoregressive filter. To make this comparison clear, we start by taking the z-transform of (A.50), expanding and rearranging for clarity:

$$S(z) = GE(z) - a_1 S(z) z^{-1} - \cdots - a_p S(z) z^{-p} \tag{A.51}$$

$$GE(z) = S(z)[1 + a_1 S(z) z^{-1} + \cdots + a_p S(z) z^{-p}] \tag{A.52}$$

$$H(z) = \frac{S(z)}{E(z)} = G \cdot \frac{1}{1 + a_1 z^{-1} + \cdots + a_p z^{-p}}. \tag{A.53}$$

From Equation (A.53) we can see the all-pole structure of $H(z)$ (which is a direct consequence of the assumptions of the LPC solution). The frequency response of this transfer function can provide an informative approximation of a spectral envelope of $s[n]$.

One may equivalently see the problem in a statistical signal processing context. Consider a prediction $\hat{s}[n]$ that is built as a linear combination of its past $p$ values. We redefine $e[n]$ in this context to be the prediction error, specifically the difference from the true signal $s[n]$

and this prediction:

$$e[n] = s[n] - \hat{s}[n] = s[n] - \sum_{k=1}^{p} a_k s[n-k]. \tag{A.54}$$

The ideal prediction signal minimizes the energy of $e[n]$. Articulating the problem in terms of error signal energy $E$, where

$$E = \sum_{n=-\infty}^{\infty} e^2[n] = \sum_{n=-\infty}^{\infty} \left( s[n] - \sum_{k=1}^{p} a_k s[n-k] \right)^2 . \tag{A.55}$$

We can find the ideal coefficients $a_k$ by reconfiguring as a minimization problem. In such a setting, we find the minimum of a convex function, here, $E$, by setting its partial derivatives to 0 with respect to the variables in question, here, $a_k$. Setting the partial derivatives with respect to $a_k$ to 0 results in a system of linear equations:

$$\sum_{k=1}^{p} a_k \sum_{n=-\infty}^{\infty} s[n-k]s[n-i] = - \sum_{n=-\infty}^{\infty} s[n]s[n-i] \qquad 1 \leq i \leq p. \tag{A.56}$$

Surprisingly, these equations can be expressed in terms of the autocorrelation function, assuming that the signal is stationary:

$$R[i] = \sum_{n=-\infty}^{\infty} s[n]s[n-i]. \tag{A.57}$$

Noting that the autocorrelation function is an even function, we may re-articulate (A.56) as:

$$\sum_{k=1}^{p} R[|i-k|]a_k = -R[i], \tag{A.58}$$

which can, in turn, be expressed succinctly in matrix form:

$$\mathbf{Ra} = -\mathbf{r}, \tag{A.59}$$

or explicitly:

$$\begin{bmatrix} R[0] & R[1] & \dots & R[p-1] \\ R[1] & R[0] & \dots & R[p-2] \\ R[2] & R[1] & \dots & R[p-3] \\ \vdots & \vdots & \ddots & \vdots \\ R[p-1] & R[p-2] & \dots & R[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} R[1] \\ R[2] \\ R[3] \\ \vdots \\ R[p] \end{bmatrix} \tag{A.60}$$

Here **R** is a symmetrical Toeplitz matrix built from R[$i$], that is, a matrix having the autocorrelation function across the top row and which is symmetrical across its diagonals. The vector **a** is the vector of linear coefficients to be found, and **r** is the autocorrelation function itself, starting at index 1 and going to $p$.

In practice, **R**, **a** and **r** are all estimates, thus a perfect solution is never possible. The best-fitting **a** can be solved by using any least-squares algorithm. In the application space of LPC, the Levinson-Durbin algorithm is a common choice (Makhoul, 1975).

Because it models an all-pole filter, LPC does not efficiently capture the spectral dips that are well-characterized by zeroes. In practice, such a filter can emphasize spectral regions of an underlying harmonic excitation signal, but it does not attenuate well. The effect of AMV, however, appears to rely on the relatively wide partial-specific modulation gain; this effect therefore requires not only spectral accentuation, but sharp attenuation as well. Our early explorations using time-varying spectral envelopes captured with LPC resulted in an overly mild timbral modulation that did not have any of the pitch-like characteristics of the AMV heard in the synthetically flattened signals. We turned instead to the CheapTrick algorithm, which will be described briefly in the next section.

### A.3.3 The CheapTrick algorithm

The CheapTrick algorithm (Morise, 2015) is part of the WORLD vocoder (Morise et al., 2016), a research tool designed for analysis and manipulation of speech signals. WORLD is itself an outgrowth of the TANDEM-STRAIGHT vocoder (Kawahara et al., 2008). Assuming an underlying source-filter structure, the algorithm attempts to smooth out regularities that are present in both the temporal and spectral axes of time-varying power spectra.

**Removing temporal influence using pitch-synchronous windowing.** The algorithm depends on a pre-calculated pitch trajectory, using pitch-synchronized analysis windows to extract spectral information. Such windows exert a minimal influence on the time-varying power of the underlying signal.

Consider a harmonic signal $y(t)$ with a period of $T_0$, multiplied by a Hann window $w(t)$ having a length of three times this period (noting that here we consider an ideal case in continuous time; in practice, the window length is rounded to the nearest sample value).

Such a window has the expression:

$$w(t) = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi t}{3T_0}\right)\right], \qquad 0 \leq t \leq 3T_0. \tag{A.61}$$

Because the underlying signal is perfectly periodic, we can consider its power by integrating over three temporal subdivisions, each the length of one period:

$$\int_0^{3T_0} (y(t)w(t))^2 \mathrm{d}t = \int_0^{T_0} y^2(t)w^2(t)\mathrm{d}t + \int_0^{T_0} y^2(t)w^2(t+T_0)\mathrm{d}t + \int_0^{T_0} y^2(t)w^2(t+2T_0)\mathrm{d}t \tag{A.62}$$

$$= \int_0^{T_0} y^2(t)(w^2(t) + w^2(t+T_0) + w^2(t+2T_0))\mathrm{d}t \tag{A.63}$$

$$= 1.125 \int_0^{T_0} y^2(t)\mathrm{d}t. \tag{A.64}$$

As $y(t)$ is periodic and $w(t)$ is tailored to its period, this process becomes a kind of built-in overlap-add, where the contributed power of $w(t)$ adds up to a constant.

**Smoothing the spectrum of a periodic signal.** Returning to the source-filter model, let us consider an excitation signal made of periodic impulses, passing through a filter with an impulse response $h(t)$:

$$y(t) = h(t) * \sum_{n=-\infty}^{\infty} \delta(t - nT_0), \tag{A.65}$$

$$Y(\nu) = \frac{1}{T_0} H(\nu) \sum_{n=-\infty}^{\infty} \delta(\nu - n\nu_0). \tag{A.66}$$

Here $y(t)$ is the observed signal, having a period of $T_0$, and a fundamental frequency of $\nu_0 = \frac{1}{T_0}$. Given an observed instrument sound, we wish to model its spectral envelope as $H(\nu)$; however we do not have direct access to it.

A periodic excitation signal *samples* its spectral filter at multiples of the fundamental frequency $\nu_0$. Based on this idea, the CheapTrick algorithm reconsiders the source-filter model as a sampling system, and uses the theory of consistent sampling (Unser, 2000) to recover an "unsampled" spectral envelope.

In the consistent sampling approach, a continuous signal $f(t)$ and an approximation signal $\hat{f}(t)$ may be considered equivalent if they both appear the same *after being sampled*. In the analogy of the source-filter model, the observed sound power spectrum $|Y(\nu)|^2$ is the *sampled* filter spectrum. We may not be able to recover the true filter spectrum directly, but we may recover an approximation spectrum that, when it is *itself* sampled (by periodic excitation in the time domain), will re-generate the observed spectrum $|Y(\nu)|^2$. This approximation spectrum is the spectral envelope extracted by CheapTrick. The CheapTrick algorithm has a pre-processing and spectral smoothing step, followed by a "digital correction" filter.

To avoid potentially negative values produced by the spectral recovery process, CheapTrick applies all processing in the *log* power spectrum. The processed log-spectrum is then exponentiated back to a power spectrum, guaranteeing that it has only positive values. Before taking its logarithm, the observed spectrum must first be smoothed in order to remove any zeroes. Let $P(\nu)$ represent the power spectrum of the observed signal $y(t)$.

The pre-processing step widens the observed spectral peaks by averaging along a moving rectangular window:

$$P_{\text{pre}}(\nu) = \frac{3}{2\nu_0} \int_{-\frac{\nu_0}{3}}^{\frac{\nu_0}{3}} P(\nu + \lambda) \mathrm{d}\lambda. \tag{A.67}$$

In the smoothing step, CheapTrick performs liftering (Bogert, 1963) to remove oscillating components in the power spectrum due to the signal's periodic nature. Liftering is a filtering-like process performed on the cepstrum; the cepstrum is the inverse Fourier transform of the log-magnitude of the spectrum, a representation that is well-suited to spectral recovery. The cepstrum is expressed as:

$$C_y = \mathcal{F}^{-1}\{\log(|\mathcal{F}\{y(t)\}|^2)\}. \tag{A.68}$$

Let us consider two cycles of the periodic signal $y(t)$:

$$y(t) = b(t) + b(t - T_0), \tag{A.69}$$

where $b(t)$ represents one cycle. The Fourier transform of such a signal is:

$$X(\nu) = B(\nu) + \exp(-j2\pi\nu T_0)B(\nu). \tag{A.70}$$

Taking its power spectrum, we have the following expression:

$$|X(\nu)|^2 = |B(\nu)|^2(2 + 2\cos(2\pi\nu T_0)). \tag{A.71}$$

Equation (A.71) contains a clearly oscillating component of the power spectrum, having a "frequency" of the signal period $T_0$. Taking the logarithm of this expression allows for linear separability of this component:

$$\log(|X(\nu)|^2) = \log(|B(\nu)|^2) + \log(2 + 2\cos(2\pi\nu T_0)). \tag{A.72}$$

Taking the inverse Fourier transform of this function brings the periodic signal into the cepstral domain, where the oscillation at $\cos(2\pi\nu T_0)$ is readily isolated as a single *cepstral* peak; this peak may be removed with liftering. CheapTrick lifters by multiplying the cepstrum with a sinc function having zeros at multiples of $\nu_0$:

$$l_s(\tau) = \frac{\sin(\pi\nu_0\tau)}{\pi\nu_0\tau}, \tag{A.73}$$

where $\tau$ is the time index (or "mite" index) in the cepstral domain. This action corresponds to a convolution in the frequency domain with a rectangular window of length $\nu_0$.

Liftering results in a slightly smeared spectrum, which no longer adheres to the rules of consistent sampling: this smeared spectrum, if "resampled" with a harmonic excitation signal, would *not* re-generate the original observed signal, due to errors introduced at multiples of $\nu_0$ Hz. This effect can be undone with a digital correcting filter.

Consistent sampling models a discrete signal that has passed through a pre- and post-sampling filter, with an optional digital correcting filter. In the CheapTrick paradigm, the post-filter is the *smoothed* power spectrum of the analysis window, $|H(\nu)|^2 * r_{\nu_0}(\nu)$, where $|H(\nu)|^2$ is the power spectrum of the analysis window (in time), and $r_{\nu_0}(\nu)$ is the rectangular window (in frequency) used for spectral smoothing in the previous step. The pre-filter is modeled as a delta function. The digital correction filter is modeled as the *inverse* filter corresponding to this pre- and post- sampling. Following Kawahara et al. (2008), this operation is performed on the log spectrum as:

$$P_R(\nu) = \exp\left(q_0 \log(P_{\text{smoothed}}(\nu)) + q_1 \log(P_{\text{smoothed}}(\nu+\nu_0)P_{\text{smoothed}}(\nu-\nu_0))\right), \tag{A.74}$$

where $P_R(\nu)$ is the "recovered" version of the smoothed power spectrum $P(\nu)_{\text{smoothed}}$. Morise empirically tuned the coefficient values to be $q_0 = 1.18$ and $q_1 = -0.09$. Because of the introduction of negative values, this step is calculated in the cepstral domain as a simple multiply, using the cepstral equivalent:

$$l_r(\tau) = q_0 + 2q_1 \cos(2\pi\tau\nu_0) \longleftrightarrow q_0\,\delta(\nu) + q_1(\delta(\nu+\nu_0) + \delta(\nu-\nu_0)), \tag{A.75}$$

where $\tau$ is the time index in the cepstral domain. The final cepstral processing, including the pre-processing, smoothing, and spectral recovery takes the form:

$$P_{\text{CheapTrick}}(\nu) = \exp(\mathcal{F}\{l_s(\tau)l_r(\tau)C_x(\tau)\}), \tag{A.76}$$

where $C_x(\tau)$ is the cepstrum of the underlying signal:

$$C_x = \mathcal{F}^{-1}\{\log(P(\nu))\}. \tag{A.77}$$

The name CheapTrick comes from the notion that the algorithm is no more than a few simple "tricks" performed on the cepstrum, each practically tuned, though grounded theoretically in consistent sampling theory. Regardless, we found that the spectral envelopes produced by CheapTrick were much more effective in conveying the AMV percept than those extracted with LPC, when used with our synthesis engine.

We suspect that this result is a function of CheapTrick's ability to capture envelope values close to zero; this property is itself indebted to its processing on the cepstrum. The AMV illusion seems to depend upon large amplitude fluctuations, modulations that can approach near-zero values. As Mellody and Wakefield (2000) discovered, the spectrogram underestimates amplitude fluctuations and can't capture these trajectories; likewise, the LPC algorithm does not model zeros.

# Appendix B

# Python code

## B.1 build_stimuli.py

```
import numpy as np
import os
from scipy.io import wavfile
from tqdm import tqdm

from src import macro
from src.analysis import single_cycles
from src.defaults import SAMPLE_RATE, SYN_PATH
from src.util import midi_to_hz, safe_mkdir


# Helper.
def quick_write(_file_path, _filename, _data):
    """Write file as 16bit mono PCM."""

    write_path = os.path.join(_file_path, _filename)

    amplitude = np.iinfo(np.int16).max
    _data *= amplitude

    wavfile.write(write_path, SAMPLE_RATE, _data.astype(np.int16))
```

```
# Experiment parameters.
num_subjects = 200
num_blocks = 2
repeats_per_block = 4

# Use this to start counting from a subject number greater than 0.
starting_subject = 200

# Load env as linear amplitude. (CheapTrick calculates the power spectrum.)
env = single_cycles[0]['env']
env = np.sqrt(env)

# Synthesis parameters.
synthesis_params = {
    'num_partials': 70,
    'f0': midi_to_hz(48),
    'fm_depth': 0.1314,
    'length': 2.5,
    'mod_rate': 5.,
    'mod_hold': 0.,
    'mod_fade': 0.,
    'audio_fade': 0.25,
    'env': env,
}

for s in range(num_subjects):
    s += starting_subject
    print(f"\nGenerating stimuli for subject {s}...")

    # Make subject directory.
    subject_path = os.path.join(SYN_PATH, f"subject_{s}/")
    safe_mkdir(subject_path)

    # Open log.
    log_path = os.path.join(subject_path, f"stimlog_subject_{s}.txt")
    log = open(log_path, "w")

    log.write(f"Subject: {s}\n" + "-" * 10 + "\n")

    for b in tqdm(range(num_blocks)):
```

```
    # Make block directory.
    block_path = os.path.join(subject_path, f"block_{b}/")
    safe_mkdir(block_path)

    log.write("\n" + "="*7 + f"\nBlock {b}\n" + "="*7 + "\n")

    for r in range(repeats_per_block):
        # Generate one of each kind of stimulus.

        # BASIC.
        tmp_x = macro.make_basic(synthesis_params)
        quick_write(block_path, f"BASIC_{r}.wav", tmp_x)

        # FROZEN.
        tmp_x = macro.make_frozen(synthesis_params)
        quick_write(block_path, f"FROZEN_{r}.wav", tmp_x)

        # FM-ONLY.
        tmp_x = macro.make_fm_only(synthesis_params)
        quick_write(block_path, f"FM_ONLY_{r}.wav", tmp_x)

        # SHUFFLE and SHUFFLE RAF.
        tmp_x, tmp_x_raf = macro.make_shuffle(synthesis_params, log)
        quick_write(block_path, f"SHUFFLE_{r}.wav", tmp_x)
        quick_write(block_path, f"SHUFFLE_RAF_{r}.wav", tmp_x_raf)

        # SIMPLE and SIMPLE RAF.
        tmp_x, tmp_x_raf = macro.make_simple(synthesis_params, log)
        quick_write(block_path, f"SIMPLE_{r}.wav", tmp_x)
        quick_write(block_path, f"SIMPLE_RAF_{r}.wav", tmp_x_raf)

        # RAG and RAG RAF.
        tmp_x, tmp_x_raf = macro.make_rag(synthesis_params, log)
        quick_write(block_path, f"RAG_{r}.wav", tmp_x)
        quick_write(block_path, f"RAG_RAF_{r}.wav", tmp_x_raf)

        # PAM.
        tmp_x = macro.make_pam(synthesis_params)
        quick_write(block_path, f"PAM_{r}.wav", tmp_x)

        # Control.
        tmp_x = macro.make_control(synthesis_params)
```

```
            quick_write(block_path, f"CONTROL_{r}.wav", tmp_x)

    log.close()
```

## B.2 macro.py

```
"""
Convenience macros for generating stimuli as per thesis.

See ‘synthesis.py‘ for condition descriptions.
"""

from src.defaults import SAMPLE_RATE, PITCH_RATE
from src.synthesis import EnvelopeMorpher, StimulusGenerator

# Instantiate one generator for all stimuli.
generator = StimulusGenerator(
    sr=SAMPLE_RATE,
    pr=PITCH_RATE,
    random_rate_lower_limit=4.,
    random_rate_upper_limit=12.,
)


def make_basic(args):
    return generator(
            f0=args['f0'],
            fm_depth=args['fm_depth'],
            env=args['env'],
            num_partials=args['num_partials'],
            length=args['length'],
            mod_rate=args['mod_rate'],
            mod_hold=args['mod_hold'],
            mod_fade=args['mod_fade'],
            audio_fade=args['audio_fade'],
    )


def make_frozen(args):
    return generator(
            f0=args['f0'],
            fm_depth=0.,
            env=args['env'],
            num_partials=args['num_partials'],
            length=args['length'],
            mod_rate=args['mod_rate'],
            mod_hold=args['mod_hold'],
            mod_fade=args['mod_fade'],
            audio_fade=args['audio_fade'],
    )


def make_shuffle(args, log):
    morpher = EnvelopeMorpher(args['env'], args['f0'])
    morpher.shuffle_phase(num_shifts=4)

    x = generator(
        f0=args['f0'],
        fm_depth=0.,
        env=morpher(),
        num_partials=args['num_partials'],
        length=args['length'],
        mod_rate=args['mod_rate'],
        mod_hold=args['mod_hold'],
        mod_fade=args['mod_fade'],
        audio_fade=args['audio_fade'],
    )

    x_raf = generator(
        f0=args['f0'],
        fm_depth=0.,
        env=morpher(),
        num_partials=args['num_partials'],
        length=args['length'],
        mod_rate=args['mod_rate'],
        mod_hold=args['mod_hold'],
        mod_fade=args['mod_fade'],
        audio_fade=args['audio_fade'],
        synth_mode='raf',
    )

    log.write("\nSHUFFLE\n" + "_"*7 + "\n")
    log.write(f"\n{morpher._log}\n")
    return x, x_raf


def make_simple(args, log):
    morpher = EnvelopeMorpher(args['env'], args['f0'])
    morpher.rap()
    morpher.shuffle_phase(num_shifts=4)

    x = generator(
        f0=args['f0'],
        fm_depth=0.,
        env=morpher(),
        num_partials=args['num_partials'],
        length=args['length'],
        mod_rate=args['mod_rate'],
        mod_hold=args['mod_hold'],
        mod_fade=args['mod_fade'],
        audio_fade=args['audio_fade'],
    )

    x_raf = generator(
        f0=args['f0'],
        fm_depth=0.,
        env=morpher(),
        num_partials=args['num_partials'],
        length=args['length'],
        mod_rate=args['mod_rate'],
        mod_hold=args['mod_hold'],
        mod_fade=args['mod_fade'],
        audio_fade=args['audio_fade'],
        synth_mode='raf',
    )

    log.write("\nSIMPLE\n" + "_" * 7 + "\n")
    log.write(f"\n{morpher._log}\n")

    return x, x_raf


def make_rag(args, log):
    morpher = EnvelopeMorpher(args['env'], args['f0'])
    morpher.rap(max_random_gain=10)
    morpher.shuffle_phase(num_shifts=4)

    x = generator(
        f0=args['f0'],
        fm_depth=0.,
        env=morpher(),
        num_partials=args['num_partials'],
        length=args['length'],
        mod_rate=args['mod_rate'],
        mod_hold=args['mod_hold'],
        mod_fade=args['mod_fade'],
        audio_fade=args['audio_fade'],
    )

    x_raf = generator(
        f0=args['f0'],
        fm_depth=0.,
        env=morpher(),
        num_partials=args['num_partials'],
        length=args['length'],
        mod_rate=args['mod_rate'],
        mod_hold=args['mod_hold'],
        mod_fade=args['mod_fade'],
        audio_fade=args['audio_fade'],
        synth_mode='raf'
    )

    log.write("\nRAG\n" + "_" * 7 + "\n")
    log.write(f"\n{morpher._log}\n")

    return x, x_raf


def make_fm_only(args):
    morpher = EnvelopeMorpher(args['env'], args['f0'])
    morpher.time_average()

    return generator(
        f0=args['f0'],
        fm_depth=args['fm_depth'],
        env=morpher(),
        num_partials=args['num_partials'],
        length=args['length'],
        mod_rate=args['mod_rate'],
        mod_hold=args['mod_hold'],
        mod_fade=args['mod_fade'],
        audio_fade=args['audio_fade'],
        synth_mode='default',
    )


def make_control(args):
    return generator(
        f0=args['f0'],
        fm_depth=0.,
        env=args['env'],
        num_partials=args['num_partials'],
        length=args['length'],
        mod_rate=args['mod_rate'],
        mod_hold=args['length'],
        mod_fade=0.,
        audio_fade=args['audio_fade'],
        synth_mode='pam',
    )


def make_pam(args):
    return generator(
        f0=args['f0'],
        fm_depth=0.,
        env=args['env'],
        num_partials=args['num_partials'],
        length=args['length'],
        mod_rate=args['mod_rate'],
        mod_hold=args['mod_hold'],
        mod_fade=args['mod_fade'],
        audio_fade=args['audio_fade'],
        synth_mode='pam',
    )
```

## B.3 synthesis.py

```
"""
StimulusGenerator reads from an array of spectral envelopes.
```

```
EnvelopeMorpher allows for distortions ot the envelope before synthesis.
"""

from copy import copy
import math
import numpy as np

from defaults import EPS, SAMPLE_RATE, PITCH_RATE
from util import (
    add_fade, midi_to_hz, normalize, plot_envelope, stft_plot, remove_dc,
    resample
)


class StimulusGenerator:
    """
    Generate modulating tones from a cycle of spectral envelopes.
    """
    def __init__(
            self,
            sr: int = SAMPLE_RATE,
            pr: int = PITCH_RATE,
            random_rate_upper_limit: float = 12.,
            random_rate_lower_limit: float = 4.,
    ):
        assert sr > 0
        self.sr = sr

        assert pr > 0
        self.pr = pr

        assert random_rate_upper_limit >= 0
        self.random_rate_upper_limit = random_rate_upper_limit

        assert random_rate_lower_limit <= random_rate_upper_limit
        self.random_rate_lower_limit = random_rate_lower_limit

        self.f0 = None
        self.fm_depth = None

        self.env = None
        self.num_partials = None

        self.length = None
        self.mod_rate = None
        self.mod_hold = None
        self.mod_fade = None

        self.synth_mode = None
        self.audio_fade = None

        self.processed_env = None

    def __call__(
            self,
            f0: float,
            fm_depth: float,
            env: np.ndarray,
            num_partials: int,
            length: float,
            mod_rate: float,
            mod_hold: float,
            mod_fade: float,
            synth_mode: str = 'default',
            audio_fade: float = 0.,
    ) -> np.ndarray:
        """Generate a spectral- and frequency- modulated tone.

        Args:
            f0: Fundamental pitch of the output, in Hz.
            fm_depth: Depth of pitch modulation, in semitones.
            env: Array of spectral envelopes (time x real frequency).
            num_partials: Number of partials for resynthesis.
            length: Synthesis length in seconds.
            mod_rate: Rate of spectral- and frequency- modulation, in Hz.
            mod_hold: Time before applying modulation, in seconds.
            mod_fade: Time to ramp modulation (from 0 to 1), in seconds.
            synth_mode: Synthesis type ->
                'default' is normal behaviour.
                'pam' is the Pure Amplitude Modulation condition (tremolo).
                'raf' is the Random Amplitude modulation Frequency condition.

        Returns:
            Numpy array. A normalized, one-dimensional, audio rate stimulus.
        """

        # Argument checking.
        assert f0 > 0
        self.f0 = f0

        assert fm_depth >= 0
        self.fm_depth = fm_depth

        assert env.ndim == 2
        self.env = env

        assert num_partials > 0
        assert (num_partials * f0) <= (self.sr // 2)
        self.num_partials = num_partials

        assert length > 0
        self.length = length

        assert 0 < mod_rate <= (self.pr // 2)
        self.mod_rate = mod_rate

        assert mod_hold >= 0
        assert mod_fade >= 0
        assert (mod_hold + mod_fade) <= length
        self.mod_hold = mod_hold
        self.mod_fade = mod_fade

        assert synth_mode in ['default', 'pam', 'raf']
        self.synth_mode = synth_mode

        assert 0 <= audio_fade <= length

        # Resample, loop and extend spectral envelope.
        self.process_env()

        # Output.
        x = self.synthesize()
        x = remove_dc(x)
        x = normalize(x)
        # Fade in/out.
        x = add_fade(x, audio_fade, self.sr)
        x = add_fade(x, audio_fade, self.sr, fade_out=True)
        return x

    def process_env(self):

        if self.synth_mode == 'raf':
            tmp_env = self.get_raf_env()
        else:
            tmp_env = self.cycle_and_resample_env()

        tmp_env = self.apply_spectral_fade(tmp_env)

        self.processed_env = tmp_env

    def get_raf_env(self):
        """
        Generate envelope with each partial amp-modulated at a different rate.
        """
        num_frames = self.env.shape[0]
        num_partials = self.num_partials

        num_samples = self.get_num_samples()
        out_ = np.zeros([num_samples, num_partials])

        tmp_env = copy(self.env)

        for k in range(num_partials):
            frequency = (k + 1) * self.f0

            random_rate = self.get_random_rate()
            num_cycles = math.ceil(self.length * random_rate)
            frame_rate = num_frames * random_rate

            # Calculate partial trajectory.
            tmp = self.get_amp_from_frequency(frequency, tmp_env)

            # Cycle to desired synthesis length and resample.
            tmp = np.tile(tmp, num_cycles)
            tmp = self.loop(tmp)
            tmp = self._resample(tmp, frame_rate)
            # Truncate and place in output.
            out_[:, k] = tmp[:num_samples]

        return out_

    def get_random_rate(self):
        upper = self.random_rate_upper_limit
        lower = self.random_rate_lower_limit
        x = np.random.rand()
        return (upper - lower + 1.)**x + lower - 1.

    def cycle_and_resample_env(self):

        # Preliminaries.
        num_frames = self.env.shape[0]
        num_cycles = math.ceil(self.length * self.mod_rate)
        num_samples = self.get_num_samples()

        # Calculate only required harmonic partials.
        tmp_env = copy(self.env)
        tmp_env = self.reduce_to_relevant_partials(tmp_env)

        # Extend in time to desired output length.
        tmp_env = np.tile(tmp_env, [num_cycles, 1])

        # Wrap-around first value to extend interpolation.
        tmp_env = self.loop(tmp_env)

        # Resample.
        frame_rate = num_frames * self.mod_rate
        tmp_env = self._resample(tmp_env, frame_rate)

        # Truncate.
        tmp_env = tmp_env[:num_samples, :]

        return tmp_env

    def reduce_to_relevant_partials(self, tmp_env):
        """
        Extract only spectral information relevant to synthesis.
        """
        num_frames = tmp_env.shape[0]
        num_partials = self.num_partials

        out_ = np.zeros([num_frames, num_partials])

        for k in range(self.num_partials):
            frequency = (k + 1) * self.f0
            out_[:, k] += self.get_amp_from_frequency(frequency, tmp_env)

        return out_

    def get_amp_from_frequency(self, frequency, tmp_env):
        """
        Linear interpolate to extract frequency-wise amplitude envelope.
        """

        # Expand to facilitate broadcasting if necessary (code for 1- or 2-d).
        if tmp_env.ndim == 1:
            tmp_env = tmp_env[None, :]

        num_frames = tmp_env.shape[0]
        amp_envelope = np.zeros(num_frames)

        # Find the (possibly fractional) bin corresponding to ‘frequency‘.
        bin_num = self.get_bin_num(frequency)
        bin_fraction = bin_num % 1

        bin_floor = math.floor(bin_num)
        bin_ceil = math.ceil(bin_num)

        # Read amplitude envelope based on the desired frequency.
        if bin_fraction == 0:
            amp_envelope += tmp_env[:, bin_num]
        else:
            # Linear interpolation between adjacent bins.
            amp_envelope += (1 - bin_fraction) * tmp_env[:, bin_floor]
            amp_envelope += bin_fraction * tmp_env[:, bin_ceil]

        return amp_envelope

    def apply_spectral_fade(self, tmp_env):
        """
        Fade in spectral modulation, taking approx pi/2 of the cycle as neutral.
        """

        fade = self.get_depth_trajectory()

        mid_env = self.get_mid_env()

        # Fade out middle spectrum.
        mid_env = np.outer((1 - fade), mid_env)

        # Fade in spectral modulation.
        tmp_env = tmp_env * fade[:, None]

        # Cross-fade.
        tmp_env += mid_env

        return tmp_env

    def get_mid_env(self):
        """Retrieve spectrum from 1/4 of cycle.

        This is approximately pi/2 or 3pi/2, i.e. where the vibrato trajectory
        is in the middle of its throw (not max nor min).
        """

        mid_cycle_index = round(self.env.shape[0] // 4)
        tmp = self.env[mid_cycle_index, :]

        out_ = np.zeros(self.num_partials)

        # Interpolate for fractional bin values.
        for k in range(self.num_partials):
            frequency = (k + 1) * self.f0

            out_[k] = self.get_amp_from_frequency(frequency, tmp)

        return out_

    def _resample(self, tmp_env, frame_rate):
        return resample(tmp_env, frame_rate, self.sr)
    def synthesize(self):
        num_samples = self.get_num_samples()
        x = np.zeros(num_samples)

        if self.synth_mode == 'default' or self.synth_mode == 'raf':
            x = self.standard_synthesis(x)
        elif self.synth_mode == 'pam':
            x = self.pam_synthesis(x)
        else:
            raise ValueError("Unknown mode: {}.".format(self.synth_mode))
        return x

    def standard_synthesis(self, x):
        for k in np.arange(self.num_partials):
            x += self.make_partial(k)
        return x

    def pam_synthesis(self, x):
        """
        Returns a stimulus with a static spectral envelope, but having a global
        amplitude envelope of an equivalent spectrum-modulated signal.
        """

        average_gains = np.mean(self.processed_env, axis=0)

        # Sum all partial amplitudes into one master envelope.
        amp_envelope = np.sum(self.processed_env, axis=1)

        # Apply master envelope to each partial, scaling partials to average.
        for k in np.arange(self.num_partials):
            frequency = (k + 1) * self.f0
            gain = average_gains[k]

            x += gain * amp_envelope * self.make_carrier(frequency)
        return x

    def make_partial(self, k):
        frequency = (k + 1) * self.f0

        amp_envelope = self.processed_env[:, k]
        carrier = self.make_carrier(frequency)
        return amp_envelope * carrier

    def get_bin_num(self, frequency):
        return frequency / (self.sr // 2) * self.env.shape[1]

    def make_carrier(self, frequency):
        t = np.arange(int(self.length * self.sr))/self.sr

        # Always begins at top of cycle (i.e. cos(0)), as per analysis.py.
        trajectory = np.cos(2. * np.pi * self.mod_rate * t)

        # Shape modulation.
        trajectory *= self.get_fm_coefficient()
        trajectory *= self.get_depth_trajectory()
        trajectory += 1.

        # Apply modulation.
        trajectory *= frequency

        # Randomize initial phase.
        phi = 2 * np.pi * np.random.rand()

        phase = np.cumsum(2 * np.pi * trajectory / self.sr) + phi
        return np.cos(phase)

    def get_fm_coefficient(self):
        """
        Converts ‘fm_depth‘ from semitones into coefficient for frequency.

        Note: strictly speaking, pitch modulation should be applied in
        log-space, because pitch scales logarithmically with frequency. Here, we
        apply modulation on a linear scale. For the small modulation excursions
        associated with typical vibrato, the difference is quite minimal
        and arguably imperceptible.
        """
        return 2 ** (self.fm_depth / 12) - 1

    def get_depth_trajectory(self):
        """
        Modulation depth from 0 to 1 based on ‘mod_hold‘ and ‘mod_fade‘ times.
        """
        hold_samples = int(self.mod_hold * self.sr)
        fade_samples = int(self.mod_fade * self.sr)
        end_samples = int(self.length * self.sr - (hold_samples + fade_samples))

        hold = np.zeros(hold_samples)
        fade = np.linspace(0, 1, fade_samples, endpoint=False)
        end = np.ones(end_samples)

        return np.concatenate((hold, fade, end))

    def get_num_samples(self):
        return int(self.length * self.sr)

    @staticmethod
    def loop(in_: np.ndarray) -> np.ndarray:
        return np.concatenate([in_, [in_[0]]])


class EnvelopeMorpher:
    """
    Generate variations of spectral modulation based on a prototype cycle.
    """
    def __init__(
            self,
            env: np.ndarray,
            pr: int = PITCH_RATE,
            sr: int = SAMPLE_RATE,
            f0: float = None
    ):
        assert env.ndim == 2
        self.env = copy(env)

        assert pr > 0
        self.pr = pr

        assert sr > 0
        self.sr = sr

        if f0 is not None:
            assert 0 < f0 <= (sr // 2)
            self.f0 = f0
        else:
            self.f0 = None

        # Log tracks randomization settings, and order of morphing.
        self._log = []

    def time_average(self):
        num_frames, num_bins = self.env.shape

        tmp = np.mean(self.env, axis=0)
        tmp = np.tile(tmp, [num_frames, 1])

        self.env = copy(tmp)

    def shuffle_phase(self, num_shifts: int = 4):
        """
        Randomly shuffle each column.
        """

        assert num_shifts > 0

        all_shifts = np.linspace(0, 1, num_shifts, endpoint=False)
        num_frames, num_bins = self.env.shape

        # Used for pairing bins that surround a partial of interest.
        bins_above_partials = None
        if self.f0 is not None:
            bins_above_partials = self.get_bins_above_partials()

        tmp_log = []
        last_shift = None

        for k in np.arange(num_bins):

            # If necessary, match this shift to the previous bin.
            if self.f0 and (k in bins_above_partials):
                shift = last_shift
            else:
                shift = np.random.choice(all_shifts)

            tmp_log.append(
                {
                    'bin': k,
                    'shift': shift
                }
            )

            tmp = self.env[:, k]
            tmp = self.roll(tmp, shift)

            self.env[:, k] = tmp

            last_shift = copy(shift)

        self._log.append(tmp_log)

    def get_bin_num(self, frequency):
        return frequency / (self.sr // 2) * self.env.shape[1]

    def get_bins_above_partials(self):
        """Collect bins around overtone partials (higher in frequency only).

        In shuffling conditions, this allows for some bins to be shuffled in
        sync with one another, to avoid artifacts in synthesis later on.
        """

        out_ = []

        max_partial = int(
            (self.sr // 2) // self.f0
        )

        for p in range(1, max_partial + 1):
            frequency = self.f0 * p
            bin_number = self.get_bin_num(frequency)

            if bin_number % 1 != 0:
                out_.append(math.ceil(bin_number))

        return out_

    def get_bin_frequency(self, k):
        num_bins = self.env.shape[1]
        return k / num_bins * (self.sr / 2)
```

```
def rap(self, max_random_gain: float = None):
    """Single cycle at base-rate with (possibly) randomized gains."""

    num_frames, num_bins = self.env.shape

    # Find average gain values in the cycle for each bin.
    ave_envelope = np.mean(self.env, axis=0)

    tmp_log = []

    for bin_ in range(num_bins):

        if max_random_gain:
            # Pick random gain (in dB) between 0 and ‘max_gain‘.
            mod_gain = np.random.rand() * max_random_gain
        else:
            # Approximate partial-wise gains from envelope.
            mod_gain = self.get_partial_gain(bin_)

        # Build modulator.
        modulator = np.cos(
            2 * np.pi * np.linspace(0, 1, num_frames, endpoint=False)
        )

        modulator *= self.db_to_linear_coefficient(mod_gain)
        modulator += 1.

        # Multiply by base envelope gain.
        modulator *= ave_envelope[bin_]

        tmp_log.append(
            {
                'bin': bin_,
                'mod_gain': mod_gain
            }
        )

        # Place in array.
        self.env[:, bin_] = modulator

    self._log.append(tmp_log)
```

```
def show(self, zoom=None):
    tmp = self.env

    if zoom:
        num_bins = tmp.shape[1]
        tmp = tmp[:, :(num_bins // zoom)]

    plot_envelope(tmp, show=True)

def get_partial_gain(self, bin_):
    """Calculate the modulation gain depth, by partial, in decibels."""
    max_ = np.max(self.env[:, bin_], axis=0)
    min_ = np.min(self.env[:, bin_], axis=0)
    return 20 * np.log10(max_/min_ + EPS)

def __str__(self):
    """Print the morph log."""
    s = f"""Summary\n-------\n\nTotal morphs:\t{len(self._log)}\n"""
    for morph in self._log:
        for bin_ in morph:
            s += f"\n{bin_}"
    return s

def __call__(self):
    return self.env

@staticmethod
def db_to_linear_coefficient(decibels):
    a = 10. ** (decibels / 20) - 1
    b = 10. ** (decibels / 20) + 1
    return a / b

@staticmethod
def roll(in_, shift):
    """
    Circular shift array using linear interpolation, where 0 <= ‘shift‘ < 1
    """
    num_samples = in_.size
    shift_samples = num_samples * shift
    shift_fraction = shift_samples % 1

    out_ = np.zeros(num_samples)
```

```
        if shift_samples == 0:
            out_ += in_
        elif shift_fraction == 0:
            out_ += np.roll(in_, shift_samples)
        else:
            out_ += (1 - shift_fraction) * np.roll(in_, math.floor(shift_samples))
            out_ += shift_fraction * np.roll(in_, math.ceil(shift_samples))

        return out_


if __name__ == '__main__':
    import matplotlib.pyplot as plt
    from analysis import single_cycles

    # Helper.
    def get_fm_depth(_datum):
        """
        FM depth in semitones, calculated as half the difference of pitch.
        """
        max_ = np.max(_datum['f0'])
        min_ = np.min(_datum['f0'])
        return 12 * np.log2(max_/min_) / 2

    # Synthesis parameters.
    partials = 70

    # Midi 48 -> C3.
    midi_pitch = 48

    # Debugging.
    plots = False

    synth_out = []

    for datum in single_cycles:
        fm_depth = get_fm_depth(datum)
        f0_ = midi_to_hz(midi_pitch)

        if plots:
            t = np.arange(len(datum['f0']))/PITCH_RATE
            plt.plot(t, datum['f0'])
            plt.ylabel('Pitch (hz)')
            plt.xlabel('Time (sec)')
            plt.show()

        # Bring to linear amplitude. Env is calculated as the power spectrum.
        env_ = np.sqrt(datum['env'])

        morpher = EnvelopeMorpher(env_, f0=f0_)
        morpher.rap(max_random_gain=10)
        morpher.shuffle_phase(num_shifts=4)

        generator = StimulusGenerator(sr=SAMPLE_RATE, pr=PITCH_RATE)
        x = generator(
            f0=f0_,
            fm_depth=0.0,
            env=morpher(),
            num_partials=partials,
            length=2.1,
            mod_rate=5.,
            mod_hold=0.3,
            mod_fade=0.7,
            synth_mode='default',
            audio_fade=0.25,
        )

        stft_plot(x)

        synth_out.append(
            {
                'filename': datum['filename'],
                'f0': f0_,
                'wav': x,
            }
        )
```

## B.4 util.py

```
"""
General utilities.
"""
```

```
import librosa
import librosa.display
import matlab
import matplotlib.pyplot as plt
import numpy as np
import os
import pickle
import warnings

from librosa import load
from scipy.interpolate import interp1d
from scipy.signal import butter, filtfilt, hilbert
from typing import Union

from defaults import EPS, PITCH_RATE, SAMPLE_RATE


def add_fade(
    signal: np.ndarray,
    fade_length: float,
    rate: int,
    fade_out: bool = False,
):
    """
    Adds linear fade in/out to signal.
    """

    if fade_length > 0.:
        num_samples = int(fade_length * rate)

        # Build ramp.
        ramp = np.linspace(0, 1, num_samples, endpoint=False)

        mean = np.mean(signal)
        signal -= mean

        # Fade in/out.
        if fade_out:
            signal[-num_samples:] *= ramp[::-1]
        else:
            signal[:num_samples] *= ramp
        signal += mean

    return signal


def contains_nan(in_: np.ndarray) -> bool:
    return np.isnan(np.sum(in_))


def flatten(signal: np.ndarray) -> np.ndarray:
    """
    Replace an array values with its mean.
    """
    mean_ = np.mean(signal)
    return np.tile(mean_, len(signal))


def force_mono(signal: np.ndarray) -> np.ndarray:
    """
    Forces stereo signal to mono by averaging channels.
    """
    assert len(signal.shape) <= 2, "Mono or stereo arrays only, please."
    if len(signal.shape) == 2:
        if signal.shape[0] > signal.shape[1]:
            signal = signal.T
        signal = np.mean(signal, axis=0)
    return signal


def get_amp_envelope(signal: np.ndarray, cutoff: float = 25., sr: int = 44100):
    amplitude_envelope = np.abs(hilbert(signal))
    smoothed = low_pass(amplitude_envelope, cutoff, sample_rate=sr, order=4)
    return smoothed


def hz_to_midi(hz: np.ndarray) -> np.ndarray:
    """
    Converts from Hz to linear pitch space, where midi:69 = A440.
    """
    return np.maximum(0, 12 * np.log2((hz + EPS)/440) + 69)
```

```
def load_pickle(path: str):
    assert os.path.isfile(path), f"Missing file:\t{path}..."

    with open(path, 'rb') as handle:
        return pickle.load(handle)


def low_pass(
        signal: np.ndarray,
        frequency: float,
        sample_rate: int,
        order: int = 16
):
    """
    Convenience function for butterworth lowpass filter.
    """
    Wn = frequency/(sample_rate / 2)
    [b, a] = butter(order, Wn, btype='lowpass')

    out_ = filtfilt(b, a, signal)
    assert not contains_nan(out_), "Filtering generated NaNs."

    return out_


def matlab2np(input_: matlab.double):
    """
    Convert Matlab double to numpy array.
    """
    return np.array(input_._data)


def midi_to_hz(midi: Union[float, int, np.ndarray]) -> Union[float, np.ndarray]:
    """
    Converts from linear pitch space to Hz, where A440 = midi:69.
    """
    return 440.0 * (2.0**((midi - 69.0) / 12.0))


def normalize(x: np.ndarray) -> np.ndarray:
    """
    Normalize array by max value.
    """
    return x / np.max(np.abs(x))


def np2matlab(input_: np.ndarray):
    """
    Convert numpy array to Matlab double.
    """
    return matlab.double(input_.tolist())[0]


def num2matlab(input_: Union[float, int]):
    """
    Convert single value to Matlab double.
    """
    return matlab.double([input_])


def plot_envelope(env, show=True):
    plt.imshow(env.T, aspect='auto', origin='lower')
    if show:
        plt.show()


def read_wav(path: str):
    x, sample_rate = load(path, sr=SAMPLE_RATE, dtype=np.float64)
    return sample_rate, x


def remove_dc(signal: np.ndarray) -> np.ndarray:
    return signal - np.mean(signal)


def resample(
        env: np.ndarray,
        frame_rate: float,
        sr: int,
) -> np.ndarray:
    """
    Resample spectral envelope array in time.
    """
    axis = -1
    if env.ndim == 2:
        axis = 0

    _indices = np.arange(env.shape[0]) * sr / frame_rate
    f = interp1d(_indices, env, kind='linear', axis=axis)

    num_samples = int(
        round((env.shape[0] - 1) * sr / frame_rate)
    )

    return f(np.arange(num_samples))


def safe_mkdir(path):
    if not os.path.exists(path):
        Warning(f"Creating directory {path}...")
        os.mkdir(path)


def save_pickle(path: str, data, force: bool = False):
    if force is False:
        assert not os.path.isfile(path), 'File {} already exists.'.format(
            os.path.basename(path)
        )

    print('Saving file {}...'.format(os.path.basename(path)))
    with open(path, 'wb') as handle:
        pickle.dump(data, handle, protocol=pickle.HIGHEST_PROTOCOL)


def stft_plot(
    signal: np.ndarray,
    sample_rate: int = SAMPLE_RATE,
    title: str = "",
    show: bool = True
):
    X = librosa.stft(signal)
    Xdb = librosa.amplitude_to_db(abs(X))
    plt.figure(figsize=(5, 5))
    plt.title(title)
    librosa.display.specshow(Xdb, sr=sample_rate, x_axis="time", y_axis="linear")
    if show:
        plt.show()


def time_plot(
        signal: np.ndarray,
        rate: int = 44100,
        show: bool = True,
        title: str = None
):
    t = np.linspace(0, len(signal)/rate, len(signal), endpoint=False)
    plt.plot(t, signal)
    plt.xlabel('time (s)')
    plt.ylabel('amplitude')
    if title:
        plt.title(title)
    if show:
        plt.show()


def trim_to_duration(
    signal: np.ndarray,
    time_in: float = 1.,
    duration: float = 1.,
    rate: int = 44100
) -> np.ndarray:
    """
    Trim audio signal given start time and desired duration.
    """
    in_ = int(time_in * rate)
    out_ = in_ + int(duration * rate)
    return signal[in_:out_]


def trim_silence(
    signal: np.ndarray,
    threshold: float = -35,
    cutoff: float = 25.,
    sr: int = 44100,
) -> np.ndarray:
    """
    Trims beginning of audio signal until it passes a given threshold in dB.
    """
    amp_envelope = get_amp_envelope(signal, cutoff, sr)
    log_envelope = np.log(amp_envelope + EPS)
    start_index = np.maximum(
        np.where(log_envelope >= threshold)[0][0],
        0
    )
    return signal[start_index:]


def upsample(
        hz: np.ndarray,
        sr: int = SAMPLE_RATE,
        pr: int = PITCH_RATE
) -> np.ndarray:
    _indices = np.arange(len(hz)) * sr / pr
    f = interp1d(_indices, hz, kind='cubic')

    num_samples = int(
        round((len(hz) - 1) * SAMPLE_RATE / PITCH_RATE)
    )

    return f(np.arange(num_samples))
```

## B.5 feature_extraction.py

```
"""
Tools for extracting timbral features from the stimuli.
"""

from glob import glob
import numpy as np
import matplotlib.pyplot as plt
import matlab.engine
import os
import pandas as pd
from tqdm import tqdm

from ext.auditory import strf
from src.defaults import DATA_PATH, TIMBRE_TOOLBOX_PATH, SYN_PATH
from src.util import matlab2np, np2matlab, read_wav, save_pickle
```

```
def extract_trials(df):
    df = df[df['trial_type'] == 'audio-slider-response']
    df = df[~df['stimulus'].str.contains("train")]
    return df


def init_matlab():
    print('Starting Matlab engine...')
    eng = matlab.engine.start_matlab()
    eng.addpath(eng.genpath(TIMBRE_TOOLBOX_PATH))
    return eng


def load(datapath=DATA_PATH):
    pattern = os.path.join(datapath, 'prolific/*.csv')
    files = glob(pattern)
    assert files, 'No csv data found.'

    df = pd.DataFrame()

    for file in files:
        df = df.append(pd.read_csv(file))

    return df


def make_modulation_representation(_path):
    """
    Return time-averaged, complex valued STRF. (freq x scale x rate)
    """
    sr, x = read_wav(_path)
    tmp = strf(x, sr, duration=-1)
    tmp = np.abs(tmp)
    return np.mean(tmp, axis=0)


def replace_path_to_local(_path):
    """
    Replace path with path to local file.
    """
    dir_, file_ = os.path.split(_path)
    tmp = dir_.split("/")
    tmp[0] = SYN_PATH
    tmp = '/'.join(tmp)

    return os.path.join(tmp, file_)


def timbre_toolbox(filepath, _eng):
    """
    Coupled to script ‘pyTimbre.m‘
    """

    _data = _eng.pyTimbre(filepath, nargout=5)

    descriptor_names = [
        'energy',
        'spectral_centroid',
        'spectral_crest',
        'spectral_flatness',
        'odd_even_ratio',
    ]

    out_ = {}

    for i, _datum in enumerate(_data):
        out_[descriptor_names[i]] = matlab2np(_datum)

    return out_


if __name__ == '__main__':

    # Flags.
    use_timbre_toolbox = False
    use_auditory_model = True

    # Pickle file paths.
    timbretoolbox_name = 'TT_features.pickle'
    timbretoolbox_pickle_path = os.path.join(DATA_PATH, timbretoolbox_name)

    auditory_name = 'modulation_features.pickle'
```

```
auditory_pickle_path = os.path.join(DATA_PATH, auditory_name)

# Generate data.
eng = init_matlab()

df = load()
df = extract_trials(df)

paths = df['stimulus'].tolist()

all_tt_data = []
all_aud_data = []

for path in tqdm(paths):
    localpath = replace_path_to_local(path)

    if use_timbre_toolbox:
        tt_data = timbre_toolbox(localpath, eng)
        tt_data['stimulus'] = path

        all_tt_data.append(tt_data)

    if use_auditory_model:
        aud_data = dict()
        aud_data['strf'] = make_modulation_representation(localpath)
        aud_data['stimulus'] = path

        all_aud_data.append(aud_data)

if use_timbre_toolbox:
    save_pickle(timbretoolbox_pickle_path, all_tt_data, force=False)
if use_auditory_model:
    save_pickle(auditory_pickle_path, all_aud_data, force=False)

# if debug:
#     from src.util import load_pickle
#     import pandas as pd
#
#     tmp = load_pickle(pickle_path)
#     tmp = pd.DataFrame(tmp)
#
#     test = pd.merge(tmp, df, on='stimulus')
```

```
    #     print(test)
```

## B.6 data_util.py

```
from glob import glob
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import QuantileTransformer

from defaults import DATA_PATH, MAX_INTEGER


def anova_prep(df):
    df = df.groupby(['subjectNo', 'condition'])['response'].mean()
    df = df.unstack()
    tmp = []
    for subject in df.index:
        for condition in df.loc[subject].keys():
            tmp.append(
                {
                    'subjectNo': subject,
                    'condition': condition,
                    'rating': df.loc[subject, condition],
                }
            )
    return pd.DataFrame(tmp)


def average_condition_rating_within_subject(df):
    tmp = df.groupby(['subjectNo', 'condition'])['response'].mean()
    return tmp.unstack()


def average_std_of_ratings(df):
    return df.groupby(['subjectNo', 'condition'])['response'].std().\
        groupby('condition').mean()


def box_plot(df, study_type, savefig=False, dpi=300):
    tmp = df.groupby(['subjectNo', 'condition'])['response'].mean()
    tmp = tmp.unstack()
    tmp.columns = [s.replace('_', ' ') for s in tmp.columns]
    plt.figure(figsize=(16, 6))
    sns.boxplot(data=tmp)
    plt.ylabel('rating')
    plt.xlabel('condition')
    if savefig:
        plt.savefig(f"figs/study_type_{study_type}_boxplot.png", dpi=dpi)
    else:
        plt.show()


def extract_condition(df):
    if 'condition' not in df:
        def process(x):
            return "_".join(x[-1].split('_')[:-1])

        df['condition'] = df['stimulus'].str.split('/').apply(process)
    return df


def extract_subject(df):
    if 'subjectNo' not in df:
        df['subjectNo'] = df['stimulus'].str.split("/").apply(
            lambda x: x[1].split("_")[1])
    return df


def extract_trials(df):
    df = df[df['trial_type'] == 'audio-slider-response']
    df = df[~df['stimulus'].str.contains("train")]
    return df


def filter_by_basic(df, threshold=0.6):
    """Find subjectNo where the BASIC condition was rated below threshold."""
    tmp1 = df[df['condition'] == 'BASIC'].groupby(['subjectNo'])[
                'response'].min() > threshold
    tmp2 = tmp1[tmp1]
    print(f"N = {len(tmp2)}")
    return df[df['subjectNo'].isin(tmp2.keys())]
```

```
def filter_by_control(df, threshold=0.6):
    """Find subjectNo where the CONTROL was rated greater than threshold."""
    tmp1 = df[df['condition'] == 'CONTROL'].groupby(['subjectNo'])[
               'response'].min() > threshold
    tmp2 = tmp1[tmp1]
    print(f"N = {len(tmp2)}")
    return df[df['subjectNo'].isin(tmp2.keys())]


def get_good_participants(num_reject=2):
    """Filter participants by number of rejections"""

    pattern = os.path.join(DATA_PATH, 'participant_demographic_data/*.csv')
    files = glob(pattern)

    df = pd.DataFrame()

    for i, file in enumerate(files):
        tmp = pd.read_csv(file)
        tmp['phase'] = i
        df = df.append(tmp)

    return df.query(f'status == "APPROVED" and num_rejections <= {num_reject}')[
        'participant_id']


def get_num_subjects(df):
    return len(df['subjectNo'].unique())


def get_summary(df):
    tmp1 = df[['condition', 'response']].groupby('condition').mean()
    tmp2 = df[['condition', 'response']].groupby('condition').std()

    tmp3 = pd.DataFrame()
    tmp3['mean'] = tmp1['response']
    tmp3['std'] = tmp2['response']
    return tmp3
```

```
def group_quantile_transform(series):
    quantiler = QuantileTransformer()
    return np.squeeze(quantiler.fit_transform(series.values.reshape(-1, 1)))


def isolate_study(df, study_type):
    assert not df[df['studyType'] == study_type].empty, 'Returns no trials.'
    return df[df['studyType'] == study_type]


def load(pattern='prolific/*.csv'):
    pattern = os.path.join(DATA_PATH, pattern)

    files = glob(pattern)
    assert files, 'No csv data found.'

    df = pd.DataFrame()

    for file in files:
        df = df.append(pd.read_csv(file))

    return df


def load_and_clean_data(num_reject=10000):
    df = load()
    df = extract_trials(df)
    df = normalize_slider(df)
    df = extract_subject(df)
    df = extract_condition(df)
    df = min_max_norm(df)

    # Filter by participants having few rejections.
    gp = get_good_participants(num_reject=num_reject)
    df = df[df['prolificID'].isin(gp)]

    df = df.reset_index(drop=True)
    return df.drop(
        ['view_history', 'trial_type', 'internal_node_id', 'studyID',
         'sessionID', 'url', 'slider_start'],
        1
    )
```

```
def load_tt_descriptors():
    tmp = pd.read_csv('./timbre_toolbox_features/Median_PowSTFTrep.csv')
    tmp.columns = ["STFT__" + col + "Med" for col in tmp.columns]
    stft_median_df = tmp.rename(columns={'STFT__SoundFileMed': 'stimulus'})

    tmp = pd.read_csv('./timbre_toolbox_features/IQR_HARMrep.csv')
    tmp.columns = ["HARMONIC__" + col + "IQR" for col in tmp.columns]
    harmonic_iqr_df = tmp.rename(columns={'HARMONIC__SoundFileIQR': 'stimulus'})

    tmp = pd.read_csv('./timbre_toolbox_features/IQR_PowSTFTrep.csv')
    tmp.columns = ["STFT__" + col + "IQR" for col in tmp.columns]
    stft_iqr_df = tmp.rename(columns={'STFT__SoundFileIQR': 'stimulus'})

    tmp = pd.read_csv('./timbre_toolbox_features/Median_HARMrep.csv')
    tmp.columns = ["HARMONIC__" + col + "Med" for col in tmp.columns]
    harmonic_med_df = tmp.rename(columns={'HARMONIC__SoundFileMed': 'stimulus'})

    tmp = pd.merge(stft_median_df, harmonic_iqr_df, on='stimulus')
    tmp = pd.merge(tmp, stft_iqr_df, on='stimulus')
    tmp = pd.merge(tmp, harmonic_med_df, on='stimulus')

    def reformat_stimulus(x):
        _tmp = x.replace('__', '/')
        return 'audio/' + _tmp + '.wav'

    tmp['stimulus'] = tmp['stimulus'].transform(reformat_stimulus)
    return tmp


def max_time_elapsed(df):
    """Returns the max time elapsed in minutes."""
    return df.groupby('subjectNo')['time_elapsed'].max() / 1000 / 60


def min_max_norm(df):
    min_ = df.groupby('subjectNo')['response'].transform('min')
    max_ = df.groupby('subjectNo')['response'].transform('max')
    df['response'] = (df['response'] - min_) / (max_ - min_)
    return df
```

```
def normalize_slider(df):
    already_normalized = (df[['response', 'slider_start']] <= 1).all().all()
    if not already_normalized:
        df[['response', 'slider_start']] = df[['response',
                                               'slider_start']] / MAX_INTEGER
    return df


def response_histograms(df, bins=20):
    # Get subject's average rating per condition.
    tmp = average_condition_rating_within_subject(df)
    for i, col in enumerate(tmp):
        plt.subplot(1, 2, (i % 2) + 1)
        plt.title(col)
        plt.hist(tmp[col], bins=bins)
        if i % 2 == 1:
            plt.show()


def within_subject_correlation(_df, _feature, _method):
    """Group by subject, get correlation with response, mean over subjects."""
    return _df.groupby('subjectNo')[_feature].corr(
        _df['response'], method=_method
    ).mean()
```

## B.7 analysis.py

```
"""
Trim, condition, extract pitch and spectral envelope from signals.
"""


import os
import pyworld as pw

from glob import glob
from scipy.signal import find_peaks, decimate

from defaults import ANA_PATH, PITCH_RATE
from util import (
    force_mono,
    get_amp_envelope,
    low_pass,
    normalize,
    read_wav,
    trim_to_duration,
    trim_silence
)

# Flags.
VERBOSE = False

# Analysis parameters.
excerpt_in = 0.75
excerpt_dur = 1.75
silence_db = -5
pitch_lp = 10
which_peak = 4
amp_env_cutoff = 25.

pattern = os.path.join(ANA_PATH, '*.wav')
audio_files = glob(pattern)
assert audio_files, "Pattern {} yields no results.".format(pattern)

# Calculations.
pitch_period_ms = 1/PITCH_RATE * 1000

single_cycles = []

for path in audio_files:
    basename = os.path.basename(path)

    if VERBOSE:
        print('Reading {}...'.format(basename))

    sr, x = read_wav(path)

    x = force_mono(x)
    x = normalize(x)
    x = trim_silence(x, threshold=silence_db, sr=sr)
    x = trim_to_duration(x, excerpt_in, excerpt_dur)

    if VERBOSE:
        print('WORLD analysis...')

    _f0, t = pw.dio(x, sr, frame_period=pitch_period_ms)
    f0 = pw.stonemask(x, _f0, t, sr)
    sp = pw.cheaptrick(x, f0, t, sr)
    ap = pw.d4c(x, f0, t, sr)

    # Remove the first sample, which is always 0 Hz.
    tmp_f0 = f0[1:]
    tmp_f0 = low_pass(tmp_f0, pitch_lp, PITCH_RATE)

    # Add ‘1‘ to compensate for the subtracted index above.
    peaks = find_peaks(tmp_f0)[0]
    peaks += 1

    # Extract one vibrato period.
    assert which_peak < len(peaks)
    if VERBOSE:
        print('Detected {} peaks. Choosing peak {}.'.format(
            len(peaks), which_peak
        ))

    start = peaks[which_peak]
    end = peaks[which_peak + 1]

    start_sr = int(round(start / PITCH_RATE * sr))
    end_sr = int(round(end / PITCH_RATE * sr))

    # TODO
    # amp_env = get_amp_envelope(x, cutoff=amp_env_cutoff, sr=sr)
    # amp_env = amp_env[start_sr:end_sr]

    single_cycles.append(
        {
            'filename': basename,
            'env': sp[start:end, :],
            'f0': f0[start:end],
            'sr': sr,
        }
    )
```

## B.8 defaults.py

```
"""
Defaults and globals.

Note, users will have to specify their own path to the Timbre Toolbox.
"""

import os


class RealPath:
    """
    Convenient way to generate absolute file-paths.
    """
    def __init__(self):
        self.here = os.path.dirname(__file__)

    def __call__(self, relative_path):
        return os.path.realpath(
            os.path.join(self.here, relative_path)
        )


# Small value.
EPS = 1e-8

# Large value for data analysis.
MAX_INTEGER = 2**53 - 1

# Sample rates.
SAMPLE_RATE = 44100
PITCH_RATE = 200

# Relevant file paths.
real_path = RealPath()

ANA_PATH = real_path('../audio/ana')
SYN_PATH = real_path('../audio/syn')
DATA_PATH = real_path('../data')
TIMBRE_TOOLBOX_PATH = real_path('../matlab/timbretoolbox')
```