

Interactive Learning of Signal Processing Through Music

*Meinard Müller**, *Brian McFee†*, *Katherine M. Kinnaird‡*

*International Audio Laboratories Erlangen, Germany, meinard.mueller@audiolabs-erlangen.de

†New York University, USA, brian.mcfee@nyu.edu

‡Smith College, Northampton MA, USA, kkinnaird@smith.edu

In this article, we show how music may serve as a vehicle to support education in signal processing. Using Fourier analysis as a concrete example, we show how the music domain provides motivating and tangible applications that make learning signal processing an interactive pursuit. Furthermore, we indicate how software tools, originally developed for music analysis, provide students multiple entry points to delve deeper into classical signal processing techniques, while bridging the gap between education and cutting-edge research.

Introduction

Music is a ubiquitous and vital part of our lives. Thanks to the proliferation of digital music services such as Spotify, Pandora, and iTunes, we can enjoy music anytime and anywhere, interacting with it in a variety of ways, both as listeners and active participants. Aside from human speech, music may be the most familiar form of structured audio to most people. Conversely, as a scientific discipline, signal processing can be obtuse and unfamiliar to newcomers. Conceptual and practical understanding of signal processing requires a rather sophisticated knowledge of advanced mathematics, which can make the subject intimidating even at the introductory level. In this article, we show how music may serve as a vehicle to make learning signal processing an interactive pursuit, whether through concrete examples, hands-on exploration, or through experimentation. The inclusion of music bridges the gap between the humanities and more typical signal processing communities such as mathematics, computer science, and engineering. This is the reason why one can find music as an integral part of books on multimedia and audio signal processing [1, 2]. Throughout this article, we show how music yields an intuitive entry point to support education on various levels. This leads to a learning approach that Guzdial [3] calls a “contextualized educational experience” for signal processing.

This article presents a scaffold for incorporating interactive music-based examples and music technology into an existing signal processing course. The proposed pipeline moves students through Bloom’s taxonomy (Figure 1), helping them transition from passive learners to engaged researchers and practitioners [4,5]. This scaffolding resembles “legitimate peripheral

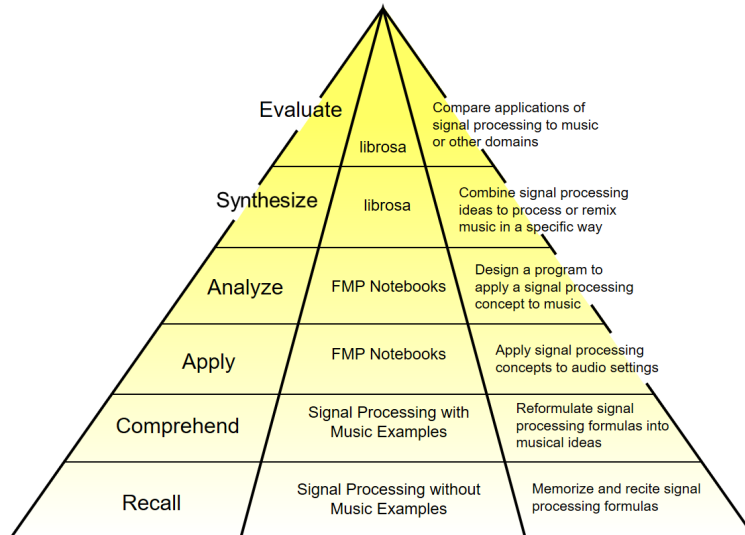


Figure 1: Adaptation of Bloom’s taxonomy [4], based on levels from Starr et al. [5]. Students begin at the lowest level. Layering music examples, the FMP notebooks, and **librosa** can help them transition to the highest levels of understanding.

participation” [6] aligning classroom learning more closely to the apprentice model of how signal processing is actually researched and practiced. Presented as a series of small but meaningful steps, this scaffold adds music to one’s signal processing course, transitioning it from a standard lecture signal processing course into one that is interactive and project-based. Each step provides a more context-rich signal processing course than the previous step. The examples in the presented scaffold are informed by the field of *music information retrieval* (MIR), which has interests in extracting semantic content from audio signals.

The article is organized in two main parts. In the first part, we highlight how music processing can serve as a tangible and approachable real-world application of signal processing methods. As such, music-based examples aid students moving from recalling and reciting signal processing concepts (the lowest level of Bloom’s taxonomy) towards comprehension and application (the second and third levels). In particular, we give a gentle introduction to Fourier analysis motivated by music and discuss basic properties of music signals via Fourier analysis.

In the second part, we discuss the role of software tools in signal processing education and detail how they add varying depths of interaction, supporting students’ development towards independent work in signal processing. We first explore constrained interaction through the FMP notebooks [7], which closely follow parts of the textbook “Fundamentals of Music Processing” [8] and which provide interactive activities to enhance teaching and learning classical signal processing techniques. As such, the FMP notebooks carry students

further up Bloom’s taxonomy, solidifying their abilities to apply signal processing concepts in musical examples. From the instructor’s perspective, the inclusion of the FMP notebooks adds interactive elements to the typical lecture-based signal processing classroom with little effort on the instructor’s part. We then discuss incorporating the Python package `librosa` [9], which enables broader experimentation with signal processing concepts. With the experience gained using the FMP notebooks, students have the technical skills and the conceptual fluency to synthesize their signal processing knowledge. Then, adopting and creating programming scripts from the elements in `librosa` allows students to delve deeper into the implications—in a music-context—of altering and exploring the role of various parameters common in signal processing. From the instructor’s perspective, including `librosa` into a course can add depth to examples in a typical lecture-based course as well as provide the ingredients for an end-of-semester project.

Music Domain

A typical introductory course on digital signal processing covers a range of topics, including but not limited to: sampling theory, Fourier analysis and the discrete Fourier transform (DFT), convolution and filtering, time-frequency representations, and so on. Although these topics are each fundamental and broadly applicable to a wide array of settings, they can also be conceptually difficult to grasp for newcomers.

When teaching a challenging concept, instructors attempt to find a compelling example that their students can hold onto through the sea of equations and subtleties. For signal processing, music can be that motivating example, and help anchor the abstract concepts in a concrete, familiar *context*. This kind of contextualized pedagogical practice has been shown to improve student retention in computer science curricula [3]. As such, using music as a context for signal processing gives students an avenue for explaining signal processing in their own words, which moves students from simply recalling formulas and reciting signal processing facts (the lowest level of learning on Bloom’s taxonomy) to deeper levels of comprehension.

As a multimedia domain, music offers a wide range of data types and formats including text, symbolic data, audio, image, and video [8, 10]. For example, music can be represented as printed sheet music (often available in the form of digitized images), encoded as MIDI (Musical Instrument Digital Interface) or MusicXML files (structured textual data), and played back as audio recordings. In this article, our primary focus is on *music signals* or *audio representations*, which encode acoustic waves as generated by an instrument (or voice) and

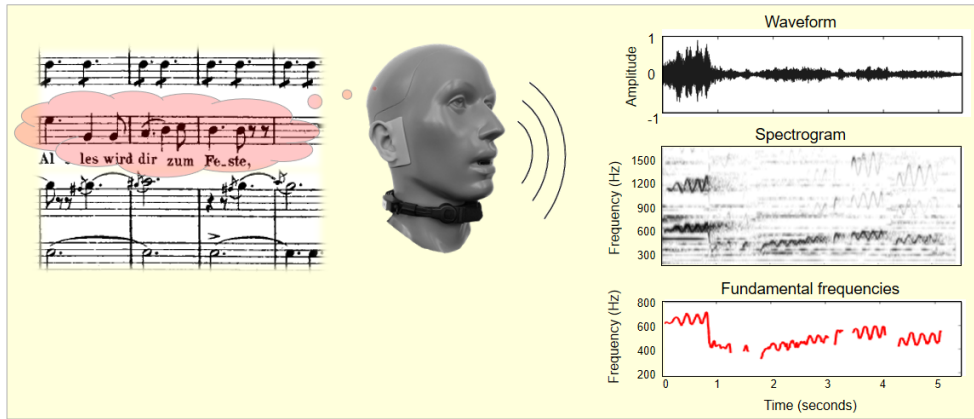


Figure 2: Singing scenario, illustrated by a short excerpt of the romantic opera “Der Freischütz” by Carl Maria von Weber. A singer performs a melody accompanied by some instruments. Analyzing the resulting music recording (waveform) leads to challenging signal processing problems (e.g., extracting the singer’s fundamental frequency trajectory).

transmitted through the air as pressure oscillations. As opposed to score and most symbolic representations, an audio representation encodes information needed to reproduce a specific acoustic realization of a piece of music. This includes the temporal, dynamic, and tonal micro-deviations that make up the particular performance style of a musician. However, in an audio representation, note parameters such as onset times, pitches or note durations are not given explicitly.

Audio representations of music connect to several standard concepts in signal processing. In Figure 2, we have the musical score, which is performed to create an audio representation. We can then derive several signal processing concepts, including the waveform, spectrogram, and fundamental frequencies (as shown in the right side of Figure 2). Constructing these different representations and concepts can be a challenging signal processing problem, which in turn provide a motivating example to understand and apply signal processing concepts appropriately. As a small side note, we want to mention in this context that the design of computational approaches for converting an acoustic music signal into some form of music notation—a task commonly referred to as automatic music transcription—is one of the most challenging and fascinating research problems in signal processing and artificial intelligence. As detailed by Benetos et al. [11], it comprises several subtasks, including multipitch estimation, onset and offset detection, instrument recognition, beat and rhythm tracking, interpretation of expressive timing and dynamics, and score typesetting, to name a few. In the remainder of this section, we explore the connection between music and signal processing and how incorporating music examples into a standard signal processing course provides students with a method for a more nuanced understanding of signal processing.

Understanding Fourier Analysis Through Music

Music signals present numerous opportunities to pose natural questions which may be understood by a novice. At very short time-scales, one might want to know the fundamental frequency of a played note; at longer time-scales, one may ask about the timbre of different instruments; moving to even longer time-scales allows us to inquire about higher-level concepts such as melody and rhythm. Each of these questions can be addressed by different applications of *Fourier analysis*, and exposure to the same basic principle in multiple related contexts can help solidify a student's understanding. Both the familiarity and complexity of music makes it an ideal motivation and a vehicle for learning signal processing.

The key idea underlying Fourier analysis is to represent arbitrary signals as combinations of *sinusoids*. When introducing Fourier analysis, it is natural to start with *simple* combinations, consisting of only a single sinusoid, i.e., a *pure tone*. A sinusoid is completely specified by three parameters: its *frequency* (the number of oscillations per second, measured in Hertz denoted Hz), its *amplitude* (the peak deviation of the sinusoid from its mean), and its *phase* (determining where in its cycle the sinusoid is at time zero). Thinking of frequency as the rate of vibration, it is easy to understand that the higher the frequency of a sinusoidal wave, the higher it sounds. This physical analogy yields an intuition for signal processing. For example, a sinusoid having a frequency of 440 Hz (physical attribute) corresponds to the pitch A4 (musical and perceptual attribute). Similarly, the amplitude of a sinusoidal wave relates to the intensity of the sound from a musical instrument. With this intuition in mind, the aim of Fourier analysis can be interpreted as a kind of reverse engineering problem. Given a music signal, the aim is to measure the intensity with which a sinusoidal wave of a given frequency occurs in (or, more precisely, correlates with) the signal. The collection of intensity values for all frequencies (concealing the role of the phase for the moment) is commonly referred to as *Fourier transform*. Plotting the intensity values over a frequency axis yields a visualization that reveals the signal's frequency spectrum.

As an example, let us consider the note C4 having a fundamental frequency of 261.6 Hz. When playing this note on different instruments, we hear different sounds. Figure 3 shows the waveforms for C4 when played on a piano, trumpet, violin, and flute. Looking at the respective Fourier transform (frequency–intensity plot), one can observe peaks at the fundamental frequency $f = 261$ Hz and its harmonics $2f$, $3f$, $4f$, and so on. However, these plots are not identical. While the peak values drop with increasing frequency for the piano

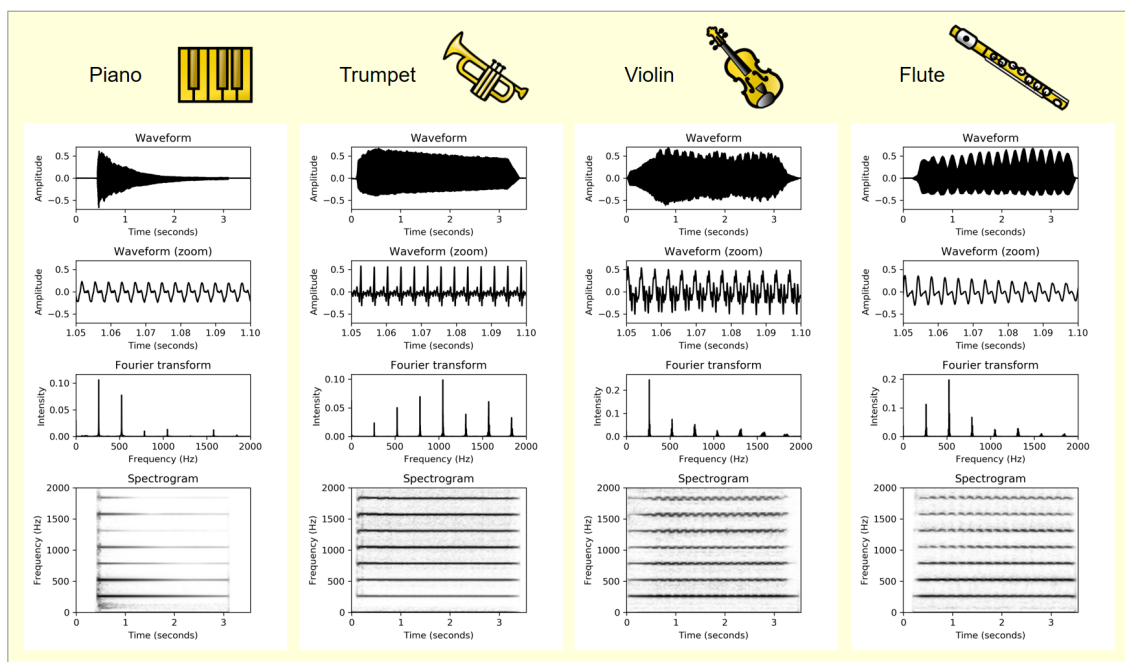


Figure 3: Waveform, Fourier transform, and spectrogram for different instruments playing the same note C4 (fundamental frequency 261.6 Hz).

and the violin after $f = 261$ Hz, the highest peak occurs at the fourth harmonic ($4f = 1046$ Hz) for the trumpet and at the second harmonic ($2f = 523$ Hz) for the flute. The distribution of the signal's energy across the harmonics is one important characteristic for the *timbre* or *tone color* of an instrument.

The Fourier transform yields frequency information that is integrated over the entire time domain, but most signals are not stationary, and their frequency contents change over time. This observation leads to another central technique in signal processing: *short-time Fourier transform* (STFT). Instead of considering the entire signal at once, the main idea of the STFT is to partition the signal into small sections in time and consider each smaller section individually. To this end, one fixes a *window function*, which is a function that is nonzero for only a short period of time (defining the considered section). The original signal is then multiplied with the window function to yield a *windowed signal*. To obtain frequency information at different time instances, one shifts the window function across time-axis and computes a Fourier transform for each of the resulting windowed signals. The STFT yields a two-dimensional representation of the original signal, which can be visualized by means of a two-dimensional image known as *spectrogram*. In this image, the horizontal axis represents time and the vertical axis represents frequency. Musical signals provide a concrete demonstration of the utility of the STFT: changes in pitch, loudness, or other musically salient characteristics are directly observable in the spectrogram image.

The distinction between the STFT and a standard Fourier transform can often be confusing to students. Additionally, the STFT introduces new parameters not present in the standard Fourier transform, notably the window length (also called *frame length*) and the frame rate (as dictated by the hop length, or number of samples between successive frames). Without a grounding context, these parameters have no obvious default setting, and students may not immediately grasp the effect of these parameters on the resulting analysis. However, musical signals provide a means for demonstrating the effects of these parameters, and by appealing to basic psychoacoustics, provide a way to connect the values of these parameters to real phenomena: for example, the frame length can be connected to a minimal (perceptible) non-trivial analysis frequency. In this setting, music provides a distinct advantage over other example stimuli (e.g. speech or environmental sound), as it is relatively easy to find (or construct) musical examples which probe the extremal cases of STFT parameters to demonstrate behavior.

Returning to Figure 3, which shows the spectrograms for note C4, one can observe horizontal lines that are stacked on top of each other for all four instruments. These equally spaced lines correspond to partials—sinusoidal sound components that are not necessarily but often close to harmonics. In case of the piano, the higher partials contain less and less of the signal’s energy. Furthermore, the decay of a piano sound over time is reflected by the fading out of the horizontal lines. For the trumpet sound, the spectrogram shows that the signal’s energy is concentrated more in the higher partials. Also, opposed to the piano sound, there does not seem to be any intensity decay over time, indicating that the trumpet player keeps the volume of the sound constant. While this is also the case for the violin and flute sounds, one can observe other phenomena that typically go along with certain playing styles such as *vibrato*. For example, when looking at the waveform, one can observe periodic variations in amplitude, also referred to as *amplitude modulation*. In the spectrogram (in particular visible in the flute example), these variations appear as regular pulsation of intensity values along the time dimension. Amplitude modulations often go along with *frequency modulations*, which are regular, pulsating changes of frequency over time. In the spectrogram (in particular visible in the violin example), these modulations appear as wave-like oscillations along the time dimension. Both amplitude and frequency depend on two parameters: the extent of the variation and the rate at which the amplitude or frequency is varied. Even though being simply local changes in intensity and frequency, the modulations do not necessarily evoke a perceived change in loudness or pitch of the overall musical tone. Rather, they are features

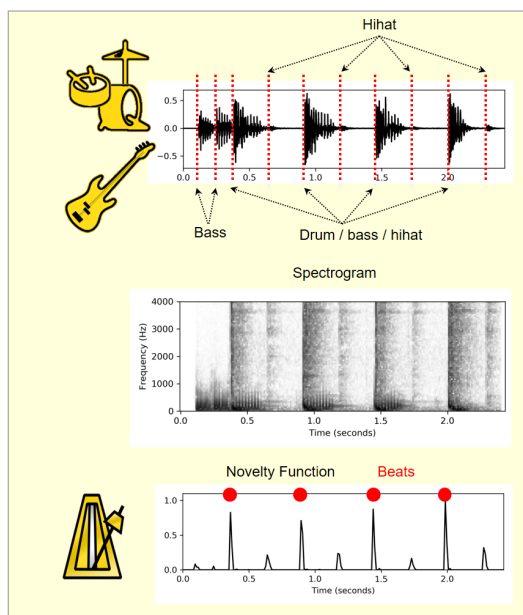


Figure 4: Short excerpt of “Another One Bites the Dust” by Queen. The figure shows the waveform with annotated onset positions, a spectrogram, and a novelty function with annotated beat positions.

that are used by musicians to influence the timbre of a musical tone.

Practical Application of Fourier Analysis to Music

So far, we have established that music can provide intuition for the Fourier transform and its short-time version, and we have demonstrated how the Fourier transform and STFT can provide an intuitive understanding of a music signal’s properties. With this connection between music and signal processing established, music scenarios such as singing (see Figure 2) can motivate students to explore the potential of Fourier analysis in an interactive and playful fashion. Most students and schoolchildren are familiar with music video games (e.g, SingStar or Rock Band), where the task is to sing along with music in order to score points. To compare the singer’s input waveform with the game’s reference melody, one could employ Fourier analysis. This makes it possible to convert the waveform into a sequence (or trajectory) of fundamental frequency values. Such trajectories are often made visible even in the video games, superimposed with piano-roll like visualizations of reference pitches. One can mimic the basic idea of such games by employing realtime-capable software for visualizing the frequency content of sounds while singing. Deepening the understanding of signal properties, such software also allows students to experiment with algorithmic parameters that control the STFT’s time and frequency resolution as well as the intensity visualization (e.g., switching from a linear to a decibel scale).

Besides analyzing melodic properties of music signals, the Fourier transform can be applied

for many more music processing tasks including harmony analysis, instrument recognition, rhythmic analysis, and source separation. In the following, we examine *beat tracking*. Temporal and structural regularities are perhaps the most important incentives for people to get involved and to interact with music [8, 12]. It is the *beat* that drives music forward and provides the temporal framework of a piece of music. Intuitively, the beat corresponds to the pulse a human taps along when listening to music [13]. The term *tempo* (often specified in beats per minute or BPM) refers to the rate of the pulse and is given by the reciprocal of the beat period.

The beat tracking task seeks to extract beat and tempo information from audio recordings, and it is one of the central and most well-studied research problems in MIR. We will now discuss why beat tracking is an instructive, challenging, and multi-faceted application for teaching and learning signal processing. Most approaches to beat tracking are based on two assumptions: first, the beat positions correspond to note onsets (often percussive in nature) and, second, beats are periodically spaced in time. Note that, for certain types of music, these two assumptions may be questionable. For example, in passages with syncopation beat positions may not go along with any onsets, or the periodicity assumption may be violated for romantic piano music with strong tempo fluctuations (played *rubato*). The explicit modeling of such simplifying assumptions is at the core of researching and teaching music processing.

Our two assumptions above motivate approaching beat-tracking in two steps. Consider the short excerpt of “Another One Bites the Dust” by Queen, shown in Figure 4, for a concrete visualization as we examine each of these steps. In the first step, one often estimates the positions of starting times of notes or other musical events as they occur in a music signal—a task commonly referred to as *onset detection*. As shown in Figure 4, onsets often go along with a sudden change in a signal’s properties. Such changes may be seen as sharp amplitude increases in the waveform. For notes with soft onsets or complex music with several instruments playing at the same time, the detection of individual note onsets becomes much harder. In these cases, converting the signal into a spectrogram turns out beneficial. In particular, percussive onsets, as produced by a drum or hi-hat, result in vertical lines in the spectrogram. This phenomenon comes from Fourier analysis: the energy of transient events is spread across the entire spectrum of frequencies, thus yielding broadband spectral structures. To detect these structures, one basic idea is to compute a kind of distance between subsequent column vectors of the spectrogram. This results in a *novelty function* (also known as the *spectral flux*), which captures sudden changes in the signal’s frequency distribution.

The peaks of such a novelty function are good indicators for note onset candidates.

In a second step, based on the assumption that beats are periodically spaced in time, the novelty function is analyzed with regard to reoccurring patterns. This is particularly suitable, intuitive, and concrete setting for studying different techniques of periodicity analysis—a central concern of signal processing and time-series analysis. One approach based on autocorrelation analysis aims to detect periodic self-similarities by comparing a novelty function with time-shifted copies [14]. An alternative approach uses a bank of comb filter resonators, where a novelty function is compared with templates consisting of equally spaced spikes, each template representing a specific tempo [15]. A third approach compares the novelty function with sinusoidal templates, each corresponding to a specific frequency. This is exactly the idea of Fourier analysis, yielding a frequency representation of the novelty function. We will come back to this last approach when introducing the FMP notebooks (see also Figure 7).

Starting with playing and listening to music, a teacher can smoothly transition to introducing basic concepts in signal processing. Singing analysis and beat tracking are two tangible example tasks that help transition learning signal processing from one of rote memorization to a contextually meaningful pursuit. Additional examples can be found in the “Fundamentals of Music Processing” textbook [8]. Although our focus in this section has been on applications of basic Fourier analysis, many music analysis tasks can be easily adapted to more advanced topics, such as wavelet theory [16]. In the next section, we will discuss educational software tools for teaching and learning such concepts.

Educational Software Tools

In addition to motivating and tangible music-based scenarios, the availability of suitably designed software packages that make signal processing more accessible are crucial in view of interactive learning [17]. Over the last 20 years, as MIR developed as a research field, so did computational accessibility, and the MIR community has contributed with several excellent toolboxes that provide modular source code for processing and analyzing music signals. Prominent examples are `essentia` [18], `madmom` [19], `Marsyas` [20], or the `MIRtoolbox` [21]. These toolboxes are mainly designed for research-oriented access to audio processing, yielding code for audio feature extraction as well as for various MIR applications. Here we focus on two concrete software examples: the FMP notebooks [7] which have an explicit educational lens, and the Python package `librosa` [9] which has become a standard in MIR research,

and recently has also been incorporated into introductory MIR courses. We indicate how these tools facilitate multiple entry points to delve deeper into classical signal processing techniques, while bridging the gap between education and cutting-edge research.

FMP Notebooks

The FMP notebooks offer an interactive foundation for MIR and for teaching and learning fundamentals of music processing (FMP) [7], which when used in a traditional signal processing course can enhance students' understanding of signal processing. By closely following the eight chapters of the textbook [8], the FMP notebooks provide an explicit link between structured educational environments and current professional practices, inline with current curricular recommendations for computer science [22]. Furthermore, these notebooks provide a vehicle for students to transition between comprehending signal processing ideas in their own words (the second level of Bloom's taxonomy) towards applying these ideas interactively to music examples (the third and fourth levels of Bloom's taxonomy). For many MIR tasks, fundamental algorithms and signal processing techniques are discussed in detail. An overview of the main topics covered by the FMP notebooks is shown in Figure 6. Besides the treatment of the theory, the notebooks demonstrate how these techniques can be implemented by providing specific Python code examples.

The FMP notebooks leverage the Jupyter notebook framework [23], which has become a standard in industry as well as in educational settings. This open-source web application allows users to create documents that contain live code, text-based information, mathematical formulas, plots, images, sound examples, and videos. Jupyter notebooks are often used as a publishing format for reproducible computational workflows [23]. They can be exported to a static HTML format, which makes it possible to generate web applications that can be accessed through standard web browsers with no specific technical requirements. By leveraging the Jupyter framework, the FMP notebooks bridge the gap between theory and practice by interleaving technical concepts, mathematical details, code examples, illustrations, and sound examples within the unifying Jupyter framework (see Figure 5). Additionally, the notebooks are essentially self-contained in terms of content by including introductions for each MIR task, providing important mathematical definitions, and describing the computational approaches in detail.

One primary purpose of the FMP notebooks is to provide audio-visual material as well as Python code examples that implement the computational approaches step by step.

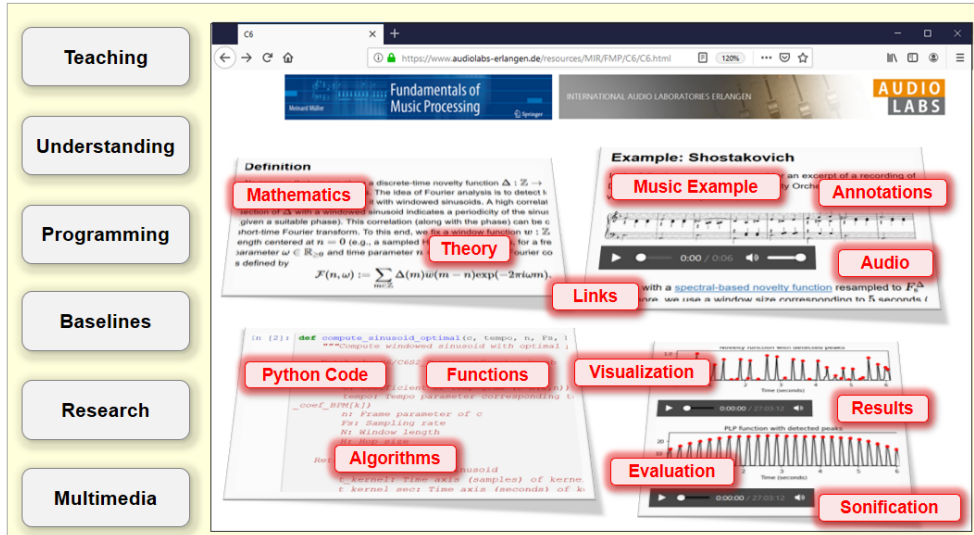


Figure 5: Overview of didactical aspects of the FMP notebooks and their implementation using the interactive Jupyter notebook framework.

Additionally, the FMP notebooks provide an interactive framework that allows students to experiment with their own music examples, to explore the effect of parameter settings, and to gain an understanding of the computed results by suitable visualizations and sonifications. These functionalities are examples of “procedural literacy” [24] by centering theoretical discussions around computational procedures.

The FMP notebooks, even though containing a library of MIR functions (called `LibFMP`), are not designed to be a toolbox per se. Instead, for a given music processing pipeline, the FMP notebooks introduce the code in a step-by-step fashion interleaved with explanations. This allows a student to access, visualize, and understand the intermediate steps. We illustrate this principle by coming back to our beat tracking scenario. Discussing Figure 4, we introduced a spectrum-based novelty function the peaks of which indicating note onset candidates. We now apply the same concept to an orchestral recording of the Waltz No. 2 by Dimitri Shostakovich’s Suite for Variety Orchestra No. 1. Figure 7 shows the score (in a piano-reduced version) as well as the novelty function of a short excerpt of this piece. Note that the first beats (downbeats) of the 3/4 meter are played softly by nonpercussive instruments, leading to relatively weak and blurred onsets. In contrast, the second and third beats are played sharply (“staccato”), supported by percussive instruments. These properties are also reflected by the spectral-based novelty function: the peaks that correspond to downbeats are hardly visible or even missing, whereas the peaks that correspond to the percussive beats are much more pronounced. As for beat tracking, we again use Fourier analysis—this time applied to the novelty function rather than to the audio signal. As for the


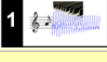



Part	Title	Notions, Techniques & Algorithms	HTML	IPYNB
	Basics	Basic information on Python, Jupyter notebooks, Anaconda package management system, Python environments, visualizations, and other topics	[html]	[ipynb]
	Overview	Overview of the notebooks (https://www.audiolabs-erlangen.de/FMP)	[html]	[ipynb]
	Music Representations	Music notation, MIDI, audio signal, waveform, pitch, loudness, timbre	[html]	[ipynb]
	Fourier Analysis of Signals	Discrete/analog signal, sinusoid, exponential, Fourier transform, Fourier representation, DFT, FFT, STFT	[html]	[ipynb]
	Music Synchronization	Chroma feature, dynamic programming, dynamic time warping (DTW), alignment, user interface	[html]	[ipynb]
	Music Structure Analysis	Similarity matrix, repetition, thumbnail, homogeneity, novelty, evaluation, precision, recall, F-measure, visualization, scape plot	[html]	[ipynb]
	Chord Recognition	Harmony, music theory, chords, scales, templates, hidden Markov model (HMM), evaluation	[html]	[ipynb]
	Tempo and Beat Tracking	Onset, novelty, tempo, tempogram, beat, periodicity, Fourier analysis, autocorrelation	[html]	[ipynb]
	Content-Based Audio Retrieval	Identification, fingerprint, indexing, inverted list, matching, version, cover song	[html]	[ipynb]
	Musically Informed Audio Decomposition	Harmonic/percussive separation, signal reconstruction, instantaneous frequency, fundamental frequency (F0), trajectory, nonnegative matrix factorization (NMF)	[html]	[ipynb]

Figure 6: Overview of the main topics covered by the FMP notebooks (adapted from [7, 8]).

STFT, the idea is to locally compare a given novelty function with windowed sinusoids. This time, the frequency of the sinusoid is interpreted in terms of BPM (e.g., an oscillation rate of 1 Hz corresponds to 60 BPM). The resulting spectrogram is then called *tempogram*, where the frequency axis is interpreted as tempo axis. Besides the frequency, we also use the phase information of the complex STFT coefficients to determine for each time position a windowed sinusoid that best captures the local peak structure of the novelty function. This is illustrated by Figure 7. Instead of looking at the windowed sinusoids individually, the idea is to employ an overlap-add technique by accumulating all locally optimal sinusoids over time. As result, one obtains a single function that can be regarded as a local periodicity enhancement of the original novelty function. Revealing predominant local pulse (PLP) information, this representation is referred to as a *PLP function* [25]. Having a pronounced peak structure, the beat positions can now be obtained from the PLP function using a simple peak picking strategy.

By looking at this concrete example, we illustrated how the FMP notebooks yield explicit access to all intermediate steps, starting with a musical score and ending with a sonification of the detected beat positions superimposed with the original audio recording. Furthermore, this example showed how Fourier analysis can be applied for periodicity enhancement, while

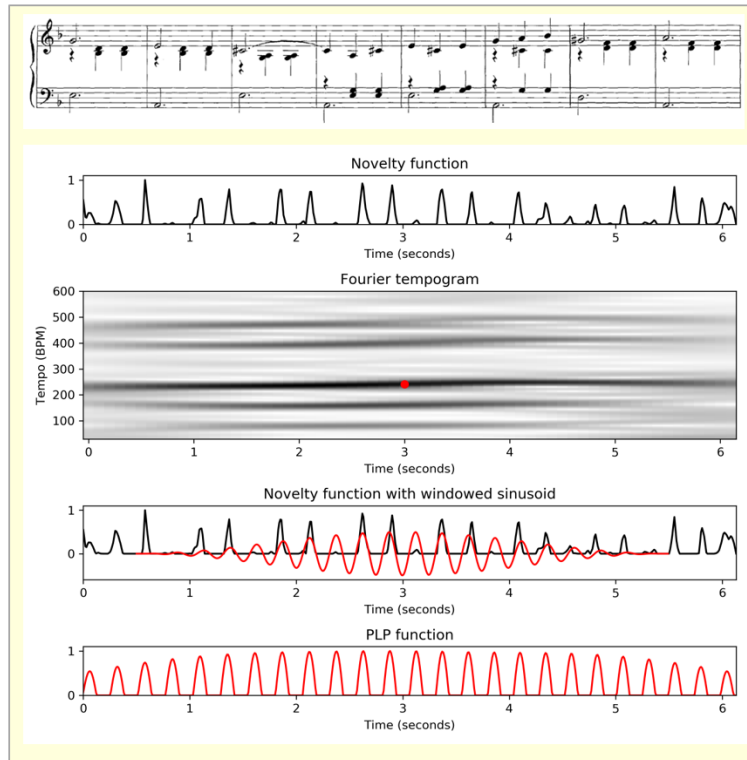


Figure 7: Illustration of the music processing pipeline for computing the predominant local pulse (PLP) function given a novelty function (see also Figure 4). The figure shows a small excerpt of the Waltz No. 2 by Dimitri Shostakovich’s Suite for Variety Orchestra No. 1.

highlighting the role of the phase. When teaching and learning signal processing, we advocate that it is essential to have a holistic view on the MIR task at hand, the algorithmic approach, and its practical implementation. Looking at all steps of the processing pipeline sheds light on the input data and its biases, possible violations of model assumptions, and the short-coming of quantitative evaluation measures. Only by an interactive examination of all these aspects, students will acquire a deeper understanding of the concepts, transitioning from merely explaining concepts (the lowest level of Bloom’s taxonomy) to applying their signal processing approaches both conceptually and in code. Furthermore, the imprecise definitions of MIR tasks allow for richer discussion and cognitive interaction with signal processing and music. For example, in beat-tracking, there are a number of questions that naturally arise: What is an onset? Can it be described by a single time instant? Does beat tracking make sense for certain musical passages (e.g., music with rubato)? Would humans agree when asked to specify a single tempo value? Is the evaluation metric relevant for a given application? In a curriculum on signal processing, wrestling with such questions illuminates to students the challenges of computational approaches in the applied sciences.

librosa Python Package

The FMP Notebooks provide an interactive framework in which students can learn about and experiment with signal processing and MIR algorithms. However, when students transition from learning to professional practice and research, we expect them to outgrow the FMP notebooks, and begin developing their own DSP methods and programs, corresponding to students' arrival at the top of Bloom's taxonomy. However, this transition can be difficult without proper (software) infrastructural support. The `librosa` package was designed to fill this role, providing standardized and flexible reference implementations of many common methods in MIR [9].

Whereas the FMP notebooks are designed to introduce fundamental concepts in signal processing, `librosa` is intended to facilitate high-level composition of basic methods into complex pipelines. As its original intended audience was the MIR research community, it was designed to facilitate the development of experimental research code. In the seven years since its first release, numerous scholarly publications have used `librosa` to provide the underlying signal processing framework, and many of these publications include open source software which students and researchers can download, use, and extend the work. The availability of open research software provides an avenue to train new researchers, as they can directly see how prior work was done, and have a significantly less work to do if modifying or extending it to achieve a new goal.

Conversely, `librosa` itself provides both reference implementations of many previously published methods (various feature extractors, phase retrieval methods, spectrogram decompositions, beat tracking algorithms, etc.), which can be used to independently replicate a method with relatively little effort. To demonstrate this, a collection of *advanced examples* are provided in the documentation, several of which demonstrate how to fully reproduce a published method by combining building blocks provided by `librosa`. These examples can also be exported as Jupyter notebooks, which a user can download and run on their own machine. This feature of the documentation serves a similar, interaction-oriented goal as the FMP notebooks, except that is aimed principally at researchers and software developers already familiar with the fundamentals.

The design of librosa

The application programming interface (API) of librosa was intentionally designed to present a low barrier to entry for new users, eschewing complex class hierarchies and object-oriented interfaces in favor of simple data structures (numerical arrays) and functions. This kind of function-oriented design can be easier for new users to learn, as functions (in contrast to objects) have well-defined entry and exit points, and no internal state for the programmer to understand: all parameters are explicitly visible in the call signature. Similarly, variable names are consistently defined across the package and human-readable (e.g., `n_fft` instead of `N` for the number of analysis frequencies in a Fourier transform).¹

Figure 8 provides a brief example code listing which follows the rhythm analysis example given in Figure 7. While the library allows a user to directly construct the PLP function from an audio signal (line 7), a user can also explicitly construct intermediate representations such as the novelty function directly (lines 10–11). The resulting code is still compact and high-level, but it also facilitates exploration and experimentation. For example, a user can easily change the calculation of the novelty function and leave the remainder of the PLP analysis fixed, allowing users to carefully measure the result of their interventions. This design philosophy is not limited to this one example, but rather is seen throughout the library as a whole.

Beyond its API design, the library developers strive for complete and thorough documentation, fully worked code examples for each function, and well-documented, readable source code. The latter point is enforced by stringent code review, and ensures a high standard of quality for all source code contributions. Well-written source code can be instructional, both in demonstrating clearly how any particular method works, as well as providing more general examples of how to structure complex programs and libraries. The intent behind the emphasis on code quality is to allow any knowledgeable user to look into the code, and with minimal effort, quickly be able to understand how it works, and potentially extend it with new contributions. More generally, the library itself provides an example of many software engineering best practices, which can be readily adopted in research settings, such as version control, code review processes, and continuous integration testing [27].

Both the FMP notebooks and librosa create opportunities for interaction in a traditional signal processing course. Adding just the FMP notebooks adds interactions similar to a

¹These design principles, among many others found in the scientific Python community, were clearly articulated by Gaël Varoquaux’s 2017 keynote address at the annual SciPy conference [26].


```

1 import librosa
2
3 # Load an audio file, returns the signal and sampling rate
4 y, sr = librosa.load('Shostakovich_Waltz.wav')
5
6 # Compute the PLP pulse directly from the signal
7 pulse = librosa.beat.plp(y=y, sr=sr)
8
9 # Or we can use a pre-computed novelty function
10 nov = librosa.onset.onset_strength(y=y, sr=sr)
11 pulse = librosa.beat.plp(onset_envelope=nov, sr=sr)
12
13 # Compute the Fourier tempogram
14 tgram = librosa.feature.fourier_tempogram(y=y, sr=sr)

```

Figure 8: Example usage of `librosa` to reproduce the analysis illustrated in Figure 7. All results are stored as numerical arrays, can be directly manipulated by the user. Functionality is grouped into different submodules (`onset`, `feature`, `beat`), and the interfaces for high-level analyses (e.g. PLP) support exploration by allowing users to pre-compute intermediate representations.

“coding worksheet” that allows students to dynamically play with examples. Additionally, students become familiar with the Jupyter framework and Python syntax. Including `librosa` allows for broader experimentation with signal processing and provides a common language for cumulative course projects, where students can demonstrate their familiarity and creativity with a number of signal processing topics. Furthermore, leveraging tools like the FMP notebooks and `librosa` provides opportunities for learning beyond signal processing. By employing these Python and Jupyter-based tools, students are passively learning about open access and reproducibility, two topics that cut across disciplines.

Interactivity and Learning

In this article, we have presented a scaffold to naturally transition students from starting to learn about signal processing to beginning independent research. Using music examples and technologies provide what Guzdial [3] calls a contextualized educational experience for signal processing. Concretely, we have shown how music-based examples can make Fourier analysis more tangible to beginners. We then demonstrated how the FMP notebooks provide a constrained scaffold for interacting with music examples for signal processing. Building on the skills and conceptual understanding gained through concrete and interactive music-based examples, students can explore more advance applications of Fourier analysis through broader experimentation via the Python library `librosa`.

The incorporation of music into a signal processing course allows students an avenue for “inauthentic legitimate peripheral participation” [28] in the field of signal processing. Extending the work of Lave and Wenger [6] defining “legitimate peripheral participation” which builds

a scaffold for introducing students to concepts similar to the apprentice structure, Guzdial and Tew [28] connect this model to computing courses that have media-based examples (such as photographs) as the basis for the coursework. In this work we have modeled a similar extension, applying their framework to the specifics of music as a vehicle for learning signal processing by first engaging the highly structured FMP notebooks and then leveraging the range of tools in the `librosa` package. In effect, this structure brings students through Bloom’s taxonomy, helping them to reach the deepest understanding of SP concepts.

Signal processing is about finding structure in signals. Audio is a familiar signal modality, and music is explicitly and intentionally structured audio. Leveraging the familiarity of music, Fourier analysis becomes more concrete, and examples from textbooks such as [1, 8] can be easily incorporated into a traditional signal processing course. For instructors seeking to transition to an interactive alternative to the lecture-based classroom model with a “sage on a stage” simply depositing knowledge into students’ brains, the FMP notebooks provide one such alternative. Created in the Jupyter framework with an explicit connection to the “Fundamentals of Music Processing” textbook [8], the FMP notebooks provide an interactive environment where students are invited to grapple with concepts through small structured coding examples. The Jupyter framework underlying the FMP notebooks provide an experimental playground for students to test signal processing concepts on music and to manipulate music using signal processing ideas. Once students are familiar with introductory signal processing and MIR concepts, they can continue experimenting via the examples in `librosa` presented as coding notebooks.

In this paper, we leverage the inherent familiarity of music to motivate theoretical signal processing and extend these examples to an interactive learning experience through the FMP notebooks and `librosa`. We have discussed how the interplay between music and signal processing leads to a variety of kinds of interaction: interaction through applications, hands-on interaction with the material through experimentation, and interaction between the structure of a classroom and the experimentation of research. We have provided resources that instructors can use in their classrooms, and we have been diligent to describe how these resources can be implemented in terms of course activities from enhancing lectures to incorporating cumulative class projects.

Acknowledgments

The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Fraunhofer-Institut für Integrierte Schaltungen IIS. Meinard Müller thanks the German Research Foundation (DFG) for various research grants that allow him for conducting fundamental research in music processing. Katherine M. Kinnaird is the Clare Boothe Luce Assistant Professor of Computer Science and Statistical and Data Science at Smith College and as such, is supported by Henry Luce Foundation's Clare Boothe Luce Program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Luce Foundation. Finally, we thank Roger Dannenberg for helpful comments.

Bibliography of Authors

Meinard Müller (meinard.mueller@audiolabs-erlangen.de) received the Diploma degree (1997) in mathematics and the Ph.D. degree (2001) in computer science from the University of Bonn, Germany. Since 2012, he holds a professorship for Semantic Audio Signal Processing at the International Audio Laboratories Erlangen, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). His research interests include music processing, music information retrieval, audio signal processing, and motion processing. He wrote a monograph titled *Information Retrieval for Music and Motion* (Springer-Verlag, 2007) and a textbook titled *Fundamentals of Music Processing* (Springer-Verlag, 2015, www.music-processing.de). In 2020, he was elevated to IEEE Fellow for contributions to music signal processing.

Brian McFee (brian.mcfee@nyu.edu) received the B.S. degree (2003) in Computer Science from the University of California, Santa Cruz, and M.S. (2008) and Ph.D. (2012) degrees in Computer Science and Engineering from the University of California, San Diego. Currently, he is an Assistant Professor of Music Technology and Data Science at New York University. His work lies at the intersection of machine learning and audio analysis. He is an active open source software developer, and the principal maintainer of the librosa package for audio analysis.

Katherine M. Kinnaird (kkinnaird@smith.edu) received her A.B. degree (2008) in mathematics from Wellesley College, and her A.M. (2010) and her Ph.D. (2014) in mathematics from Dartmouth College. Her research takes a mathematical lens to machine learning

approaches on cultural artifacts including music and TED talk transcripts. Currently, she is the Clare Boothe Luce Assistant Professor of Computer Science and Statistical & Data Sciences at Smith College. Throughout her career, she has participated in professional development opportunities focused on excellence in teaching, including Dartmouth College’s NSF GK-12 program and the Mathematical Association of America’s Project NExT (New Experiences in Teaching).

References

- [1] A. Lerch, *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. John Wiley & Sons, 2012.
- [2] S. V. Vaseghi, *Multimedia Signal Processing: Theory and Applications in Speech, Music and Communications*. Wiley, 2007.
- [3] M. Guzdial, “Does contextualized computing education help?” *ACM Inroads*, vol. 1, no. 4, pp. 4–6, 2010.
- [4] B. S. Bloom and M. D. Engelhart, *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain*. David McKay Company, 1956.
- [5] C. Starr, B. Manaris, and R. H. Stalvey, “Bloom’s taxonomy revisited: Specifying assessable learning objectives in computer science,” *SIGCSE Bulletin*, vol. 40, no. 1, pp. 261–265, 2008.
- [6] J. Lave and E. Wenger, *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, 1991.
- [7] M. Müller and F. Zalkow, “FMP Notebooks: Educational material for teaching and learning fundamentals of music processing,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 573–580.
- [8] M. Müller, *Fundamentals of Music Processing*. Springer Verlag, 2015.
- [9] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “Librosa: Audio and music signal analysis in python,” in *Proceedings the Python Science Conference*, Austin, Texas, USA, 2015, pp. 18–25.
- [10] G. Wiggins, D. Müllensiefen, and M. Pearce, “On the non-existence of music: Why music theory is a figment of the imagination,” *Musicae Scientiae, Discussion Forum*, vol. 5, pp. 231–255, 2010.
- [11] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, “Automatic music transcription: An overview,” *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.
- [12] R. Parncutt, “A perceptual model of pulse salience and metrical accent in musical rhythms,” *Music Perception*, vol. 11, no. 4, pp. 409–464, 1994.
- [13] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*. MIT Press, 1983.

- [14] M. E. P. Davies and M. D. Plumbley, “Context-dependent beat tracking of musical audio,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1009–1020, 2007.
- [15] A. P. Klapuri, A. J. Eronen, and J. Astola, “Analysis of the meter of acoustic musical signals,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 342–355, 2006.
- [16] S. Mallat, *A Wavelet Tour of Signal Processing – The Sparse Way*, 3rd ed. Academic Press, 2009.
- [17] M. Guzdial, “Exploring hypotheses about media computation,” in *International Computing Education Research Conference (ICER)*, La Jolla, CA, USA, 2013, pp. 19–26. [Online]. Available: <https://doi.org/10.1145/2493394.2493397>
- [18] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra, “Essentia: An audio analysis library for music information retrieval,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Curitiba, Brazil, 2013, pp. 493–498.
- [19] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “madmom: A new python audio and music signal processing library,” in *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, Amsterdam, The Netherlands, 2016, pp. 1174–1178.
- [20] G. Tzanetakis, “Music analysis, retrieval and synthesis of audio signals MARSYAS,” in *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, Vancouver, British Columbia, Canada, 2009, pp. 931–932.
- [21] O. Lartillot and P. Toiviainen, “MIR in MATLAB (II): A toolbox for musical feature extraction from audio,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Vienna, Austria, 2007, pp. 127–130.
- [22] Joint Task Force on Computing Curricula of the Association for Computing Machinery (ACM) and IEEE Computer Society, *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA: Association for Computing Machinery, 2013.
- [23] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, and J. development team, “Jupyter notebooks—a publishing format for reproducible computational workflows,” in *Proceedings of the International Conference on Electronic Publishing*, Göttingen, Germany, 2016, pp. 87–90.
- [24] A. Vee, “Understanding computer programming as a literacy,” *Literacy in Composition Studies*, vol. 1, no. 2, 2013.
- [25] P. Grosche and M. Müller, “Extracting predominant local pulse information from music recordings,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1688–1701, 2011.

- [26] G. Varoquaux, “Coding for science and innovation,” 2017, accessed: 2020-06-08. [Online]. Available: <https://pyvideo.org/scipy-2017/keynote-coding-for-science-and-innovation.html>
- [27] B. McFee, J. W. Kim, M. Cartwright, J. Salamon, R. M. Bittner, and J. P. Bello, “Open-source practices for music signal processing research: Recommendations for transparent, sustainable, and reproducible audio research,” *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 128–137, 2019.
- [28] M. Guzdial and A. E. Tew, “Imagineering inauthentic legitimate peripheral participation: An instructional design approach for motivating computing education,” in *International Computing Education Research Workshop (ICER)*, R. J. Anderson, S. Fincher, and M. Guzdial, Eds. Canterbury, UK: ACM, 2006, pp. 51–58.