Algorithmic Generation Classical-Style Music Using Wave-Based Concepts

The project I wish to build is an algorithmic compositional tool for generating music in the style of the Classical era (roughly 1750-1800) of music, specifically that of Wolfgang Amadeus Mozart. It only takes a quick online search to discover that many tools already exist for reproducing various forms of music, especially the more well-defined genres such as J.S. Bach's fugues and chorales. These software tools arrive at their results using a wide variety of techniques, such as using counterpoint rules as a constraint. Several common methods are outlined in “Six Techniques for Algorithmic Music Composition”, including note-to-note interpolation, fractal patterns, L-grammars, and “stochastic binary subdivision” for rhythm generation, among others (Langston 2).

Mozart himself is often cited as contributing to the field of algorithmic composition with his “Musikalisches Würfelspiel” (K. 516f), which is a musical dice game in which each phrase is selected by rolling a pair of dice until a full and logical piece has been written (Cope 22). One particularly successful automated composition technique has been termed “recombinant music”, which at the most basic level involves chopping musical works into short vertical segments, identifying the important characteristics in each, then pasting them back together in some new order, along with various other steps to improve the aesthetic appeal of the result. This technique is used in David Cope's Experiments in Musical Intelligence, which was known to achieve impressive results in many different styles (Cope 27-28). However, few if any of the notes are actually composed by the software, and if one is familiar with the original work it is easy to recognize where many of the segments come from.

My project will use a new approach to composition based on several observations about Mozart's music along with some musical aspects that tend to receive less attention than others. First and foremost, rhythm appears to be a subject that is not as developed or modeled as it could be, and one can see this by scanning through the recently-published “AI Methods in Algorithmic Composition: A Comprehensive Survey” (Fernández et al), which categorizes and describes many publications throughout the history of algorithmic composition up through 2013. While there are certainly an assortment of papers that address rhythm in some form, the task of figuring out a model for note timings is almost always assigned to the computer, usually using Markov chains (such as an Ames, C., & Domino, M. paper mentioned on Fernández 527), other forms of statistics-based probability, artificial neural networks (such as a Gibson, P. M., & Byrne, J. A. paper mentioned on Fernández 545), or genetic algorithms that use a database of real compositions as a fitness function (such as a Spector and Alpern study mentioned on Fernández 550). While these methods can be flexibly applied to a vast array of genres and have achieved various degrees of success in their imitative results, machine learning alone cannot perfectly recreate or help us identify concrete patterns and rules embedded within pieces of music.

I propose providing models of rhythmic generation to the computer through hard
code, rather than through data input. By narrowing my scope to classical music, several important patterns are already well-studied and well-defined in music theory, such as periods and sentences, which are two common phrase structures found in pieces of this time. In other words, while music theory's rules of counterpoint have been applied to pitch generation, to this point it seems that few people if any have implemented common patterns of timing to algorithmic rhythmic generation of classical music, instead relying on the machine to figure out the patterns. This does make sense, though: counterpoint rules are thorough and strict, while rhythmic patterns are less rigorously defined and not usually referred to as “rules”. However, many music theorists and composers have studied and conceptualized temporality in their writings, such as Ernst Kurth, Hugo Riemann, Viktor Zuckermandl, Wallace Berry, and dozens of others mentioned in “Rhythm in twentieth-century theory” (London). Implementing some of these advanced ideas into code would certainly provide a technical challenge.

In particular, I would like to draw inspiration from Kurth, Zuckermandl, and Berry who focused on music as a type of motion, with musical gestures as a combination of waves of several sizes, from “constituent waves” to “developmental waves” to “symphonic waves” (qtd. in London 696). While this implies representing a sequence of notes as a Fourier transform using computer science and math terminology, these waves are highly punctuated (only one half of a sine-like wave) and not necessarily evenly spaced between crests (as seen in a Zuckermandl diagram on London 698). Additionally, that discrete nature of pitches and rhythms in Mozart's music perhaps would not lend itself well to precise mathematical waves, although it may be worth experimenting with if there is time. Regardless, this concept fits into what one of my former music teachers would always say: a note is either going somewhere or leaving somewhere. Rarely do we see true stasis in classical music.

The algorithm I have in mind is one that is recursive, starting with a few basic musical events and embellishing these punctuated wave-points with so-called “pickups” leading up to them, and terraced “fall-downs” leading away from them. These patterns can happen at many durational levels (weak phrases between strong phrases, weak measures between strong measures, weak quarter notes surrounding strong quarter notes, weak eighth notes between strong eighth notes, and so on) and can stretch near or far (sometimes there are seven ascending eighth notes leading to a downbeat, while sometimes there is only one). The algorithm will start with a few notes, then expand their lengths and fill in the extra space with more notes that trail away in some fashion like “waves”, then repeat this process until some proper length is met (determined by the user).

Maintaining patterns throughout this recursive algorithm will be part of challenge: if the number of pickups and fall-downs we assign to each note is purely random at each level, the result will be noisy and hard to follow. Repetitions and sequences occur often in classical music, so at each iteration we will have to mark related notes as “similar” to each other in some fashion (chord, contour, rhythm, etc.), so that future levels will embellish them in a “similar” manner. The goal of this pattern-building is not just aesthetic, but more importantly to be able to reproduce periods and sentences mentioned earlier (which are built out of smaller modules that share certain similarities in relation to each other). A side effect of this goal will hopefully be an overall “simple” feel in which there are many common patterns or repetitions for the listener to grab onto throughout a
A variety of phrases. Additional constraints or heuristics will have to be introduced as well, because each level will have different tendencies. For example, if we expand out a lengthy piece, we want to avoid leaving behind a big empty space, because in many Mozart pieces it is rare to see more than a measure go by without a new note, so this would probably be an additional constraint).

This construction alone provides enough depth for a full semester project, as each duration level could be studied and tweaked in complicated ways to guarantee “interesting” rhythms. However, rhythm is only one piece of the puzzle, and other aspects like key and contour share a connected role along the way. Contour fits itself well into the concept of a wave, with pitches leading away in either direction and then coming back to the “punctuated” or “strong” spots. Keys and chords are somewhat less analogous – while there is a measurable distance between each key based on the number of shared notes (i.e. the circle of fifths), chords in classical musical have specific functional purposes such as cadential gestures (IV → V → I). Perhaps by implementing a multidimensional distance function between chords it would be possible to result in the same functional harmony chains we expect to hear leading to and from tonic (I), but ultimately these functions may end up being hard-coded, with more emphasis on rhythm and contour. So far a single-voice melody generator has been described; additional voices would add a significant degree of complexity. I hope these aspects will become clearer as my project develops.

As far as implementation details go, I will be using Haskell to write my algorithm, along with the Euteropia library for a set of convenient functions and musical representations, including methods for outputting MIDI files and playback. The modular and functional nature of Haskell means that this algorithm will not be one massive blob of code; instead, a number of smaller “tools” will naturally form within the project, which might mean functions that do anything from transforming motives to analyzing chords to generating arpeggios between two notes. So regardless of the aesthetic outcome, a small suite of perhaps-useful musical code will likely be an added bonus of this project.

Thus the main deliverable item will be Haskell code which has some kind of interface for generating music, using the recursive algorithm described to generate rhythmic patterns and modules, pitch-wise contour, and possibly chord progressions, all under the idea of music as a set of punctuated waves as described by several music theorists. Sample output will of course be provided, and the hope is that it approximates Mozart's style of music. A detailed paper will also be attached which describes the code and decision process.

References


