Score Generation in Chord Spaces

Michael Gogins (michael.gogins@gmail.com)

March 12, 2017

1. Musical Motivation

- I am a composer, not a mathematician.
- I like to use algorithms, e.g. fractals, that recursively generate a finished score in one pass.
- Because current algorithms work in spaces without inherent musical significance, generated events must be mapped onto time and pitch.
- Therefore, generated scores are often unstructured over larger scales of time and pitch.
- It seems desirable to create algorithms that work directly with voice-leadings, chords, and so on.
- The goal is neither to model music theory nor to reproduce existing styles, but simply to use generative procedures in spaces with some built-in musical semantics.

2. Summary

- The *Tonnetz* is the graph of minimal voice-leadings between the major and minor triads.
- The *Tonnetz* can be generalized as "chord space," an orbifold (i.e. a quotient space) in which every point is a chord, and the smoothness of a voice-leading equals the closeness of its chords (Tymoczko 2006).
- From chord space I derive related geometries to represent all trichords, higher arities of chord, voice-leading solutions, chord progressions, *etc.*
- I present a Lindenmayer system that recursively rewrites an axiom by applying symbol replacement rules. The final production of the system is a string of commands for a turtle that moves within either a chord progression space, or a voice-leading space, to write a score.

- I conclude by discussing other possibilities for score generators in mathematical spaces with built-in musical structure.

3. Orbifolds

- An *orbifold* is a quotient space whose points are equivalent under a group action (any combination of translation, rotation, glide rotation, and reflection) that defines a symmetry.
- The group action permutes the corners of the orbifold, to glue faces of the quotient space together.
- For example, a strip of paper is a quotient space of the plane; gluing two ends together creates ring, which is an orbifold.
- Giving the ends a half twist before gluing them creates a Moebius band, another orbifold.

4. *Musical* Orbifolds

- Music theorists distinguish various equivalence classes with respect to pitches: octave equivalence, permutational equivalence, inversional equivalence, and transpositional equivalence.

- For each equivalence class, and each *combination* of classes, there exists an orbifold in chord space. The equivalence defines which points of the space are identified to form the orbifold (Callender, Quinn, and Tymoczko 2006).

- The orbifold most familiar to musicians is pitch under octave equivalence, or pitch-class set space, $\mathbf{R}/12\mathbf{Z}$, where the octave is defined as 12 semitones.

5. The *Tonnetz* as Orbifold

- Tymoczko and colleagues (Tymoczko 2006; Callender, Quinn, and Tymoczko 2006) are developing a geometric approach to music theory. Chords are represented as points in a linear *chord space*, which has one dimension of pitch per voice, distinguishing inversions, octaves, and order of voices. This space extends up and down to infinity from a reference pitch at the origin. Chord space is continous, and every equally tempered and non-equally tempered chord can be defined in it.

- (Tymoczko 2006) has identified the orbifold that divides pitch-class set space for n voices by the symmetry group for n voices:

  $(\mathbf{R}/12\mathbf{Z})^n / \mathbf{S}_n$.

  Its fundamental domains are $n-1$ dimensional simplexes.

- The points of the orbifold are identified by a group action that consists of a rotation (for odd numbers of voices) or a rotation plus a reflection (for even numbers of voices). Figure 1 shows this orbifold for trichords. The lines connecting chords indicate movements of one semitone, so that joined chords are closest neighbors and the lines indicate all minimal voice-leadings.

- The classical *Tonnetz* of Oettingen, Euler, and Riemann is simply the six columns of major (red) and minor triads (blue) surrounding the central column of 4 augmented triads (white) that defines the orthogonal axis of symmetry of this orbifold. One end of the prism can also be visualized as being rotated 120° and glued to the other end to form a torus.

- As a geometric representation of the *Tonnetz*, $(\mathbf{R}/12\mathbf{Z})^n/\mathbf{S}_n$ illuminates basic symmetries and constraints of Western music.

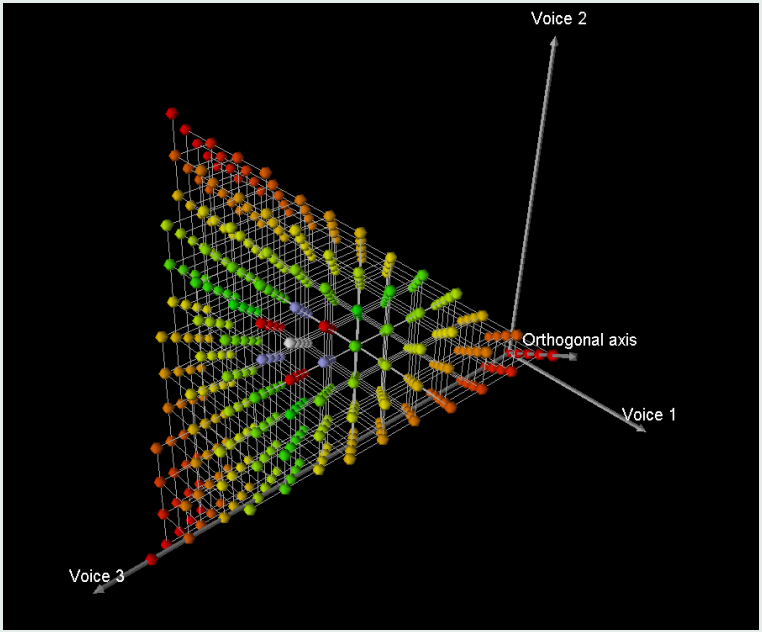

Figure 1: ***Tonnetz*** **for trichords**

- Within this orbifold, there is a one-to-one correspondence from voice-leadings to progressions between unordered pitch-class sets.

- The major and minor triads are flexible with respect both to harmonic progression and to voice-leading because they not only surround the orbifold's axis of symmetry, but also lie near each other. Similarly, for tetrachords, commonly used seventh chords surround the orthogonal column of diminished seventh chords.

6. Scores and Operations

- The present work represents musical scores as functions of time onto chord space. Arpeggios, passing tones, counterpoint, etc. are represented as succcessions of more or less fleeting chords.

- Chord spaces themselves are not sufficient to serve as spaces for composing:

  - For automatic score generation it is necessary to represent octaves, which breaks the octave equivalence of the *Tonnetz*.

  - Whereas a primitive operation such as a translation vector that does not cross a face of the orbifold has of course always the same orientation, if the same translation *does* cross a face, it can change orientation as it is reflected from a symmetry hyperplane. It is easier to visualize progressions in spaces in which there are no mirrors, so that (for example) adding a vector to a chord always moves it along the same orientation.

7. Variations on Chord Space

- Consequently, I have found the following variations on chord space to be useful for primitive operations in score generation:

  Ranged chord space Chord space for n voices under equivalence for range R defines the orbifold $(\mathbf{R}/R\mathbf{Z})^n$. Voices that move past the top of the range re-appear at the bottom. Figure 2 shows ranged chord space for 3 voices. Because its opposing faces are identified, the cube is actually a 3-dimensional torus.

  Normal chord space This space can be formed by dividing the fundamental domains of the completed *Tonnetz* into symmetrical "kites" and stacking them on top of each other, as shown in Figure 3 for trichords. The column is formed by the inversion of each chord that is closest to the orthogonal axis. This is related to, but not the same as, the theoretical notion of "normal form."

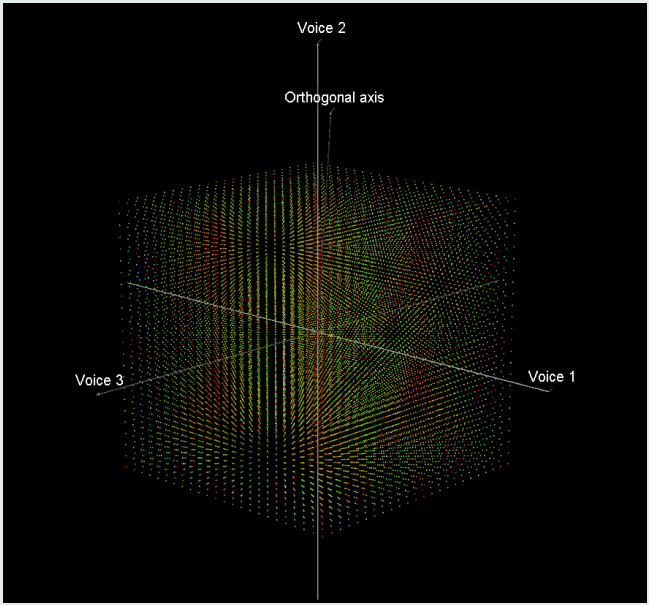

Figure 2: **Ranged chord space for 3 voices, 2 octaves**

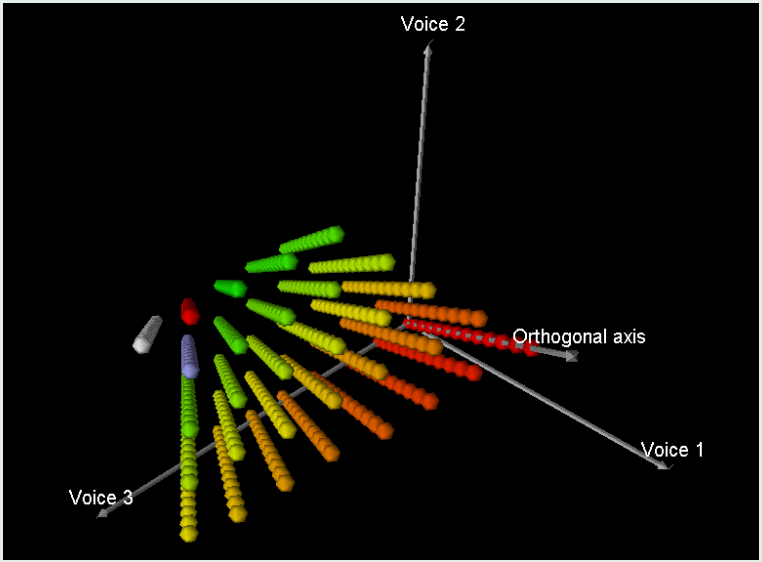

Figure 3: **Normal chord space for trichords**

8. Voice-Leading in Ranged Chord Space

- Tymoczko defines a normlike total preorder for voice-leadings within chord spaces, first by smoothness (distance between chords, using either the taxicab norm or the Euclidean norm), and then by parsimony (fewer or shorter movements of voices). Every known measure of voice-leading size creates a normlike total preorder (Tymoczko 2006).

- I have used this normlike ordering to implement a simple algorithm for finding the closest voice-leading in ranged chord space from a source chord to a target pitch-class set. Such algorithms can be very useful in score generators. Keep in mind that each voice of the source chord could be in any octave of the space's range. As both the octave-equivalent and ranged chord spaces inherit their metric from the same parent, Euclidean space, it follows that the normlike total preorder for voice-leadings also exists in ranged chord space.

8.0.1. Voice-Leading Algorithm

1. Store the source chord and target pitch-class set.
2. Generate the lattice of all chords within ranged chord space that match the target pitch-class set, by iterating octavewise through all points in the space.
3. Compare the voice-leading from the source chord to each chord in the lattice, first by smoothness, then by parsimony (it is easy to add the option of excluding parallel fifths).
4. Return the target chord with the closest voice-leading.

- My current implementation performs a brute force search to compare voice-leadings, but as long as this need not be done in real time, it is quite fast enough.

9. Lindenmayer Systems

- A Lindenmayer system consists of an axiom (initial string of atoms), some rules each specifying how one atom is to replaced with a string of atoms, an implicit rule that an atom with no replacement is replaced by itself, and the number of iterations for the recursion (Prusinkiewicz and Lindenmayer 1991).

- Some atoms are commands for a "turtle" in a drawing system. `F` might mean move one step while drawing a line, `f` move one step without drawing a line, `+` turn right, `-` turn left, `[` push the turtle state onto a stack, and `]` pop the turtle state from the stack.

- Repeated replacements usually expand the initial axiom into a long string of atoms — the *production* of the system.

- This is then interpreted as a program for the turtle, which draws a figure in the space.

9.1. *OL* Lindenmayer Systems

- In the simplest type of Lindenmayer system, or *OL* system, the replacement rules do not depend on the state of the production on either side of the current atom, and take no parameters.

- *OL* systems have been used for some time to generate musical scores in spaces where time is one dimension of the space (Holtzman 1981; Gogins 1992; McCormack 1996).

- I adapted *OL* systems to generate scores in chord spaces where each dimension is a voice, and time is simply the sequence of chords.

- I implemented the score generator using Python (van Rossum 2006), the SciPy package which provides efficient matrix arithmetic for Python (Oliphaunt, Peterson, Jones, et al. 2005), and CsoundVST, an extended version of Csound 5 with Python scripting and some facilities for mathematically-based algorithmic composition (Vercoe, ffitch, et al. 2006).

9.2. An OL Lindenmayer System for Musical Orbifolds

- The turtle is a position vector in n-dimensional chord space, the step is another vector, movement adds the step to the turtle, the step can be multiplied by a rotation matrix to point in a new direction.
  - Sometimes the turtle is moving voicewise in ranged chord space (`V` commands), or voice-leading space.
  - Sometimes the turtle is moving chordwise in normal chord space (`P` commands), or chord progression space.
  - Chords are created with the `C` commands.
    * The `Ca` command creates a chord at the current absolute position of the turtle in ranged chord space.
    * The `Cv` command creates a chord with the target pitch-class set, but the voice-leading that is closest by Tymoczko's

measure, using the algorithm described above.

- Because each operation is defined on one or another quotient of chord space, all operations can actually be performed on points in chord space alone, using ordinary matrix arithmetic. The actual quotient space is defined by applying a specific modulus or some other action that identifies points in the space.

- Time is defined as the sequence of chords, and the turtle state includes the size of the time step (`T` commands). The paper contains the complete table of turtle commands.

9.3. A Musical Example

- The `Pv` and `Cv` turtle commands shown in Table 1 set up a C major 7^{th} chord, then create a IM7 ii7 V7 VIM7 progression (Figure 4).

Iterations	1
Axiom	`C=CM7 Cv Pv1,2,1,2 Cv Pv5,5,5,6 Cv Pv2,2,3,2 Cv`

Table 1: **Progression**

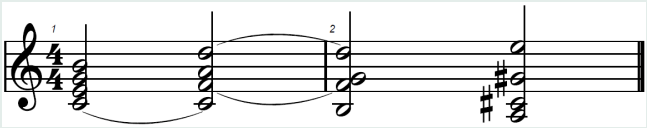

Figure 4: **Progression**

- A sequence of these progressions can be created using a simple Lindenmayer system with one substitution rule (Table 2, Figure 5).

Iterations	1
Axiom	`C=CM7 A A A`
Replace `A`	`Cv Pv1,2,1,2 Cv Pv5,5,5,6 Cv Pv2,2,3,2 Cv`

Table 2: **Sequence**

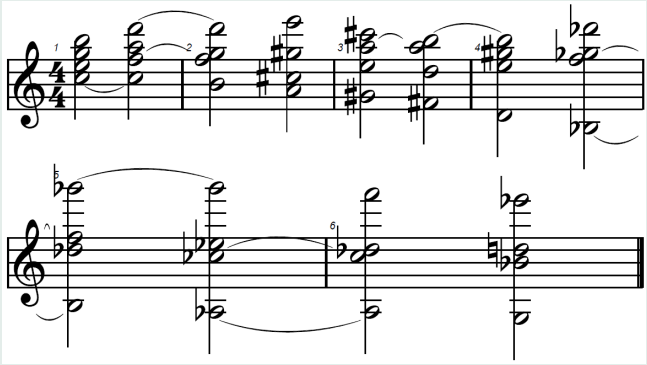

Figure 5: **Sequence**

Iterations	2
Axiom	`C=CM7 B`
Replace `B`	`Td1.125 Cv Va0,2 Cv Va1,3 Va2,2 Cv B Vs3,4 Cv B Va0,2 Cv Va1,3 Va2,4 Cv Va3,3 Cv Cv Tm1.125`

Table 3: **Independent Voices**

- Table 3 and Figure 6 show how the `V` commands can move voices one at a time to produce arbitrary counterpoint. In the Lindenmayer system's interpreter, voices with the same pitch in different chords are tied. In this system, the replacement rule for `B` itself includes references to `B`, so that 2 iterations cause a recursive expansion of the motive. The `T` commands also cause the 2nd replacement to move at a faster tempo.

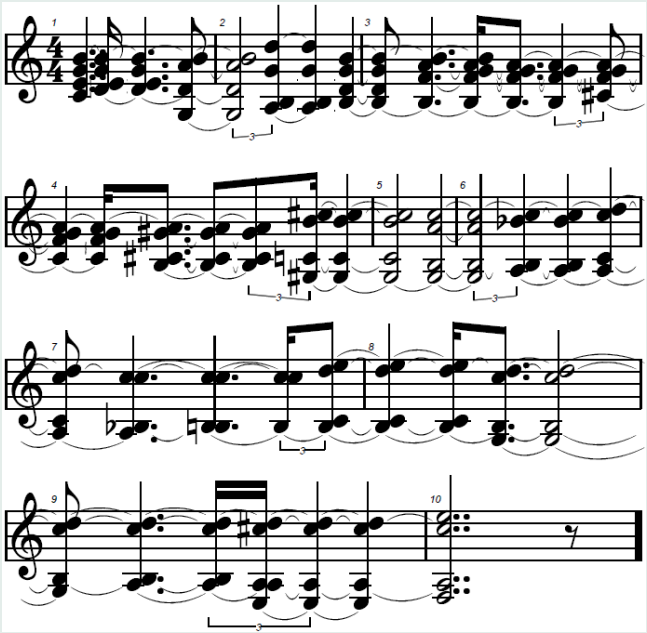

Figure 6: **Independent Voices**

- Finally, Table 4 represents the Lindenmayer system for a piece of 3 minutes 45 seconds. The axiom creates a major 7^{th} chord, then specifies an $ABB'A$ form. The [and] commands push and pop the turtle state from a stack, so that the chord appearing before [re-appears after]. The A section is constructed from the Table 2 sequence above, as well as another similar sequence. The B section consists of the independent voices section described in Table 3, recursively constructed at faster and faster tempos. In the B' section, one of the voices is moved before rule B is applied, which affects all subsequent voice-leading. At the ends of the A, D, and F sections, the E section generates arpeggiations within some chords.

Iterations	3
Axiom	`C=CM7 V=4,0 V=5,1 V=6,2 V=7,3 A Cv Cv [ V=4,4 V=5,5 V=6,6 V=7,7 B Cv Cv B Cv Cv ] A Cv Cv`
Replace `A`	`F F E D D E F E Cv Cv`
Replace `B`	`Td1.125 Cv Va0,2 Cv Va1,3 Va2,2 Cv B Vs3,4 Cv B Va0,2 Cv Va1,3 Va2,4 Cv Va3,3 Cv Cv Tm1.125`
Replace `D`	`Cv Pv1,2,1,2 Cv Pv2,3,2,3 Cv D Cv`
Replace `E`	`Cv [ V=4,4 V=5,5 V=6,6 V=7,7 Td2.  Vs3,12 Ca Vs2,12 Ca Vs1,12, Ca Vs0,12 Vs1,12 Vs2,12 Vs3,12 Ca E Tm2.0 ] Cv`
Replace `F`	`Cv Pv1,2,1,2 Cv Pv5,5,5,6 Cv Pv2,2,3,2 Cv F Cv`

Table 4: **Sample Piece**

10. Discussion

10.1. Limitations

- This approach knows something about voice-leading and chord progression.
- This approach knows nothing about counterpoint proper, imitative procedures, tonal key, or serial procedures.
- These spaces always represent a fixed number of voices, though dropping voices can be simulated by doubling or silencing some.
- The voice-leading algorithm is inflexible; the target chord stays close to the source chord in range, yet may wander up and down the score, or widen or narrow in range.
- In short, basic movements within musical spaces do not model the fine points of musical hearing or of any musical style.

10.2. Strengths

- One reader of an earlier version of this noted that the operations described here could have been implemented any number of ways — why, therefore, do I feel that this geometric approach is better?

- I think it is better because representing chords as points makes it possible to represent all time and pitch information in any score as a function from time onto voice-leading space, or onto chord space.

- This in turn makes it easy to manipulate *all* the time and pitch information in a score recursively (as with the Lindenmayer system), or even globally, using purely mathematical operations.

- To me, the most interesting musical advantage of this system is how easily recursive patterns of movement in chord space can create top-down, hierarchical pitch structures of arbitrary depth.

11. Future Directions

- It should be easy to modify the voice-leading algorithm to control range and spacing.
- It would be useful to have a concise way to represent changing numbers of voices.
- These procedures are not complex, but in the context of algorithmic composition their very simplicity is a virtue, enabling the concepts to be adapted to a wide variety of compositional procedures. These could include:
  - Chaotic dynamical systems
  - Applying musical filters to images or scientific data
  - Xenakis sieves or other stochastic generators
  - *et cetera*

- The use of geometry to model voice-leading and chords opens the composition of pitch structures up to all the resources of geometric algebra, and perhaps even topology and algebraic geometry. My impression is that these branches of mathematics have not seen as much musical application as, say, group theory.

- There are a number of ways that chord space can be factored into subspaces. I am evaluating several different sets of subspaces regarding their suitability as geometric bases for atomic operations in algorithmic composition.

12. Acknowledgments

- I thank Dmitri Tymoczko for his generous patience in discussion and for helping me with the mathematics.

- I also benefitted from discussing earlier stages of these ideas with Drew Krause and the members of the New York Csound Users Group.

- Any errors that remain here are of course my own.

References

Callender, C., I. Quinn, and D. Tymoczko (2006). Generalized chord spaces. Unpublished, http://music.princeton.edu/~dmitri. 6, 7

Gogins, M. (Winter 1992). Fractal music with string rewriting grammars. *News of Music 13*, 146–170. 18

Holtzman, S. R. (1981). Using generative grammars for music composition. *Computer Music Journal 5*(1), 51–64. 18

McCormack, J. (1996). Grammar based music composition. In R. Stocker et al. (Eds.), *From Local Interactions to Global Phenomena*, Complex Systems 96, Amsterdam. ISO Press. 18

Oliphaunt, T., P. Peterson, E. Jones, et al. (2005). Scipy — scientific tools for python. http://www.scipy.org. 18

Prusinkiewicz, P. and A. Lindenmayer (1996 [1991]). *The Algorithmic Beauty of Plants*. New York: Spring-Verlag. Available online at http://algorithmicbotany.org/papers. 17

Tymoczko, D. (2006). The geometry of musical chords. *Science 313*, 72–74. 3, 7, 15

van Rossum, G. (2006). Python. http://www.python.org. 18

Vercoe, B., J. ffitch, et al. (2006). The canonical csound reference manual. http://www.csounds.com/manual/Csound5.00_manual.pdf. 18