pcsets:
Pitch Class Sets for Python

Bruce H. McCosar

August 19, 2007

Contents

Chapter 1

Introduction

This tutorial is an introduction to two subjects at once:

- **Pitch Class Sets**, and
- the Python module **pcsets**.

I'm hoping that the two will reinforce each other. That is, the standard method of learning about Pitch Class Sets might be to read a book[1] or take a class. Very few times do you have the opportunity to actually *play* with them—to get some hands-on experience. Even if some brilliant inspiration strikes, sometimes the aggravation of working these set operations out manually can drag the idea to ground before it even has the chance to fly high.

Well, here's an opportunity. A lot of the examples can be run in Python interactive mode. You can not only learn the concept, you can *try* the operation, immediately. You can develop your own ideas or theories and prove or disprove them quickly. For the more programming-oriented, I'm also including a few coding challenges that make use of the material you're learning.

But above all, try some of these ideas out: *play* in the musical sense. If you play an instrument, let these concepts be a springboard to developing original music of your own. I'll give you some examples of how I'd approach this, but keep in mind that I come from a jazz background. You come from your own musical background, and you should find your own path.

1.1 Summary

Pitch Class Sets are a mathematical model for analyzing and composing music.[2] Each note 'C' through 'B' has an equivalent pitch class number 0 through 11. Sets

[1] Actually, there are some very good books. I've listed the best in the **References** section (p. 7). I'll point out some of the most relevant as I go along.

[2] The classic work of Pitch Class Set theory is Allen Forte's *The Structure of Atonal Music* [For73]. Straus has also written a very readable introduction and overview [Str05].

of these numbers may be operated on by mathematical functions, leading to new combinations and creations.

The basic **pcsets** modules free you from the drudgery of computing Pitch Class Set operations by hand. Moreover, incorporated into a programming language such as Python, the package allows application of the Pitch Class Set concepts to the broader use of musical interpretation and creation.

1.2 Why Pitch Class Sets?

Basically, one day, I got sick of II-V progressions. I'd read a lot on chord-scale theory, but I wasn't satisfied that was the whole of the musical universe. Sometimes I would play something, realize it worked, then think, *Hmm—now what was that?* Standard music theory just didn't cover it.[3]

Looking back through my music notebooks a few weeks ago, I found the seed that started this all. On one page, I'd tried to work out a new chord progression, then wrote a final thought on the page:

Is there a periodic table for chords?

It turns out that there was. You'll read about it in a later section (§ **??**, p. **??**). Since then, I've found them to be a general use creative tool, not just a static table.

1.3 Why pcsets?

I decided to write this Python module after finding most of the programs available online were GUI-only / interactive only (or worse ... applets). For various reasons, I needed to be able to set up long computational chains on a group of pitch class sets. Typing them into a web browser one at a time and poking buttons was *not* an option!

I released the module to the public under the GPL (Appendix **??**, p. **??**) for three reasons:

1. It might serve as an educational tool for music theory and Pitch Class sets.
2. The addition of functions to connect set theory to the more traditional chord / scale theory might lead to innovative, new types of music software.
3. There wasn't a module available that provided the same functionality.

[3] This is something a lot of jazz musicians run into, eventually. Mark Levine was the first author to bring it to my attention [Lev89, p. 250] with an unusual Herbie Hancock chord, notated "E♭$7^{\flat 9}$/F". There aren't really any standard scales having E♭, E♮, and F.

1.4 The "Uh Oh" Section

I'm assuming, if you're reading this, that you're familiar with:

- At least a little music theory;
- the programming language Python; and
- mathematical concepts such as *function* and *identity*.

I'm also assuming you've read the distribution's README and INSTALL files, and have the **pcsets** package installed on your system properly. I mean, after all—considering the range of the above requirements, you must be pretty good at whatever you do![4]

[4] All six of you who are reading this, that is. ☺

Chapter 2

The Basics

2.1 Pitch Classes

Pitch Class values are the "atoms" of the **pcset** world. However, they do not cover every possible musical situation.

2.1.1 Restrictions

In the literature, and in the **pcsets** package, Pitch Class values are defined for *12 tone equal temperament* only.

There are, of course, other methods of intonation, and other intonation systems around the world. In fact, a PcSets module tweaked for some sort of microtonal music might be pretty interesting.[1]

But not today.

2.1.2 Definition

A *Pitch Class* is a single integer that represents a particular musical note on the 12-tone equal temperament scale. 'C' is defined as zero; ascending the scale, each note is assigned consecutive integers until 'B' (11).

2.1.3 Enharmonic Equivalence

Because of the equal temperament basis, it is reasonable that enharmonic equivalance translates to *exact* Pitch Class equivalence. No distinction is made between C♯ and D♭: they both have Pitch Class value 1.

[1] David Lewin demonstrated this concept quite well in Chapter 2 of his book [Lew07], which is highly recommended reading if you are more into the mathematics of set theory.

2.1.4 Octave Equivalence

Perhaps the only Pitch Class value rule that may seem strange to a musician is the principle of **octave equivalence**. Any note with the name 'C'—whether it be played on the fifth string of a 7/8 scale upright bass or the highest key on a piano—is considered to be pitch class zero.

Musicians are accustomed to thinking of notes being in a particular order, or in particular positions relative to other notes. Using Pitch Class values, the "relativity" information is lost.

Pitch Classes, therefore, are an *abstraction* of musical structures. The information which we lose here is not really "gone". A musical structure translated to pitch class values has been reduced to some sort of essential structure; if we make any alterations to the structure, *we*, human beings, have to **construct a new meaning** from the result.

I wish to emphasize this as a particularly important point. We will see a lot of pitch class set transformations later on. Taken at face value, *they mean nothing.* They only acquire meaning through our interpretation.

I can write a Python module to do the math, but it's up to the user to provide the magic.

2.1.5 Pitch Class Value Table

The above rules lead to a fairly simple note name to Pitch Class conversion table:

note name	C	$\frac{\text{C}\sharp}{\text{D}\flat}$	D	$\frac{\text{D}\sharp}{\text{E}\flat}$	E	F	$\frac{\text{F}\sharp}{\text{G}\flat}$	G	$\frac{\text{G}\sharp}{\text{A}\flat}$	A	$\frac{\text{A}\sharp}{\text{B}\flat}$	B
pitch class	0	1	2	3	4	5	6	7	8	9	10	11

Please note, however, one simplification, which will be consistent throughout the **pcsets** module: I have omitted from the table any "accidental" notes that have a natural equivalent. There is no F♭ on the table; E is always E. Similarly, there are no double flats or double sharps.

I call this the *natural rule*. There is no particular mathematical basis for it; in fact, there is no particular reason to have or use note names at all, other than convenience. For this reason, the core `pcsets.pcset` module is a separate entity from the module which deals with traditional note names (`pcsets.noteops`).

Bibliography

[For73] Allen Forte. *The Structure of Atonal Music.* Yale University Press, New Haven, CT, 1973.

[Lev89] Mark Levine. *The Jazz Piano Book.* Sher Music Company, Petaluma, CA, 1989.

[Lew07] David Lewin. *Generalized Musical Intervals and Transformations.* Oxford University Press, Oxford, 2007.

[Str05] Joseph E. Straus. *Introduction to Post-Tonal Theory.* Pearson Prentice-Hall, Upper Saddle River, NJ, third edition, 2005.

Index