Matthias Smith
P498POM
Final Project

Pat Metheny in a Box

Introduction

The project I worked on for the semester concerns itself with using analytical methods to produce a competent jazz simulation program. I used a range of ideas that I believed could produce viable jazz lines based on an arbitrary chord progression. Matlab provided my weapon of choice for the programming and production of midi sound files. The most obvious starting point for such an endeavor would be a simple probability based method with a single value for the probability of each note occurring. I used this method as my basis for alteration in further methods. The next idea I had was to consider each pitch as an entity coupled to the two preceding notes and create a weighted three dimensional array of probabilities. I have always found chaos fascinating, so I decided to implement simple analysis treating the musical structure as a complex system relating to the nonlinear regime of the logistic equation. I gathered data for pitch sequences and rhythm structure from a Pat Metheny Songbook. After the initial prototype became functional, I conceived a few more subtle changes that could improve the content generated by the jazz simulator.

Data Acquisition

In order to start programming I needed data to construct probabilities matrices. All the data came directly from a Pat Metheny Songbook containing some 140 songs. I used over 30 for my pitch data. The chords were simplified into 4 basic jazz chords. All 7th based chords were considered as only 7th's. For example Am11 or G13b5#9 would be simply considered as Am7 or G7 respectively. I only considered the four basic jazz chords, i.e. minor, major, dominant and diminished 7th's, to provide a solid basis probabilistic constructed of jazz lines. Pat Metheny must have a personal disdain for the diminished 7th, for though it is fairly common in jazz, he only used it twice in 30 songs. Therefore I supplemented the data for the diminished 7th with arpeggio content from the chord.

For the rhythm data I scoured about 10 songs. The rhythm was existent throughout all content unlike separate chords, so it did not require sampling from as many jazz charts. I considered 3 separate cases for the rhythm data. The rhythms were classified as either melodic, color/texture, or solo based. Such classification did introduce a subjective element to the data I took, but while it was biased it should be consistent as I gathered all of the data.

Functions

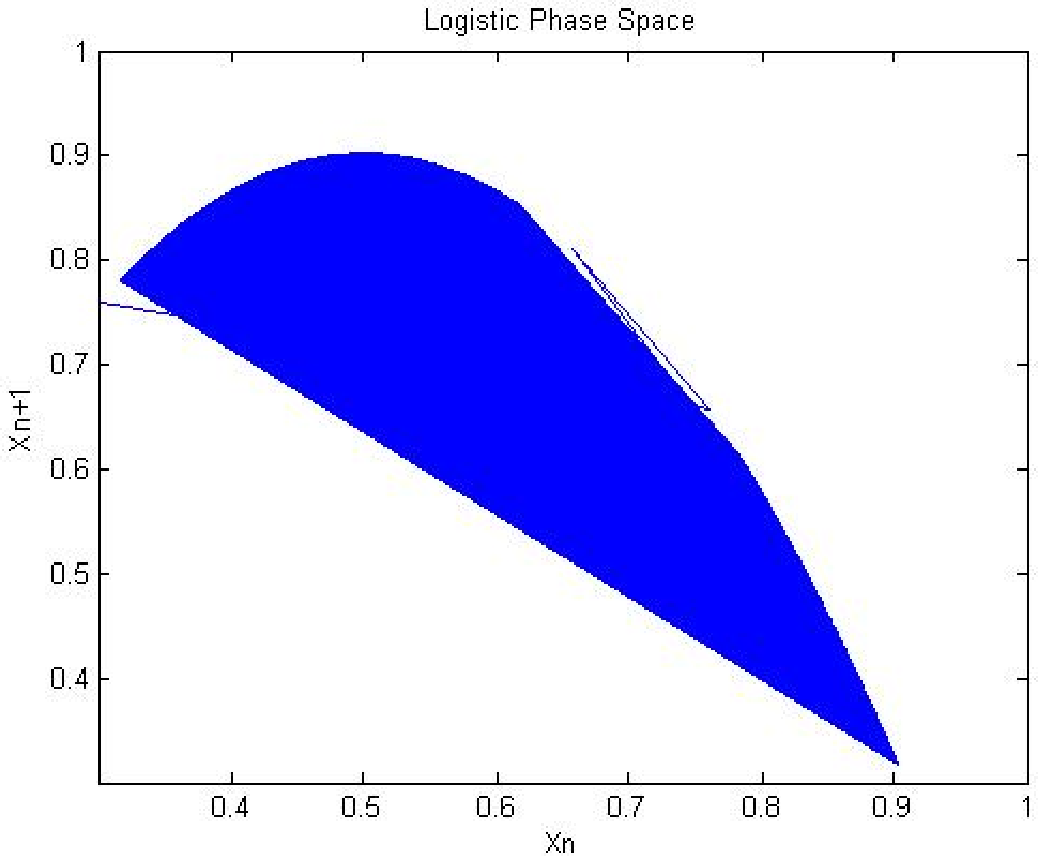
Rhythm:

The first step in creating a jazz simulator was developing a method for assigning a rhythm structure for the music sequence. I considered similar methods to those being used for the generating pitch, but I felt they were all inadequate for the purpose of rhythm. The inherent randomness in straight probability could yield rhythmic measures that were simply too random, such as a sixteenth note followed by a quarter note triplet then a half note and another sixteenth. Instead I decided to employ a two-fold case structure that broke each measure into a group of eighth notes, then added further substructure to each group of eighth notes. For example, one measure could be broken into a 3-3-2 emphasis pattern then the 3's into 3 and 1-1-1, then the 2 into 1-½ -½. The resulting rhythm function produced satisfactory results based on user input about the length of the chord progression, and I moved on to work on functions for assigning pitch.

Pitch:

The first method for generating pitch for the rhythm data generated considered only a simple note density for each chord. The probability matrix used for this method is a simplification of the chord data taken from Pat Metheny's songs. Each matrix leading to a specific note in the three dimensional array was reduced to its magnitude for each chord. Then using a function which generates a random number and checks its placement among a normalized probability vector generated a note from 1 to 12 with 1 being the root of the chord underlying the melody. The resulting musical sequence was not typically impressive.

Another method I explored used the logistic equation, $x_{n+1} = Rx_n(1 - x_n)$, in its nonlinear domain, $(R > 3.47)$. The idea was that hopefully this method would add some natural sequencing into the randomness produced by simple probability. By analyzing a phase space plot I constructed a probability vector that nearly matched the simple note density vector for all four chords when summed over 10,000 iterations of the logistic equation.



The notes were extracted in a similar manner as with note density, but instead of using random numbers, iterations of the logistic equation from a random starting point were used. The resulting musical sequence sounded better than I expected. It seems that the complex analysis method did introduce some patterns while generating pitches.

The final method I employed was probability with a deeper structure. Every important chord had its own 12 by 12 by 12 matrix for extracting notes from. Each note has 144 separate probabilities from each of the combinations of two notes that could possibly precede it. The idea is that the resulting note progression should have more pieces of the patterns used by Pat Metheny from the data taken. The resulting progression did in fact contain more pieces of patterns and seemed less random than the other methods. Overall this method seemed like the most effective method for synthesizing jazz lines.

Improvements:

All of the previous methods for generating pitches in the note sequence contain the same limitation. Each only allows for the possibility of 12 notes in the octave above the reference pitch for the root note. In order to make the music seem less synthetic I decided to implement a supplement which allows the pitches to move up or down from the previous pitch, but with limited magnitude for the jumps. My solution was to put the "pitch on a leash." I allowed jumps less than an octave by giving the pitch changes two options an octave apart when deciding the next pitch. The probability of each jump is inversely proportional to the magnitude of the jump with a small additional weighting for the smaller jump. For example take a change of +4 semitones. Possible outcome: +4 or -8 semitones while $P(+4)=\alpha(1/4)$ and $P(-8)=(1/8)$ with some normalization to make the total 1. For my function I made $\alpha=1.2$. The "pitch on a leash" function greatly improved the quality of the lines produced by my jazz simulator.

Conclusion

I endeavored to produce a jazz AI rather naively with only a few ideas. My ideas provided a good foundation, but they were more difficult to implement than I had imagined. I kept adjusting and fine tuning the methods until I produced workable prototypes. Overall I learned a lot about applying probability in programming as well as relearned much of the programming I had previously forgotten. I also feel that I have a better understanding of the way jazz players think by examining which methods were relatively more effective than others. After listening to numerous samples from each method of jazz simulation, I consider the sequenced probability method the most effective method for synthesizing jazz lines.