Simulated speaker feedback using Python

Jason Kaszpurenko
UC Davis Physics Department
June 9, 2009
jkaszpurenko@ucdavis.edu

In this paper I will attempt to create a speaker feedback loop with multiple speakers, different gains in each speaker, and even demonstrate the Doppler effect with moving speakers. The applications can range from data storage, noise generation, and creating unique audio effects. A mathematical framework will be given that will relate this system to an analogous system of ring buffers or circular buffers that are used for data storage in computer systems. Results showing the various sound effects that can be achieved using this simple setup will be shown.

Introduction/Background:

Speaker feedback is a commonly seen phenomenon in daily life, a speaker at a presentation accidentally makes a loud screech when they walk in front of the speakers with their microphone, a person hears a small echo when talking on a speaker phone, or a person calling a radio station leaves a loud screech. All of these things occur when the speaker outputs its signal to a microphone that retransmits it back to the speaker, creating a loop. I originally investigated this system as an intellectual curiosity but in my research I have found that other scientist and engineers had already discovered it and have been using it for quite some time.

The system exhibits several interesting behaviors that are worthwhile studying, first it is non-linear. In figure 1 we see a basic speaker feedback loop. If the speaker is far enough away, and the gain from the speaker isn't large enough, the sound will be repeated many times but will fade out after a number of cycles. This can be thought of as an echoing behavior, a repeating signal but continually getting smaller and eventually going away. However if the source is to close to the microphone it will continue to repeat itself getting louder and louder. This is the same process that creates a screech from a speaker at a conference.

Figure 1
A microphone that is connected to a speaker that is pointed towards it

The system is also not limited to using linear amplification sources or one speaker. There may be regions where the signal never dies out fully or explodes to infinity with multiple speakers. The amplification can also be a more complicated function such as a logistic function or a piecewise function designed so that the signal never decays to zero or diverges off to infinity.

An analogous data structure called a ring buffer, (also known as a circular buffer), has been used as a data storage and output architect as far back as the 1960's, possibly even before that. As well as being used in current engineering and scientific equipment. The data structure allows for an infinite (analog) or finite dimensional system (digital), see figure 2. In this setup new information about the world is constantly replacing old information within the system, however a certain number of bits of information are still being stored. Essentially for every degree of freedom you add to the system you create an extra bit of information that can be read, stored, recorded over or analyzed.

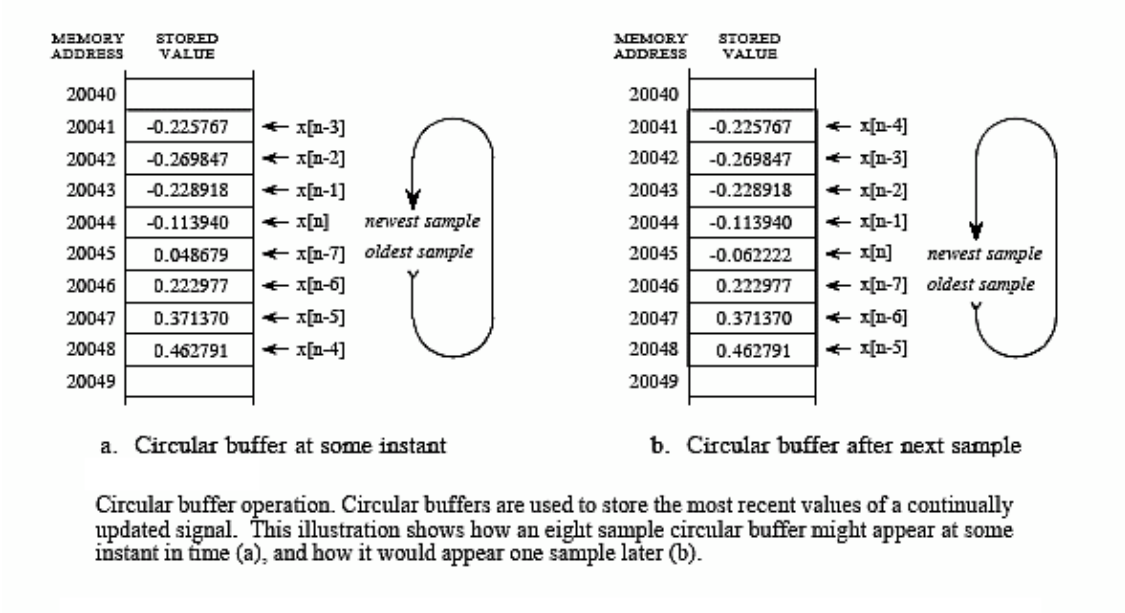

Figure 2
Circular buffer diagram showing how certain computer data structures use a similar concept of a repeating feedback loop

When playing around with the system I was able to create a wide variety of sounds, many of which were purely noise. However I was able to create several interesting rhythms, change pitches and produce some very odd signals. These signals are just the beginning, a person feeling inclined enough could modify there own pitch several times over with multiple speakers and create there very own one man chorus.

Dynamical system:

In a real speaker and microphone setup you will need an infinite number of points to describe the sound at a given time in the future, making it an infinite dimensional system.

This infinite dimensional property is from the fact that you need to completely describe the initial system to describe the system at any time t after but the inputted signal must be continuous since it is a real world system. This can be discretized in simulation by discretizing time and creating a finite number of spaces in-between the speaker and the microphone. This number can easily be in the thousands, so the dimensionality of the system is still much higher than we are used to dealing with in our daily lives. This can be very intimidating at first but you can view the system as a bit shift for each time step that you take, see figure 3. By doing this we create a simple mechanism to describe the system, equation 1 and 2.

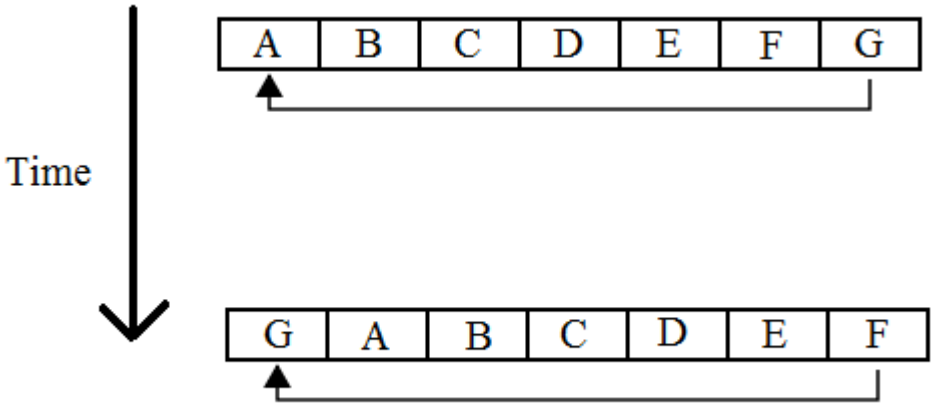

Figure 3
A basic diagram for a bit shift, very similar to figure 2

$$g(t+1,i) = g(t,i-1) \text{ Eq. 1}$$
$$g(t+1,N) = f(g(t,0)) \text{ Eq. 2}$$

In this description of the system $g(t,i)$ is the state at a given time t and position i. When i=0 you are at the microphone and i=N is the signal at the speaker. The term $f(x)$ is the function that you choose to run between the microphone and the speaker. The statements above literally read that the state at a given position is the state of its adjacent space (the one closer to the speaker) at one time step before. And the state at the speaker is the state at the microphone at the previous time step passed through some function of our choosing.

This works for one speaker but it should be modified for multiple speakers. Equations 3, 4 and 5 are a set of equations for multiple speakers in a row.

$$g(t+1,i) = g(t,i-1) \text{ Eq. 3}$$
$$g(t+1,N_i) = f_i(g(t,0)) + g(t,N_i+1) \text{ Eq. 4}$$
$$g(t+1,N_f) = f_f(g(t,0)) \text{ Eq. 5}$$

The term N_i represents the i[th] speaker away from the microphone. N_f is the speaker that is the furthest away from the system. Essentially, the description above is still valid except at certain points where new speakers are located at. The sound will be from the bit closest to the outside of the system plus a signal from the speaker at that point.

Another point worth mentioning is that these equations are for a discrete system but as we set the limit of the size of the steps to zero we see that we will create a continuous system.

Methods:

Although the equations for the system describe the state at any given state at a given time t later it can take a large amount of time to describe every single point between the speakers. If there are 4,000 spaces (N) between the furthest speaker and the microphone and I want to run the simulation for 20 seconds in real time with a low sample rate of 6000 Hz I'm left with keep track of 480 million points. But it is possible to view the system only from the microphones point of view. Although this greatly reduces the size of the system we will still keep all the dimensionality of the system. The system can be described by ~1/N of the size as listed above. To do this I propose a new function $h(y,z)$.

$$h(y,z) = \begin{cases} 0 & z < 0 \\ y & z \geq 0 \end{cases} \quad \text{Eq. 6}$$

We call our initial function $\lambda(t)$, essentially our seed for the system. The sound located at the microphone at a given time later can be described as:

$$g(t) = \lambda(t) + \sum_i f_i\left(h_i(g(t-\tau_i), t-\tau_i)\right) \quad \text{Eq. 7}$$

In this τ_i is the time delay of the sound of the i[th] speaker to the microphone. As long as we record the function $g(t)$ we can keep track of all system for any previous time and can know the signal at any given future time provided we iterate the system enough times.

In a computer the system has very large maximum numbers in it and certain system immediately go off to infinity. To counteract this problem I implemented a maximum signal output. (This is not actually used in the results in the results section). This maximum signal code replaces the g(t) at a given time by a ±1 if the abs(g(t)) is greater than 1. This corresponds to a physical limitation of the microphone to record sounds greater than this value. So g(t) is the sound recorded at a given time by the microphone into the speaker.

This method does leave something to desire however, even if the microphone is unable to detect signals of that size it doesn't change the fact that the actual signal at a given point

could be greater than that which we record. Although not implemented, this could be rectified by including another function that displays the actual current value of the system as compared to what the microphone records.

Another advancement I have made with the model is changing the position of the microphone as a function of time. This can be done by implementing a code that as the system iterates after a certain number of iterations goes by τ_i is increased by a user control value.

Results:

One of the first functions that I ran through the system was a short ramped signal through a simple linear amplification using 3 different speakers with the same gain. I placed the furthest speaker about 3 times as far away from the source. This is shown in figure 4 and doesn't have a maximum value for the microphone. In addition I ran a long ramp. What is neat between the two is that the signal will never return back to zero for the long ramp. It looks like a growing version of the original wave for the shorter pulse. I ran the signal for a long time and it would always go to zero at one point.

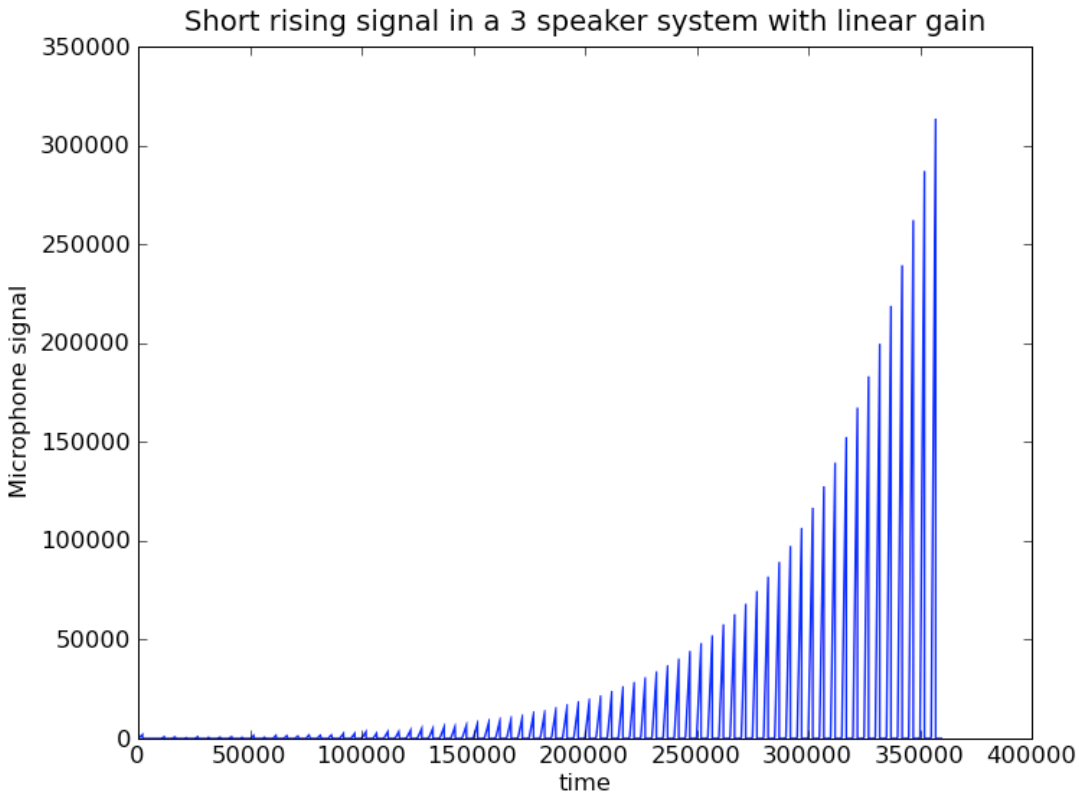

Figure 4
A ramped function, even as time progresses you are still able to see the original signal shape as it grows larger and larger

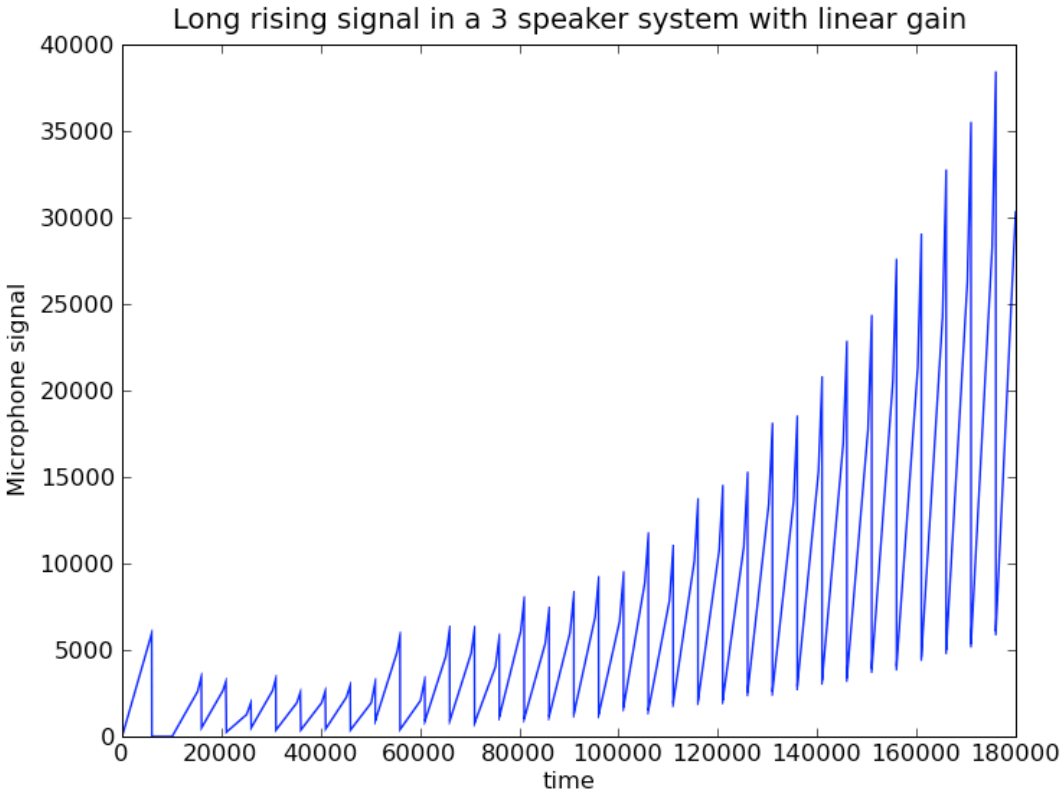

Figure 5
As time progresses the signal moves away from the x-axis and will continue to grow

Another attempt was to see what happens with 3 functions with the logistic with r at 1.3. Now I've experimented with this with values converging with 0<r<1 and diverging at values r>1.3. What I found particularly interesting was for a short pulse in the logistic function it appears that I've created noise, as seen in the graph as a function of time. But when I plot time vs time +1 I find a very striking linear relationship with several values around zero. I'm assuming those are from the fixed point at zero. But when I use a ramped function I create a very interesting time vs time +1 graph and I have a hard time describing precisely what it is.

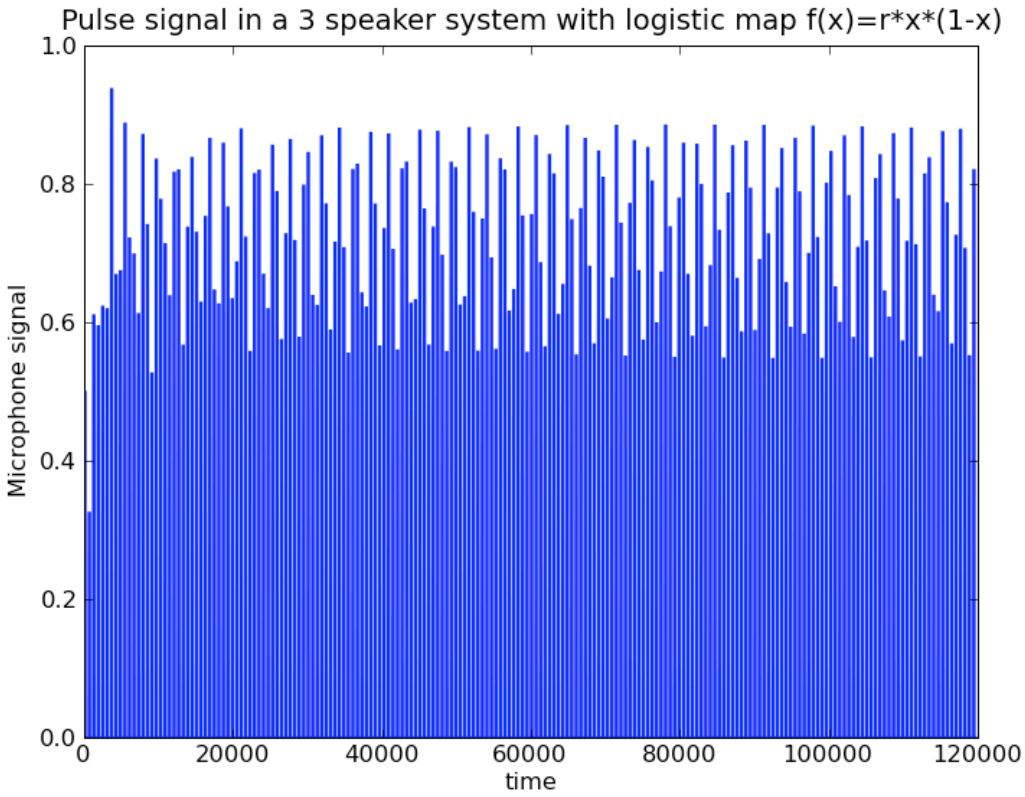

Figure 6
A 3 map logistic function, it rapidly goes to a region that never exceeds 1 or goes below zero. This signal is originally from a constant 0.5 signal

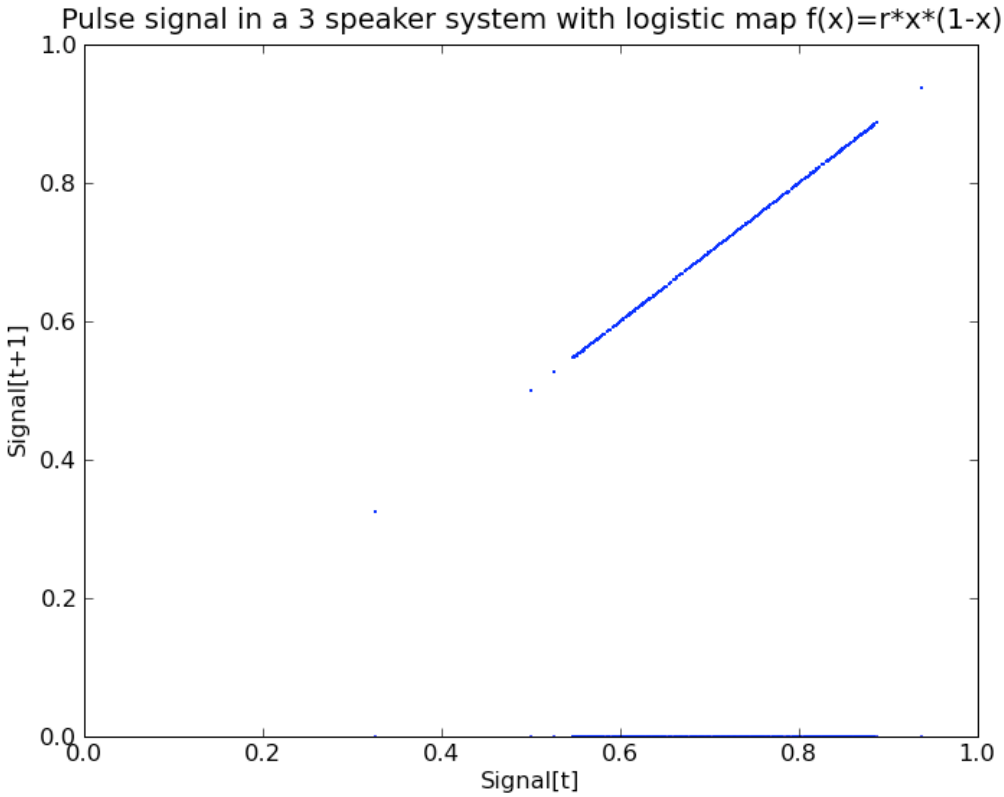

Figure 7

Plotting the same logistic curve against the signal at t vs the signal at t+1 we notice a very linear behavior. The past position depends heavily upon the previous position in a linear fashion. There are some values between ~.5 and ~.9 on the Signal[t] that lie on the zero point, these most likely correspond to the fixed point at zero

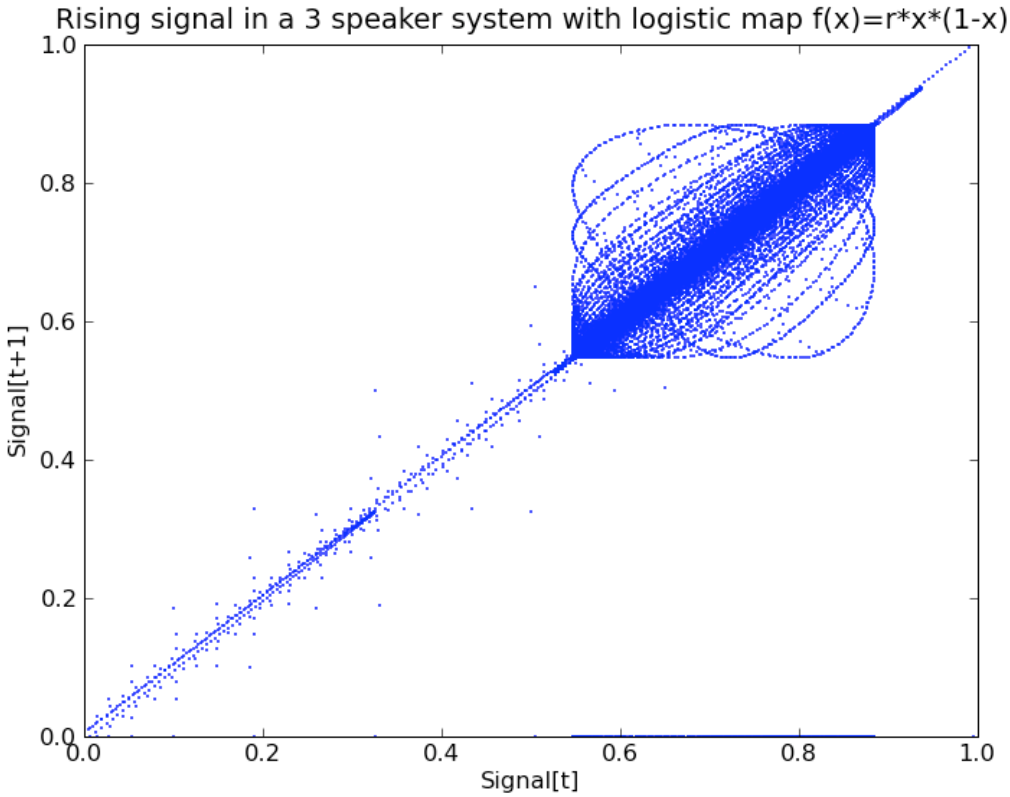

Figure 8

This is the same logistic system above but this time with a ramped input signal. I am unable to identify that pattern that it is making but it does appear to be somewhat chaotic to me. More investigation would be needed to confirm if it is indeed chaotic.

Another effect that I was able to create was the Doppler effect with a single speaker. I took it and every fourth iteration I moved the speaker away from the microphone one spot. As time goes on you can see the signal get longer and when you hear it you hear a very definite change in tone. The amplification was linear and r was set to 1.

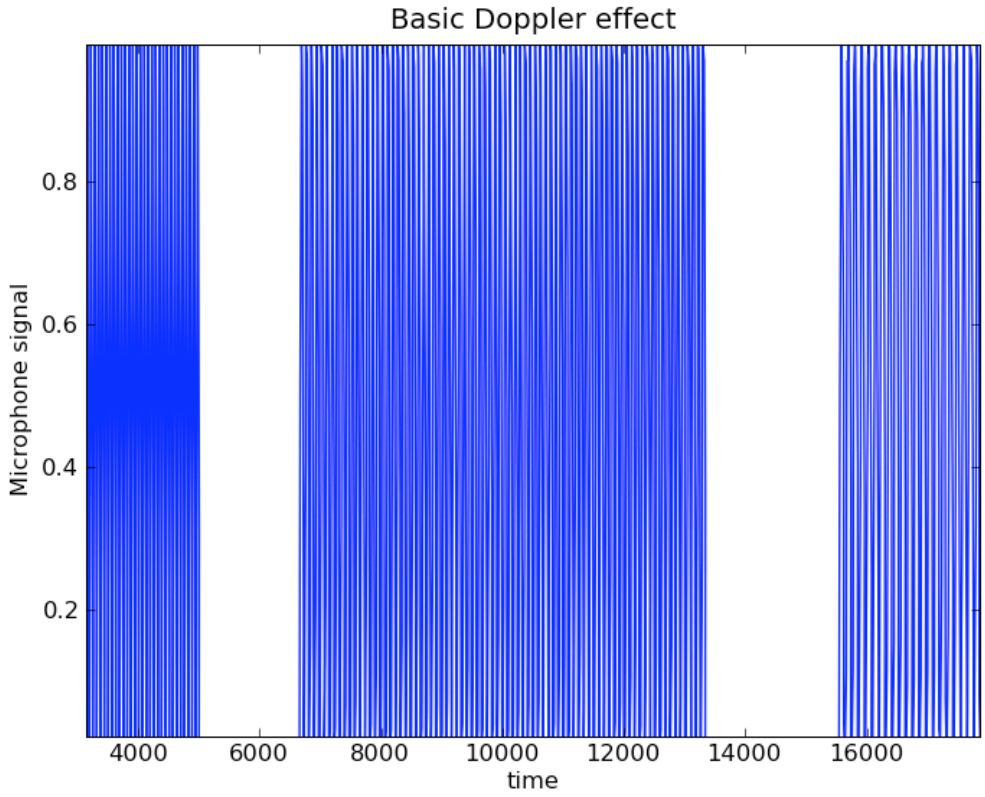

Figure 9
An example of the Doppler effect using my simulation. Although zoomed in it is difficult to see the fact that the different sections correspond to different frequencies. This system used only one speaker.

Finally I tried to see if I could make some odd noises using just a simple two speaker setup. I set the gains to linear an played a simple sine wave through it. I later implemented some code to confine most of the signal between +-1. This allows a better conversion to a wave file. There are several different frequencies heard and a changing rhythm can also be heard.

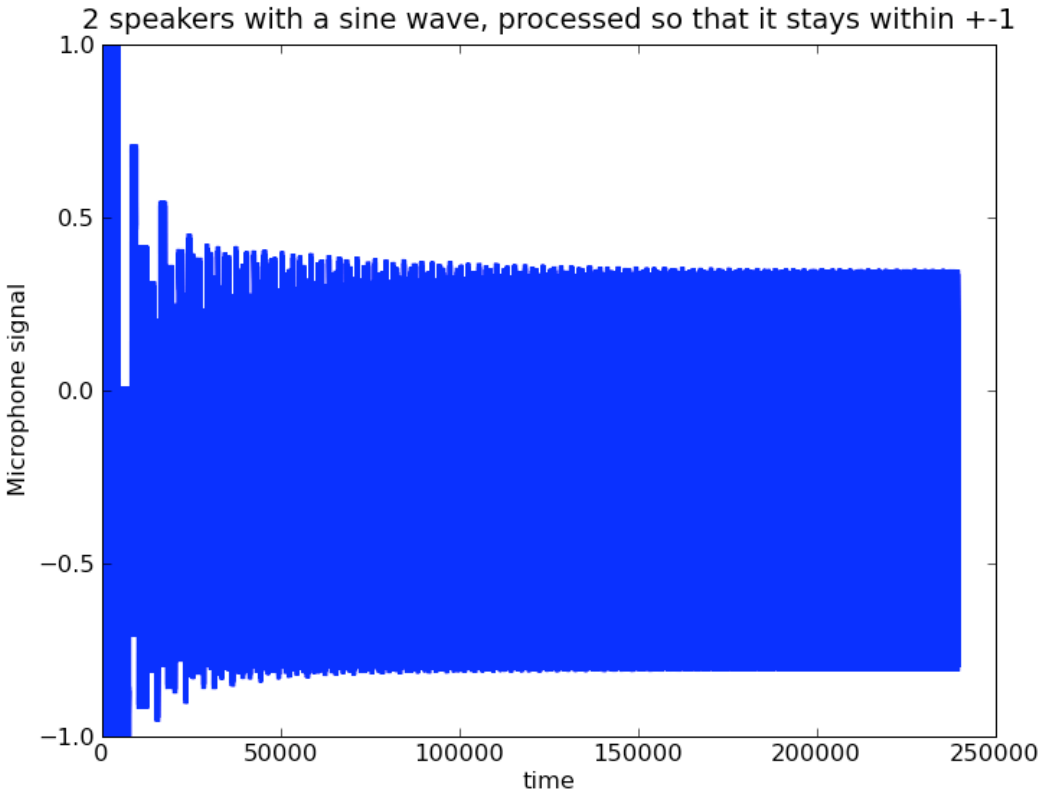

Figure 10
This is a signal that was produced with an initial signal of a sine wave that was than modified to be confined with +-1. Even if it isn't confined you can hear a noise, but it is very noisy making it harder to hear the rhythm.

Most of these finding are better heard than viewed on a diagram, for that reason I have included the Python file, so you can create your own, and several of the sounds that I have produced.

Conclusion:

In starting this project the goals were very abstract and not done for any particular application. The whole experiment was started from an observation of two cell phones put on speakerphone and held next to each other and my personal fascination with audio. More of a thought experiment to see if I can could create a dynamic feedback system with a non-linear behavior. But through my exploration of the system I've found that it has applications in memory management and can be used to create unique sounds.

I believe I have come up with a simple model computer model to describe feedback loops within Python's framework. This system is versatile and allows the user to add many more speakers than I did, as well as more microphones and different effects they can try with the speakers, (moving them, different amplifications, amplifications varying over

time....) In the end the tool I created can be used to allow the user to explore with there own creativity.

Bibliography:

Cite background materials and current related papers and books.

Images for figure 1:
http://www.acclaim-music.com 6/9/09
http://www.hometheaterhifi.com 6/9/09

Images for figure 2:
http://www.dspguide.com 6/9/09

Images for figure 3 were modified images from wikipedia:
http://en.wikipedia.org/wiki/Circular_buffer

Nonlinear Dynamics and Chaos: with applications to physics, biology, chemistry, and engineering, S. H. Strogatz, Second Edition, Addison-Wesley, Reading, Massachusetts (2001)