DSP AUDIO EFFECTS

Vicerut Nonzee
Piya Poongbunkor

ECE 320 Final Project Paper
May 3, 2001

Section F4

Introduction

Many musicians are now looking towards using digital signal processing of audio signals instead of using analog effects boxes that were more common just a few years ago. There are many reasons why this new trend towards using digital signal processors (DSPs) is occurring. By digitizing an analog audio signal and using a DSP, an audio signal can be manipulated in an infinite amount of ways that can sound pleasing to the human ear and benefit the musician. A single DSP can do the work of hundreds of analog processors, and can do things that analog processors are unable to do, as well. It is very convenient for musicians to use DSPs for audio effects because DSPs condense all of the functions and space of analog circuits. For our project, we decided to explore the realm of audio effects, using digital signal processing to manipulate audio signals. We were particularly interested on replicating some of the more "classic" guitar effects that are normally implemented by analog signal processors.

There are many different effects to implement digitally, but we finally decided to pick the two audio effects known as "flanging" and "wah". We chose these two effects because the ways to implement both of these techniques were very different. Flanging theory is an effect based on oscillating time delay, while wah theory was an effect based on using filters to alter the frequency response of an audio signal. Also, we thought that implementing both of these effects would have very unique and pleasing sounding results. Flanging can be described as a circular sweeping sound going across an audio signal. The listener is able to hear a pitch change oscillate throughout an audio signal from a low frequency to a high frequency, then down back to low, etc. Mixing the audio signal with a very slightly delayed version of itself and oscillating the length of delay creates this. The wah effect involves using filters to simulate a resonance on the

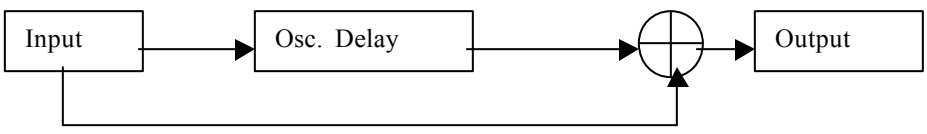
signal to go up and down the frequency of the signal. This creates a "wah" sound for a guitar signal. Both of these effects are used widely by electric guitarists and bassists.

Our goal was to take what we learned from this class so far this semester, coupled with our experience with the project labs, and create code that would implement both of these effects. After finishing writing our code, we would put an audio signal through the DSP and run the code, while listening to the output effect through speakers. We had some different options for what to use as our audio input. We decided to use a 1 kHz square wave as the audio input to the wah code, and also decided to use a microphone having our voices be the input for the flanging code. Using an electric guitar as input was also an option that we considered.

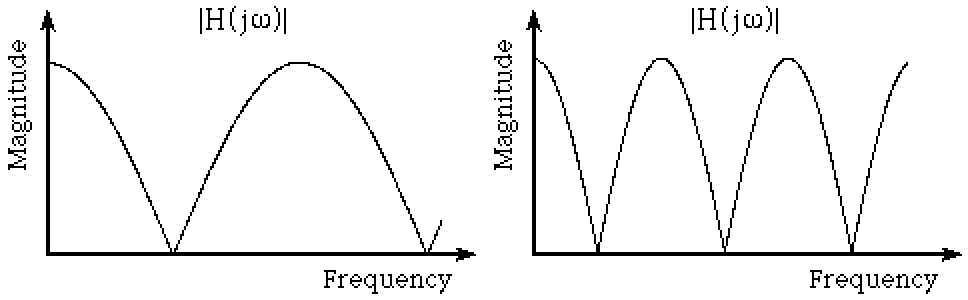
Description of Implementation Details

Flanging Audio Effect

The flanging audio effect produces variations in pitch along a sweeping motion. The effect will accentuate specific frequencies and dampen others, and as time passes, the effect will widen and narrow its region of accentuation. Accentuation of a certain band of frequencies can be implemented with a series of filters, however the motion across the input cannot be easily achieved with simple stationary filters. Many coefficients would need to be loaded, and organizing the placement timing of these filters would prove to consume memory and valuable calculation time. Instead of implementing this arduous task, a simpler, yet equivalent, approach is taken. The flanging audio effect is created by adding an input signal to a slightly delayed signal of itself, as shown in the block diagram below.



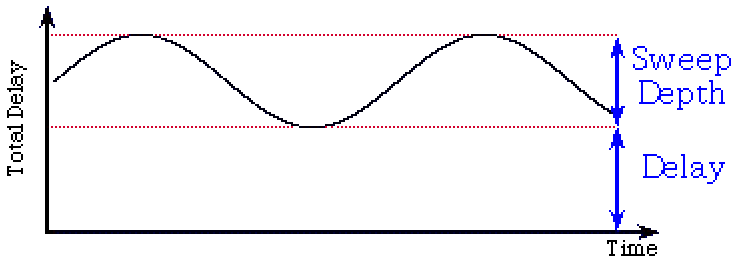
This connection results in a comb filtering of the input signal. The delayed input signal destructively interferes with the original input signal and causes notches in the frequency response, see figure below.



These notches correspond to the dampening of certain frequencies, while the other frequencies remain accentuated. Also present in this approach is the use of an oscillating delay. Each length of delay corresponds to notches at different frequencies. So, as the delay changes, this approach produces the sweeping motion of the flanging effect. The final result of this approach is the completed flanging effect.

The flanging audio effect consists of two major components, the delay and the low frequency oscillator (LFO). The delay consists of techniques used in project lab 1. Large buffers were created in the external memory of the digital signal processor. These buffers served as storage areas for the delayed input signals. By writing and addressing the buffers correctly, the program could recall those values at any time. However, timing was controlled by a counter within the program memory. After each sample the counter would increment, and when the desired delay length was reached the counter reset itself for the next delay length. In project lab1, the delay length was a set constant, however, for flanging, it is necessary to have an oscillating delay length. This oscillating delay length is controlled by the LFO. The LFO should be a periodic waveform with a characteristically low frequency. A sine wave is a perfect

function for this project; however, a simpler waveform that requires fewer processing calculations is a triangular wave. A triangular wave implementation consists of a counter, a subtraction function, and an addition function. The addition and subtraction functions serve to oscillate the wave. The counter length controls the time between function execution and thus increases or decreases the frequency of the oscillator. In addition to just creating a basic triangular wave, the flanging effect required the wave to shift up to a finite delay. This ensures that a notch filter will be created in the frequency response of the output. An illustration of the LFO as a sine wave appears in the diagram below.

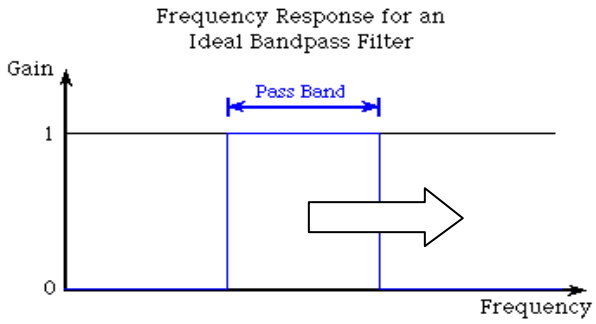


In this diagram, the wave remains bounded between a maximum and minimum value and varies with time. In the flanging effect implementation, boundary conditions were also created. Conditional statements at each of the boundaries directed the use of the addition and subtraction functions. With both the delay and the LFO working correctly, the flanging effect is completed with the addition of the input signal and the oscillating delay signal. The layout of this implementation would achieve the desired flanging audio effect.

Wah Audio Effect

The wah audio effect produces an impression of motion in pitch. The effect amplifies a small band of frequencies, and as time passes, the amplified band shifts toward higher

frequencies. Over time, the wah effect creates the perception that the input signal is raising its pitch. However, the wah effect merely accentuates a different section of frequency at a different point in time. In other words, the center frequency of a filter that moves from lower to higher frequencies causes the wah audio effect. To accomplish this effect, rotation of filter coefficients is implemented. Each set of filter coefficients corresponds to a band pass filter around a specific center frequency, see graph below.



As the filter coefficients change the pass band will travel right into the higher frequencies. This motion will create the illusion that the pitch is changing, essentially the wah effect. By controlling when these filter coefficients execute upon the input signal, the wah effect can be produced.

Implementation of the wah audio effect consists of adding the band pass signal to the original input signal. However, the band pass signal is always changing with time. In order to control the changing filtered signal, organizing the use of the coefficients becomes the most important task. A counter determines the organization of the filter coefficients. This counter delegates the time between the loading of each set of coefficients. The 44.1kHz sampling period allows the easy conversion from sampling time to real time. By loading filter coefficients according to increasing center frequency location at incrementing times, the wah effect presents itself.

Description of the Code

Flanging Code

The flanging code is centered around the execution of the delay section. All of the other subroutines serve to manipulate variables of the delay section. The most important variable of the delay section is the length of the delay. The set up for this variable becomes the focus LFO subroutines. After the initializations, the LFO counter is the first section calculated. The LFO counter varies the length of time it takes for the triangular waveform to change. Essentially, the longer for the LFO counter to add up to the finish value, the lower the frequency of the LFO. The code then just performs the previous delay length on the input, and it outputs the original input plus the delayed input. Once it becomes time for the delay length to change, the code compares the delay length to the boundary values, and then, it directs the delay length to increment or decrement based on its location within the boundaries. There are subcommands to increment and decrement which then jump the program to performing the delay length and outputs. Also a graphical interface from MATLAB send data through the serial port, which the code reads and updates the corresponding variables, such as the LFO counter length (i.e. the frequency of the LFO).

Wah Code

The wah code focuses on loading a different set of filter coefficients at increments of time. This is executed by first creating all of the filter coefficients in MATLAB. Band pass filter coefficients were made in MATLAB. Finite impulse response (FIR) filter coefficients were chosen because of the ease of calculation and 20 taps were sufficient enough to create the necessary filter. So, by using the remez command, thirty-one band pass filter coefficients with

gradually increasing center frequencies (from 3kHz to 12 kHz) were generated. The wah code copies these coefficients into memory and the proceeds to initialize variables. A counter where AR4 is pointing is set up. This counter is used to determine when the next set of coefficients is to be loaded. Comparisons to certain values correspond to time values. Once a time value is hit, the next set of filter coefficients is loaded and the band pass filtering of the input is executed. A gain variable adjusts the amplitude of the filtered input. Then the output just consists of the addition of the original input to the filtered input resulting in increased amplitude of the band passed frequency. The code loops back incrementing the counter and gets ready for the next set of coefficients. This implements a continuous rotation of the filter coefficients and a continuous wah effect. A serial interface to a MATLAB GUI also appears to control variables such as the amplitude gain of the filtered input

Accomplishments/Failures

Overall, we were disappointed with the outcome of our final project, but we did make some accomplishments after all was said and done. First off, we were able to hear a pitch change in the output of our audio signal (voice going through microphone) when we tried the flanging code. Although this is not exactly what a flanger is supposed to do, the basis of flanging is to create a pitch change, and this is the result we got from our code. This phase shift was due to the mixing of the original signal with a slightly delayed copy of itself, but we were not able to control the pitch change as we intended. This is thought to be our main failure in the flanging code. For some reason, our LFO (low frequency oscillator) did not function as we would like, and we did not get the correct flanging output. We did try to debug this by stepping through our code and checking if the conditions for counting up and down worked correctly, which was the

main part of our LFO. Making sure the code is going to the right places in our code at the correct times was very important.

We did not have much success with the wah effect at the project demo. But after going through the code again, we were able to hear a short "wah" sound when putting a 1 kHz square wave through the DSP. It was functioning correctly after we debugged the program. The problem with our code was that when our "compare" register (AR4 that we used to signal when to use the next set of coefficients) was not equal to one of the values we set, it would go through the loop and then be reset to 0. This meant that our coefficients for our filter never got to be used, and thus, our wah code did not output the proper "wah" sound. Again, we were able to find this by stepping through our code and looking at if the instructions did what we wanted to. The main problems that we had on our project came from problems with the code.

Extensions

If we had more time to complete this final project, there would be many things that we could improve upon. First of all, our code would be more properly debugged, and we would have had our code running our project to our liking. A better LFO for the flanging code could have possibly been developed; one that more replicated a sine wave than the triangle wave that we implemented. More extensive debugging would have also fixed our original triangle wave LFO. By again stepping through the code and looking at what the various registers and accumulators were doing, we could see if the LFO was working correctly. More time to analyze the behavior of the LFO also would have led to successful debugging. For the wah code, more extensive research and testing could be done on what type of filters we should have used. There

are many different sets of coefficients of filters to be considered. Butterworth? Elliptical? FIR? IIR? 2^{nd} order? 4^{th} order? The type of filter to use would have a large impact on the performance of a wah.

Also, an important concept could have been developed to complement our code, if we had more time: friendly user interface. We could have used the 2^{nd} project lab for audio effects as a guide and developed GUI controls to use in conjunction with Matlab through the use of a serial port. Controls for the flanger such as depth gain and LFO period could be very useful for the user. The same thing is true with the wah. Having a slider control the resonance frequency is a must for a wah pedal. Creating an easy user interface with the DSP is very important to create a marketable audio effects box.

Finally, if we had more time, we definitely would have liked to experiment with other various guitar based effects. Creating effects concerning time delay such as reverberation, chorus, and vibrato are certainly not outside the realm of this class, and would not be very different from the digital delay and flanging codes that we worked with. Other effects, such as distortion, ring modulation, and talk box would be interesting to research and try to implement. Digital signal processing is and will continue to be a very important part of audio effects.

Pitfalls

There were a couple of things that we would have liked to have known before starting to work on our code that would have helped speed the process of implementing our final project. One of the problems that slowed down the debugging and development of the code was figuring out how to use the conditional commands. Looking up how to use these commands took up some time. There are many different status bits and different ways of using these commands to

keep track of. Also, for the flanging part, knowing to make use of another counter beforehand would have also helped. For the wah effect, we also had found that using an IIR filter to create resonance in the audio signal would have been better to use. Also, using a low pass filter would have worked just as well as using a bandpass filter. It would have been easier to implement the low pass filter, and we would have been able to spend less time developing the filter. Knowing this would have saved the trouble of considering and developing other coefficients to use for the filter.

References

Harmony Central
www.harmony-central.com/Effects

Guitar Effects Oriented FAQ
www.geofex.com

Guitar Effect Newsgroup
alt.guitar.effects

Paras Naik, Signal Processing
webug.physics.uiuc.edu/courses/phys398/ Student_Projects/PNaik/

.