Multimedia
Module No: CM0340
Laboratory Worksheet Week 8:
MATLAB Digital Audio Effects

Dr. D. Marshall

Aims and Objectives

After working through this worksheet you should be familiar with:

- The basic theories of digital audio effects
- Implementation of a variety digital audio effects through filters, delays and modulation of their control parameters, *e.g.* delay period, frequency.
- The distinction between different audio effects classes.
- Familiarity with the basic *sound* of each digital effect.
- The basic implementation of digital audio effects in MATLAB.

None of the work here is part of the assessed coursework for this module ALTHOUGH many of the exercises below will help in parts of of your solution for the assessed coursework

MATLAB Digital Audio Effects

1. **Equalisers**: Using the shelving filter MATLAB code discussed in lectures (*http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/shelving_eg.m and shelving.m*) modify the `shelving.m` function to implement a ***peak*** filter. Try your peak filter out on some audio and vary the centre frequency, 'Q' and gain values and listen to the results.

2. **Phaser**: Using the Wah-wah MATLAB code discussed in lectures (*http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/wah_wah.m* but changing the filter to notch filter (instead of a bandpass filter). Implement a phaser effect. Try this out on some input audio changing the frequency parameters.

3. **M-fold wah-wah**: Using the Wah-wah MATLAB code discussed in lectures (*http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/wah_wah.m*) as starting point Implement an ***M-fold wah-wah*** effect — create M delayed bandpass filters with different centre frequencies and modulate each centre frequency as with the single wah-wah. Try this out on some input audio changing the frequency parameters. Try and create a ***'bell'*** type effect with a large value of M.

4. **Chorus**: Using the flanger MATLAB code discussed in lectures (*http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/flanger.m*) as a basis: Implement an ***chorus*** effect — by having *several copies* of the base effects unit and randomly modulating each unit's delay time within a suitable range. Try this out on some input audio changing the delay time accordingly. Listen to the results.

5. **Resonator/Slapback/Echo**: Using the flanger MATLAB code discussed in lectures (*http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/flanger.m*) as a basis: Implement a ***Resonator/Slapback/Echo*** effect — by removing the modulation and using an appropriate delay time range. Try this out on some input audio changing the delay time accordingly. Listen to the results, compare the different effects.

6. **Ring Modulation**: Using the ring modulation lecture examples (*http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/ring_mod.m*)and changing the carrier (and sine wave modulator in the example) familiarise yourself with the ring modulator effect. In the case of the sine wave example predict the audio tones you hear.

7. **Ring Modulation**: Using the ring modulation lecture examples (*http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/ring_mod.m*) and changing the carrier waveform from a sine wave to other simple waveforms and audio samples experiment with the ring modulation effect.

8. **Amplitude Modulation**: Using the ring modulation lecture examples (*http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/ring_mod.m*) as well as the amplitude modulation tremolo example (*http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/tremolo1.m*) as a basis implement a **two sine wave amplitude modulation**. Predict the audio tones you hear.

   Try amplitude modulation with other non-sinusoidal waveform modulations.

9. **Limiter/Compressor/Expander**: Using the example MATLAB code for each effect (*http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/limiter.m*, *compexp.m*, *compression_eg.m* and *expander_eg.m*) experiment with different parameter settings to audition and observe via MATLAB plots the effects of such changes.

10. **Exciters/Enhancers**: Using the Fourier transform implement a simple ***exciter*** and ***enhancer***

11. **Reverb**: Implement a 10 comb filter 2 allpass filter version of Schroeder's reverb algorithm in MATLAB. Use *http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/Reverb/schroeder2.m* and *http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/Reverb/reverb_schroeder_eg.m* as examples. Compare the output with the more classic 4 comb filter 2 allpass filter version.

12. **Reverb**: Implement a 12 comb filter allpass filter version of Moorer's reverb algorithm in MATLAB. Use *http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/Reverb/moorer.m* and *http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/Reverb/moorer_eg.m* as examples. Compare the output with the 6 comb filter example given in the lectures.

13. **Convolution Reverb**: Using the *http://www.cs.cf.ac.uk/Dave/Multimedia/Lecture_Examples/Digital_Audio_FX/Reverb/reverb_convolution_eg.m* and fconv.m examples discussed in the lecture apply convolution:

    - Find some free impulse response on the Internet and experiment with these in place of the ones given in the lecture, *e.g.*:
      http://www.voxengo.com/impulses/
      http://www.cksde.com/p_6_250.htm
      http://www.prosoniq.net/
      Try and find some odd ones that do not model reverb and try these.
    - Try convolution reverb using any short piece of audio as the impulse response.