Multimedia Module No: CM3106 Laboratory Worksheet Lab 4 (Week 5): MATLAB Digital Audio Synthesis (Part 2) Digital Audio Effects

Prof. D. Marshall

Aims and Objectives

After working through this worksheet you should be familiar with:

- The basic theories of digital audio synthesis: Sample-based, Additive Wavetable, Granular and Physical Modelling Synthesis
- The basic implementation of digital synthesis (Sample-based, Additive Wavetable, Granular and Physical Modelling) techniques in MATLAB.
- The basic theories of digital audio effects: Equalisers and Time-Varying/Delay Based Effects (Wah-Wah, Flanger, Chorus), *etc.*.
  - The distinction between different audio effects classes.
  - Familiarity with the basic *sound* of each digital effect.
  - Implementation of these digital audio effects through filters, delays and modulation of their control parameters, *e.g.* delay period, frequency.
  - 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 Synthesis: Sample-based, Additive Wavetable, Granular and Physical Modelling Synthesis

Zip file for all Digital Audio Synthesis examples is available here.

1. **Sample-based Synthesis**: Write a function that takes as input a short sampled piece of audio, a sample loop start and end point, and number of loop repeats and outputs the suitably looped audio.

   Refine this looping function by searching for *zero-crossings* in the audio in a small window around the start and end loop points — this should reduce any audible clicks in the loop.

2. **Wavetable Synthesis** Modify the lecture example code *wavetable_synth.m* to implement **sequential enveloping** on the waveform.

3. **Granular Synthesis** Modify the lecture example code *granulation.m* to implement ***pitch synchronous granular synthesis*** by instead of just random sampling from any point in the grain sample sampling is constrained to within a small random offset from the current sample position and grains overlap and the overlapping audio *smoothed* via some envelope.
   (Hint: Refer back to the phase vocoder (Lab 2) example). All Granular Synthesis zip files *here*.

4. **Physical Modelling Synthesis**: Experiment with the Karplus-Strong Algorithm discussed in the lectures (*karplus.m*) in the following ways:

   - Produce a sequence of notes with different pitch and duration.
   - Change the input noise source so some sample audio data or even some generated wave form.

   All Karplus Synthesis zip files *here*.

MATLAB Digital Audio Effects: Equalisers and Time-Varying/Delay Based Effects, Further Modulation and Reverb

Zip file for all Digital Audio Effects is available *here*.

1. **Equalisers**: Using the shelving filter code discussed in lectures (*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 code discussed in lectures (*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 code discussed in lectures (*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 code discussed in lectures (*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 code discussed in lectures (*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 (*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 (*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 (*ring_mod.m*) as well as the amplitude modulation tremolo example (*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 (*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 *schroeder2.m* and *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 *moorer.m* and *reverb_moorer_eg.m* as examples. Compare the output with the 6 comb filter example given in the lectures.

13. **Convolution Reverb**: Using the *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
      Try and find some odd impulse responses that do not model reverb and try these.
    - Try convolution reverb using any short piece of audio as the impulse response.

    All reverb example code is available *here*.

Revision Questions

The following questions are examples of the sort of questions you will be asked in the exam — in fact practically all these questions are taken from past exam papers. For full exam paper questions and solutions please see the CM3106 web page: www.cs.cf.ac.uk/Dave/Multimedia/EXAMS/

1. **Exam Paper 2014, Q2** Solution:
   www.cs.cf.ac.uk/Dave/Multimedia/EXAMS/Multimedia_BSC_Exam_2013SOLNS.pdf

   (a) What is the *difference* between *reverb* and *echo*?

   (b) Describe *two filter based* approaches to simulating the *reverb effect* in digital audio, explaining how one approach builds on the other and how filters are used to achieve the desired effect.

   (c) Describe, briefly how *Convolution Reverb* is implemented. What the is fundamental theorem that underpins this approach?

   (d) A new audio application requires that reverb be simulated as would be heard at a precise location within an acoustic space. This location must be allowed to vary and will be user-defined. Describe how this may efficiently be implemented via *Convolution Reverb*. Clearly state what challenges this approach presents for standard convolution reverb approaches and outline how your solution addresses such problems.

   (e) An audio engineer has been presented with a rare audio recording at an established concert hall. The concert hall is still in existence but the performer is not. The audio engineer has been tasked to create a new piece of music incorporating this recording with a new studio recorded backing. The *problem* is that the reverberation of the concert hall recording *does not match* the reverb of the new recording. How may the audio engineer achieve a better match to the reverb audio characteristics of the two recordings?

2. **Exam Paper 2009 Q3** Solution:
www.cs.cf.ac.uk/Dave/Multimedia/EXAMS/MM_BSC_SOLNS_2009.pdf

   (a) State *one alternative* approach to *reverb simulation* that does not employ filters.

   Briefly, giving **no** mathematical detail, describe how this approach is implemented.

   (b) For each of the *three reverb* methods discussed in the lectures how, in the following two scenarios, the sounds recorded by the microphone could be *modelled*:

      i. A long hallway where the long walls are lined with a high frequency absorbing acoustic panels. The sound source is placed at one end of the hallway and a microphone is placed at the other end.

      ii. A *cardoid microphone* is a microphone that accepts sound from the front and sides but not the back of the microphone. In a square recording studio, with uniform surfaces, a cardoid microphone is placed directly facing a sound source a few feet away.

3. **Exam Paper 2008 Q4** Solution:
www.cs.cf.ac.uk/Dave/Multimedia/EXAMS/MM_BSC_SOLNS_2008.pdf

   (a) Give a definition of a *one-dimensional Fourier transform*. [2]

   (b) Explain in detail how data is represented after the Fourier transform has been applied to a signal. [2]

   (c) Outline the basic approach to performing data *filtering* with the Fourier
   transform. [4]

   (d) Describe **one** application of Fourier transform filtering methods in multimedia data compression. [8]

   (e) An *exciter* is a digital audio signal process that emphasises or de-emphasises certain frequencies in a signal in order to change its *timbre*. Describe how you could use the Fourier transform to implement such a process, giving a *practical example* and explaining how it works. [11]