1 Lecture Outline

Reading: Chapter 8 Signal Processing Applications

- tap.c (4.2)
- Digital Audio Effects (Section 8.2)
- Delays and echoes (Section 8.2.1)

2 tap.c

We will find it convenient to have a routine which can read out or "tap" the i-th element in a circular buffer.

tap.c

```
1 #include "util.h"
2
3 float tap(int M, float *w, float *p, int i)
4 {
5     return w[(p - w + i) % (M + 1)];
6 }
```

This functions operates as follows. Assume M is the order of the array, i.e. length of array is $M+1$, *w is a pointer to the base address of the array, and *p is a pointer into the array. This routine reads out the value stored in memory i elements beyond the element pointed to by *p taking care to circularly wrap the array index. Since *p is not changed by this routine is passed by value.

3 Digital Audio Effects

Audio effects, such as delay, echo, reverberation, flanging, chorusing, and equalization, are indispensable in music production and performance. Some are also available for home and car audio systems. Most of these effects are implemented using a digital signal processor (DSP). A typical audio effects signal processor is shown in Fig. 8.2.1. The processor takes in the "dry" analog input, produced by an instrument such as a keyboard or previously recorded on some medium, and samples it at an appropriate audio rate, such as 44.1 kHz (or less, depending on the effect). The sampled audio signal is then subjected to a DSP effects algorithm and the resulting processed signal is reconstructed into analog form and sent on to the next unit in the audio chain, such as a speaker system, a recording channel, a mixer, or another effects processor.

Figure 1: Orfanidis p. 350 Figure 8.2.1. audio effects signal processor.

In all-digital recording systems, the sampling/reconstruction parts can be eliminated and the original audio input can remain in digitized form throughout the successive processing stages that subject it to various DSP effects or mix it with similarly processed inputs from other recording tracks.

4 Delays, Echoes, and Comb Filters (8.2.1)

In a listening space such as a room or concert hall, the sound waves arriving at our ears consist of the direct sound from the sound source as well as the waves reflected off the walls and objects in the room, arriving with various amounts of time delay and attenuation.

5 Echo (8.2.1)

A single reflection or echo of a signal can be implemented by the following filter

$$y(n) = x(n) + ax(n-D) \tag{1}$$

where D represents the travel time (in sample periods) from the source to a reflecting wall and the coefficient a is a measure of the reflection and propagation losses ($|a| < 1$). The impulse response and transfer function are given by

$$\begin{aligned} h(n) &= \delta(n) + a\delta(n-D) \\ &\updownarrow \\ H(z) &= 1 + az^{-D}. \end{aligned} \tag{2}$$

Figure 2: Orfanidis p. 351 Figure 8.2.2 digital echo processor.

The filter has zeros located at $z_k = \rho e^{j(2k+1)\pi/D}$, $k = 0, 1, \ldots, D-1$ where $\rho = \sqrt[D]{a}$. The magnitude response and pole/zero pattern of the filter are given in Figure 8.2.3.

Figure 3: Orfanidis p. 351 Figure 8.2.3 FIR comb filter.

Given a circular buffer which contains past input samples, we can tap the buffer to retrieve $x(n-D)$ using tap.c in order to implement the I/O difference equation.

5.1 C Implementation of Echo

In this example, we will read an input text file containing interleaved right/left samples from a stereo audio signal. The right-channel samples will be echoed while the left-channel samples will be passed unprocessed. The interleaved right/left output samples will be written to an output text file. The file I/O can be replaced with code which reads and writes samples to an audio device such as a soundcard.

main.c There is no change to main.c for this program.

user_data.h

```
#ifndef USER_DATA
#define USER_DATA

void initialize_program();
void process_signal(float inputRight, float inputLeft, float *outputRight, float *outputLeft);

#define MAX_DELAY 8001          // tapped delay line length, max is 32767
#define DELAY 8000              // desired delay (in samples)

// extern variables are seen by *all* functions in *all* source files, i.e. global
// initialization of extern variables in initialize_program.c
extern float buffer[];          // buffer to store samples
extern float *oldestSamplePtr;  // pointer to oldest sample in buffer
extern int delay;              // echo delay
extern float a;                 // echo gain

#include "util.h"

#endif
```

initialize_program.c

```
#include "user_data.h"

/*************************************/
/* Global variable initializations here */
/*************************************/
float buffer[MAX_DELAY]={0};    // buffer to store samples initialized to zero
float *oldestSamplePtr=buffer;  // pointer to oldest sample in buffer
int delay=DELAY;                // echo delay
float a=0.5;                    // echo gain

void initialize_program()
{

}
```

process_signal.c

```
#include "user_data.h"

void process_signal(float inputRight, float inputLeft, float *outputRight, float *outputLeft)
{
    float delay_sample;

    /*******************************/
    /* Process right channel sample */
    /*******************************/
    *oldestSamplePtr = inputRight;  // insert newest sample into buffer
    delay_sample = tap((MAX_DELAY-1),buffer,oldestSamplePtr,delay); // get delay sample
    *outputRight = inputRight + a*delay_sample; // x(n)+ax(n-D) mix of new sample and delay sample
    cdelay((MAX_DELAY-1), buffer, &oldestSamplePtr);    // back up pointer

    /*******************************/
    /* Process left channel sample */
    /*******************************/
    *outputLeft = inputLeft;
}
```

util.h There is no change to util.h for this program.

6 Three Echoes

We can add three successive echoes using the filter

$$y(n) = x(n) + ax(n-D) + a^2x(n-2D) + a^3x(n-3D). \tag{3}$$

The impulse response and transfer function are given by

$$\begin{aligned} h(n) &= \delta(n) + a\delta(n-D) + a^2\delta(n-2D) + a^3\delta(n-3D) \\ &\updownarrow \\ H(z) &= 1 + az^{-D} + a^2z^{-2D} + a^3z^{-3D} \\ &= \frac{1-a^4z^{-4D}}{1-az^{-D}}. \end{aligned} \tag{4}$$

The magnitude response and pole/zero pattern of the filter are given in Figure 8.2.5

Figure 4: Orfanidis p. 353 Figure 8.2.5 FIR comb filter.

Given a circular buffer which contains past input samples, we can tap the buffer at three locations to retrieve $x(n-D)$, $x(n-2D)$, and $x(n-3D)$ using tap.c in order to implement the I/O difference equation.

6.1 C Implementation of Three Echoes

In this example, we will read an input text file containing interleaved right/left samples from a stereo audio signal. The right-channel samples will be echoed three times while the left-channel samples will be passed unprocessed. The interleaved right/left output samples will be written to an output text file. The file I/O can be replaced with code which reads and writes samples to an audio device such as a soundcard.

main.c There is no change to main.c for this program.

user_data.h

```
1  #ifndef USER_DATA
2  #define USER_DATA
3
4  void initialize_program();
5  void process_signal(float inputRight, float inputLeft, float *outputRight, float *outputLeft);
6
7  #define MAX_DELAY 24001          // tapped delay line length, max is 32767
8  #define DELAY 8000               // desired delay (in samples)
9
10 // extern variables are seen by *all* functions in *all* source files, i.e. global
11 // initialization of extern variables in initialize_program.c
12 extern float buffer[];           // buffer to store samples
13 extern float *oldestSamplePtr;   // pointer to oldest sample in buffer
14 extern int delay;                // echo delay
15 extern float a;                  // echo gain
16
17 #include "util.h"
18
19 #endif
```

initialize_program.c

```
1  #include "user_data.h"
2
3  /*************************************/
4  /* Global variable initializations here */
5  /*************************************/
6  float buffer[MAX_DELAY]={0};     // buffer to store samples initialized to zero
7  float *oldestSamplePtr=buffer;   // pointer to oldest sample in buffer
8  int delay=DELAY;                 // echo delay
9  float a=0.5;                     // echo gain
10
11 void initialize_program()
12 {
13
14 }
```

process_signal.c This program will be written in Homework #9.

util.h You may wish to add plain.c to util.h for this program.