ECEN 444: Matlab Assignment 4

Due April 30, 2009

1 Overview

In this assignment, we will explore topics related to the development of digital music synthesizer. This will include the design of digital filters (using the window method), the use of linear interpolation for waveform generation, the use of amplitude envelopes for realistic synthesis, and the sequencing of multiple notes.

When working on the project, please follow the instructions and respond to each item listed. Your project grade is based on: (1) your Matlab script, (2) your report (plots, explanations, etc. as required), and (3) your final results. The project report should include all your scripts and the requested plots. It is often easier to combine these using Microsoft Word or Powerpoint. For example, you can copy/paste figures from MATLAB into these applications. You must clearly display the associated problem number and label the axes and on your plots to get full credit. Submission can be done electronically in PDF format or on paper.

2 Exercises

2.1 A Flexible Signal Generator

In many DSP applications, it is necessary to generate a variety of periodic waveforms with a programmable fundamental frequency. This is usually accomplished by using table look-up and interpolation operations from one period of the basic waveform. Let $x(n)$ be a discrete-time periodic signal with period N so that $x(0), x(1), \ldots, x(N-1)$ defines a single length-N period of the basic waveform.

The goal is to design a digital signal generator whose output is passed through a D/A converter operating at F_s samples per second. Suppose we want the D/A output to have frequency F Hz. Then, one approach is to generate a discrete-time output waveform

$$y(n) = x\left(\lfloor rn \rceil\right)$$

where $\lfloor t \rceil$ denotes the rounding operation which gives the integer closest to t. The discrete-time period of the new signal is $N' = \frac{1}{r}N$ so output frequency is $F = F_s/N' = rF_s/N$ Hz. For the following steps, we will assume that $F_s = 22050$, $N = 20$, and use the waveform

$$x(n) = \sum_{k=1}^{9} \left(1 - \frac{k}{10}\right) \cos\left(\frac{2\pi kn}{N} + 10k^2\right)$$

(a) In Matlab, let `x` be a row vector containing the first $N+1$ samples of the waveform. Then, we can generate one second of `y` using the straightforward `y=x(mod(round(r*m),N)+1)` or the more general command `y=interp1(0:N,x,mod(r*m,N),'nearest')` with `m=0:22050`. Using $F = 440$ Hz, plot 5 periods of the waveform and then play the sound (with `soundsc(y,Fs)`).

(b) Now, repeat part (a) using linear interpolation `y=interp1(0:N,x,mod(r*m,N),'linear')` instead of simply rounding time index. Using $F = 440$ Hz, plot 5-10 periods of the waveform and then play the sound (with `soundsc(y,Fs)`). Why does it sound different?

(c) The optimal interpolation in DSP, in contrast to linear interpolation, is bandlimited interpolation. It turns out that linear interpolation is very close to bandlimited interpolation if the signal has very little energy at high frequencies (e.g., normalized frequency $f > \frac{1}{4}$ cycles/sample). To improve the

sound quality, we will now upsample the waveform by a factor of 4. First, design an ideal discrete-time lowpass FIR filter $h(n)$ whose normalized cutoff frequency is $\frac{1}{8}$ cycles per sample. This gives

$$h(n) = \frac{1}{2\pi}\int_{-\frac{2\pi}{8}}^{\frac{2\pi}{8}} e^{j\omega n} d\omega = \frac{\sin\left(\frac{2\pi n}{8}\right)}{\pi n}.$$

Use `freqz(h,1,0:0.001:0.4*pi)` to plot the frequency response of this filter `h` for `n=-40:40` (i.e., truncated to length 81).

(d) To improve the sidelobes of the filter, we can use a window function $w(n)$ to define a new filter $\widetilde{h}(n) = w(n)h(n)$. The gain is not entirely without cost, because it also has the effect of increasing the width of transition band. For example, try the Hann window centered at $n = 0$ with exactly 81 non-zero coefficients given by

$$w(n) = \begin{cases} \frac{1}{2} + \frac{1}{2}\cos\left(\frac{2\pi n}{82}\right) & \text{if } |n| \leq 40 \\ 0 & \text{otherwise} \end{cases}.$$

Use `freqz(h,1,0:0.001:0.4*pi)` to plot the frequency response of the filter $\widetilde{h}(n)$.

(e) To see the effect of bandlimited interpolation, let `z` be four periods of $x(n)$ upsampled by a factor of 4 (i.e., add 3 zeros between each sample). Use `fz=filter(h.*w,1,z)` to filter this vector with the windowed filter $\widetilde{h}(n)$. Extract one period of the output signal and plot it against the original signal with `plot(0:80,4*fz(121:201),'b-',0:4:80,x,'ro')`. The original signal (i.e. red circles) should line up perfectly with interpolated signal. Why do we have to multiply `fz` by 4? Why does `121:201` pick out the correct portion of signal?

(f) Now, repeat part (a) using linear interpolation `y=interp1(0:N,fz(121:201),mod(r*m,N),'linear')` with $N = 80$ and $F = 440$ Hz. Plot 5-10 periods of the waveform and play the sound (with `soundsc(y,Fs)`). Does it sound different than the sound from (b)? Why?

2.2 Sequencing Notes

To generate music using DSP, one needs to sequence a set of notes. In this section you will write a short program which uses the signal generator to play "Mary had a little lamb".

(a) To play music, we first need to generate the frequencies in our scale. The twelve tone scale starting from 440 Hz (known as A above middle C) is given by `twelve=440*2.^((0:12)/12)` and the frequencies of the A major scale is the subset given by `major=twelve([1 3 5 6 8 10 11 12])`. List these 8 frequencies.

(b) For the notes of "Mary had a little lamb", the index (relative to the major scale) is given by `mllnotes=[3 2 1 2 3 3 3 2 2 2 3 5 5 3 2 1 2 3 3 3 2 2 3 2 1]`. This means that `mllfreq=major(mll)` gives the vector of note frequencies. Write a for-loop which uses the method in 2.1f to generate an 0.5 second tone for each note and then concatenates the tones to make the vector `y`. Play the song with `soundsc(y,Fs)`.

(c) In reality, some notes should be twice as long than others. Create a vector `len` of durations for each note and modify your program to handle notes of different durations. Play the new song with `soundsc(y,Fs)`.

(d) Now, we will write the song to a .wav file. First, normalize the vector to make the maximum absolute value one with `y=y/max(abs(y))`. Then, use `wavwrite(y,Fs,16,'yourname.wav')`. Play the song outside of Matlab to make sure this works.

Extra Credit

Add something new and interesting to your synthesizer. Generate a new .wav file using your upgraded digital synthesizer. Here are a few ideas.

ADSR

A realistic synthesizer also shapes the amplitude envelope of each note. The standard shaping is based on the physics of most instruments and is known as attack, decay, sustain, release (ADSR). Consider, for example, a piano where the hammer striking the strings gives the attack, the decay is caused by losses induced by large amplitude oscillations, the sustain is due to resonance between the strings and the sounding board, and the release is caused by the felt returning to mute the strings when the key is released. For a nice picture, see

```
http://en.wikipedia.org/wiki/ADSR_envelope
```

Pick an ADSR envelope (i.e., window function) for each note to get a more realistic sounding song. For the attack try a linear ramp from 0 to 1 over 0.05 seconds. For the release, try a linear ramp from 1 to 0.8 over 0.1 seconds (an exponential decay over a longer time is more realistic). For the sustain, try a constant level of 0.8 for the part of the note that is not attack, decay, or release. For the decay, try a linear ramp from 0.8 to 0 over 0.05 seconds.

A New Sound

Pick a new periodic waveform for your synthesizer. The cooler it sounds the better. For example, use a microphone to sample your voice or any other periodic waveform and cut out one period. If you're not careful, you'll introduce a discontinuity and there will be a buzzing of clicking sound. This can be avoided by resampling (e.g., upsampling followed by linear interpolation) the signal so that your prototype waveform has a period which is exactly integer.

Multi-track

Let's face it, you're not going to impress too many people with "Mary had a little lamb". So pick a new song. If you're musically inclined, try another song with two notes playing simultaneously (e.g., a repetitive bass line with a melody above it).