ECEN 314: Matlab Project 1
Fourier Series Synthesizer

Due April 8th, 2013

1 Overview

In this project, you will develop a simple digital music synthesizer in Matlab. The continuous-time Fourier series will be used to generate the waveforms and basic Matlab will be necessary to sequence multiple notes. For more realistic sounding synthesis, one can extend this by introducing amplitude envelopes.

Students may work by themselves or in pairs. When working on the project, please follow the instructions and respond to each item listed. Your project grade is based on: (1) your Matlab scripts, (2) your report (plots, wave files, explanations, etc. as required), and (3) your final results. The project report should include all your scripts and all 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.

The following should be e-mailed (in a single e-mail) to the instructor and TA: the report in PDF format, the wave file of generated by your synthesizer, and a zipfile containing all your Matlab code. The subject line of the e-mail should be "314Project1-lastname1-lastname2" and the files should be named: "lastname1_lastname2_report.pdf", "lastname1_lastname2.wav", and "lastname1_lastname2.zip".

2 Exercises

2.1 A Signal Generator Based on CT Fourier Series

In many signal processing applications, it is necessary to generate a variety of periodic waveforms with a programmable fundamental frequency. One way to accomplish this is by storing the Fourier series coefficients of the desired waveforms. Let $\{a_k\}$ be the CT Fourier series coefficients of the desired CT waveform. Suppose we would like to produce a DT signal $x[n]$, sampled at $f_s = 8000$ Hz, whose fundamental frequency, f_0, ranges from 100 Hz to 1000 Hz. One solution would be to compute

$$x[n] = \sum_{k=-K}^{K} a_k e^{jk(2\pi f_0/f_s)n}, \tag{1}$$

for some reasonable large value of K.

Let a_k be the Fourier series coefficients of a periodic square wave with $T = 1$ defined, for $|t| \leq \frac{1}{2}$, by

$$x(t) = \begin{cases} 1 & \text{if } |t| \leq \frac{1}{4} \\ -1 & \text{otherwise.} \end{cases}$$

Let b_k be the Fourier series coefficients of a periodic triangle wave with $T = 1$ defined, for $|t| \leq \frac{1}{2}$, by

$$x(t) = \begin{cases} 1 - 4t & \text{if } t \geq 0 \\ 1 + 4t & \text{otherwise.} \end{cases}$$

Repeat the following steps using both a periodic square wave and a periodic triangle wave.

(a) Now, we will choose $K = 19$, $f_s = 8000$, $f_0 = 210$, and compute $x[n]$ for $n = 0, \ldots, 2f_s$ (i.e., 2 seconds of signal). In Matlab, we choose `K=19`, `fs=8000`, `f0=210`, `n=0:(2*fs)`. Then, we compute the signal vector `x` using for-loop over `k`. The result can be played as a sound using `soundsc(real(x),fs)`. Describe how this waveform sounds.

(b) Now, repeat part (a) using $f_0 = 420$ Hz. Describe how this waveform sounds. What is happening?

(c) Next, find a value of K where the synthesizer sounds good for $f_0 = 420$ Hz. How should one choose K as a function of f_0? Why?

2.2 Sequencing Notes

To generate music using Matlab, 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 12 13])`. 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 the vector of note frequencies is given by `mllfreq=major(mll)`. Write a for-loop which uses the previous exercise 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(real(y),Fs)`.

(c) In reality, some notes should be twice as long than others. Create a vector `dur` of durations for each note and modify your program to handle notes of different durations. For example, write a Matlab function `y=synth(freq,dur)` that concatenates `length(freq)` tones where the i-th tone has frequency `freq(i)` and duration `dur(i)`.

(d) Play the new song with `soundsc(y,Fs)`.

(e) 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(real(y),Fs,16,'yourname.wav')`. Play the song outside of Matlab to make sure this works.

(f) There should be a main Matlab script named `testsynth.m` that generates and plays the synthesized waveform.

Extra Credit

Add something new and interesting to your synthesizer. Generate a new .wav file using your improved digital synthesizer. A few ideas are listed below. Honors students are required to choose one of the below or implement their own idea.

A New Sound

Create your favorite synthesizer sound by choosing new Fourier series coefficients. The cooler it sounds the better. For example, you could use a microphone to sample any periodic waveform (e.g., your voice) and perform Fourier analysis on one period. To make this more robust, you can cut an integer number of periods (e.g., 10) and multiply by a window function before performing analysis. Let `x` be M periods of the signal you want to analyze and assume `x` has length N. Then, `xw=x.*hamming(length(x))` will give you a windowed version of the signal. Then, you can estimate the Fourier series coefficients of one period with

$$a_k = \sum_{n=1}^{N} x[n] e^{-j2\pi kM/N}.$$

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.

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).