MUMT605 Assignment 1

Johnty Wang

13 Oct 2014

Part 1

Overview

For this part, I implemented a matlab function called `A1_func` in Matlab, with its help/description as follows:

```
% A1_func generates a rectangular wave with specified parameters
% W = A1_func(f, d_cycle, ph, time, sample_rate, doPlot)
%   - f frequency in hertz
%   - d_cycle duty cycle of wave (0.0 - 1.0)
%   - ph phase offset (as portion of period, 0.0 - 1.0)
%   - time duration of output wave in seconds
%   - sample_rate output sample rate
%   - doPlot 1=plot spectrum, wave, etc 0 = no plot
%   - W the output samples
```

The internal code comments provide explanation of the process, but the overall process is as follows:

- Compute discrete spectrum
- Generate frequency independent wavetable with correct duty cycle from spectrum, that is otherwise independent of sample rate
- Produce frequency correct wavetable given sample rate, target frequency and fill up entire buffer corresponding to required duration of the synthesized output

After I obtained the initial frequency-independent wavetable, I made two attempts at interpolating towards the final target frequency:

1. *The naïve approach*: for the first attempt, I thought that by simply adjusting the duration of the final waveform, I could get different frequencies, like follows:

   The frequency of a given wavetable, at a certain sample rate is:

   $$f_0 = \frac{F_s}{N} \tag{1}$$

   To obtain a desired frequency, we can simply go:

   $$f_{\text{target}} = \frac{F_s}{N_{\text{target}}} \tag{2}$$

   Which means the desired table size of the new wavetable should be:

   $$N_{\text{target}} = \frac{F_s}{f_{\text{target}}} \tag{3}$$

   However, after I did this, it was clear it had a serious drawback. Since we're working in the discrete time domain, N_{target} must be integer values. This means that the possible output frequencies are limited to countable numbers of the form in Eq. 2. In terms of achievable frequencies, we end up with the following examples:

   For F_s of 44100 Hz:

N_{target}	F
100	441
99	445.45
98	450
97	454.64
50	882
49	900
48	918.75

   The frequency limitation is clearly inadequate for any kind of musical synthesis!

2. *Corrected approach*: The issue in the above approach is due to the limitation of interpolating by modifying the period of the wavetable. I ended up going with the phase increment approach, which adjusts the rate at which the wavetable is read according to the desired frequency. This rate can be adjusted using a continuous variable, and the interpolation is done not by resizing the original table, but instead by altering the rate that the table is scanned. For a thorough implementation more advanced techniques for interpolation should be used, but for my first implementation I simply *rounded* to the nearest index.

Putting it together

Below is a file listing of the submitted assignment:

- `A1_func`: the main synthesizer code that generates the waveform.
- `runme.m`: the tester application that does the following:
    1. Runs `A1_func` once with plotting enabled, to see what the output waveform looks like
    2. Calls additional helper functions that generates and plays back a small “song".
- `loadscore.m`: a “song generator" that takes in an input tempo, sample rate, and outputs a hard-coded score made up of frequencies and durations.
- `note2freq.m`: a quick formula for converting between MIDI notes and frequency.

Part 2

1. For a given function

$$y(t) = \begin{cases} x(t) & t \geq 0 \\ 0 & \text{elsewhere} \end{cases}$$

Then, based on the definition of $|\ |$:

$$y(|t|) = \begin{cases} x(t) & t \geq 0 \\ x(-t) & t < 0 \end{cases}$$

Therefore, for this particular example, we have:

$$x(t) = s(t) + s(-t)$$

from the property of the fourier transform:

$$\text{if } a(t) \rightarrow A(f)$$
$$\text{then } a(-t) \rightarrow A(-f)$$

$$\begin{aligned} \text{so } x(t) &\rightarrow X(f) \\ \implies X(f) &= S(f) + S(-f) \\ \implies X(f) &= \frac{1}{\alpha - 2\pi jf} + \frac{1}{\alpha + 2\pi jf} \\ &= \frac{\alpha + 2\pi jf + \alpha - 2\pi jf}{(\alpha - 2\pi jf)(\alpha + 2\pi jf)} \\ &= \frac{2\alpha}{\alpha^2 + 4\pi f^2} \end{aligned}$$

2. When we sample the spectrum $X(f)$ of the time series signal $x(t)$ using a dirac comb, it creates a "periodized" version of the signal, which can be expressed as:

$$X'(f) = X(f)\text{Ш}_{T_0}(f)$$

where

$$\text{Ш}_{T_0}(f) = \sum_{k=-\infty}^{\infty} \delta(f - kT_0)$$

$$\implies X'(f) = \frac{2\alpha}{\alpha^2 + 4\pi f^2} \sum_{k=-\infty}^{\infty} \delta(f - kT_0)$$

3. From the above, we can see that $X'(f)$ is nonzero where $f = kT_0$, which means the amplitude of the kth harmonic can be expressed as:

$$\frac{2\alpha}{\alpha^2 + 4\pi k^2 {T_0}^2}$$

4. One advantage of this method is that the calculation is very simple - by exploiting the additive and time reversal properties of the fourier transform, we did not need to evaluate the integral for the |t| case.