To: Prof. Ross Snider

From: Jeff Meirhofer

Regarding: Lab #3 – Synthesis of Sinusoidal Signals

Date: February 5, 2009

**Summary:**

The purpose of this lab is to experiment with the synthesis of sinusoidal signals. The signals will be used to construct musical notes. These notes will then be used to represent an entire song. Some advanced techniques are then implemented in order to improve the sound quality of the song.

**Musical Synthesis of Signals**

**3)** We can use the signal to generate a sound that we can hear. The frequency of the signal determines the pitch of the sound. We know that the musical note A­4 has a frequency of 440 Hz. We also know that the frequencies for all the notes on a typical piano keyboard are related by a constant rate. For example, the frequency of a key N steps above A­4 would be . Using this formula, we can calculate the frequency of every note we need to play. We will use this relation to construct a fairly complicated song.

**3 a)** The sampling frequency we chose was 11025 Hz, which should work well with the D/A converter of our Windows PC.

**3 b)** We decided we wanted a quarter note to last .7 seconds through trial and error.

**3 c)** We calculated all the frequencies for the notes we needed, and defined them at the start of our song’s m-file so they would be easier to work with.



**3 f)**

Figure 1. Plot of some sinusoids generated to represent musical notes.

**3.1 c)** We can use the “ginput” function to verify that the amplitude of each signal is 3. We also see that the frequency of each signal is 1250 Hz, which is correct. This is done by taking the reciprocal of the period, 0.8 ms, so that 1/0.8ms = 1250 Hz. Finally, we see that the phases are -74º and -164º. When converted to radians, we see that the real part has a phase shift of -0.4\*pi rad, and the imaginary part a phase shift of 0.9\*pi. This is calculated by taking the offset in seconds from ginput, and multiplying it by -2\*pi\*1250 rad/sec. The offsets in milliseconds were .16 and .35, respectively. Since the imaginary part is based on a sine function, we expect it to lag the real part (which is based on a cosine) by 90º. The measurements are consistent with this principle.

**3.2 a) and c)** Five different sinusoids were plotted and are shown below. They are:

Figure 2. Plot of four different sinusoids and their sum.

**3.2 b)** The four original signals all have an amplitude of 5 and each has the correct phase.

**3.2 d)** The magnitude and phase of x5 are measured from the plot using the “ginput” function and clicking on the peak nearest to time t=0.

**3.2 e)** z1 – z4 are now given in polar and Cartesian form.

= 5\*j

= 3.5 – 3.5\*j

= 1.5 – 4.8\*j

= -4.8 – 1.5\*j

**3.2 f)** The following plot shows that z5 = z1 + z2 + z3 + z4 in terms of vector addition.



Figure 3. Addition of the vectors z1-z4 to create z5

**3.2 g)** You can see from the plot that the magnitude and phase of z5 correlate to the values our script printed out, which are 4.69 @ 0.48π.

**4 a)** Now, the “sumcos” M-file is used to synthesize a waveform. It's fundamental frequency is 25 Hz and Zk is j/k for odd harmonics and 0 for even harmonics. Three plots are graphed which show N=5, 10, and 25. The period of the synthesized waveform is the same as the fundamental frequency. As N nears infinity, the wave converges to a square wave. One unusual thing about the waveform is that it has overshoot where it rises and falls. The graphs are shown on the next page.

Figure 4. Synthesized sine wave with varying values of N.

**4 b)** N/A

**4 c)** Now part (a) is repeated, except this time . Its result is shown below. As N goes to infinity, the shape approximates a ramp function. The period of the ramp is the same as the fundamental frequency.

Figure 5. Synthesized waveform from part 4 (c)

**Conclusion**

MATLAB is a powerful tool for working with complex numbers. It can do conversions and plot complex exponentials much easier than a human can do it by hand. We can use MATLAB for a variety of functions dealing with complex exponentials.

APPENDIX

**2.1**