DSP First

Laboratory Exercise #3

Synthesis of Sinusoidal Signals

This lab includes a project on music synthesis with sinusoids. One of several candidate songs can be selected when doing the synthesis program. The project requires an extensive programming effort and should be documented with a complete lab report. A good report should include the following items: a cover sheet, commented MATLAB code, explanations of your approach, conclusions and any additional tweaks that you implemented for the synthesis. Since the project must be evaluated by listening to the quality of the synthesized song, the criteria for judging a good song are given at the end of this lab description. In addition, it may be convenient to place the final song on a web site so that it can be accessed remotely by a lab instructor who can then evaluate its quality.

1 Overview

We have spent a lot of time learning about the properties of sinusoidal waveforms of the form

$$x(t) = A\cos(\omega_0 t + \phi) \tag{1}$$

In this lab we will synthesize waveforms composed of sums of sinusoidal signals of this form, sample them, and then reconstruct them for listening. We will use combinations of the basic sinusoid (1) to synthesize the following signals:

1. Sine waves at a specific frequency played through a D/A converter.
2. Sinusoids that create a synthesized version of *Für Elise*.
3. Any song can be used for the synthesis project. This lab write-up and the CD-ROM include information about four alternative tunes: *Jesu, Joy of Man's Desiring*, *Minuet in G*, *Beethoven's Fifth* and *Twinkle, Twinkle, Little Star*.

CD-ROM

MUSIC SYN-THESIS

The primary objective of the lab is to establish the connection between musical notes, their frequencies and sinusoids. A secondary objective is the challenge of trying to add other features to the synthesis in order to improve the subjective quality for listening. Students who take this challenge will be motivated to learn more about the spectral representation of signals—a topic that underlies this entire book.

2 Warm-up: Music Synthesis

The instructor verification sheet is included at the end of this lab.

In this lab, the sine waves and music signals will be created with the intention of playing them out through a speaker. Therefore, it is necessary to take into account the fact that a conversion is needed from the digital samples which are numbers stored in the computer memory to the actual voltage waveform that will be amplified for the speakers. The layout of a piano keyboard will also be explored, so that we have a formula that gives the frequency for each key.

2.1 D-to-A Conversion

The digital-to-analog conversion process has a number of aspects, but in its simplest form the only thing we need to worry about at this point is that the time spacing (T_s) between the signal samples must correspond to the rate of the D-to-A hardware that is being used. From MATLAB, the sound output is done by the `sound(x,fs)` function which does support variable sampling rate if the hardware on the machine has such capability. A convenient choice for the D-to-A conversion rate is 8000 samples per second, so $T_s = 1/8000$ seconds. Another common choice is 11,025 Hz which is one-quarter of the rate used for audio CDs. Both of these rates satisfy the requirement of sampling fast enough as explained in the next section. In fact, most piano notes have relatively low frequencies, so an even lower sampling rate could be used. In some cases, it will also be necessary to scale the vector `x` so that it lies between ± 1.[1]

2.2 Theory of Sampling

Even though Chapter 4 treats sampling in detail, we provide a quick summary of essential facts here. The idealized process of sampling a signal and the subsequent reconstruction of the signal from its samples is depicted in Fig. 1. This figure shows a continuous-time input signal $x(t)$, which

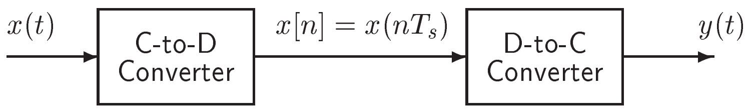

Figure 1: Sampling and reconstruction of a continuous-time signal.

is sampled by the continuous-to-discrete (C-to-D) converter to produce a sequence of samples $x[n] = x(nT_s)$, where n is the integer sample index and T_s is the sampling period. The sampling rate is $f_s = 1/T_s$. As described Chapter 4, the ideal discrete-to-continuous (D-to-C) converter takes the input samples and interpolates a smooth curve between them. The *Sampling Theorem* tells us that if the input is a sum of sine waves, then the output $y(t)$ will be equal to the input $x(t)$ if the sampling rate is more than twice the highest frequency $f_{\max}$ in the input, i.e., $f_s > 2f_{\max}$.

Most computers have a built-in analog-to-digital (A-to-D) converter and a digital-to-analog (D-to-A) converter (usually on the sound card). These hardware systems are physical realizations of the idealized concepts of C-to-D and D-to-C converters respectively, and for purposes of this lab we will assume that they are perfect realizations.

(a) The ideal C-to-D converter will be implemented in MATLAB by taking the formula for the continuous-time signal and evaluating it at the sample times, nT_s. This assumes perfect knowledge of the input signal, but we already did it this way in Lab 2.

To begin, compute a vector `x1` of samples of a sinusoidal signal with $A = 100$, $\omega_0 = 2\pi(1100)$, and $\phi = 0$. Use a sampling rate of 8000 samples/second, and compute a total number of samples equivalent to 2 seconds time duration. You may find it helpful to recall that the MATLAB statement `tt=(0:0.01:3);` would create a vector of numbers from 0 through 3 with increments of 0.01. Therefore, it is only necessary to determine the time increment needed to obtain 8000 samples in one second.[2]

[1]In MATLAB version 5, there is a function `soundsc(x,fs)` which performs that scaling.
[2]Another popular rate is 11,025 sample/sec, which is one fourth of the rate used in audio CD players.

Using `sound()` play the resulting vector through the D-to-A converter of your computer, assuming that the hardware can support the $f_s = 8000$ Hz rate (or $f_s = 11,025$ Hz). Listen to the output.

(b) Now compute a vector `x2` of samples (again, 2 seconds time duration) of the sinusoidal signal for the case $A = 100$, $\omega_0 = 2\pi(1650)$, and $\phi = \pi/3$. Listen to the signal reconstructed from these samples. How does it compare to the signal in part (a)? Put both signals together in a new vector defined with the following MATLAB statement (assuming that both `x1` and `x2` are row vectors):

```
xx = [x1 zeros(1,2000) x2];
```

Listen to this signal. Explain what you heard.

(c) Now send the vector `xx` to the D-to-A converter again, but double the sampling rate in `sound( )` to 16000 samples/second. Do not recompute the samples in `xx`, just tell the D-to-A converter that the sampling rate is 16000 samples/second. Describe what you heard. Observe how the *duration* and *pitch* of the signal changed. Explain.

Instructor Verification (separate page)

2.3 Piano Keyboard

Section 3 of this lab will consist of synthesizing the notes of a well known piece of music.[3] Since these signals require sinusoidal tones to represent piano notes, a quick introduction to the frequency layout of the piano keyboard is needed. On a piano, the keyboard is divided into octaves—the notes in each octave being twice the frequency of the notes in the next lower octave. For example, the

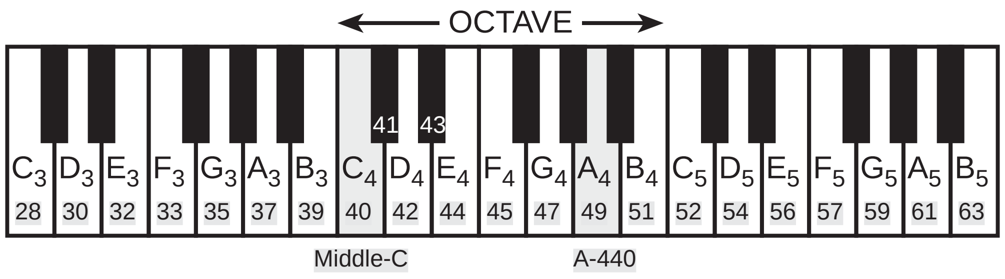

Figure 2: Layout of a piano keyboard. Key numbers are shaded. The notation C_4 means the C-key in the fourth octave.

reference note is the A above middle-C which is usually called A-440 (or A_5) because its frequency is 440 Hz. Each octave contains 12 notes (5 black keys and 7 white) and the ratio between the frequencies of the notes is constant between successive notes. Thus this ratio must be $2^{1/12}$. Since middle C is 9 keys below A-440, its frequency is approximately 261 Hz. Consult chapter 9 for more details.

Musical notation shows which notes are to be played and their relative timing (half notes last twice as long as quarter notes, which, in turn, last twice as long as eighth notes). Figure 3 shows how the keys on the piano correspond to notes drawn in musical notation.

[3]If you have little or no experience reading music, don't be intimidated. Only a little knowledge is needed to carry out this lab. On the other hand, the experience of working in an application area where you must quickly acquire knowledge is a valuable one. Many real-world engineering problems have this flavor, especially in signal processing which has such a broad applicability in diverse areas such as geophysics, medicine, radar, speech, etc.

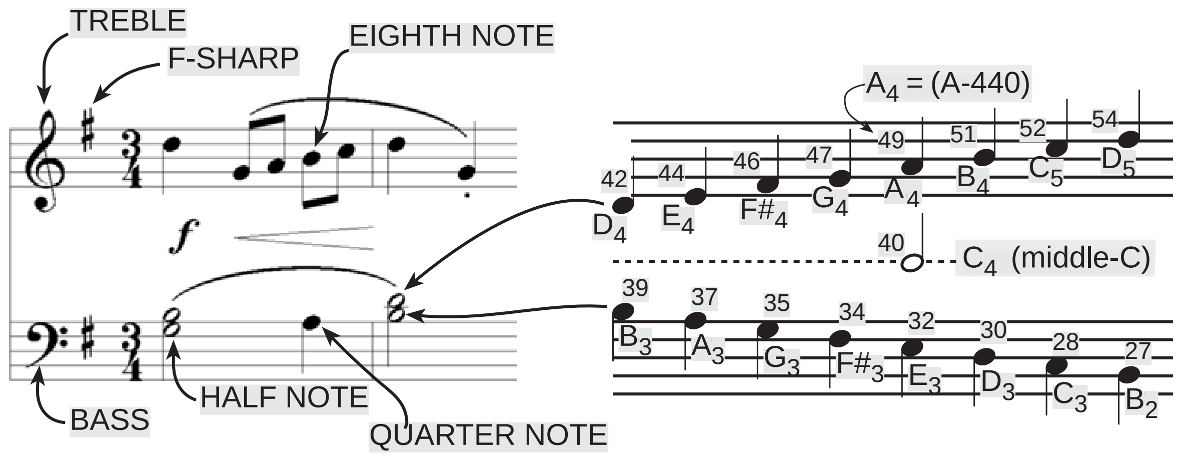

Figure 3: Musical notation is a time-frequency diagram where vertical position indicates the frequency of the note to be played.

Another interesting relationship is the ratio of fifths and fourths as used in a chord. Strictly speaking the fifth note should be 1.5 times the frequency of the base note. For middle-C the fifth is G, but the frequency of G is about 392 Hz which is not exactly 1.5 times 261.6. It is very close, but the slight detuning introduced by the ratio $2^{1/12}$ gives a better sound to the piano overall. This innovation in tuning is called "equally-tempered" and was introduced in Germany in the 1760's and made famous by J. S. Bach in the "Well Tempered Clavichord."

CD-ROM

Music GUI

You can use the ratio $2^{1/12}$ to calculate the frequency of notes anywhere on the piano keyboard. For example, the E-flat above middle-C (black key number 43) is 6 keys below A-440, so its frequency should be $f = 440 \times 2^{-6/12} = 440/\sqrt{2} \approx 311$ Hertz.

(a) Generate a sinusoid of 2 seconds duration to represent the note E_5 above A-440 (key number 56). Choose the appropriate values for T_s and `fs`. Remember that `fs` should be at least twice as high as the frequency of the sinusoid you are generating. Also, T_s and `fs` must "match" in order for the note played out of the D-to-A converter to sound correct.

(b) Now write an M-file to produce a desired note for a given duration. Your M-file should be in the form of a function called `note.m`. You may want to call the `sumcos` function that you wrote for Lab 2. Your function should have the following form:

```
function tone = note(keynum,dur)
% NOTE  Produce a sinusoidal waveform corresponding to a
%       given piano key number
%
%  usage:  tone = note (keynum, dur)
%
%        tone = the output sinusoidal waveform
%      keynum = the piano keyboard number of the desired note
%         dur = the duration (in seconds) of the output note
%
fs = 8000;     %-- use 11025 Hz on PC/Mac, 8000 on UNIX
tt = 0:(1/fs):dur;
```

```
freq =
tone =
```

For the `freq =` line use the formulas based on $2^{1/12}$ to determine the frequency for a sinusoid in terms of its key number. You should start from a reference note (middle-C or A-440 is recommended) and solve for the frequency based on this reference. For the `tone =` line generate the actual sinusoid at the proper frequency.

(c) The following is an incomplete M-file that will play scales:

```
%--- play_scale.m
%---
keys =   [ 40  42  44  45  47  49  51  52 ];
%--- NOTES: C   D   E   F   G   A   B   C
%  key #40 is middle-C
%
dur  = 0.25 * ones(1,length(keys));
fs   = 8000;           %-- use 11025 Hz on PC/Mac, 8000 on UNIX
xx   = zeros(1,sum(dur)*fs+1);
n1 = 1;
for kk = 1:length(keys)
   keynum = keys(kk);

   tone =                       %<=== FILL IN THIS LINE

   n2 = n1 + length(tone) - 1;
   xx(n1:n2) = xx(n1:n2) + tone;
   n1 = n2;
end
sound( xx, fs )
```

For the `tone =` line, generate the actual sinusoid for `keynum` by making a call to the function `note()` written previously. Note that the code in `play_scale.m` allocates a vector of zeros large enough to hold the entire scale then adds each note into its proper place in the vector `xx`.

Instructor Verification (separate page)

3 Lab: Synthesis of Musical Notes

The audible range of musical notes consists of well-defined frequencies assigned to each note in a musical score. Five different pieces are given below, but you only need to choose one for your synthesis program. Before starting the project, make sure that you have a working knowledge of the relationship between a musical score, key number and frequency. In the process of actually synthesizing the music, follow these steps:

(a) Determine a sampling frequency that will be used to play out the sound through the D-to-A system of the computer. This will dictate the time T_s between samples of the sinusoids.

(b) Determine the total time duration needed for each note.

(c) Determine the frequency (in hertz) for each note (utilize the `note.m` function written in the warm-up and Fig. 2.)

(d) Synthesize the waveform as a combination of sinusoids, and play it out through the computer's built-in speaker or headphones using `sound()`.

(e) A chord can be synthesized by adding the sinusoids for each note in the chord. This will be a vector addition of the sinusoidal values for each note. Likewise, if you have more than one melody line playing at the same time, you can produce separate signal vectors for each melody (treble and bass) and then combine them into one song by adding the signal vectors.

(f) Make a plot of a few periods of two or three of the sinusoids to illustrate that you have the correct signals for each note.

3.1 Spectrogram of the Music

Musical notation describes how a song is composed of different frequencies and when they should be played. This representation can be considered to be a *time-frequency* representation of the signal that synthesizes the song. In MATLAB we can can compute a time-frequency representation from the signal itself. This is called the spectrogram, and is implementation with the MATLAB function `specgram`. To aid your understanding of music and its connection to frequency content, a MATLAB GUI is available so that you can visualize the spectrogram along with musical notation. This GUI also has the capability to synthesize music from a list of notes, but these notes are given in "standard" musical notation, not key number. For more information, consult the `help` on `musicgui.m` which only runs in MATLAB version 5.

CD-ROM
MUSIC GUI

3.2 *Für Elise*

Für Elise is a well known piece of music written by Beethoven. You can listen to a recording of the part that you will synthesize by following the links on the *DSP First CD-ROM*, and the first few measures are shown in Fig. 4. More of the song can be found on the CD-ROM, where an entire page of the music for Für Elise is reproduced.

CD-ROM
Fur Elise

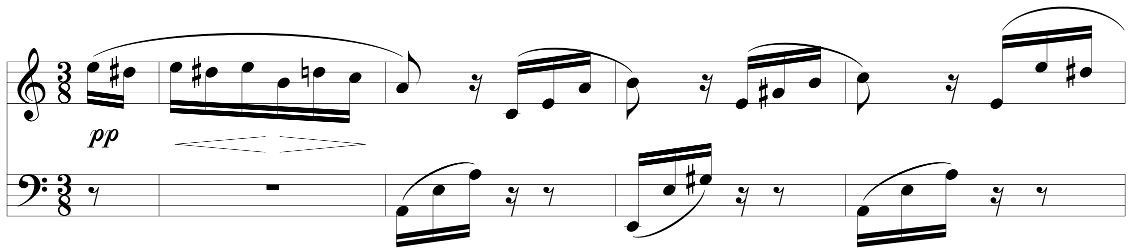

Figure 4: First few measures of Beethoven's Für Elise.

Determine the notes that are played in Für Elise, by mapping each note to a key number (Fig. 2) and then synthesize sine waves to recreate the piece. Use either the short form (Fig. 4) or the long form found on the CD. Use sine waves sampled at 8000 samples/sec (for UNIX) or 11025 samples/sec (for other platforms). From your recollection of this music, estimate the time duration needed for each note. If you define a fixed time duration for a quarter note, say T_q then all the other durations will be defined: an eighth note is $\frac{1}{2}T_q$, a sixteenth note is $\frac{1}{4}T_q$, a half note $2T_q$, and so on. After defining the time duration for all notes, you still may need to make adjustments in the timing to improve the subjective quality of the synthesized song. In addition, adding very short pauses between notes usually improves the music because it imitates the natural transition that a musician must make from one note to the next.

After you finish the project, assess the quality of your synthesized result. Suggest some other features that could be incorporated into your program if you had more time to work on it.

3.3 Musical Tweaks

The musical passage is likely to sound very artificial, because it is created from pure sinusoids. Therefore, you might want to try improving the quality of the sound by incorporating some modifications. For example, you could multiply each pure tone signal by an envelope $E(t)$ so that it would fade in and out.

$$x(t) = E(t)\cos(2\pi f_0 t + \phi) \tag{2}$$

If an envelope is used it should "fade in" quickly and fade out more slowly. An envelope such as a half-cycle of a sine wave $\sin(\pi t/\text{dur})$ is not good because it does not turn on quickly enough, so simultaneous notes of different durations no longer appear to begin at the same time. A standard way to define the envelop function is to divide $E(t)$ into four sections: attack (A), delay (D), sustain (S), and release (R). Together these are called ADSR. The attack is a quickly rising front edge, the delay is a small short-duration drop, the sustain is more or less constant and the release drops quickly back to zero. Figure 5 shows a linear approximation to the ADSR profile.

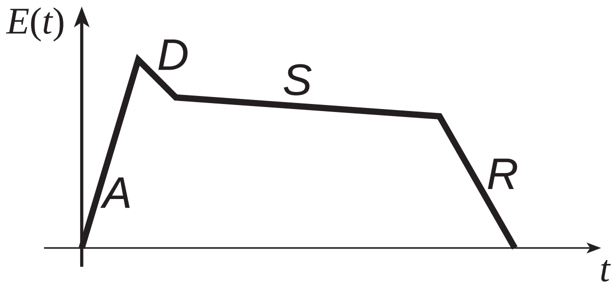

Figure 5: ADSR profile for an envelope function $E(t)$.

Some other issues that affect the quality of your synthesis include relative timing of the notes, correct durations for tempo, rests (pauses) in the appropriate places, relative amplitudes to emphasize certain notes and make others soft, and harmonics. True piano sounds contain several frequency components, such as second and third harmonics. Since we have been studying harmonics, this modification would be simple, but be careful to make the amplitudes of the harmonics smaller than the fundamental frequency component. You should experiment to see what sounds best.

3.4 Programming Tips

You may want to modify your note function to accept additional parameters describing amplitude, duration, etc. You will also want to change it to add an envelope and/or harmonics. Chords are created on a computer by simply adding the signal vectors of several notes.

For testing we have provided a MATLAB script which initializes vectors containing the note values and durations for *Für Elise*. This will save you from typing it all in yourself; but, you are free to modify the duration values or anything else. This script called `fenotes.m` just contains the melody, and it is available on the *DSP First CD-ROM*.

CD-ROM
fenotes.m

3.5 Alternate Song: *Jesu, Joy of Man's Desiring*

Follow the project description given in Section 3.2, but use the song, *Jesu, Joy of Man's Desiring* written by Bach. The first few measures are shown in Fig. 6, and you can listen to the part that you will synthesize by following the links on the *DSP First CD-ROM.* More of the song can be found on the CD-ROM, where an entire page of the music is reproduced.

CD-ROM
Jesu

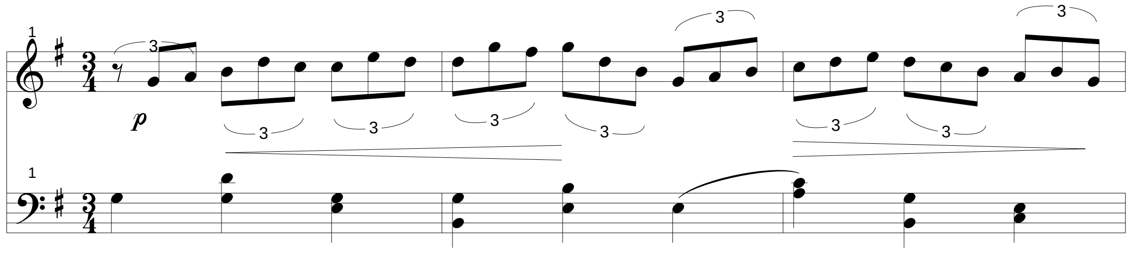

Figure 6: First few measures of Jesu, Joy of Man's Desiring.

3.6 Alternate Song: *Minuet in G*

Follow the project description given in Section 3.2, but use the song, *Minuet in G* written by Bach. The first few measures are shown in Fig. 7, and you can listen to the part that you will synthesize by following the links on the *DSP First CD-ROM.* More of the song can be found on the CD-ROM, where an entire page of the music is reproduced.

CD-ROM
Minuet in G

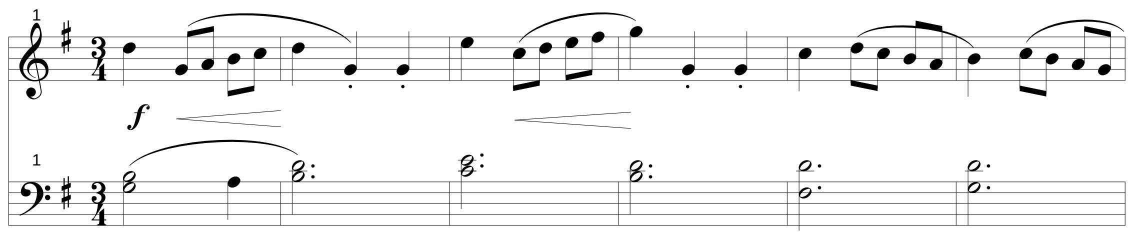

Figure 7: First few measures of the theme from Minuet in G.

3.7 Alternate Song: *Beethoven's Fifth*

Follow the project description given in Section 3.2, but use the theme from *Beethoven's Fifth.* The first few measures are shown in Fig. 8, and you can listen to the part that you will synthesize by following the links on the *DSP First CD-ROM.* More of the song can be found on the CD-ROM, where an entire page of the music is reproduced.

CD-ROM
Beethoven's Fifth

3.8 Alternate Song: *Twinkle, Twinkle, Little Star*

Follow the project description given in Section 3.2, but use the song *Twinkle, Twinkle, Little Star* written by Mozart. The first few measures are shown in Fig. 9, and you can listen to the part that you will synthesize by following the links on the *DSP First CD-ROM.* More of the song can be found on the CD-ROM, where an entire page of the music is reproduced.

CD-ROM
Twinkle, Twinkle, Little Star

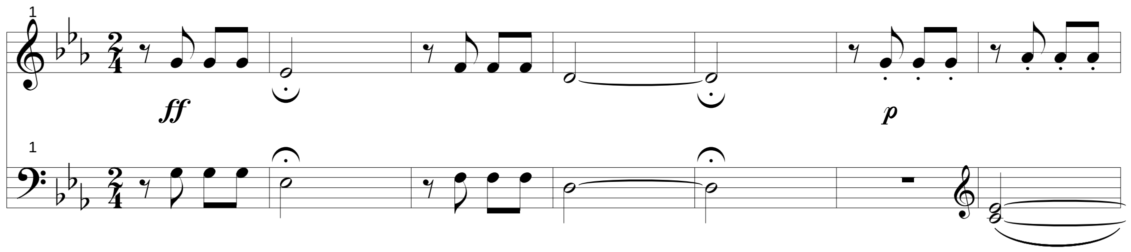

Figure 8: First few measures of the theme from Beethoven's Fifth.

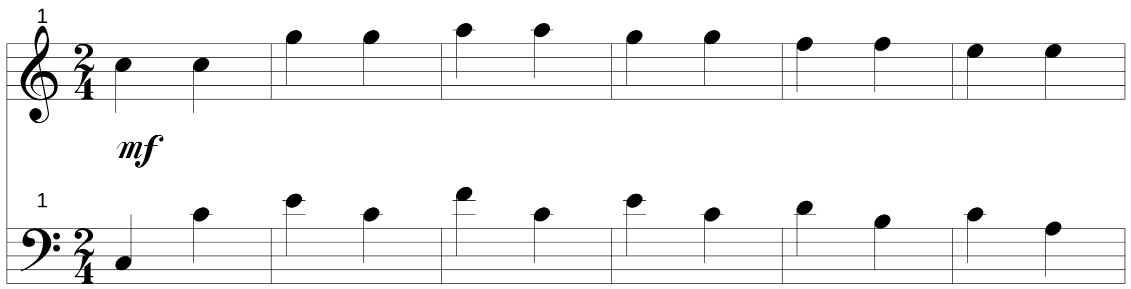

Figure 9: First few measures of the theme from Twinkle, Twinkle, Little Star.

Lab 3
Instructor Verification Sheet

Staple this page to the end of your Lab Report.

Name: ______________________________ Date: __________

Part 2.2 Explain sampling rate effects:

Verified: ______________________

Part 2.3 Complete and demonstrate `note.m` and `play_scale.m`:

Verified: ______________________

Sound Evaluation Criteria

Does the file play notes? All Notes _____ Most _____ Treble only _____

Overall Impression: ____________________

Excellent: Enjoyable sound, good use of extra features such as harmonics, envelopes, etc.

Good: Bass and Treble clefs synthesized and in sync, few errors, one or two special features.

OK: Basic sinusoidal synthesis, including the bass, with only a few errors.

Poor: No bass notes, or treble and bass not synchronized, many wrong notes.