GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

EE 2200 Winter 1999
Lab #4: Synthesis of Sinusoidal Signals

Date: week of 1 Feb 1999

This is *the official* Lab#4 description; it is similar to the one in Appendix C.3 of the text, but the piece *Jesu, Joy of Man's Desiring* has been chosen for the synthesis.

The Warm-up section of each lab must be completed in Lab and the steps marked *Instructor Verification* must also be signed off **during the lab time**. One of the laboratory instructors must verify the appropriate steps by initialing on the **Instructor Verification** line. When you have completed a step that requires verification, simply raise your hand and demonstrate the step to the instructor.

FORMAL Lab Report: You must write a formal lab report that describes your approach to music synthesis (Section 4). Staple the **Instructor Verification** sheet to the end of your lab report as evidence that the appropriate steps were witnessed by the instructor.

The report will **due during the week of 8-Feb at the start of your lab**.

---

1 Introduction

This lab includes a project on music synthesis with sinusoids. The piece *Jesu, Joy of Man's Desiring* has been selected for doing the synthesis program. The project requires an extensive programming effort and should be documented with a complete **formal** lab report.[1] 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.

2 Overview

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

$$x(t) = \sum_k A_k \cos(\omega_k t + \phi_k) \tag{1}$$

In this lab we will synthesize waveforms composed of sums of sinusoidal signals (1), sample them, and then reconstruct them for listening. Specifically, we will create a version of *Jesu, Joy of Man's Desiring* using sinusoidal synthesis. If you would like to try other songs, the CD-ROM includes information about four alternative tunes: *Minuet in G*, *Für Elise*, *Beethoven's Fifth* and *Twinkle, Twinkle, Little Star*.

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 course.

[1] Refer to the Web-CT page for more details on the required format.

3 Warm-up: Music Synthesis

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

In this lab, the sinusoids 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.

3.1 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 `soundsc(xx,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. If you are using `sound.m`, it will also be necessary to scale the vector `xx` so that it lies between ± 1.[2]

3.2 Piano Keyboard

Section 4 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 reference note is the A above middle-C

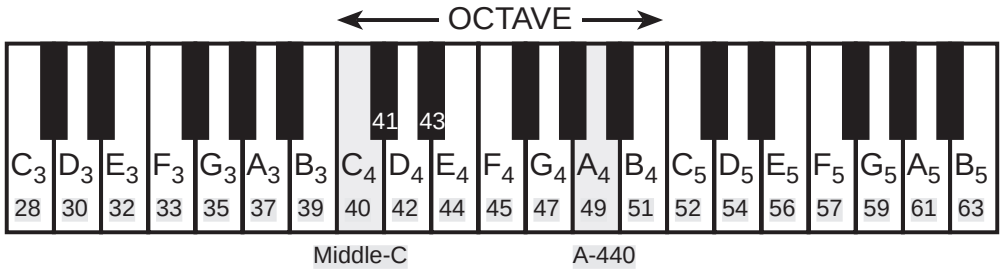

Figure 1: Layout of a piano keyboard. Key numbers are shaded. The notation C_4 means the C-key in the fourth octave.

which is usually called A-440 (or A_4) 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 even more details.

[2] In MATLAB version 5, there is a function `soundsc(xx,fs)` which performs that scaling.

[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.

Musical notation shows which notes are to be played and their relative timing (half, quarter, or eighth). Figure 2 shows how the keys on the piano correspond to notes drawn in musical notation. The white keys are all labeled as A, B, C, D, E, F, and G; but the black keys are denoted with "sharps" or "flats." A sharp such as $A^{\#}$ is one key number larger than A; a flat is one key lower, e.g., $A_4^{\flat}$ is key number 48.

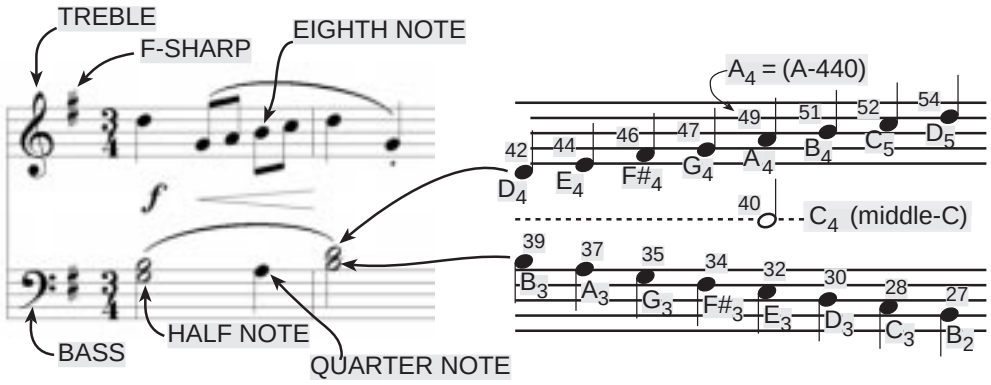

Figure 2: Musical notation is a time-frequency diagram where vertical position indicates which note is to be played. Notice that the shape of the note defines it as a half, quarter or eighth note, which in turn defines the duration of the sound.

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" or "well-tempered" and was introduced in Germany in the 1760's and made famous by J. S. Bach in the "Well Tempered Clavichord."

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 $f_{43} = 440 \times 2^{-6/12} = 440/\sqrt{2} \approx 311$ Hertz.

(a) From the previous lab you should have a `note.m` function that will synthesize the signal for a particular key number. It will be useful to have that function for doing the rest of the warm-up.

(b) 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   = 11025;                      %-- or 8000 Hz
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
soundsc( xx, fs )
```

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

Instructor Verification (separate page)

3.3 Spectrogram: Two M-files

In this part, you must synthesize an F-major chord consisting of the notes F_4, A_4 and C_5 and then display the spectrogram computed from the resulting time signal. Remember that the spectrogram displays the *frequency* content of the synthesized *time* signal.

(a) Generate a signal for the F-major chord that is at least 0.8 seconds long. Use a sampling frequency of 11025 Hz.

(b) Use the function `specgram(xx,[],fs)`. Set the arguments correctly.
Note: the second argument is made equal to the "empty matrix" so that a default value of 256 is used. This second argument is the *window length* which could be varied to get different looking spectrograms. The spectrogram will probably look better with a longer window length, e.g., 512, or 1024.[4] For some useful display functions, try `help axis` and `help zoom`.

(c) Use the function `plotspec(xx,fs)` which can be downloaded from Web-CT if necessary. Show that you get the same result as in part (b). Explain why the result is correct. If necessary, add a grid so that frequencies can be measured accurately.
Note: `plotspec()` can have a third argument which is the *window length* (default value is 256). The same comment about window length from part (b) applies here.

Instructor Verification (separate page)

4 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 in the book, but we have chosen a particular one for the synthesis program in this lab 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.

[4] Usually the window length is chosen to be a power of two, because a special algorithm called the FFT is used in the computation. The fastest FFT programs are those where the signal length is a power of 2.

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

(c) Determine the frequency (in hertz) for each note (see Fig. 1 and the discussion of the well-tempered scale in the warm-up.) A file called `j_notes.m` will be provided with this information.

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

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

(f) Include a spectrogram plot which is actually a gray-scale image. Make the spectrogram of a portion of your synthesized music–probably about 3 or 4 secs.–so that you can illustrate the fact that you have all the different notes. Since the spectrogram M-files will scale the frequency axis to run from zero to half the sampling frequency, it might be useful to "zoom in" on the region where the notes are. Consult `help zoom` for a MATLAB command that enables zooming.

4.1 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()` or `plotspec()`. 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.

MUSIC GUI

4.2 *Jesu, Joy of Man's Desiring*

Jesu, Joy of Man's Desiring is a well known piece of music written by Bach. The first few measures are shown in Fig. 3, and you can listen to the part that you will synthesize by following the links on the *DSP First CD-ROM*.

Jesu

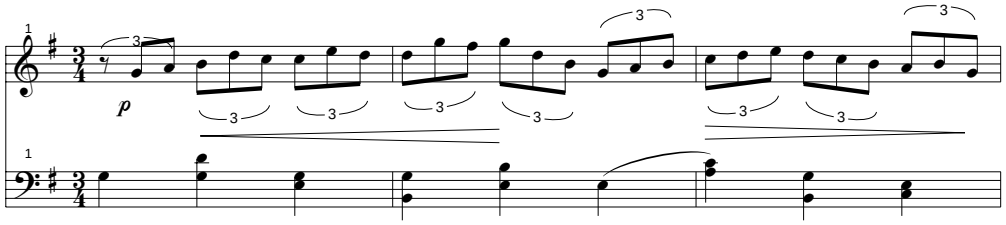

Figure 3: First few measures of Jesu, Joy of Man's Desiring.

Determine the notes that are played in this passage, map each note to a key number and then synthesize sinusoids to recreate the piece. Use sinusoids sampled at 11025, or 8000, samples per second.

4.2.1 Timing

From your recollection of this music, estimate the time duration needed for each note. You can define a time duration for quarter notes, eighth notes, and so on, but you still may need to make adjustments in the timing.

A baseline value for the time of a quarter note might be one quarter of a second (0.25 sec.). However, that might make the piece sound very slow. You should write the code so that note duration is a global parameter that can be changed easily. For example, you might let the duration of a quarter note (see Fig. 2) be defined with the statement:

```
q_dur = 0.25;
```

and then calculate all other durations in terms of `q_dur`. Half notes are twice as long as quarters; eighth notes are half as long. Triplets, as in *Jesu*, appear to be eighth notes, but should actually be a little shorter—three of them should have a total duration equal to a quarter note. If the quarter note duration is defined only once, then it could be changed: for example, making `q_dur = 0.125;` would make the whole piece play twice as fast.

Another timing issue is related to the fact that when a musical instrument is played, the notes are not continuous. Therefore, adding very short pauses between notes usually improves the musical sound because it imitates the natural transition that a musician must make from one note to the next.

4.2.2 Assessment

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. Some commonly used improvements are described in the next section.

4.3 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.

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

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 envelope 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 4 shows a linear approximation to the ADSR profile.

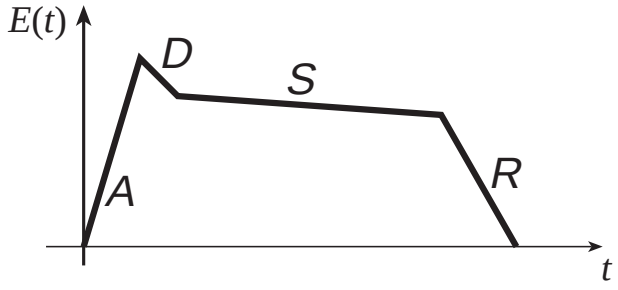

Figure 4: ADSR profile for an envelope function $E(t)$.

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. Since true piano sounds have a second and third harmonic content, and 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.

4.4 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 *Jesu, Joy of Man's Desiring*. 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 `j_notes.m` just contains the melody, and it is available on the WebCT page, under "Laboratory Assignments."

Lab #4
EE-2200
Winter-99
Instructor Verification Sheet

Staple this page to the end of your Lab Report.

Name: _____ Date of Lab: _____

Complete the on-line survey in Web-CT:

Verified:_____ Date/Time:_____

Part 3.2 Complete and demonstrate the script file `play_scale.m`:

Verified:_____ Date/Time:_____

Part 3.3 Demonstrate and explain the spectrogram of the F-major chord:

Verified:_____ Date/Time:_____

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.