Project 1 - Synthesis of Musical Notes and Instrument Sounds with Sinusoids

ECE505, Fall 2015
EECS, University of Tennessee
(Due 09/03)

1 Objective

The project serves three purposes, to experience more complicated signals related to the basic sinusoid, to get familiar with a couple of interesting phenomena produced based on basic sinusoid, and to become comfortable with MATLAB programing.

2 Background

You need a little bit background on sampling in order to best play the music you synthesized. You also need some background in how to read musical notes such that you can translate a music piece into sound played through the computer. Finally, some more complicated phase functions are needed in order to mimic interesting instrument sounds.

2.1 Sampling

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). When we record our speech using a recorder, the A-to-D converter *samples* the analog speech signal, using a sampling rate f_s, and converts it to a discrete-time signal such that the speech can be stored on computer as a sequence of numbers (or samples). When we need to play the speech signal, the D-to-A converter interpolates the discrete samples and *reconstructs* it back to its analog format such that we can hear the sound. The sampling theorem says that if the input analog signal is a sum of sinusoids, then the output signal, resulting from both A-to-D and D-to-A, will be equal to the input signal if the sampling rate f_s is more than twice the highest frequency f_{max} in the input. A convenient choice for the D-to-A conversion rate is 8000 samples per second, i.e., $T_s = 1/8000$ sec. Another common choice is 11025 Hz, which is one-quarter of the rate used for audio CDs. For PC/Mac users, use 11025 Hz; for Unix users, use 8000 Hz.

- Q1) Most of you know that audio CDs use a sampling rate of 44.1 kHz. Why? or How is this number calculated?

2.2 Piano keyboards and music notes

Music tones can be modeled mathematically by sinusoidal signals. A piano, with 88 keys (Fig. 1), is divided into *octaves* containing 12 keys each. Within an octave, the neighboring keys maintain a constant frequency ratio and the tones in each octave are twice the frequency of the corresponding tones in the next lower octave. Actually, the word *octave* means a doubling of the frequency. To calibrate the frequency scale, a reference tone is needed. By convention, the A (key 49) above middle C (key 40), which is usually called A-440 (or A_4, i.e., A in the 4th octave), is used as the reference and its frequency is 440 Hz.

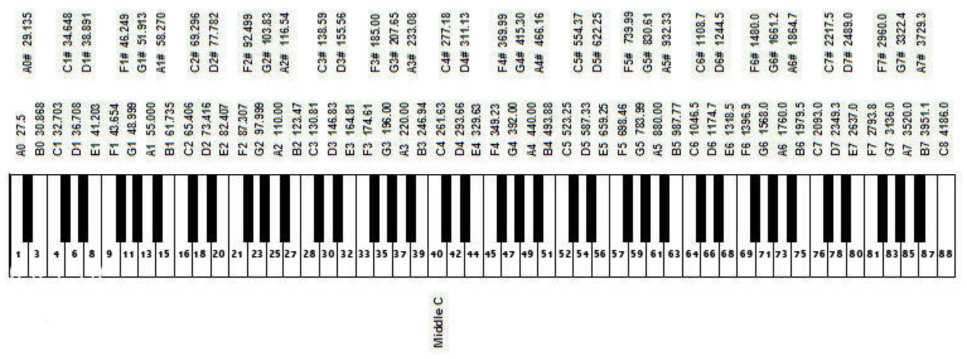

Figure 1: Piano keys numbered from 1 to 88 with their corresponding frequencies. Middle C is key 40. A-440 is key 49. The 4th octave starts from key 40 and ends at key 51.

- Q2) What is the frequency ratio between neighboring keys within the same octave?

- Q3) What is the frequency of middle C? Show detailed steps.

Musical notation shows which notes are to be played and their relative timing. E.g., half notes last twice as long as quarter notes, which, in turn, last twice as long as eighth notes. Figure 2 shows how the keys on the piano correspond to notes drawn in musical notation.

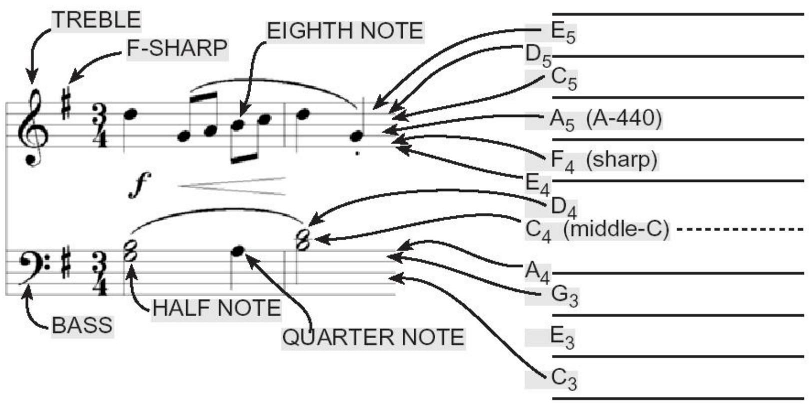

Figure 2: Musical notation and piano keys [2].

2.3 FM synthesis of instrument sounds

Frequency modulation (FM) can be used to make interesting sounds that mimic musical instruments, such as bells, woodwinds, drums, etc. We have already demonstrated the chirp signal generated with frequency changes linearly in time. FM music synthesis uses a more interesting phase, $\psi(t)$, one that is sinusoidal. Since the derivative of a sinusoidal is also sinusoidal, the instantaneous frequency will oscillate sinusoidally. The general equation for an FM sound synthesizer is

$$x(t) = A(t)\cos(2\pi f_c t + I(t)\cos(2\pi f_m t + \theta_m) + \theta_c)$$

and the instantaneous frequency

$$\begin{aligned} F(t) &= \tfrac{1}{2\pi}\tfrac{d}{dt}\psi(t) \\ &= \tfrac{1}{2\pi}\tfrac{d}{dt}(2\pi f_c t + I(t)\cos(2\pi f_m t + \theta_m) + \theta_c) \\ &= f_c - I(t) f_m \sin(2\pi f_m t + \theta_m) + \tfrac{1}{2\pi}\tfrac{dI}{dt}\cos(2\pi f_m t + \theta_m) \end{aligned}$$

where $A(t)$ is an *envelope* that modulates the amplitude of the signal, f_c is the *carrier frequency*, f_m is the *modulating frequency* that specifies the rate of oscillation of the instantaneous frequency $F(t)$, θ_m and θ_c are arbitrary phase constants, usually set to $-\pi/2$ so that $x(0) = 0$, and $I(t)$ is referred to as the *modulation index envelope*. The net result is that $I(t)$ can be used to vary the harmonic content of the instrument sound (referred to as *overtones*). When $I(t)$ is small, mainly low frequencies will be produced. When $I(t)$ is large, higher harmonic frequencies can also be produced. In addition, the ratio between f_c and f_m is important in creating the sound of a specific instrument. For example, for the bell, a good choice for this ratio is 1:2, e.g., $f_c = 110$ Hz and $f_m = 220$ Hz. For more details, see Chowning's paper [3] in 1973.

3 Tasks

3.1 Task 1: Synthesis of musical notes

- Task 1.1: Write a function `function tone = note(keynum, dur, fs)` that can produce a sinusoidal waveform corresponding to a given piano key number for any given duration with the specified sampling rate. Suppose you are using a PC, then the default sampling rate is 11025 Hz.

- Task 1.2: Write a MATLAB code to play scales, `scale.m`, e.g., the major scale, which uses the function `note`.

- Task 1.3: Improve the sound by adding harmonics. True piano sounds contain several frequency components, such as second and third harmonics. Be careful to make the amplitudes of the harmonics smaller than the fundamental frequency component. Experiment what sounds the best. Write a function `function [x] = harmonic(x, n, f0, fs)` to composite a new sinusoidal signal with n harmonics based on input `x` which, e.g., can be the output from `note.m`, and fundamental frequency, `f0`.

- Task 1.4: Improve the sound quality by applying envelope. The musical passage is likely to sound artificial, because it is created from pure sinusoids. You can improve the quality of the sound by multiplying each pure tone signal by an envelope $E(t)$ so that it would fade in and out.

  $$x(t) = E(t)\cos(2\pi f_0 t + \theta)$$

  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, the method is 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. See more details at [1]. Write a function `function [x] = adsr(dur, fs)` that generates such an envelope for given duration of time.

- Task 1.5: Synthesize *Twinkle, Twinkle, Little Star*. Figure 3 is the entire page of the music. You can just synthesize the first few measures.

3.2 Task 2: FM synthesis of instrument sounds

- Task 2.1: Write a function, `function [x, t] = mychirp(f1, f2, dur, fs)` that synthesizes a linear-FM chirp signal with f1 being the starting frequency, f2 the ending frequency. Let f1=200, f2=2000, plot the spectrogram of the signal.

- Task 2.2: Write two functions, `function [x] = bellenv(tau, dur, fs)` that produces the envelope function for the bell sound, and `function [x, t] = bell(fc, fm, I0, tau, dur, fs)` that synthesizes the bell sound. The amplitude envelope $A(t)$ and the modulation index envelope $I(t)$ for the bell are both decaying exponentials with the following form

$$y(t) = e^{-t/\tau}$$

where τ is a parameter that controls the decay rate of the exponential. Notice that $y(0) = 1$ and $y(\tau) = 1/e$, which means that τ is the time it takes for the signal to decay to $1/e = 36.8\%$ of its initial value. Therefore, τ is referred to as the *time constant*. For the bell, use $A(t) = e^{-t/\tau}$ and $I(t) = I_0 e^{-t/\tau}$. The table below lists 6 cases to get a variety of bells. For each case in the

CASE	f_c (Hz)	f_m (Hz)	I_0	τ (sec)	T_{dur} (sec)
1	110	220	10	2	6
2	220	440	5	2	6
3	110	220	10	12	3
4	110	220	10	0.3	3
5	250	350	5	2	5
6	250	350	3	1	5

table, do the following,

1. Listen to the sound. Describe how you can hear the frequency content changing according to $I(t)$. Plot $F(t)$ versus t for comparison.
2. Display a spectrogram of the signal. Describe how the frequency content changes, and how that change is related to $I(t)$. Point out the "harmonic" structure of the spectrogram, and calculate the fundamental frequency, f_0.

3. Plot the entire signal and compare it with $A(t)$.
4. Plot about 100 to 200 samples from the middle of the signal and explain what you see, especially the frequency variation.
5. Experiment different ratio between f_c and f_m and see which parameters sound best to you.

4 What to Turn in

- a tar file or a zip file with the following items:
  - Task 1: note.m, scale.m (with harmonics and envelope added), adsr.m, harmonic.m, twinkle.m
  - Task 2: mychirp.m, bellenv.m, bell.m
  - Content.m: explaining what each code is for
  - (5) Completeness of the above listed files
- a pdf file with the written report that includes the following
  1. (10) On sampling
     - (10) Explain in general how the choice of different sampling rates (other than 8000 or 11025 Hz) affects the sound effect. This needs to be done through experiments. But you do not need to turn in test code. Just the explanation.
  2. (45) On synthesis of musical notes
     - (5) Answer Q2 and Q3 on page 2
     - (10) Correctness of note.m, scale.m
     - (10) Plot the ADSR envelope. Plot the spectrogram of the major scale before and after adding improvements and explain the difference
     - (5) Sound quality of the major scale
     - (15) Sound quality of twinkle, twinkle, little star
  3. (40) On synthesis of instrument sounds
     - (15) Correctness of mychirp.m, bellenv.m, and bell.m
     - (25) Answer the five listed tasks at the end of the Task 2 description
  4. Source code print

Acknowledgement

This project is designed based on labs provided in McClellan, Schafer, and Yoder's *DSP FIRST: A Multimedia Approach*.

References

[1] Adsr envelope. `http://en.wikipedia.org/wiki/ADSR_envelope`.

[2] Piano keyboard. `http://www.engr.udayton.edu/faculty/jloomis/ece561/notes/piano/piano.html`.

[3] J. M. Chowning. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society*, 21(7):526–534, Sept. 1973.

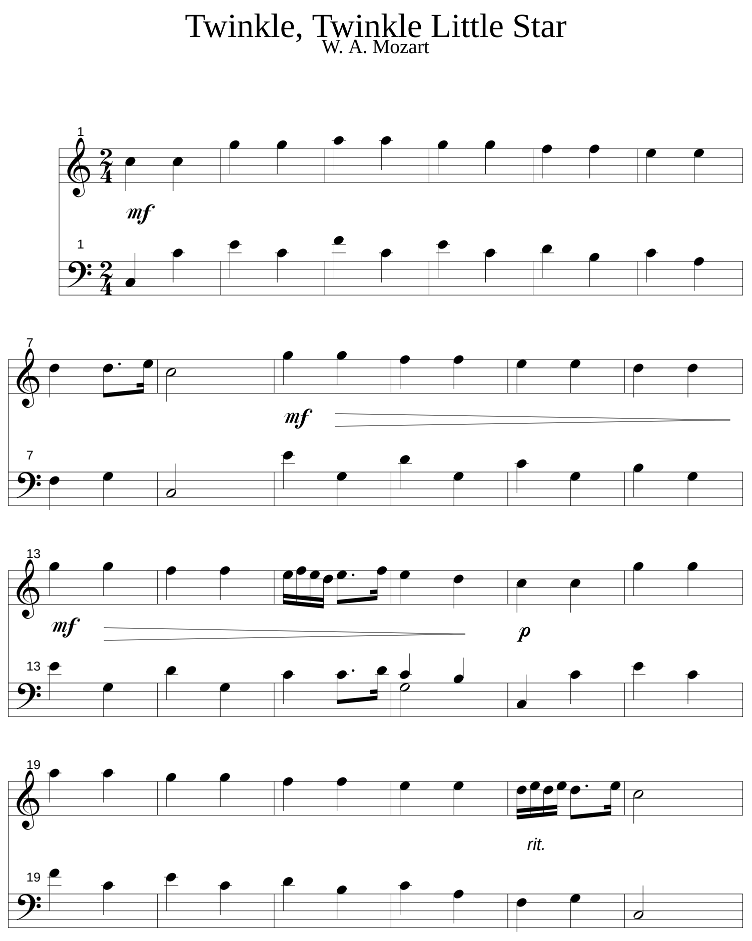

Figure 3: Twinkle, Twinkle, Little Star.