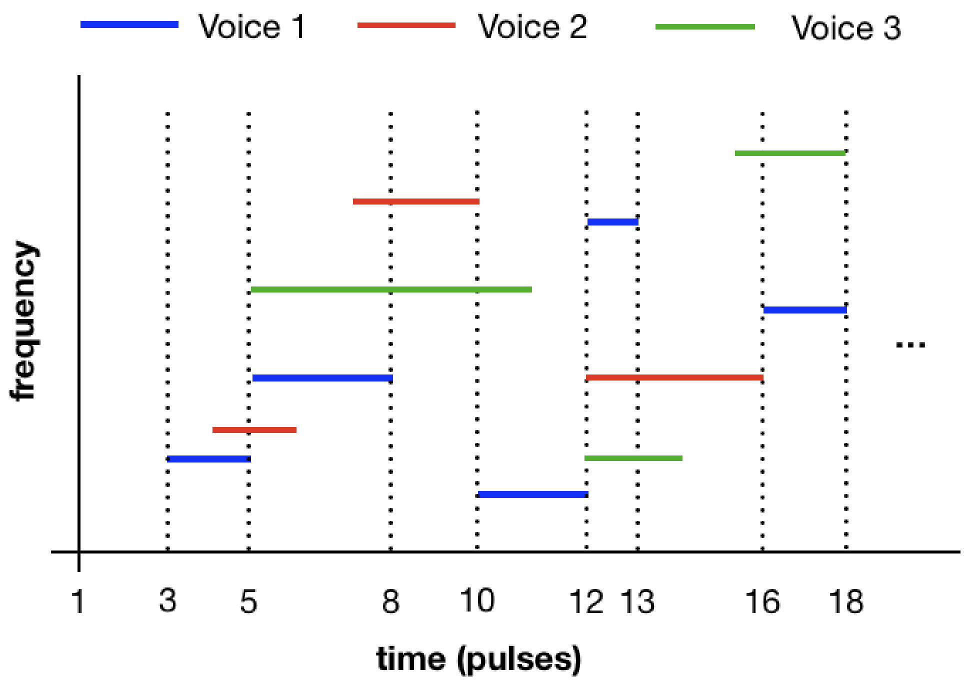
Lab 6 Prelab
ECE203
Music Synthesis 2

Introduction

In the previous lab you used sinusoids to synthesize music consisting of a single note at a time. In this week's lab you will extend your code to synthesize music involving multiple notes played at a time (chords) and improve the sound quality.

Playing Multiple Notes Simultaneously

We will include multiple notes by changing the struct `theVoices` to be an array struct. That is, `theVoices(1).noteNumbers`, `theVoices(1).startPulses`, and `theVoices(1).durations` contains the note numbers, start pulses, and durations for the first "track" of notes or voice, `theVoices(2).noteNumbers`, `theVoices(2).startPulses`, and `theVoices(2).durations` contains the note numbers, start pulses, and durations for the second voice, and so on. The figure below illustrates the time-frequency characteristics of a short section of a song containing three voices. Note that none of the notes in any one voice occur at the same instant in time, but multiple notes can occur simultaneously (producing a chord) when notes from two or three voices occur at the same time.



The example illustrated in the figure would correspond to the following struct:
`theVoices(1).noteNumbers(1)=43, theVoices(1).startPulses(1)=3, theVoices(1).durations(1)=2`

```
theVoices(1).noteNumbers(2)=45, theVoices(1).startPulses(2)=5, theVoices(1).durations(2)=4
theVoices(1).noteNumbers(3)=42, theVoices(1).startPulses(3)=10, theVoices(1).durations(3)=3
theVoices(1).noteNumbers(4)=49, theVoices(1).startPulses(4)=12, theVoices(1).durations(4)=1
theVoices(1).noteNumbers(5)=47, theVoices(1).startPulses(5)=16, theVoices(1).durations(5)=2

theVoices(2).noteNumbers(1)=44, theVoices(2).startPulses(1)=4, theVoices(2).durations(1)=2
theVoices(2).noteNumbers(2)=50, theVoices(2).startPulses(2)=7, theVoices(2).durations(2)=3
theVoices(2).noteNumbers(3)=45, theVoices(2).startPulses(3)=12, theVoices(2).durations(3)=4

theVoices(3).noteNumbers(1)=48, theVoices(3).startPulses(1)=5, theVoices(3).durations(1)=6
theVoices(3).noteNumbers(2)=43, theVoices(3).startPulses(2)=12, theVoices(3).durations(2)=2
theVoices(3).noteNumbers(3)=52, theVoices(3).startPulses(3)=15, theVoices(3).durations(3)=3
```

You will incorporate these additional voices into the vector representing the sound by adding a loop over the array index for `theVoices`. That is, you will have a nested loop - an outer loop over the array index for `theVoices` and an inner loop over the notes within each voice. If there are three voices, then your code will look something like this:

```
 for k = 1:3 % loop over voices
   for i = 1:length(theVoices(k).startPulses)
      tone = key2note( ...  ); % note for voice k, start pulse i
      start_indx = ....; % start index of note
      stop_indx = ....; % stop index of note
      xx(start_indx:stop_indx) = xx(start_indx:stop_indx) + tone;
   end
end
```

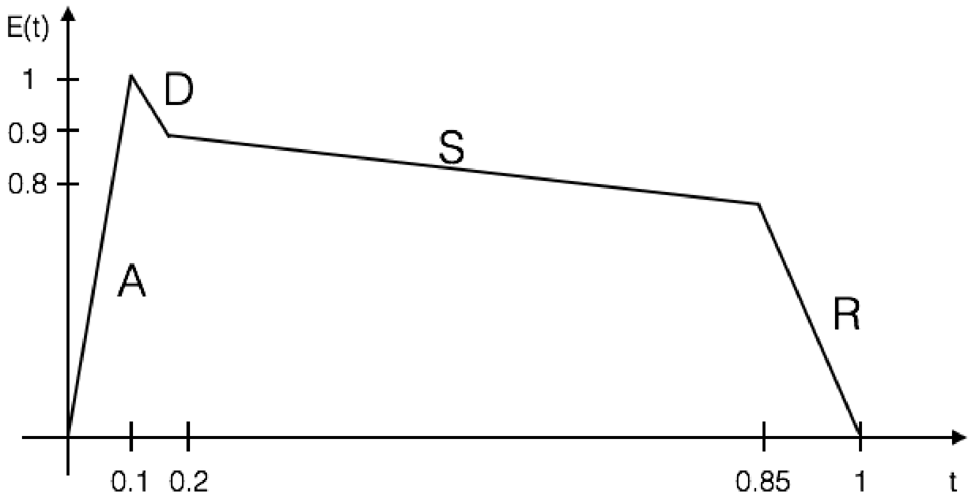
Here `xx` is the vector containing the sound. The loop on `i` synthesizes the notes for the ith voice, while the loop on `k` synthesizes all three voices. In the figure, the notes in blue (Voice 1) are created when `k=1`, the notes in red (Voice 2) are created when`k=2`, and the notes in green (Voice 3) are created when `k=3`. The inner loop inserts all the notes for the first voice when `k=1`, then adds all the notes for the second voice when `k=2`, and finally adds all the notes for the third voice when `k=3`.

Sound Quality - Envelope

The notes you have generated so far have an abrupt onset and stop. That is, the sound amplitude instantly changes from zero to full amplitude at the note start, and from full amplitude to zero at the end of the note. Real instruments build amplitude quickly at the beginning of a note and exhibit some amplitude decay before the note ends. We will simulate this effect by multiplying each tone with an envelope signal $E(t)$ that fades in and fades out

$$x(t) = E(t)\cos(2\pi f_{note} t)$$

A standard way to define the envelope function is to divide $E(t)$ into four sections: attack (A), decay (D), sustain (S), and release (R). Together these are called ADSR. The attack is a quickly rising front edge, the decay is a small short-duration drop, the sustain is more or less constant and the release drops quickly back to zero, as illustrated in the figure below. The time axis on the figure assumes the note is one unit long. The actual envelope is scaled in time to match the duration of the note. The figure illustrates the A phase being the first 10%, the D phase the second 10%, the S phase lasting from 20% to 85%, and the R phase as the last 15% of the note.



You will synthesize $E(t)$ in MATLAB using line segments and apply it to the note within the `key2note` function.

Sound Quality - Harmonics

Musical instruments are rich in harmonics. Harmonics give the sound a rich, pleasing quality. Harmonics also differentiate a saxophone from a clarinet or trumpet playing the same note. Pianos have relatively strong second and third harmonics.

You will add second and third harmonics to each note within `key2note`. Be sure the amplitude of the harmonics are less than the amplitude of the fundamental.