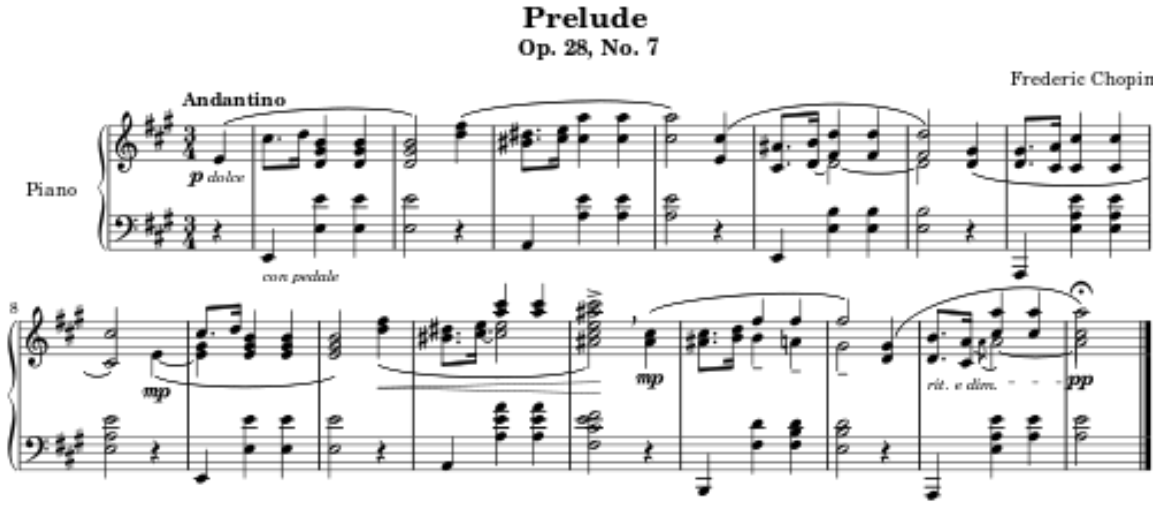
Lab 5 Prelab
ECE203 - Van Veen
Music Synthesis 1

Introduction

In this lab you will use sinusoids to synthesize music consisting of a single note at a time. You will learn about structured variables in MATLAB as well as some features useful for debugging MATLAB code. Next week's lab will extend your code to synthesize music involving multiple notes played at a time (chords) and improve the sound quality.

The simplest view of music is a sequence of tones. Each tone or "note" has a frequency associated with it, a starting time, and a duration. Musical notation, such as the example shown below, codes this information so the musician can efficiently read it and play their instrument accordingly.



Music notation is essentially a time-frequency representation. Time progresses from left to right and the vertical position of a note determines its frequency. We will be using an equivalent notation that is easy for the computer to read. Instead of using letters, like 'A', 'B', 'C', etc and vertical position on a staff for notes, we will use numbers. We will also define the start time and duration of each note.

Frequency of Musical Notes

Music is based on octaves - a doubling of frequency produces the same perceived "note", but with higher pitch. This is a consequence of human perception of sound being based on a logarithmic frequency scale. In western music there are twelve logarithmically-spaced steps per octave. This means that the frequency of the next higher note is $2^{1/12}$ that of the previous. Twelve such steps $(2^{1/12})^{12} = 2$ results in doubling the frequency.

The note "Concert A" is used for tuning and has frequency 440 Hz. Given this as a reference point and our knowledge that each step is $2^{1/12}$ higher in frequency, we can

compute the frequency of any note. In this lab we will use integers to represent notes and 49 to represent concert A in our code. This choice is arbitrary. Thus, note 51 is two steps higher than concert A and has frequency $440 * (2^{1/12})^2 = 493.88$ Hz. Similarly, note 47 is two steps lower and has frequency $440 * (2^{1/12})^{-2} = 392$ Hz. The "A" in the next higher octave is twelve steps higher (note 61 in our convention) and has frequency 440*2 = 880 Hz.

Starting Time and Duration

Music notation specifies the starting time of a note relative to the notes preceding it. The duration of each note is determined by a graphic symbol. The pace at which a piece is played may be specified using terms like "Allegro" or more precisely in terms of beats per minute. The pace, relative position, and symbol for the note determine its starting time and duration. We will specify absolute starting time and duration for ease of interpretation with the computer.

The convention adopted in this lab is to specify the absolute starting time of a note and its duration in terms of a unit called "pulses". We shall assume there are four pulses per beat and that the beats per minute are given. This information may be used to determine the start and end time of a note in seconds. The beginning of the piece is at pulse number 1.

Suppose there are 60 beats per minute. Then there are 240 pulses per minute or 4 pulses per second. So each pulse is 0.25 seconds long. Thus, a note that starts at pulse 40 and is 2 pulses long would start at $(40-1)*0.25 = 9.75$ seconds from the beginning of the piece at last 0.5 seconds. (Recall that the first pulse starts at time 0 seconds.) If the beats per minute is 120, then each pulse is 0.125 seconds long. A note that starts at pulse 60 and is 3 pulses long would begin $(60-1)*0.125 = 7.375$ seconds from the beginning of the piece and last 0.375 seconds.

You will be representing sound using a vector in MATLAB, so you will need to translate the start time and duration in seconds to indices of the corresponding vector. This is accomplished using the sampling frequency, or the interval between samples. We will assume a sampling frequency of 22,050 Hz (samples per second). If a note starts 5.125 seconds from the start of the piece, then it will start at index `notestart` = round$\{5.125 * 22050\} + 1$ where the round function rounds to the nearest integer. Remember, vector indices **must** be integers. If the tone representing the note is `lengthoftone` samples long, it will end at index `notestart` + `lengthoftone` - 1.