Processing Audio with Octave and LTSpice

Tony Richardson

Introduction

The Octave numerical analysis program and the LTSpice circuit simulation program can be used to process audio files. This papers describes how to read and write audio files from those programs.

Audio File Formats

There are several different audio file formats. Octave only supports reading/writing the AU and WAV formats. LTSpice only supports reading/writing the WAV format. To work with other audio file formats you will need an audio format converter program. I recommend Audacity (audacity.sourceforge.net). It is free and runs under most operating systems.

The WAV Audio File Format

The WAV file format is a fairly general file format. It supports a number of data types (integer and floating point) and sampling frequencies. The format can even support up to 65535 channels.

Most audio players that play WAV files are only capable of playing a subset of the data types, sampling frequencies and channels supported by the format. If you intend to play a WAV file you should use one of the following sampling frequencies 8000 Hz (telephone), 11025 Hz, 16000 Hz, 22050 Hz and 44100 Hz. Files should be created with only one (mono) or two (stereo) channels.

Audio Players

Audacity can also be used to play audio files created by Octave or LTSpice. Octave can be configured to play audio data directly from the Octave command prompt. The process of configuring the Windows version of Octave to directly play audio data using the VLC player (www.videolan.org/vlc) is described below.

Octave

Audio support is provided in the **audio** package from the Octave forge. This package must be loaded to have access to the functions described here.

Audio data can be loaded into Octave using the **auload** function:

[data, fs] = auload(“audiofile.wav”);

data will be a matrix with a number of columns equal to the number of channels in the audio file. The number of rows will be equal to the number of samples. **fs** is a scalar variable containing the sample frequency.

You can play audio data using the **sound** function:

sound(data, fs);

If this function does not work, Octave may not be configured to use an audio player. The configuration process is described below.

Save Octave audio data to a file using **ausave**:

```
ausave("oct_output.wav", data, fs, format);
```

Format is a string equal to one of ulaw, alaw, char, short, long, float or double.

The following code illustrates how to create a two-second stereo file containing 1000 Hz and 2000 Hz tones in the separate channels. The sample rate is 8000 Hz and the samples size is 16 bits.

```
fs = 8000; f1 = 1000;  f2 = 2000;
t_duration = 2;

k = [0:t_duration*fs]';                    % column vector containing sample indexes

x = [ cos(2*pi*f1*k/fs)    cos(2*pi*f2*k/fs)];

sound(x, fs);                              % listen to the tones

ausave('output.wav', x, fs, 'short');      % save data to file
```

Refer to the Octave on-line documentation for complete descriptions of the **auload**, **sound** and **ausave** functions.

Note that it is up to the Octave user to keep track of the actual sampling frequency associated with a matrix of audio data. Many of the Octave functions will assume a sampling frequency of 8000 Hz if a sampling frequency is not specified. Also note that aliasing may occur if a sample frequency is not selected in accordance with the Nyquist criterion.

Configuring Octave to Play Sound
Under Windows you can configure the **sound** function to play audio using the **vlc** player with the following commands:

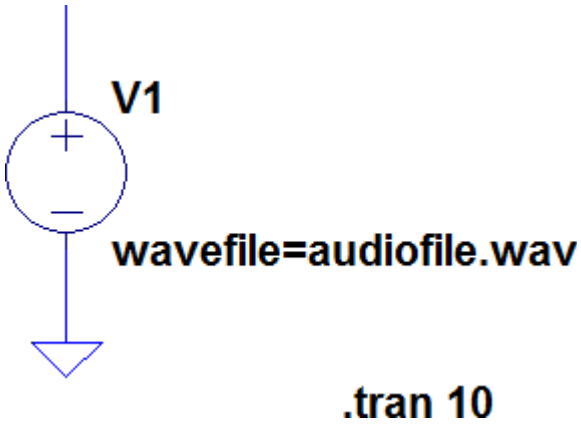
```
global  sound_play_utility;
VLC_PROGRAM="c:/Program Files (x86)/VideoLan/VLC/vlc";
VLC_OPTIONS="-q  --play-and-exit  -";
sound_play_utility=["cmd /c \"" VLC_PROGRAM "\" " VLC_OPTIONS];
clear   VLC_PROGRAM    VLC_OPTIONS
```

The **VLC_PROGRAM** string should contain the path to the **vlc** program. The quotes, spaces, slashes and backslashes are all important in get the **sound** function configured to use **vlc** correctly. You will want to copy these commands into an Octave script file so that they can be easily executed when ever you want to play audio from the Octave prompt. You may want to include them in your Octave startup file so that they are executed automatically every time Octave starts.

The **sound** function will not typically require any configuration under other operating systems.

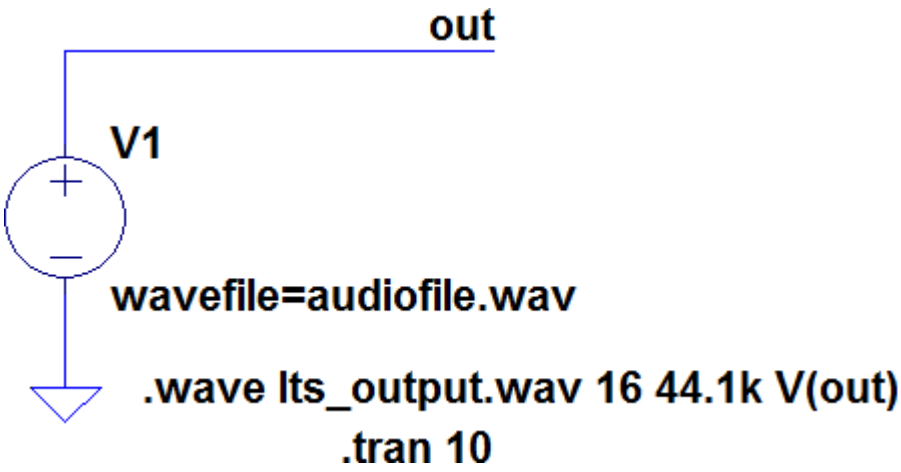
LTSpice

LTSpice voltage or current sources can be configured to read their data from a WAV file. Just change the voltage (or current) value to "wavefile=audiofile.wav". The figure below provides an example:



Voltages from WAV files are always in the range from -1 V to +1 V.

Any LTSpice voltage can be written to a WAV file by including a "dot" command on the schematic. The circuit schematic below writes the voltage at node "out" to an audio file with 16-bit resolution and a 44.1 kHz sampling rate. (The input file contains audio data sampled at 11025 Hz. LTSpice is being used as a sample rate converter in this "circuit").



Multiple channels can be included in the file by specifying multiple voltages at the end of the "dot wave" line in the schematic. The output voltage must be within the range -1 V to +1 V or clipping will occur. Dependent voltage sources are useful for creating voltages within the proper range. LTSpice supports writing WAV files with sample rates between 1 Hz and 4,294,967,295 Hz. Most of these sample rates are not supported by standard audio players, but the files can be read by LTSpice. The support for arbitrary sample rates makes it possible to read in audio data, modulate it and then write the modulated data out to a file at a high sample rate (so that the Nyquist criterion is met). The modulated data can then be used as input to an LTSpice demodulation circuit. The output of the demodulator can be written to a file at a standard sampling so that it can be listened to using a standard audio player.