Scientific Audio Processing, Part I

Octave, the equivalent software to Matlab in Linux, has a number of functions and commands that allow the acquisition, recording, playback and digital processing of audio signals for entertainment applications, research, medical, or any other science areas. In this tutorial, we will use Octave V4.0.0 in Ubuntu and will start reading from audio files through writing and playing signals to emulate sounds used in a wide range of activities.

Note that the main focus of this tutorial is not to install or learn to use an audio processing software already established, but rather to understand how it works from the point of view of design and audio engineering.

.

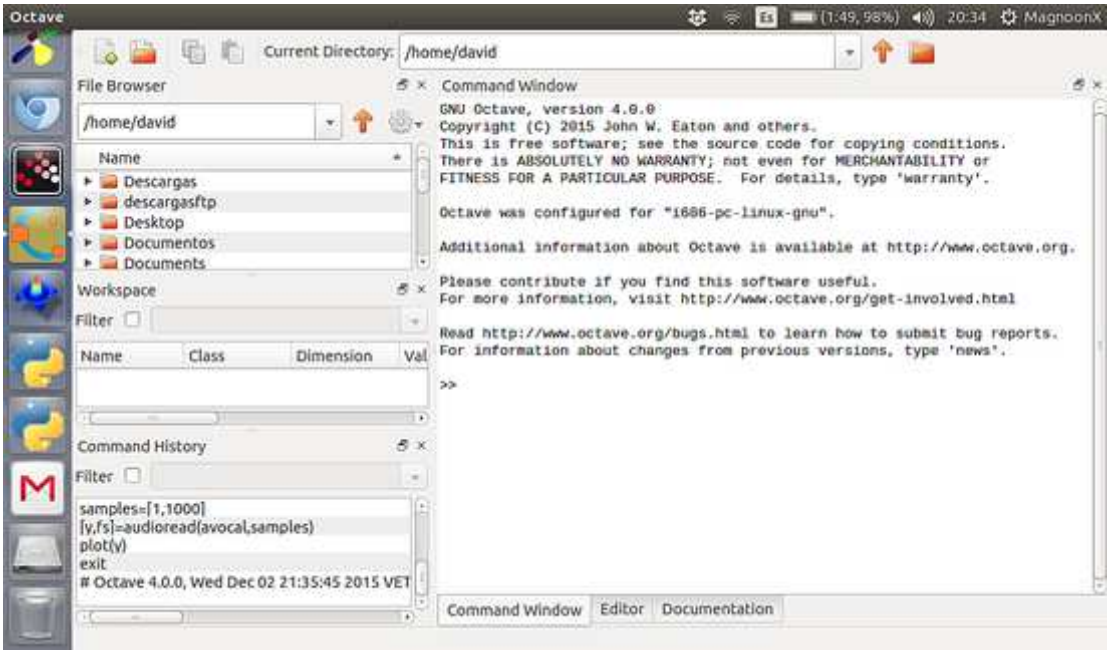
Prerequisites

The first step is to install octave. Run the following commands in a terminal to add the Octave PPA in Ubuntu and install the software.

sudo apt-add-repository ppa:octave/stable
sudo apt-get update
sudo apt-get install octave

Step 1: Opening Octave.

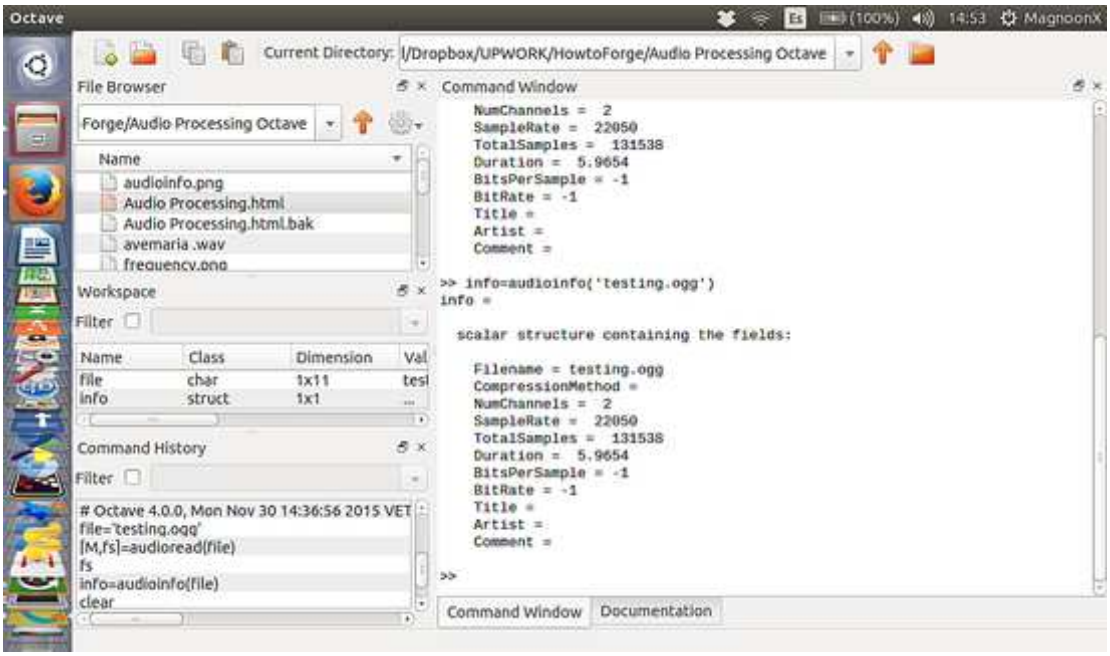
In this step we open the software by clicking on its icon, we can change the work directory by clicking on the File Browser dropdown.



Step 2: Audio Info

The command "audioinfo" shows us relevant information about the audio file that we will process.

>> info = audioinfo ('testing.ogg')



Step 3: Reading an audio File

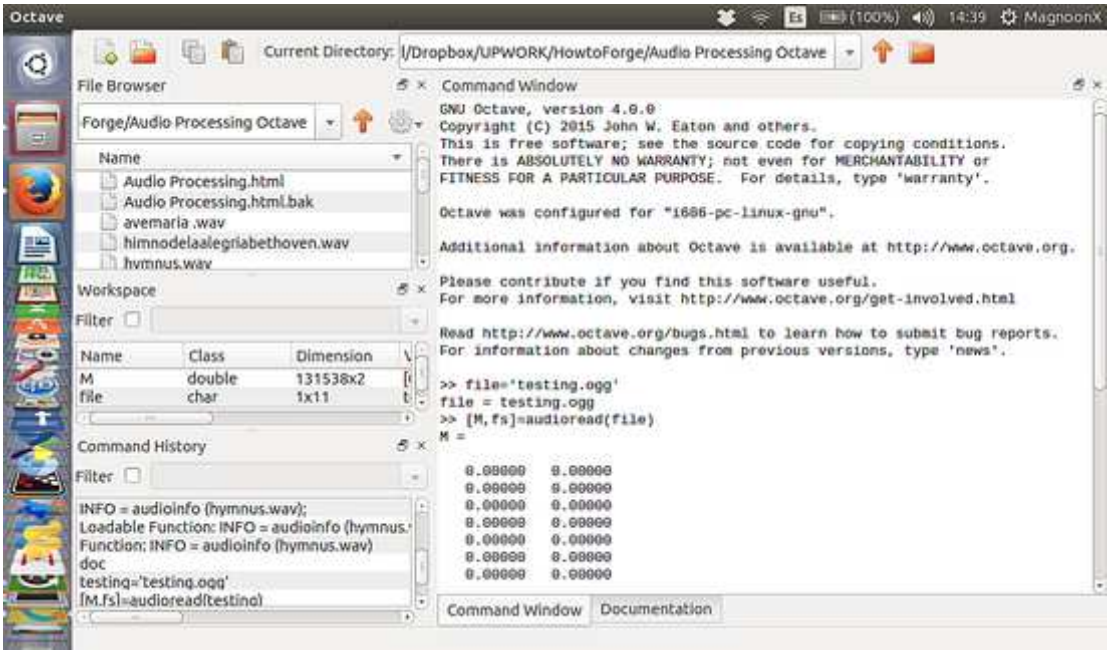
In this tutorial I will read and use ogg files for which it is feasible to read characteristics like sampling , audio type (stereo or mono), number of channels, etc. I should mention that for purposes of this tutorial, all the

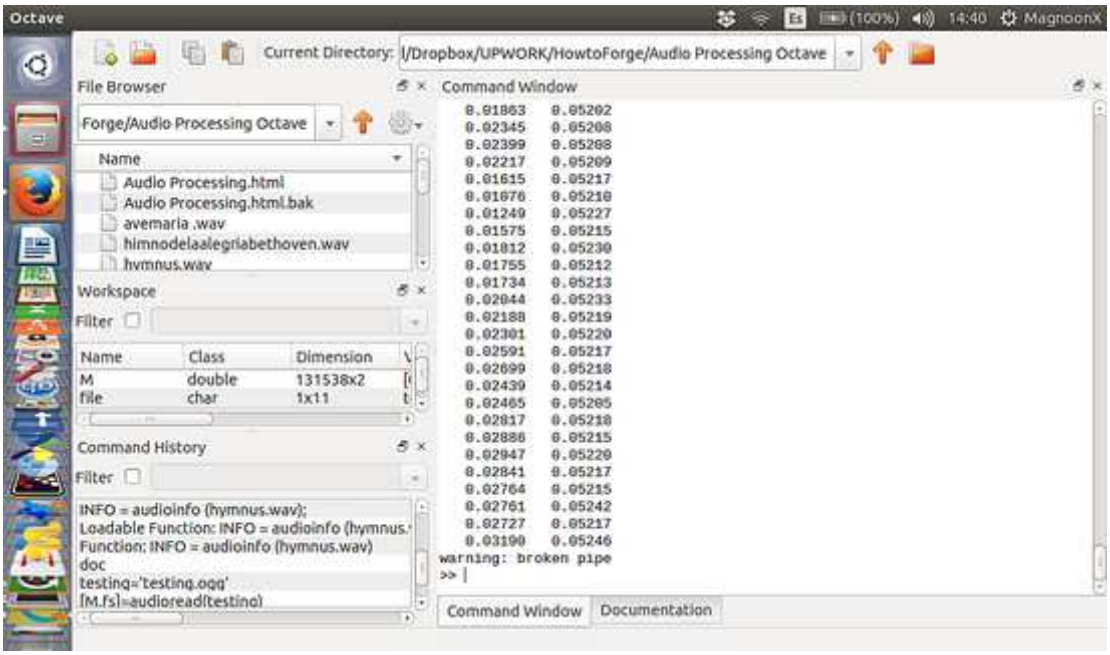
commands used will be executed in the terminal window of Octave. First, we have to save the ogg file in a variable. Note: it´s important that the file must be in the work path of Octave

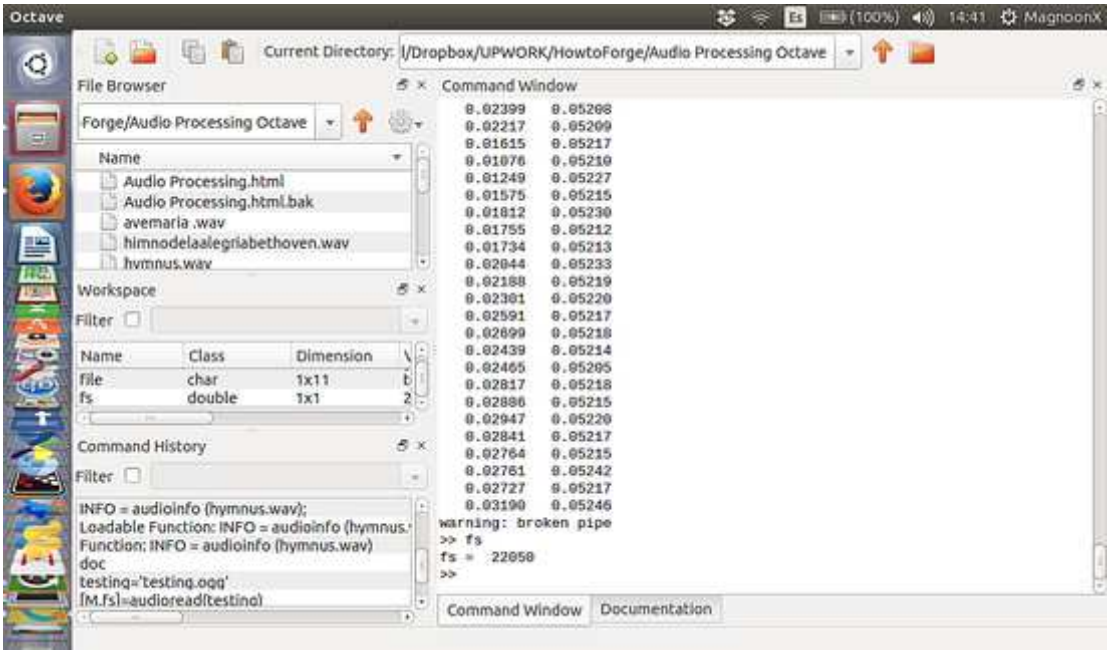
```
>> file='yourfile.ogg'

>> [M, fs] = audioread(file)
```

Where M is a matrix of one or two columns, depending on the number of channels and fs is the sampling frequency.







There are some options that we can use for reading audio files, such as:

>> [y, fs] = audioread (filename, samples)

>> [y, fs] = audioread (filename, datatype)

>> [y, fs] = audioread (filename, samples, datatype)

Where samples specifies starting and ending frames and datatype specifies the data type to return. We can assign values to any variable:

>> samples = [1, fs)

>> [y, fs] = audioread (filename, samples)

And about datatype:

>> [y,Fs] = audioread(filename,'native')

If the value is 'native' then the type of data depends on how the data is stored in the audio file.

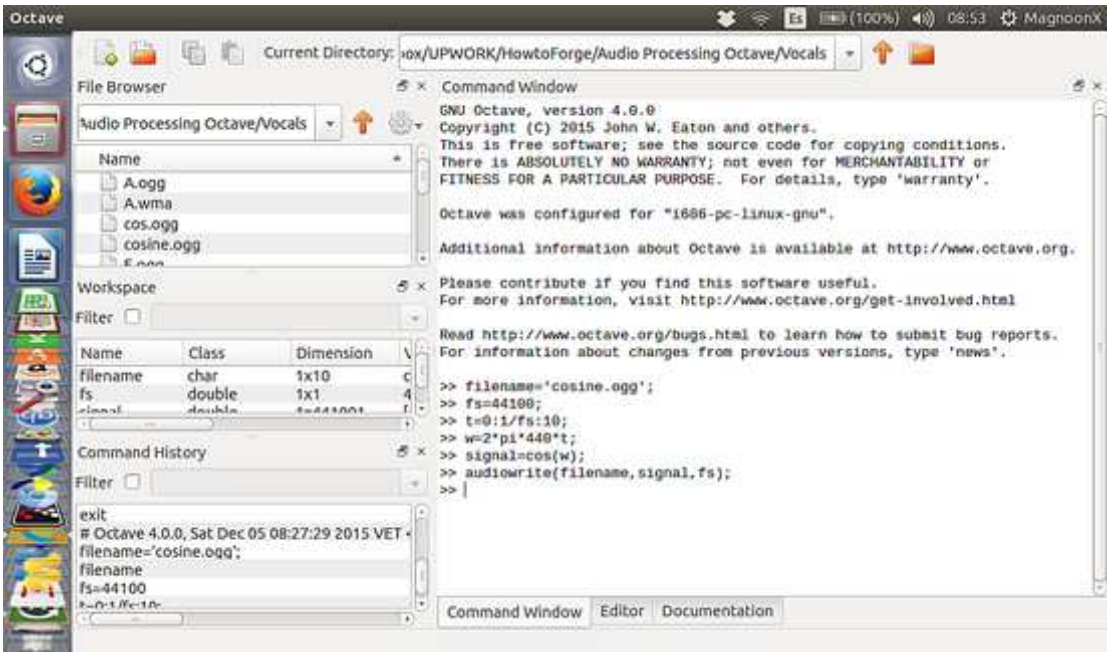
Step 4: Writing an audio file

Creating the ogg file:

For this purpose, we are going to generate an ogg file with values from a cosine. The sampling frequency that I will use is 44100 samples per second and the file will last for 10 seconds. The frequency of the cosine signal is 440 Hz.

```
>> filename='cosine.ogg';
>> fs=44100;
>> t=0:1/fs:10;
>> w=2*pi*440*t;
>> signal=cos(w);
>> audiowrite(filename, signal, fs);
```

This creates a file named 'cosine.ogg' in our workspace that contains the cosine signal.



If we play the 'cosine.ogg' file then this will reproduce a 440Hz tone which is equivalent to an 'A' musical tone. If we want to see the values saved in the file we have to 'read' the file with the 'audioread' function. In a further tutorial, we will see how to write an audio file with two channels.

Step 5: Playing an audio file

Octave, by default, has an audio player that we can use for testing purposes. Use the following functions as example:

```
>> [y,fs]=audioread('yourfile.ogg');
```

```
>> player=audioplayer(y, fs, 8)

  scalar structure containing the fields:

    BitsPerSample =  8
    CurrentSample = 0
    DeviceID = -1
    NumberOfChannels =  1
    Running = off
    SampleRate =  44100
    TotalSamples =  236473
    Tag =
    Type = audioplayer
    UserData = [](0x0)
>> play(player);
```

In the next parts of the tutorial, we will see advanced audio processing features and possible use cases for scientific and commercial use.