Lab 7: Complex Amplitude Modulation, Callback Functions

DSP Lab (EE 4163 / EL 6183)

Last Edit: Tuesday 25th October, 2016 11:33

1 Complex Amplitude Modulation

Previously, we used amplitude modulation (AM) to affect a speech signal. This method computes the output signal as

$$y(t) = x(t)\cos(2\pi f_1 t). \tag{1}$$

This AM method can lead the spectrum of the signal x to overlap itself. To shift a speech signal to a higher frequency without causing spectral overlapping as in Figure 1, we can use complex AM. This method was shown in class and in the Matlab demo programs.

1.1 Exercises

1. Using an input wave file of your choice, use Matlab to implement complex AM (like in the Matlab demo file).

2. Use Python to obtain the same result as in Matlab using the same input wave file as in the preceding part. Your Python output should be the same as your Matlab output.

3. **Real-time complex AM.** Implement real-time complex AM in Python with PyAudio. Your program should take the microphone signal as input and produce an output audio signal (on speakers or headphones). Compare the sound with simple AM in equation (1) implemented earlier in the course. It is best to use headphones or earbuds to compare the audio of the two methods because integrated laptop computer speakers might not be of sufficiently quality to properly hear the difference. SUBMIT

2 The Callback method

In previous PyAudio programs we used `stream.read` and `stream.write` to read from the microphone and to write to the speaker. This is known as *blocking* mode. Another approach is the *callback* method which does not use these functions. Instead, the callback method calls a user-defined *callback* function. The simple demo program `simple_wire_gain.py` shows how to read input from the microphone and write output to the speaker using the callback method. This demo does not process the signal; it

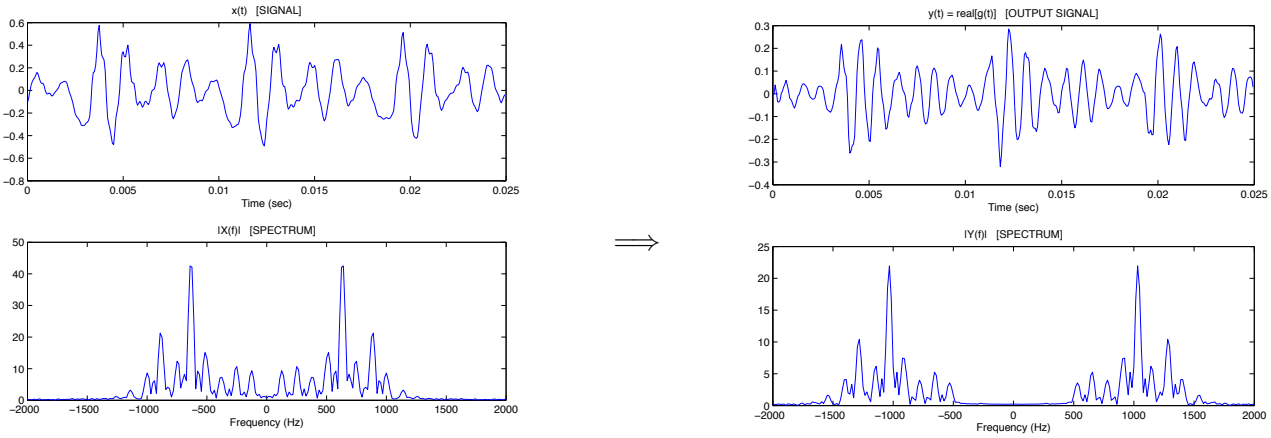

Figure 1: Complex AM

just plays the input from microphone in real-time. To avoid feedback, use headphones or an external microphone (otherwise you may hear an artifact that gets louder and louder).

For the callback method, we use the functions `start_stream` and `stop_stream`. The `start_stream` function initiates an ongoing calling of the callback function whenever a new block of input audio becomes available. The callback function will continue to be called until the `stop_stream` function is invoked. In the demo program, the `sleep` function temporarily prevents the program from continuing to `stop_stream` which stops the ongoing callback function.

For more details about PyAudio and its callback:

https://people.csail.mit.edu/hubert/pyaudio/docs/

For more details about `time` module:

https://docs.python.org/2/library/time.html

The demo program `simple_wire_gain_stereo_AM.py` shows how an audio effect can be implemented in the callback.

The demo program `record_and_play_vibrato.py` implements the vibrato effect using the callback method.

2.1 Exercises

1. Run and verify the demo programs on the course site, including:

   ```
   simple_wire.py
   simple_wire_gain.py
   simple_wire_gain_stereo.py
   simple_wire_gain_stereo_AM.py
   ```

2. Modify the demo program `simple_wire_gain.py`. Use a print statement inside the callback function to print the length of the input string and the value of `block_size`. This program does

not specify the block size, so the printed value is the default value.

3. The block size can be set using the `frames_per_buffer` parameter in the `p.open` statement. (See demo programs.) Set this value to a value different from the default value and verify by a print statement in the callback function that the input block has the specified length.

4. The demo program `play_randomly.py` does not use the callback method. Rewrite this demo program so it uses the callback method. When opening the audio stream, do not set the parameter `frames_per_buffer`. Instead, use the default block size. SUBMIT

5. In the demo program `simple_wire_gain.py`, there is one channel. If `CHANNELS` is set to 2, does the program work? Why or why not?

6. If we set `frame_per_buffer = 1` and `CHANNELS = 2`, then what is the value of `block_size` in the callback function?

7. Write a program to implement the vibrato effect using the callback method. The input should be from the microphone. The output should be to the speakers (no wave files). The parameter `frames_per_buffer` should be set to 1024 or left unspecified (default value). (Note that some demo programs set the parameter `frames_per_buffer` to 1. For this exercise, change this to `frames_per_buffer = 1024` or omit it.) SUBMIT

3 Notes

3.1 Stereo

For stereo signals, the stream should contains interlaced samples: L R L R .... For example, see the demo program `simple_wire_gain_stereo_AM.py`. For a stereo signal to be played correctly, the binary string written to the audio output should have this structure (L R L R ...).

3.2 NumPy and matplotlib

Depending on your Python installation, these libraries may need to be downloaded an installed on your computer system before you can import them into Python.

- The matplotlib library defines functions for plotting.

  http://matplotlib.org

- The NumPy library defines an array data type and allows for vector operations (instead of loops).

  http://www.numpy.org

  Tutorials about NumPy:

  http://cs231n.github.io/python-numpy-tutorial/

  http://www.python-course.eu/numpy.php