Outline

Pyaudio on the web:
https://people.csail.mit.edu/hubert/pyaudio/

Basic difference equation

In Python

    filter_16: 16 bits per sample

    Experiment with gain - change gain up and down
        if gain too low, then volume is low
        if gain too high, then error

    Assignment: insert an if-statement to avoid overflow error,
        e.g., set to maximum value (what is max value?)

    filter_32.py: 32 bits per sample
        Note use of paInt32 instead of paInt16
        Note use of 'i' instead of 'h'
        Why does 'gain' need to be increased for 32 bits compared to 16 bits?

    Assignment: Use 8 bits per sample

        Solution: use paInt8 (signed) and 'b' in pack
        signal values are -128 to +127

    filter_16:
        Try different values of Fs.
        The value of Fs affects the duration. Why?

    How to control the duration? (set r correctly)

In Matlab

make_filter_01: how to make a filter with specified pole position
compare two impulse responses, two pole-zero diagrams, two frequency response plots.

In Python

filter_16_r.py: show that changing r changes duration (time-constant)

What is a suitable formula for r?

How long does it take for the amplitude envelope to decay
to 1% of its initial value?

```
r^N = 0.01
=>
N = log_r (0.01)
N = log(0.01) / log(r)
=>
r = 0.01^(1/N)
```

In Matlab

make_filter_02: Uses audio sampling rate Fs = 8000

In Python

filter_16_T:
Prescribe the time-constant (duration),
vary Fs, listen to output, verify that duration is maintained
as Fs is varied.

Fourth-order system:

One way to implement a fourth-order system is to
simply cascade two second-order systems.

In Python: filter_twice.py
In Matlab: make_filter_03.m

Questions:

Controlling the initial amplitude: In the difference equation, how should b0 be set so that the impulse response of the second-order system has an initial value of 1.0?

How to set the cascade of two filters (same frequency) but with control of the rise-time and duration? (The two filters should have different pole radii).

Activities for students

• Modify filter_16.py to avoid overflow errors, regardless of the value of gain. To do this, insert an if statement to verify the sample value is in the allowed range. If it is not, set the value to its maximum (positive or negative) allowed value, before writing it to the audio stream. Test your program by setting the gain to a high value. What effect does this have on the sound produced by the program?

• In filter_16.py, the filter transfer function is
```
   H(z) = 1 / (1 + a1/z + a2/z^2).
```
Modify the filter so that the transfer function is
```
   H(z) = B(z) / (1 + a1/z + a2/z^2)
```
and set B(z) so that the impulse response is
```
   h(n) = r^n cos(om1 n).
```
You can consult a table of Z-transforms!

How should the gain be set to ensure the impulse response does not exceed the maximum allowed value of 2^15-1 ?

• Write a version of filter_16.py using 8 bits/sample.
You may use paInt8 as the PyAudio format (signed data).

• How to design two second-order filters (with same resonant frequency om1) so that the rise-time and decay-time of the impulse response are different?
Write a version of filter_16.py that applies two second-order filters in cascade, for the purpose of generating an impulse response with short rise-time and slow decay-time.