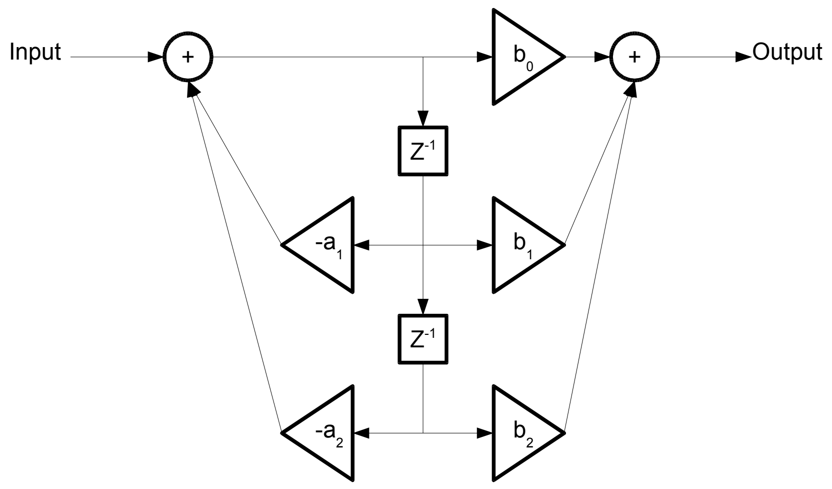
Second-order IIR filters in BasicDSP

by Niels A. Moseley PE1OIT

Second-order filters offer a sharper frequency response compared to the first-order filter. First order filters, which have a 6 dB per octave slope, are often used for quick-and-dirty smoothing. If you want a steeper slope, such as 12 dB per octave or higher, you need a higher order filter.

A simple way to achieve a higher order is to put two first-order filters in series. This is similar to putting two analog RC filters in series. Unfortunately, this isn't the best filter you can make, given the number of components. Better filters (fewer components) can be made by including an inductor L, which leads to an RLC filter.

The digital equivalent of an RLC filter is the second-order IIR filter:



The 'Z^{-1}' blocks are a 1-sample delay. The following BasicDSP script implements this filter:

```
temp = f_in - a1*state1 - a2*state2
f_out = b0*temp + b1*state1 + b2*state2
state2 = state1
state1 = temp
```

,where 'state1' and 'state2' are the values stored in the 1-sample delay blocks. The input and output of the filter are 'f_in' and 'f_out', respectively.

For lowpass filters, the filter coefficients (a1, a2, b0, b1 and b2) can be calculated using the following formulas [1]:

```
pi = 3.1415927
w0 = 2*pi*f0/FS; (where f0 is the desired cutoff frequency, and FS is the sample rate)
alpha = sin(w0)/(2*Q); (where Q is typically 0.5*sqrt(2)=0.707)
b0 = (1-cos(w0))/(1+alpha)
b1 = 2*b0
b2 = b0
a1 = -2*cos(w0)/(1+alpha)
a2 = (1-alpha)/(1+alpha)
```

Combining the filter and the calculating of the filter coefficients, we can make a second order filter where the cutoff frequency f0 is settable through slider1:

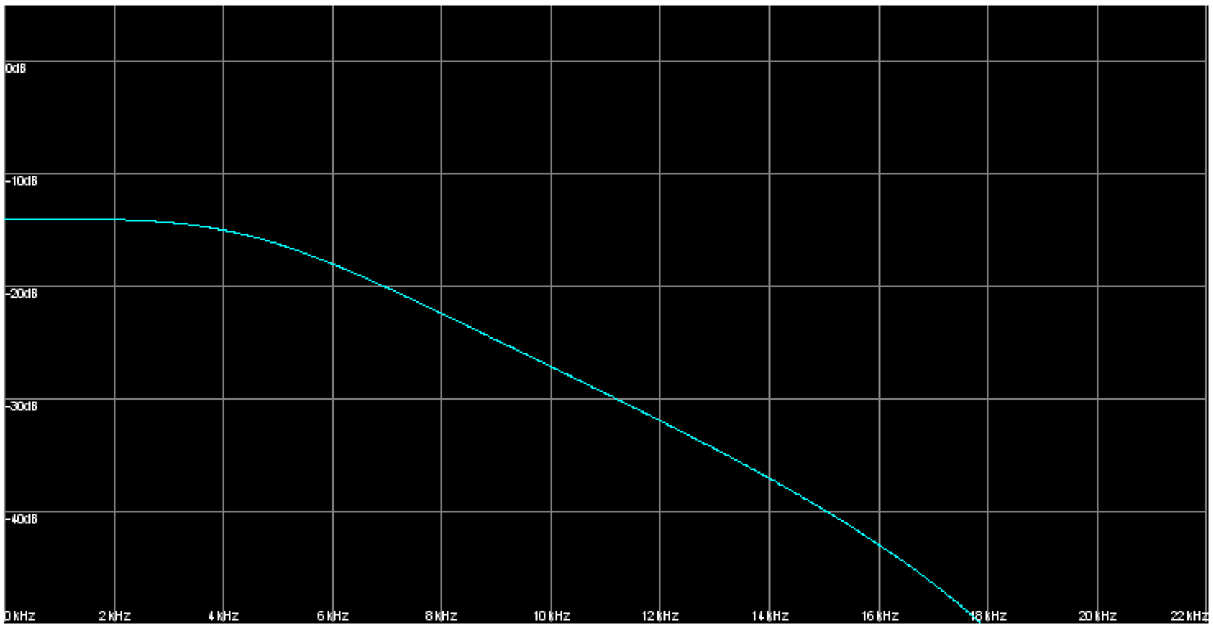
```
samplerate = 44100;
f_in = in;

; calculate filter coefficients
w0 = 3.1415927*slider1
alpha = sin(w0)/(2.0*0.707)
b0 = (1-cos(w0))/(1+alpha)
b1 = 2*b0
b2 = b0
a1 = -2*cos(w0)/(1+alpha)
a2 = (1-alpha)/(1+alpha)

; calculate filter
temp = f_in - a1*state1 - a2*state2
f_out = b0*temp + b1*state1 + b2*state2
state2 = state1
state1 = temp

out = f_out * 0.1
```

Here's a screenshot of the filter spectrum (input = impulse & slider1 = 0.2476):



References:

[1] Filter design equations http://www.musicdsp.org/files/Audio-EQ-Cookbook.txt