Downsampling with LP filter

This algorithm performs downsampling with anti-aliasing filtering. It has two steps. First, the signal is filtered at one half of the new sampling rate. Then the signal gets decimated using linear interpolation.

1. Filtering

The filter is the 6-th order low pass Butterworth filter. The filter coefficients are computed on the fly but for improved performance they can be pre-computed for each corner frequency and kept in a table.

$$
\begin{aligned}
LP6(X, ASR, f_0) := \;& XT_0 \leftarrow 0 \\
& XT_1 \leftarrow 0 \\
& V_0 \leftarrow 0 \\
& V_1 \leftarrow 0 \\
& \text{for } i \in 0..\,\text{last}(X) \\
& \quad XT_{i+2} \leftarrow X_i \\
& \alpha_0 \leftarrow -0.51764 \\
& \alpha_1 \leftarrow -1.41421 \\
& \alpha_2 \leftarrow -1.93185 \\
& om \leftarrow \tan\left(\pi \cdot \frac{f_0}{ASR}\right) \\
& \text{for } j \in 0..\,2 \\
& \quad denom \leftarrow om^2 - \alpha_j \cdot om + 1 \\
& \quad b0_j \leftarrow \frac{om^2}{denom} \\
& \quad b1_j \leftarrow 2 \cdot b0_j \\
& \quad b2_j \leftarrow b0_j \\
& \quad a1_j \leftarrow \frac{2 \cdot (om^2 - 1)}{denom} \\
& \quad a2_j \leftarrow \frac{om^2 + \alpha_j \cdot om + 1}{denom} \\
& \quad \text{for } k \in 2..\,\text{last}(XT) \\
& \quad\quad V_k \leftarrow (b0_j \cdot XT_k + b1_j \cdot XT_{k-1} + b2_j \cdot XT_{k-2}) - (a1_j \cdot V_{k-1} + a2_j \cdot V_{k-2}) \\
& \quad\quad X_{k-2} \leftarrow V_k \\
& \quad XT \leftarrow V \\
& X
\end{aligned}
$$

X – input data array

ASR – acquisition sampling rate, in samples per second, of the data acquisition device (ASR = 48000 for the sound card)

f_0 – filter corner frequency in Hz. f_0 = SR/2, where SR (in samples per second) is the sampling rate required by the application.

2. Decimation

The signal must be low pass filtered at SR/2 before decimation

$$
\mathrm{downSample}(X, ASR, SR) := \left|
\begin{array}{l}
R \leftarrow \dfrac{ASR}{SR} \\
\mathrm{newLength} \leftarrow \mathrm{floor}\left(\dfrac{\mathrm{length}(X)}{R}\right) \\
\text{for } j \in 0..\mathrm{newLength} - 1 \\
\quad \left|
\begin{array}{l}
LO \leftarrow \mathrm{floor}(j \cdot R) \\
HI \leftarrow \mathrm{ceil}(j \cdot R) \\
V_j \leftarrow X_{LO} + (j \cdot R - LO) \cdot \left(X_{HI} - X_{LO}\right)
\end{array}
\right. \\
V
\end{array}
\right.
$$

This algorithm returns a new array that is shorter than the input array by the ratio R.