SSSP Homework #1

Marco Furio Colombo - Student ID: 10537094
Matteo Pettenò - Student ID: 10868930

1 Implementation

We implemented the Leslie speaker emulation by processing each sample of the input signal through the given block diagram which consists of three main components described in the following subsections.

1.1 Crossover network

In order to separate frequencies above and below the given cutoff frequency ($f_c = 800Hz$), allowing slightly different processing between bass and treble, the signal first goes through a crossover network that is mimicked by a LPF and a HPF. The filters are implemented as fourth order *Butterworth* filters and we compute their coefficients with the help of the MATLAB function ***butter***.
Being the filters a *M-th* order FIR approximation of IIR *Butterworth* filters, the output at a given time instant n can be computed by using the finite difference equations:

$$y_f(n) = \sum_{i=0}^{M} b_i x(n-i) - \sum_{j=1}^{M} a_j y(n-j)$$

where a_j and b_i are the coefficients respectively of the denominator and the numerator of the filter's transfer function and $x(n)$ is the input signal. Thus it is evident that to compute the current output we need to know the last $M+1$ samples of the input signal and the last M samples of the output itself and this is why the sample-by-sample processing is done with the help of two auxiliary rotating buffers that are updated at each iteration with the **circshift** function (the most recent sample is stored in the first position of the buffer).
Although we have not implemented it, we note that a small optimization can be accomplished by using a single buffer for the input signal, common to both filters.

1.2 Modulations

To recreate the characteristic Leslie effect we need to apply both frequency and amplitude modulations. The modulating signals are defined as:

$$m'(n) = M_s \cdot m(n) + M_b, \text{ with } m(n) = \sin(2\pi \cdot freq \cdot t) ,$$

where $freq$ is the rotation speed of the drum and horns. The modulators are in common between the modulation blocks but they are different for the bass and treble bands (the scaler M_s and the bias M_b serve to this purpose). Also, for the treble band, the rotation speed is set 0.1 Hz higher than the bass one, in order to simulate the rotations of two different masses powered by the same engine.

The frequency modulation is implemented for each band with a Spectral Delay Filter (SDF) using the difference equation:

$$y_n = \sum_{i=0}^{N} \binom{N}{i} m'^i(n)[y_f(n-(N-i)) - y(n-i)]$$

where y_f is output of the Butterworth filter and $y(n-i) = 0$ if $i = 0$. The equation clearly shows that we need to know the previous N samples of both the input and output signals to compute the current output, thus we apply the same strategy used for the crossover and we allocate two auxiliary buffers.

The last step is to apply amplitude modulation following the relation:

$$y_{AM}(n) = [1 + \alpha m'(n)]y_n(n) ,$$

where α is a given scaling factor, y_n is the output of the SDF and $m'(n)$ is the value of the modulation signal and y_n is the output of the SDF at the current sample.
The output y is the sequence of the estimated samples $y_{AM}(n)$.

1.3 Results

Lastly we compared the quality of the obtained estimation by computing the *Mean Squared Error* (MSE) between the algorithm output and the ground-truth audio signals (one per rotation speed):

MSE Chorale	MSE Tremolo
$3.113 \cdot e^{-}10$	$3.10989 \cdot e^{-}10$

2 Questions

2.1 Q1 - Vibrato effect alternative approach

An alternative approach to delay-based audio effects is having an interpolation-based delay line, usually placed inside a feedback system.
As displayed in Figure 1, every sample time n the newest input sample $x(n)$ is accepted in the delay line, while the oldest sample is discarded off the right-hand side. That action defines the delay line. The output is shown above the block as an arbitrary tap. Because the arbitrary delay tap is moving smoothly about the tap center, it becomes necessary to interpolate in real time in between the discrete samples of the delay line. The audible effect of the tap movement is vibrato.

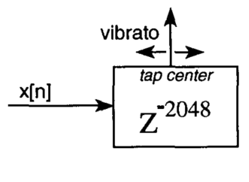


Figure 1: Delay line of length 2048 samples.

More complex schemes that make use of the same block can be designed to achieve other delay-based effects as *Flanger*, *Chorus* and *Doubling*. The scheme in Figure 2 is an industry standard chorus effect circuit with feedback and interpolation. Adjusting its parameters according to Figure 3, it is possible to obtain different delay-based modulation effects, as shown in the table.

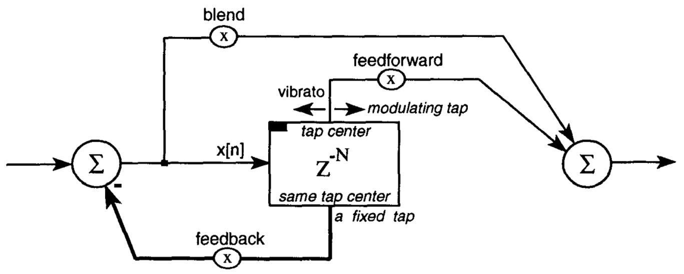
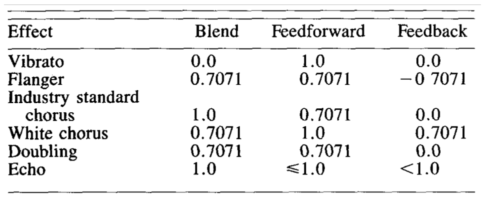


Figure 2: Standard delay-bȧsed modulation effect with using a feedback scheme.



Effect	Blend	Feedforward	Feedback
Vibrato	0.0	1.0	0.0
Flanger	0.7071	0.7071	−0 7071
Industry standard chorus	1.0	0.7071	0.0
White chorus	0.7071	1.0	0.7071
Doubling	0.7071	0.7071	0.0
Echo	1.0	⩽1.0	<1.0

Figure 3: Standard audio effect parameters.

2.2 Q2 - Modulation Stability

SDF can be represented as a cascade of first order all-pass filters, stable iif each filter in the chain is stable. For first order filters, the stability is indeed a concern, $|a_1| \leq 1$ being required for filter stability. The modulator signal has boundaries too, to be more specific must be minor than 1.
Therefore, the all-pass filter is stable if and only if $|m(n)| \leq 1 \ \forall n$, m being the modulator signal.

2.3 Q3 - Group Delays

Having modeled bandlimited discrete-time delay (with $n = tF_s$) as $y(n) = x(n - D)$ and all-pass filter transfer function:

$$A(z) = \frac{a_1 + z^{-1}}{1 + a_1 z^{-1}}, \qquad a_1 < 1.$$

The group delay of a filter is defined as the negative derivative of the filter's phase response $\phi(\omega)$ with respect to frequency:

$$\tau_g(\omega) = -\frac{d\phi(\omega)}{d\omega}.$$

For the first-order all-pass filter the phase response is:

$$\phi(\omega) = -\omega + 2 \arctan\left(\frac{a_1 \sin(\omega)}{1 + a_1 \cos(\omega)}\right).$$

Substituting and deriving we obtain the group delay of the first-order all-pass filter in closed form, expressed as:

$$\tau_g(\omega) = \frac{1 - a_1^2}{1 + 2a_1 \cos(\omega) + a_1^2}.$$

For lower frequencies the group delay is approximately linear with group delay approximately equal to:

$$\tau_g(\omega) = \frac{1 - a_1}{1 + a_1} \quad \text{and follows} \quad a_1 = \frac{1 - D}{1 + D},$$

where D is the delay in term of samples.
The Figure 4 has been generated with the MATLAB script *group_delay.m* included in the delivery folder.

As we can see from the plots, for the given input signal $u(n)$, the delay of the two components ω_1 and ω_2 is never truly equal for the two frequencies.
There are instead regions where the group delay growth is almost constant, namely for frequencies greater than about 10 kHz for positive a_1 coefficients or lower than 10 kHz for negative a_1 coefficients. We also noticed that as the a_1 coefficient get closer to zero, the group delay graph flattens.

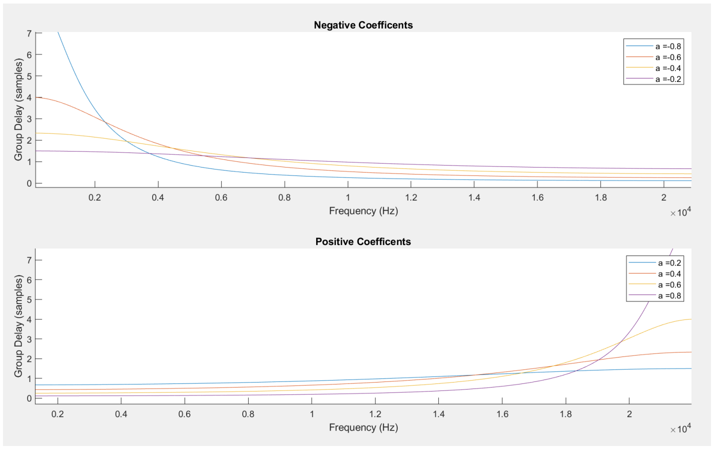


Figure 4: Group Delay

2.4 Q4 - Group Delays

As shown in Figure 5, both magnitude and group delay are influenced by the designed filter order N. What we aim for is the optimal filter, which has magnitude and group delay approximately linear. Knowing that we need to always set $N > \lfloor D \rfloor$, we then tried filters of higher than third order (being the sample delay $D = 3.3$). As expected in our case the optimal filter that is achieved with the filter of fifth order, satisfying the optimality condition:

$$N = 2 \cdot round(D) \quad \text{for N even,}$$
$$N = 2 \cdot \lfloor D \rfloor + 1 \quad \text{for N odd}$$

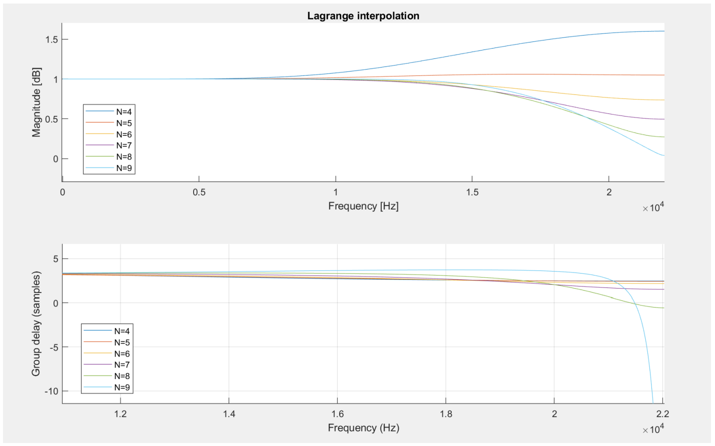


Figure 5: Lagrange Interpolation