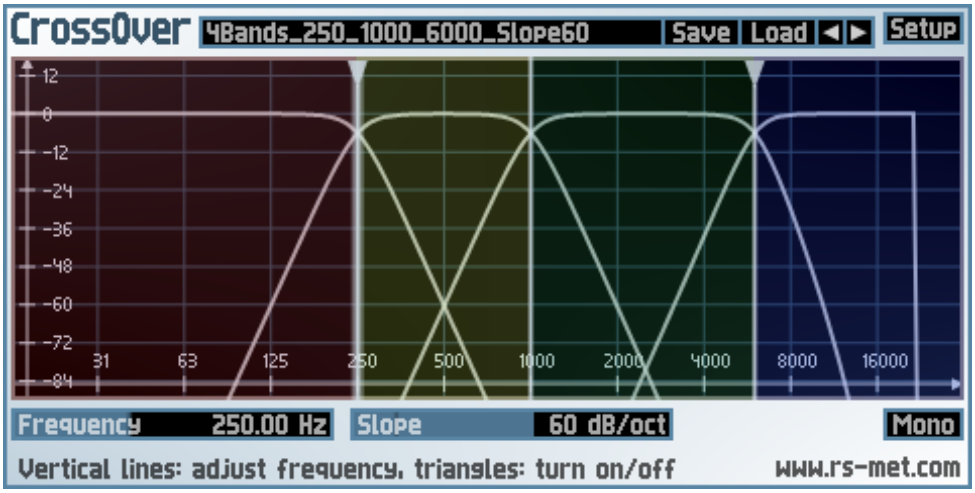
CrossOver - User Manual



What is CrossOver?

CrossOver is a plugin to split an incoming signal into several (at most 4) frequency ranges. It can be used to build your own multiband effects within a modular (sub)host environment. By using only 2 bands, you can also create bass-effects. These are effects which (counterintuitively) keep the low frequency range dry and affect only the high frequency range. The band-splitting is done by complementary pairs of Linkwitz-Riley lowpass/highpass filters, the slope of which is adjustable between 12 and 96 dB/oct. For configurations with more than 2 output bands, the signal is first split into 2 bands (low and high) and then the low or high or both bands is/are split further, thus the splitting is done in a hierarchical manner.

Setting it up

The Frequency-Response Plot: The main area of the GUI is devoted to a frequency response plot which shows the frequency responses (or, more precisely, magnitude-responses) of the individual band outputs. There are always at least 2 bands and at most 4 bands. The vertical lines in the plot can be grabbed and dragged left/right to adjust the crossover frequencies. When there are only 2 bands, only the middle line will be drawn solidly, the other ones will be drawn dashed. You can activate the hierarchical splitting to more bands by clicking on the triangles at the top of the left and right vertical lines (which then become drawn solidly as well). The currently selected vertical line (if any) will be highlighted by drawing it with a kind of glow around it.

Frequency: This slider lets you adjust the crossover frequency of the selected split.

Slope: This slider lets you adjust the slope (in dB/oct) of the selected split.

Mono: In cases where your input signal comes from a mono source, you can activate this switch to save some CPU power. With this switch active, the splitting will not be done for both input channels separately, instead only the left input channel is band-splitted and the results will be copied into the right output channels.

Output Channel Routing:

The plugin has 8 ouput channels (corresponding to 4 stereo output pairs). During operation, the user may change the number of bands into which the incoming signal is split such that - depending on the configuration - not always all 4 output pairs are needed. This calls for some strategy to route the actual number of output bands to the 4 availabe otput pairs. CrossOver uses the following strategy here: the output channel-pairs are always filled up from the first to the last with actual output bands (going from low to high). The unused channel-pairs will feed silence.

Background: Linkwitz-Riley Filters:

Splitting into 2 bands: When designing a filter pair for splitting audio into a low and high frequency range, we want a pair of a complemetary lowpass and highpass filter. We ideally want to have this filter pair satisfy the following conditions:

1. one filter should be lowpass and the other one should be highpass in nature
2. the sum of both filter outputs should be our original signal

Requirement one can be fulfilled by using just about any pair of lowpass/highpass filters tuned to the same cutoff frequency. However, the summed output of some randomly chosen lowpass/highpass pair will in general not be our input signal. Specifically, due to interference between the two signals, we typically see some boost, dip or wiggle around the common cutoff frequency in the magnitude response of the filter-sum. The ideal stated above - to exactly get back to our original signal when summing both filters outputs - is attainable only by use of linear phase FIR filters. Although this is perfectly doable in the digital domain, it is not possible in the analog world. Here we mostly deal with a compromise known as Linkwitz-Riley structure. Whereas it would have been possible to design a digital crossover with FIR-filters fulfilling condition 2 perfectly, this plugin here follows the analog design approach and implements the crossovers by the digital version of the Linkwitz-Riley structure which has the advantage of being more efficient and not introducing latency (which an FIR implementation would do). A Linkwitz-Riley filter consists of two identical Butterworth (IIR) filters in series with their half-power frequency (-3.01 dB) adjusted to the desired crossover frequency. When summing the outputs of a Linkwitz-Riley pair, we don't get our original signal back exactly, but at least the magnitude response of the sum-filter will be flat. That is: the sum of the filter pair yields an allpass filter. This is much better than having to live with strange behaviour in the magnitude response around the crossover frequency and that's why Linkwitz-Riley filters are so popular in the context of analog crossover designs. Because of the series connection of two identical Butterworth filters each of which having a slope of some integer multiple of $6dB/oct$, Linkwitz-Riley filters come with slopes of integer multiples of $12dB/oct$. Because the Butterworth filters have a $-3.01dB$ gain at the crossover frequency, the corresponding Linkwitz-Riley filter will have a $-6.02dB$ gain there. Sometimes, Linkwitz-Riley filters are also called Butterworth-squared filters, because their magnitude response is that of a Butterworth filter squared (putting two identical filters in series yields squaring of the magnitude response).

More than two bands: One approach to split a signal into more than 2 bands is to use a bank of parallel bandpass-filters (and a lowpass and highpass for the bottommost and topmost bands). Here, however, we will discuss another approach: we take the lowpass- and/or highpass output of a 2-way crossover and split that signal further. When doing this with Linkwitz/Riley filters, we unfortunateley loose the desirable property that the sum of the output-channels sums flat - adding up the bands will not yield an allpass filter anymore. Consider the case where we first split the original signal into a low and high band. Adding these two bands produces an allpass filter, as explained before. Now we further split the low band into low-low and low-high. Adding low-low, low-high and high is now not allpass anymore because low-low + low-high does not equal our low-signal from the first splitter-stage. Instead, the addition of low-low and low-high is itself an allpass filtered version of the low-signal from the first splitter stage. In order to make our overall sum flat again, we have to compensate this allpass filtering by applying the exact same allpass filter to our high band as well. Similarly, we would have to insert a compensation allpass on the low band, if we had split the high band further. And when we split both bands further, we need compensation allpasses on both branches. ...and this is exactly what this CrossOver plugin does.