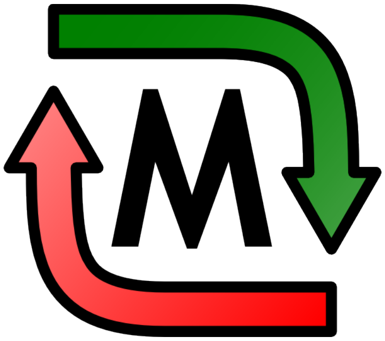
KVR DC'12

DEVELOPER CHALLENGE 2012

VST Plug-in Preset Match

Version 1.0.0



Welcome

Thank you for downloading my entry for the KVR Developer Contest 2012. With **VST Plug-in Preset Match** it is possible to match a VST 2.x plug-in to a given reference VST plug-in for a certain piece of music. With the included ASIO VST plug-in it is also possible to match a VST plug-in to an external hardware device.

! **NOTE:** This tool needs a lot of CPU and especially time for matching. Please don't expect immediate results! !

In order to get the most out of the **VST Plug-in Preset Match**, please spend a few moments reading this brief manual.

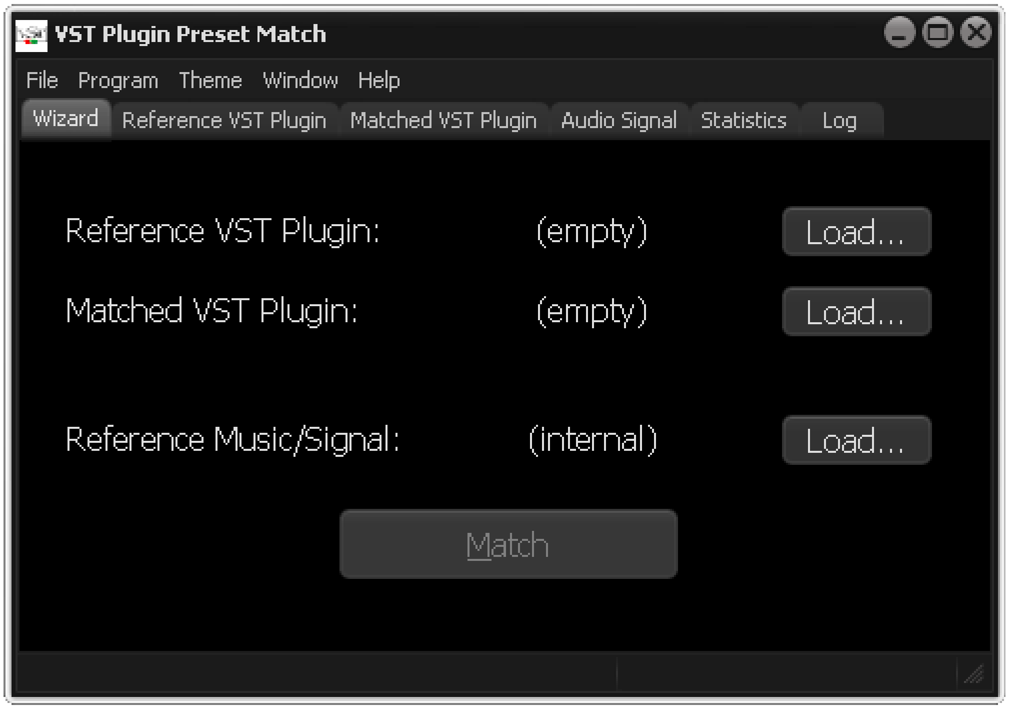
License

The **VST Plug-in Preset Match** has a very simple license:

1. **VST Plug-in Preset Match** is freeware. This means that you are free to use this application in any context. Also you are free to share it on a personal base (ie. give it to friends). However, only the entire unaltered archive, including this document, may be shared. Public redistribution is only allowed on request.
2. Copyright of the code and the **VST Plug-in Preset Match** tool remain property of *Christian-W. Budde*.
3. This tool is provided at no cost; therefore the author *Christian-W. Budde* assume no responsibility for any negative effects that may occur to the end user or the equipment used to run the tool.
4. Magazine editors are welcome to include the tool on cover mount discs or similar media; However, it is mandatory to inform the author *Christian-W. Budde* about this. A copy of the publication is always appreciated, but not expected.
5. The included example VST plug-ins, contained with this tool apply to the same license with the exception, that these VST plug-in may not be used outside this suite in a commercial context.

Usage

The ***VST Plug-in Preset Match*** tool is designed to be as simple as possible. A wizard, which is located on the first tab guides through the different steps to setup and run the tool.



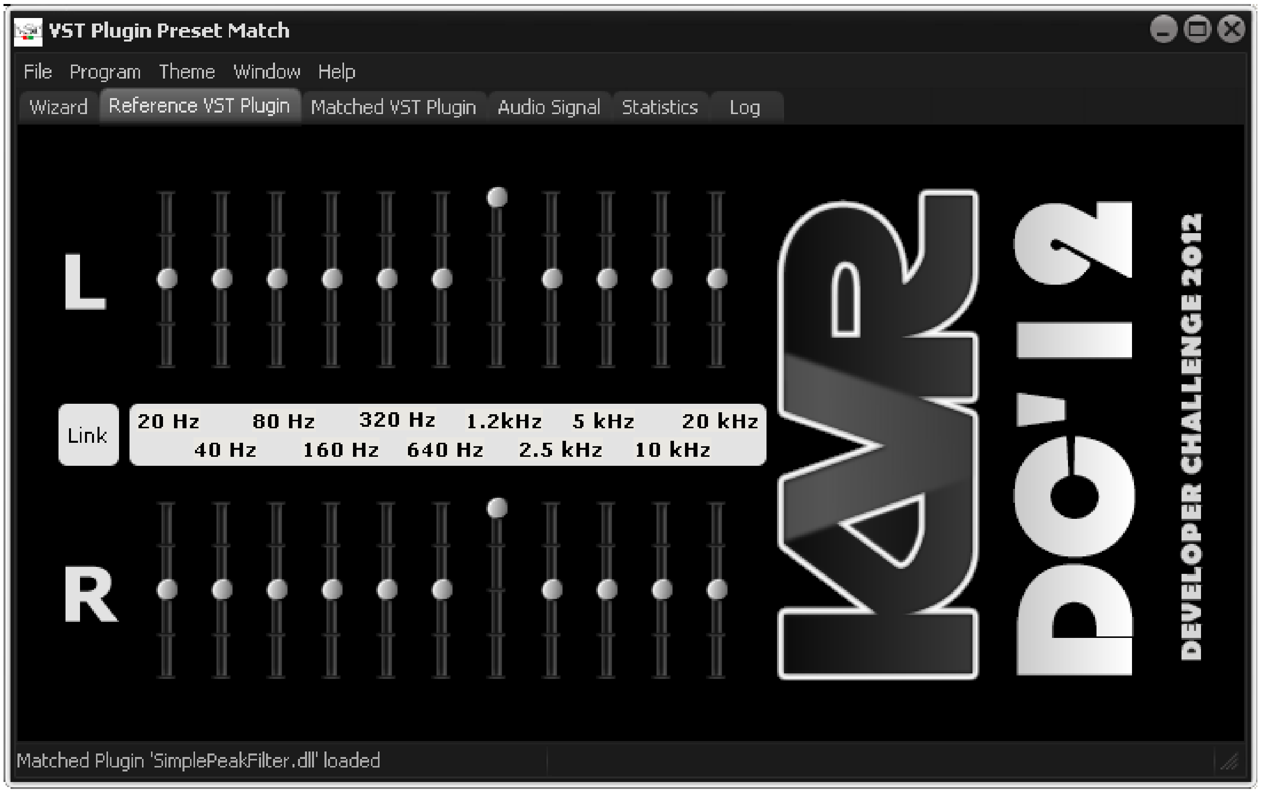
Reference VST Plugin: (empty)
Matched VST Plugin: (empty)
Reference Music/Signal: (internal)

On the wizard tab the reference plug-in and the plug-in, whose preset shall be matched to the reference can be loaded. Furthermore a reference signal can be chosen. By default an internally generated sine sweep will be used.

As plug-in any VST version 2.x effect plug-in can be loaded. Instruments or synths are not yet supported, but might also work if they can act as effect plug-in even without any midi input. Keep in mind, that the tool has been developed for classical effects such as filters (EQs) and dynamics (gate compressor, limiter). Despite some warnings, there are no further limitations, but the matching is never guaranteed.

! Make sure that the selected VST plug-in is suitable before wasting precious energy (CPU cycles) by trying to match apples with oranges! **!**

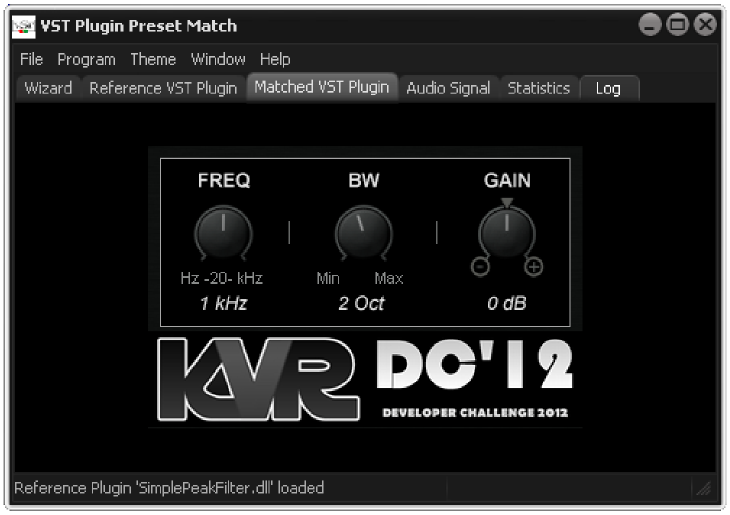
Once the plug-in has been loaded, it can be tweaked on the second tab. Alternatively, any program (or even entire banks) can be loaded from the 'Program' menu item. The recent program (sometimes called preset) is stored as [Plug-in name].fxp as soon as the application is closed.



Matched Plugin 'SimplePeakFilter.dll' loaded

As example the GraphicEQ.dll, which is shipped with the tool has been loaded. The plug-in implements a simple graphic EQ with a band distance of one octave (= roughly twice the previous frequency). To trial the matching, just pick any band and dial a random gain (e.g. maximize the 1.2 kHz band).

Next, switch back to the wizard or use the 'File' menu to load a plug-in that shall be matched to the reference plug-in. In this demonstration we are using the SimplePeakFiler.dll, which implements a single parametric peak filter.



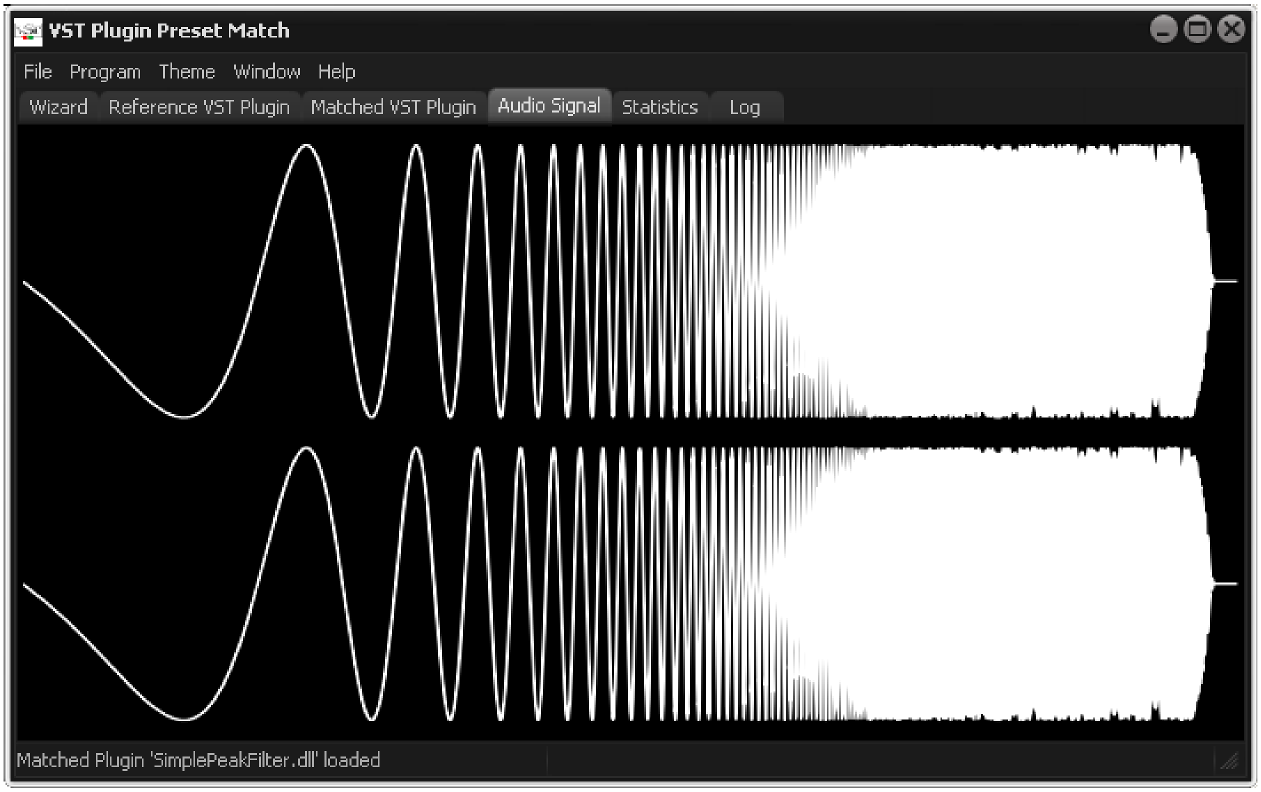
FREQ 1 kHz

BW 2 Oct

GAIN 0 dB

Reference Plugin 'SimplePeakFilter.dll' loaded

Since both selected plug-ins implement LTI systems, the internally generated reference signal can be used for matching. The advantage of the test signal compared to other signals is the fact, that it is short, has a high RMS level and will hardly produce the so-called inter-sample clippings.



Matched Plugin 'SimplePeakFilter.dll' loaded

The sweep is recommended for all LTI systems and all other time invariant systems that only contain a limited amount of non-linearities. This includes filters, EQs and reverb/hall/delay plug-ins.

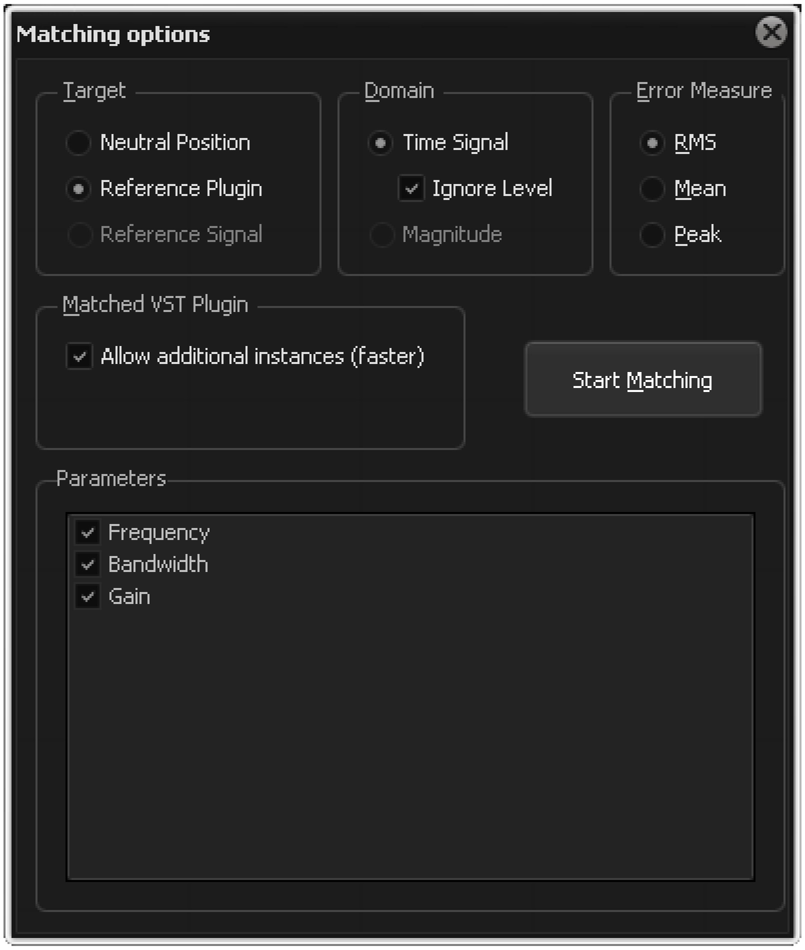
For matching dynamics or distortion effects, it might be more realistic to match the preset for a certain piece of music. If the piece is long enough and contains different content (fast and slow - staccato and legato parts) , it is likely to be valid for other types of music.

! Please note that the longer the reference signal is, the more time and CPU cycles are required to calculate the matching. !

Typically start with a short reference signal. If you are unpleasant with the results, select a different (longer) reference signal.

Once the setup has been configured, the matching can be started. Either hit the [Match] button on the wizard tab or select the 'File'->'Match' from the main menu. The shortcut for matching (start and stop) is F10.

Before the matching process starts, a dialog is shown on which the matching options are shown:



Matching options

Target
○ Neutral Position
● Reference Plugin
○ Reference Signal

Domain
● Time Signal
☑ Ignore Level
○ Magnitude

Error Measure
● RMS
○ Mean
○ Peak

Matched VST Plugin
☑ Allow additional instances (faster)

Parameters
☑ Frequency
☑ Bandwidth
☑ Gain

In the matching options dialog the target for matching can be selected. The default setting is reference plug-in. If no reference plug-in has been specified either the neutral position (reference signal without further processing) or a preprocessed reference signal (if available) can be chosen.

So far, only a time domain method ('Time Signal') is available under domain. It compares every sample from the target with the processed signal (by the plug-in, which should be matched). To improve matching, the level difference can between reference and processed audio can be ignored. In fact, if checked, the level is assumed to be an additional parameter as well. With this switch enabled, polarity issues can also be solved.

! Unfortunately, the time domain method is very prone to phase and time shifts (caused by latencies within the plug-in). !

Due to the above reason, it might be impossible to match a plug-in with the time domain method, even if matching by ear seems to be very simple (since the human ear ignores phase and short latencies). In particular it is not possible with this method to match a linear phase EQ with a minimum phase EQ!

A second method 'Magnitude' has already been developed. However, due to time constraints, it could only be verified to be working in the 32-bit version. If you want to test it in the 64-bit application on your own risk, please double click the 'Domain' box to make it selectable.

Related to the time domain method it is possible to chose the error measure between 'RMS', 'Mean' and 'Peak'. It is recommended to use RMS, but in certain situations other error measures can be useful as well.

To speed up processing, the matching can take place on additional, hidden instances of the plug-in. For now, only a second instance is used, but with multi-thread support, even more instances might be instantiated. In case the plug-in runs on external hardware, such as the UAD and the external hardware runs out of resources, it might be useful to deactivate this option, but in every other case it should be allowed.

The last step before matching actually takes place is to chose the parameters. A list of available parameters is shown on the bottom. Make sure at least one parameter is checked.

! For parameters, which acts as selectors or switches, the matching will be slow (brute force), so exclude these parameters and tweak by hand! !

To start matching, simply press the [Start Matching] button

The dialog can be avoided by either holding down the [Shift] key on the keyboard or by deselecting 'Window'->'Always Show Options Dialog' from the main menu.

Once started, the matching can be stopped at any time by pressing [F10] again or by hitting the (now relabeled) [Stop] button.

Matching

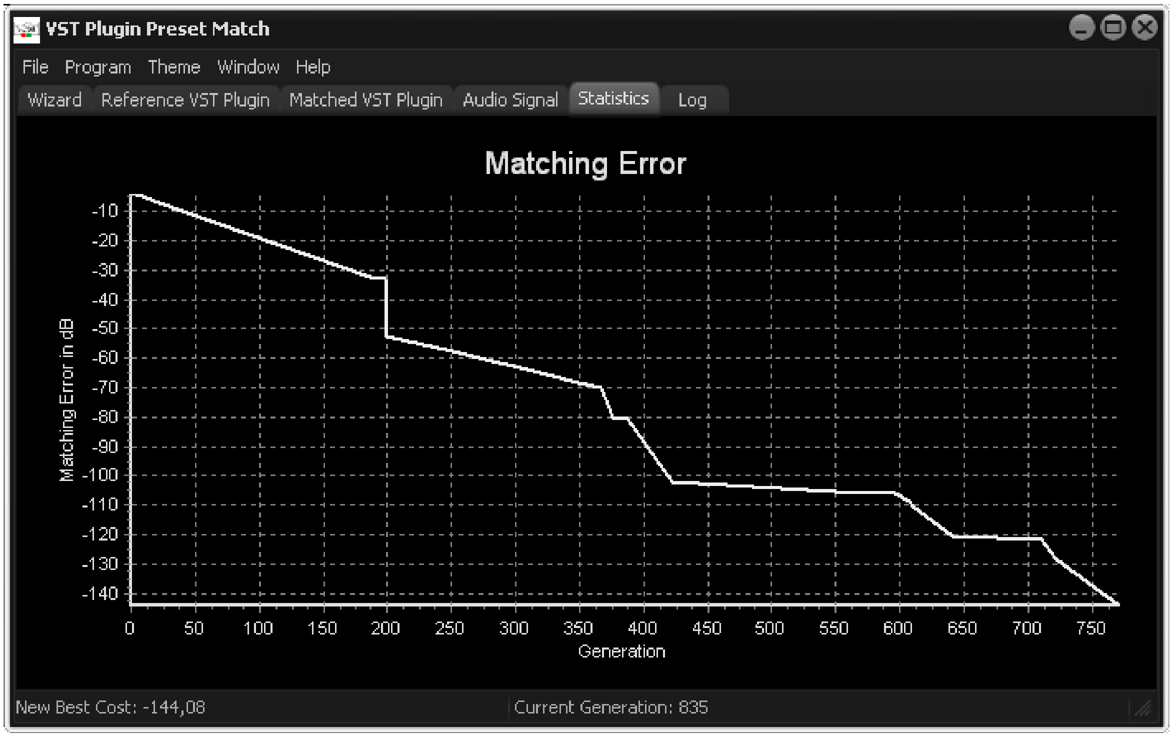
The matching is performed using a differential evolution algorithm. From WIKIPEDIA we know:

> *“In computer science, differential evolution (DE) is a method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Such methods are commonly known as metaheuristics as they make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics such as DE do not guarantee an optimal solution is ever found.”*

In short means we are shuffling our VST parameters dozens of times, select the best matches and shuffle again (dozens of times) based on these results. This is repeated until the user stops the matching process.

The shuffling generates a set of shuffled parameters, called the generation. The population (number of parameters sets) of a generation depends on the number of parameters.

The statistics below show the development of the matching error for 835 generations. As can be seen the error continuously decreases over time (generation). Typically it will stuck sooner or later (only little progress).



New Best Cost: -144,08

Current Generation: 835

With the setup described above, the matching works very well. After 835 generations, the matching error is down to -144,03 dBFS, which means that most sound cards can not make it audible any more (perfect 24-bit soundcards have a dynamic range of 144 dB, but typically far less, due to resistor noise).

If we continue matching a limit at about -400 dB will be reached, which is a limit of the single precision floating point arithmetic used (in particular it's the de-normal prevention).

As can be seen, the selected 1.2 kHz from the graphic EQ is actually a 1.25 kHz peak filter with a bandwidth of 1 octave and 15 dB gain.

Performance

On the Core2Duo reference machine the matching was performed in less than a minute.

The good performance is due to the fact, that the matched plug-in was kept very simple, the reference signal was short and only 3 parameters were matched. Also the plug-in was optimized using handwritten assembler code.

As written above, more parameters will result in longer matching time. In fact, if you double the number of parameters to match it takes 4 times until you have calculated the same number of generations (each generation is bigger). Furthermore the matching process itself will be more complex and thus more time-consuming, as it is assumed that the parameters might interact.

If it is known that the parameters are independent from each other, it can be useful to match all parameters separately.

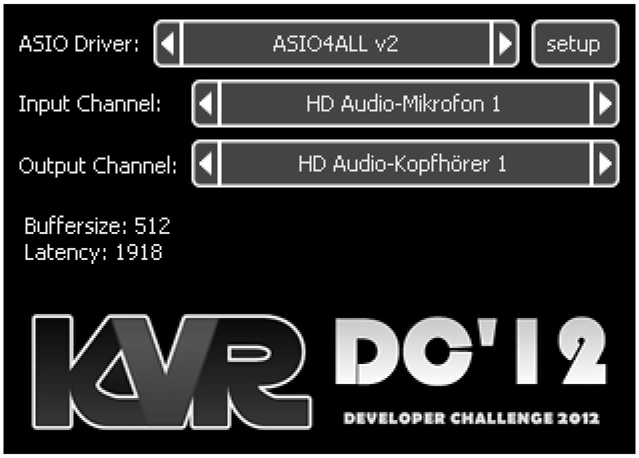
! Before wasting precious CPU-cycles, make sure that matching is possible, by only matching certain parts and by using short signals! **!**

ASIO VST Plug-in

With the help of the ***ASIO VST Plug-in*** it is possible to treat external hardware as if it was a VST plug-in. However, since both interfaces were never designed to work this way, it is likely that it will not run without issues.

In particular the **VST Plug-in Preset Match** is very sensitive about additional latencies. If the latency reported by the ASIO sound card does not match the real latency – in particular due to additional hardware delay caused by A/D and D/A converters – matching especially in time domain only might probably fail.

If the delay is a real issue for matching, it is also possible to capture and preprocess the reference signal externally and provide it instead of at reference VST plug-in.


Driver Select

ASIO Driver: ASIO4ALL v2
Input Channel: HD Audio-Mikrofon 1
Output Channel: HD Audio-Kopfhörer 1
Buffersize: 512
Latency: 1918

On the plug-in's GUI it is possible to select an ASIO driver, input channel and output channel. Since the VST plug-in is mono, it is only possible to capture one channel (no stereo effects can be matched).

Known Issues

The algorithm does not contain magic and thus it's hardly possible to match a compressor plugin with an EQ plugin. If you are unsure if the plugins work, you can load them into the VST Plugin Analyser first to check some basic properties.

For example if a chosen plugin contains a non-linearity (which can be identified by checking the harmonic distortion), the other plugin should be capable of reproducing the non-linearity (distortion) as well. If it does not, the match can never be perfect!

If the plugin contains such non-linearities, it is often also oversampled to reduce aliasing. In the likely case, that the oversampling filters of both plugins are not identical, matching can also never be perfect!

With several 3rd party VST plug-ins there are still matching issues, even if the effect is of the same type. Not seldom, the reason for this is that the developers of the VST plug-ins did not make everything right in the first place. Although the ear excuses many issues easily (especially in regards of the often inaudible phase), technical applications which are evaluating the output based on physical measures, fail easily. If the VST plugin has some latencies (aka delays), which the developers forget to compensate, it's hardly audible in the mix (it maybe sounds like the instrument is some (centi-) meters farther away).

This is also the reason, why linear and minimum phase EQs can hardly be matched (at least not with the time domain method). As the phase is nearly inaudible, both may sound equal, but technically, they are very different. Selecting the magnitude domain will ignore the phase information, which improves matching (makes it possible), but the result might not be perfect.

In some cases, VST plug-in developers forget to implement the reset for the VST plug-in. If used in a DAW, this will hardly be audible, but for this application, it really complicates the matching, as it is inevitable important to always start from scratch, when comparing different preset settings. If the last trial has an influence on the next trial, it's likely that wrong assumptions about the quality of matching will be made.

A workaround for this issue would be to reload the plug-in for every trial or feed it with zeroes. Both will increase the time for matching dramatically.

What does 'cost' mean?

To evaluate the quality of the matching, a cost function is introduced. It represents the physical difference, between the reference and the matched signal (expressed in dB) the A perfect match would mean error measures of below -100 dBFS, but often a lower error measure is also satisfying. Since the error is shown independently from the input signal, it might be overestimated. Not seldom, an error measure of 'only' about -24 dBFS is already inaudible. If the combined error (with with the input signal) is evaluated, the difference may only be little. Since the dB measure is a relative measure, the difference related to a full scale input signal would then be only 0.5 dB, which is most often already inaudible for inexperienced ears.

Feedback / Bug Reports

I am always eager to hear feedback or have bugs reported. The easiest way is to send me a mail to: Christian@aixcoustic.com

Beyond the scope of the KVR Developer Contest it is not yet planned to develop this tool further. However, the code base may be integrated into version 2.0 of the ***VST Plug-in Analyser***, for which I'm currently collecting funds.

If you like this tool and if you want its features to be added to the new ***VST Plug-in Analyser 2.0***, feel free to contact the author under the above email address.

Version History

1.0.0	First release!

Credits

- Programming: Christian W. Budde
- Special Thanks: Swen Müller
- Favourite music artist during work: Earth, Wind & Fire
- Documentation based on a template by Greg Pettit