JSFX SOX Plugins for the Reaper DAW

Dr. Thomas Tensi

November 16, 2019

Contents

1 Introduction

1.1 Overview

The JSFX-SOX software package provides JSFX (Jesusonic) plugins for being used in the Reaper DAW [REAPER]; they implement some of the audio processing effects from SOX. It is also possible to use them in any VST compatible DAW using the freely available ReaJS VST plugin [REAPLUGS] that is able to interpret JSFX files.

SOX ([SOXDOC]) is a command line audio processing tool for Unix, Windows and Mac OS that transforms source audio files in several formats into other audio files. It provides several standard audio effects (like e.g. filters or reverb) in good quality and with a transparent, open-source implementation.

The effects provided here are a complete rewrite of the sox algorithms for producing *(bit-exact) identical* renderings in the DAW. This can easily be checked by rendering some audio externally via SOX and internally with the plugins and subtracting the results. Apart from roundoff errors (SOX often uses 32bit integer processing, while JSFX always uses double floating point processing) the results cancel out with typically a residual noise of less than -120dBFS.

The main motivation for this package is to be able to play around with effects in Reaper and be sure that an external rendering by sox will produce exactly the same results. Although sox does not always provide the "best" effects, it still is a reliable and well-defined audio tool.

Because sox very often uses rich command line options for its effects, not every effect configuration from sox can be transported into the slider oriented GUI for JSFX. E.g. the compander of sox allows the definition of a transfer function having multiple segments. Although the internal engine of the JSFX compander implements exactly the same internal segment logic, the user interface only allows the typical definition of a threshold and a compression ratio (with three segments).

Note also that a spiffy user interface is *not at all* a priority for this package.

The sox effects have been rewritten and restructured for easier maintenance, because in the original source there is some redundancy and unnecessary complexity due to its several contributors. Nevertheless the effects provided here faithfully model the SOX processing.

1.2 Available Effects

The following effects are supported:

allpass:
a biquad allpass filter two-poled with filter frequency and the filter bandwith (in several units)

band:
a biquad bandpass filter with center filter frequency and the filter bandwith (in several units) and an option for unpitched audio

bandpass/bandreject:
a biquad filter for bandpass or bandreject with center filter frequency and the filter bandwith (in several units)

bass/treble:
a biquad filter for boosting or cutting bass or treble with a shelving characteristics with settings for filter frequency and the filter bandwith (in several units)

biquad:
a generic biquad (iir) filter with 6 coefficients b0, b1, b2, a0, a1 and a2

compand:
a compander with attack, decay, input gain shift, threshold and compression and soft knee; this is a reduced version of SOX compand with only a simple transfer function and a combined attack/decay setting

equalizer:
a biquad filter for equalizing with settings for the pole count, the filter frequency and the filter bandwith (in several units)

gain:
a volume changer by *exact* decibels…

lowpass/highpass:
a biquad filter for either lowpass or highpass with settings for the pole count, the filter frequency and the filter bandwith (in several units)

mcompand:
a multiband compander with a Linkwitz-Riley crossover filter and for each band a compander with attack, decay, input gain shift, threshold and compression and soft knee; again the companders only allow a simple transfer function and a combined attack/decay setting

overdrive:
a simple tanh distortion with gain and colour specification

phaser:
a phaser effect with sine or triangle modulation

reverb:
a reverb effect (based on Freeverb) with several parameters for the room (like size and HF damping) as well as a possible predelay

tremolo:
a tremolo effect with sine modulation using a double-sideband suppressed carrier modulation

2 Installation of the JSFX-SOX Effects

The installation in Windows is as follows:

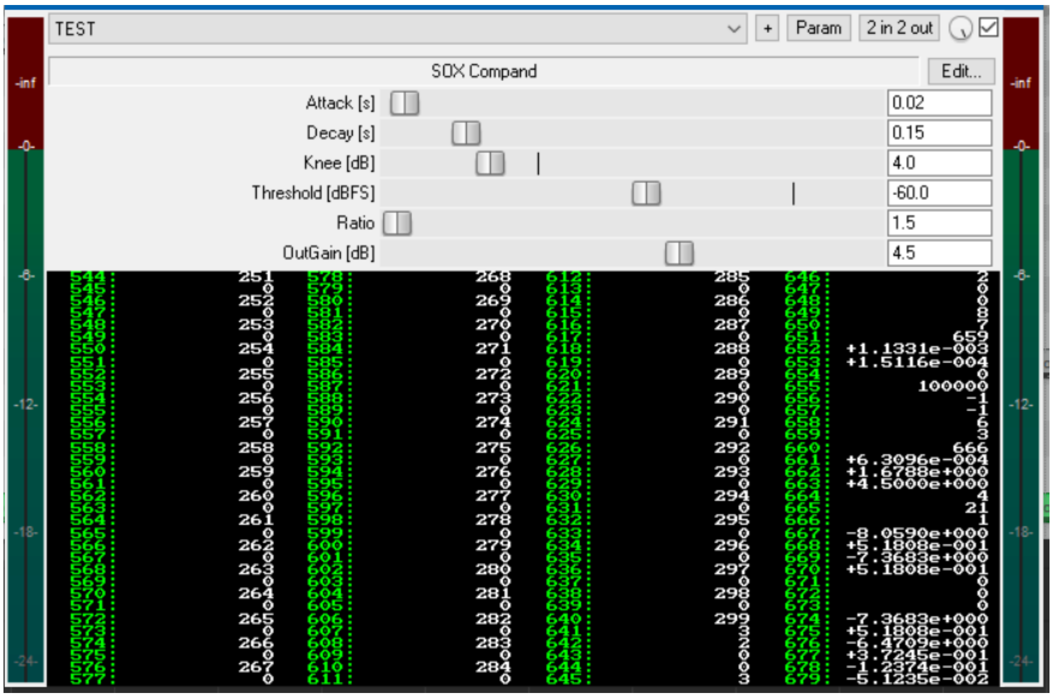
1. Load the JSFX-SOX files from the repository in [JSFXSOX].
2. Close the Reaper application or your other DAW using ReaJS (if open).
3. Make a subdirectory Dr_TT\JSFXSOX in either the Effects directory of the Reaper installation (typically in \Program Files\Reaper\Effects) or — if your DAW uses some different directory for JSFX files — in that directory.
4. Copy over all *.jsfx and *.jsfx-inc files from the distribution src directory into the target directory. If helpful, also add the documentation from the root directory.
5. If helpful, also copy the test files from test to the effects sub-directory Dr_TT\JSFXSOX\Test (see section 4).
6. Restart Reaper (or your other DAW). You should now be able to select the plugins from the JSFX folder (they are all prefixed with "SOX").

Alternatively you can use the ReaPack plugin [REAPACK] and do an automatic install via the index.xml file in the repository [JSFXSOX].

3 Description of the Effects in JSFX-SOX

3.1 General Remarks

As mentioned in the introduction this package provides several audio tools written in JSFX for emulating SOX bit-exactly.



```
Attack [s]         0.02
Decay [s]          0.15
Knee [dB]          4.0
Threshold [dBFS]   -60.0
Ratio              1.5
OutGain [dB]       4.5
```

```
544:   251   578:   268   612:   285   646:             2
545:     0   579:     0   613:     0   647:             0
546:   252   580:   269   614:   286   648:             0
547:     0   581:     0   615:     0   649:             8
548:   253   582:   270   616:   287   650:             7
549:     0   583:     0   617:     0   651:           659
550:   254   584:   271   618:   288   652: +1.1331e-003
551:     0   585:     0   619:     0   653: +1.5116e-004
552:   255   586:   272   620:   289   654:             0
553:     0   587:     0   621:     0   655:        100000
554:   256   588:   273   622:   290   656:            -1
555:     0   589:     0   623:     0   657:            -1
556:   257   590:   274   624:   291   658:             6
557:     0   591:     0   625:     0   659:             3
558:   258   592:   275   626:   292   660:           666
559:     0   593:     0   627:     0   661: +6.3096e-004
560:   259   594:   276   628:   293   662: +1.6788e+000
561:     0   595:     0   629:     0   663: +4.5000e+000
562:   260   596:   277   630:   294   664:             4
563:     0   597:     0   631:     0   665:            21
564:   261   598:   278   632:   295   666:             1
565:     0   599:     0   633:     0   667: -8.0590e+000
566:   262   600:   279   634:   296   668: +5.1808e-001
567:     0   601:     0   635:     0   669: -7.3683e+000
568:   263   602:   280   636:   297   670: +5.1808e-001
569:     0   603:     0   637:     0   671:             0
570:   264   604:   281   638:   298   672:             0
571:     0   605:     0   639:     0   673:             0
572:   265   606:   282   640:   299   674: -7.3683e+000
573:     0   607:     0   641:     3   675: +5.1808e-001
574:   266   608:   283   642:     2   676: -6.4709e+000
575:     0   609:     0   643:     0   677: +3.7245e-001
576:   267   610:   284   644:     0   678: -1.2374e-001
577:     0   611:     0   645:     3   679: -5.1235e-002
```

Figure 1: Memory View for JSFX-SOX Effects in Reaper

This goal is reached up to a certain precision (of about -120dBFS), because SOX often uses 32bit integer processing while JSFX-SOX uses 64bit float processing.

Where noted in the following description, some simplifications have been done to take care of the limited user interface and also some parameters were omitted.

Note again that the focus of this toolset is the faithful reimplementation and somehow a redesign of SOX; a spiffy user interface is *not at all* a priority in this project.

For the same reasons none of the effects of JSFX-SOX displays anything; they just process audio parametrized by their slider settings.

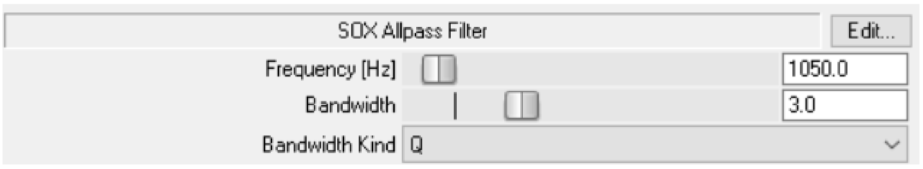
Nevertheless for debugging purposes, some textual display is supported via keypresses in the effects window as follows:

- Internal effect data can be displayed by pressing “d”(ata).
- The table of allocated strings is shown via key “s”(trings).
- The local memory is displayed via “m”(emory) (see figure 1).
- All those displays scale automatically and are scrollable via “+” and “-” and reset via “0”.

- The display of the effect (if any) is shown via “e”(ffect) which is the default (and — as mentioned before — empty).

All effects of JSFX-SOX are discussed in alphabetical order in the following chapter. Note that the effects description is mostly taken from the SOX documentation [SOXDOC] except for specifics in the JSFX-SOX effects.

3.2 SOX Allpass Filter



Frequency [Hz] 1050.0
Bandwidth 3.0
Bandwidth Kind Q

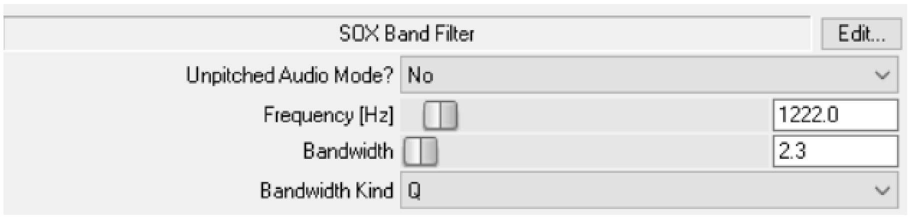
Figure 2: JSFX Panel for SOX Allpass

Parameter	Description	Unit
frequency	the center frequency of the filter	Hz
bandwidth	the bandwidth modulus of the filter	—
bandwidthKind	the bandwidth unit of the filter	frequency / octaves / quality / butterworth

This effect is a two-pole all-pass filter with frequency as center frequency and filter-width bandWidth with unit bandwidthKind. The bandwidth kinds are a relative *frequency*, a specification of *octaves*, the filter *quality* or the *butterworth* quality (with fixed quality $q = \sqrt{2}/2$).

An all-pass filter changes the audio’s frequency-to-phase relationship without changing its frequency-to-amplitude relationship. The detailed filter description can be found in [RBJFILT].

3.3 SOX Band Filter



Unpitched Audio Mode? No
Frequency [Hz] 1222.0
Bandwidth 2.3
Bandwidth Kind Q

Figure 3: JSFX Panel for SOX Band

Parameter	Description	Unit
usesUnpitchedAudio	flag to tell whether special processing for unpitched audio is applied	Boolean
frequency	the center frequency of the filter	Hz
bandwidth	the bandwidth modulus of the filter	—
bandwidthKind	the bandwidth unit of the filter	frequency / octaves / quality / butterworth

This effect is a band-pass filter. The frequency response drops logarithmically around frequency, the bandWidth and bandwidthKind parameters gives the slope of the drop; frequencies at *frequency+width* and *frequency-width* will be half of their original amplitudes. The effect defaults to a mode that is oriented to pitched audio, i.e. voice, singing, or instrumental music.

When the option useUnpitchedAudio is set, an alternate mode for un-pitched audio (e.g. percussion) is applied. Note that this option introduces a power-gain of about 11dB in the filter, so beware of output clipping; the option introduces noise in the shape of the filter, i.e. peaking at the center frequency and settling around it.

3.4 SOX Bandpass and Bandreject Filter

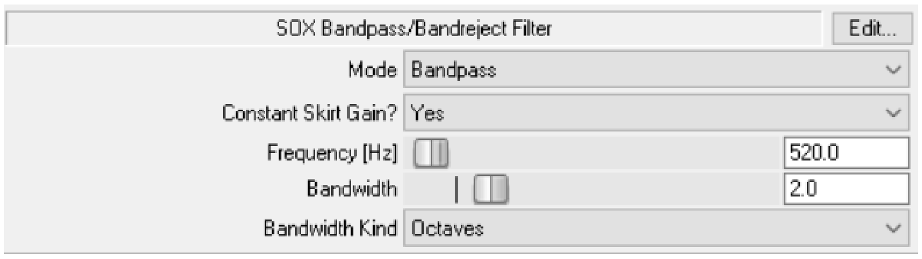

SOX Bandpass/Bandreject Filter
Edit...
Mode: Bandpass
Constant Skirt Gain?: Yes
Frequency [Hz]: 520.0
Bandwidth: 2.0
Bandwidth Kind: Octaves

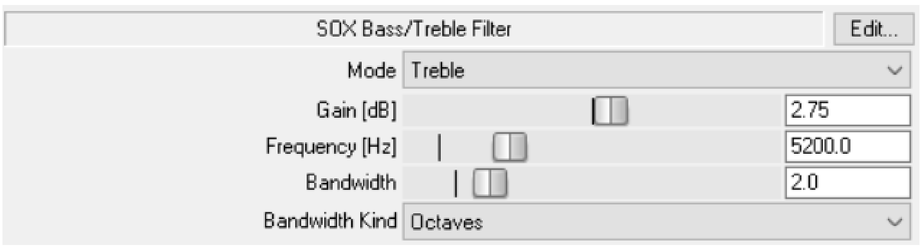
Figure 4: JSFX Panel for SOX Bandpass

Parameter	Description	Unit
mode	selects between filter modes	bandpass / bandreject
constantSkirtGain	flag to tell whether a constant skirt gain is applied	Boolean
frequency	the center frequency of the filter	Hz
bandwidth	the bandwidth modulus of the filter	—
bandwidthKind	the bandwidth unit of the filter	frequency / octaves / quality / butterworth / slope

By selecting the appropriate mode, those effects are either a two-pole Butterworth band-pass or band-reject filter with frequency as central frequency, and (3dB-point) band-width given by bandWidth and bandwidthKind. The constantSkirtGain option applies only to a bandpass and selects a constant skirt gain (peak gain = Q) instead of the default, which is a constant 0dB peak gain. The filters roll off at 6dB per octave (20dB per decade).

The detailed filter description can be found in [RBJFILT].

3.5 SOX Bass and Treble Filter



SOX Bass/Treble Filter

Mode: Treble

Gain [dB]: 2.75

Frequency [Hz]: 5200.0

Bandwidth: 2.0

Bandwidth Kind: Octaves

Figure 5: JSFX Panel for SOX Treble

Parameter	Description	Unit
mode	selects between filter modes	bass / treble
gain	gain of filter at 0Hz (for bass) or 22kHz (for treble)	dB
frequency	the center frequency of the filter	Hz
bandwidth	the bandwidth modulus of the filter	—
bandwidthKind	the bandwidth unit of the filter	frequency / octaves / quality / butterworth

By selecting the appropriate mode, this effect boosts or cuts either the bass (lower) or treble (upper) frequencies of the audio using a two-pole shelving filter with a response similar to that of a standard hi-fi's tone-controls. This is also known as shelving equalisation (EQ).

The parameters are as follows:

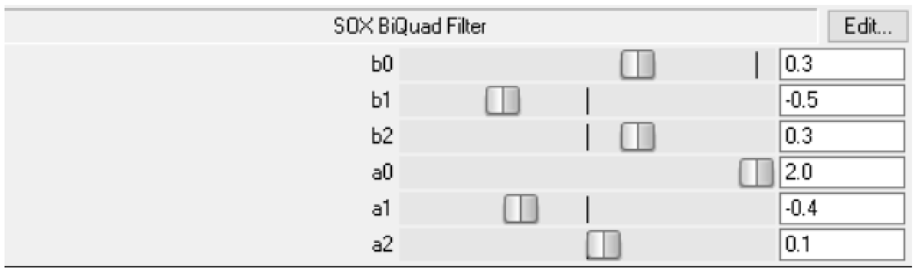
- gain gives the gain at 0Hz (for bass), or whichever is the lower of 22kHz and the Nyquist frequency (for treble). Its useful range is about -20 (for a large cut) to +20 (for a large boost). Beware of clipping when using a positive gain.

- frequency sets the filter's central frequency and can be used to extend or reduce the frequency range to be boosted or cut.

- The band-width given by parameters bandWidth and bandwidthKind determines how steep is the filter's shelf transition. In addition to the common width specification methods described above, “slope” may be used. The useful range of slope is about 0.3, for a gentle slope, to 1 (the maximum), for a steep slope.

The detailed filter description can be found in [RBJFILT].

3.6 SOX Biquad Filter



b0 0.3
b1 -0.5
b2 0.3
a0 2.0
a1 -0.4
a2 0.1

Figure 6: JSFX Panel for SOX Biquad

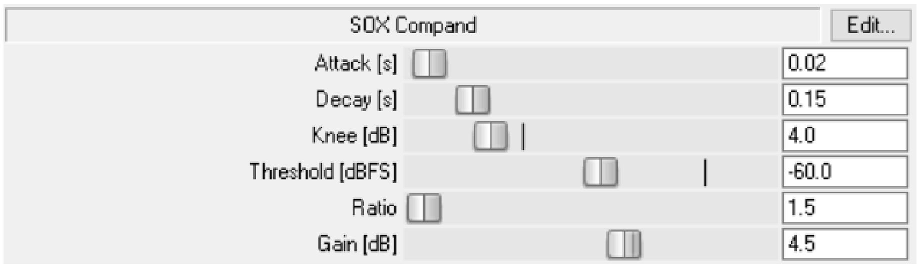
Parameter	Description	Unit
b0	coefficient for x_n	—
b1	coefficient for x_{n-1}	—
b2	coefficient for x_{n-2}	—
a0	coefficient for y_n	—
a1	coefficient for y_{n-1}	—
a2	coefficient for y_{n-2}	—

This effect is a biquad IIR filter with the given coefficients (see [DBIQFILT]). It implements the (direct form) function

$$y_n = \sum_{i=0}^{2} b_i x_{n-i} - \sum_{i=1}^{2} a_i y_{n-i}$$

and is the basis for the other biquad filters (like e.g. the "SOX Equalizer").

3.7 SOX Compand



Attack [s] 0.02
Decay [s] 0.15
Knee [dB] 4.0
Threshold [dBFS] -60.0
Ratio 1.5
Gain [dB] 4.5

Figure 7: JSFX Panel for SOX Compand

Parameter	Description	Unit
attack	the attack time of the compander	s
decay	the decay time of the compander	s
knee	the rounding of the corners in the transfer function	dB
threshold	the threshold of the compander	dBFS
ratio	the compression factor of the compander	—
gain	the compander gain before processing	dB

This effect implements a compander to compress or expand the dynamic range of the audio.

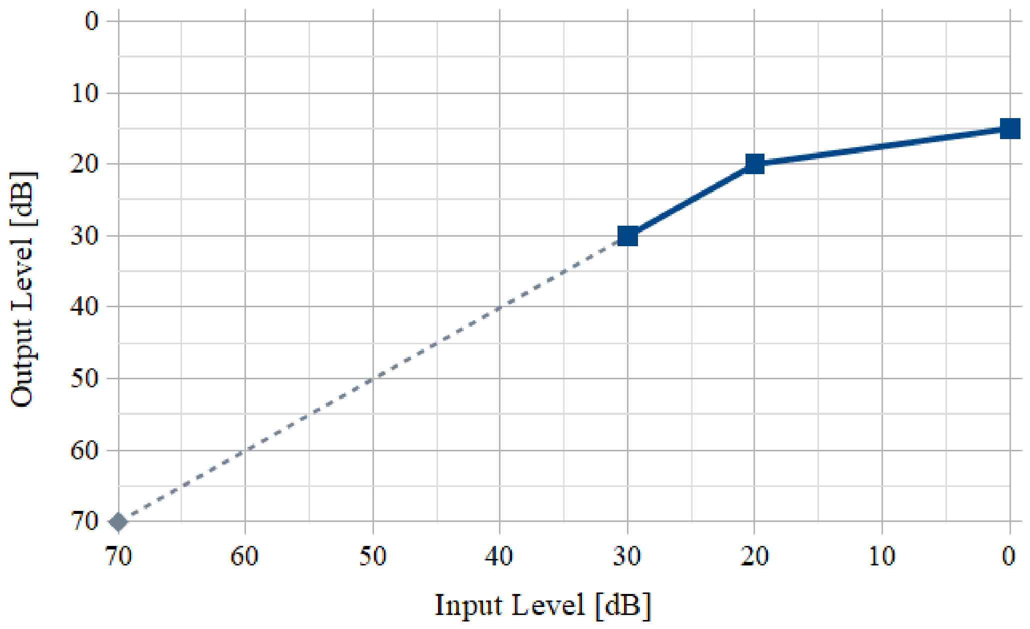


Figure 8: Example Transfer Function (Threshold 20dBFS, Ratio 4:1)

The parameters attack and decay (in seconds) determine the time over which the instantaneous level of the input signal is averaged to determine its volume; attacks refer to increases in volume and decays refer to decreases. For most situations, the attack time (response to the music getting louder) should be shorter than the decay time because the human ear is more sensitive to sudden loud music than sudden soft music. Typical values are 0.3s for attack and 0.8s for decay.

The transfer function of the compander is given by parameters threshold, ratio and knee. The compander leaves the original level unchanged, when it is below threshold and compresses it by ratio beyond this threshold. So e.g. for a threshold of 20dBFS, a knee of 0dB and a ratio of 4:1 the transfer function is a graph shown in figure 8. Note that for technical reasons SOX uses a linear lead-in segment with size 10dB below threshold value.

If the parameter knee is greater than 0, the corner points of the transfer function will be rounded by that amount.

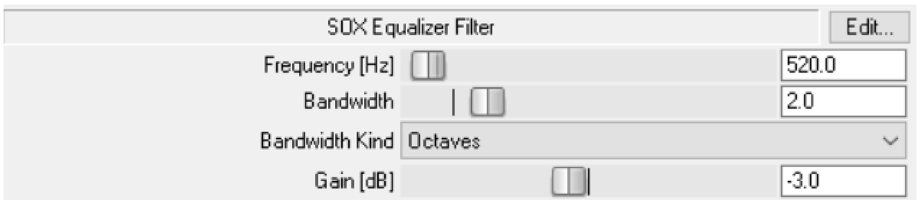
The parameter gain is an additional gain in dB to be applied at all points on the transfer function and allows easy adjustment of the overall gain.

Restriction: Only one overall pair of attack/decay parameters may be specified (where SOX allows one pair per channel).This is in principle supported by the effects engine of JSFX-SOX, but not supported in the current user interface.

Restriction: The original SOX allows an arbitrary multi-segmented transfer function. This is in principle supported by the effects engine of JSFX-SOX, but not supported in the current user interface.

Restriction: There is no delay parameter for delayed compansion.

3.8 SOX Equalizer Filter



Frequency [Hz] 520.0
Bandwidth 2.0
Bandwidth Kind Octaves
Gain [dB] -3.0

Figure 9: JSFX Panel for SOX Equalizer

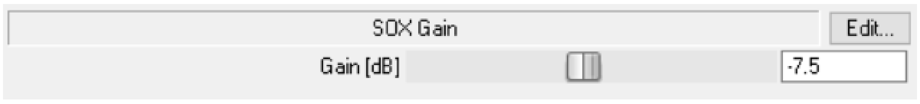
Parameter	Description	Unit
frequency	the 3dB point frequency of the filter	Hz
bandwidth	the bandwidth modulus of the filter	—
bandwidthKind	the bandwidth unit of the filter	frequency / octaves / quality / butterworth
gain	gain of filter at frequency	dB

This effect implements a two-pole peaking equalisation (EQ) filter. With this filter, the signal-level at and around a selected frequency can be increased or decreased, whilst (unlike band-pass and band-reject filters) that at all other frequencies is unchanged.

The parameter frequency gives the filter's central frequency in Hz, parameters bandWidth and bandwidthKind the bandwidth and gain the required amplification or attenuation in decibels. Beware of clipping when using a positive gain.

The filter is described in detail in [RBJFILT].

3.9 SOX Gain



Gain [dB] -7.5

Figure 10: JSFX Panel for SOX Gain

Parameter	Description	Unit
gain	the amplification or attenuation factor	dB

This effect is an amplifier or attenuator for the audio signal with a single gain parameter in decibels. The gain factor applies to all channels identically.

Nothing special, but note that the calculation is exact, hence a gain of -6dB does *not* halve the signal (but a gain of -6.0206dB does quite well).

3.10 SOX Highpass and Lowpass Filter

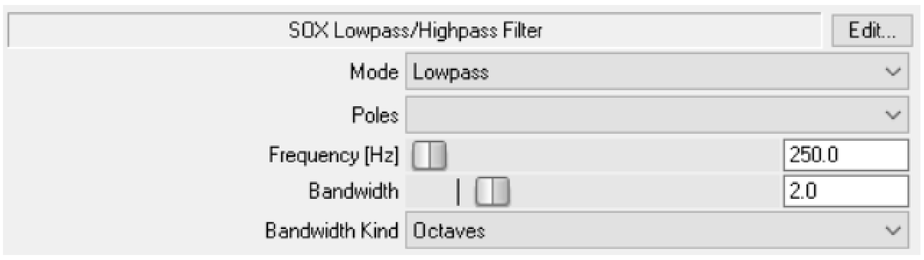

SOX Lowpass/Highpass Filter
Edit...
Mode Lowpass
Poles
Frequency [Hz] 250.0
Bandwidth 2.0
Bandwidth Kind Octaves

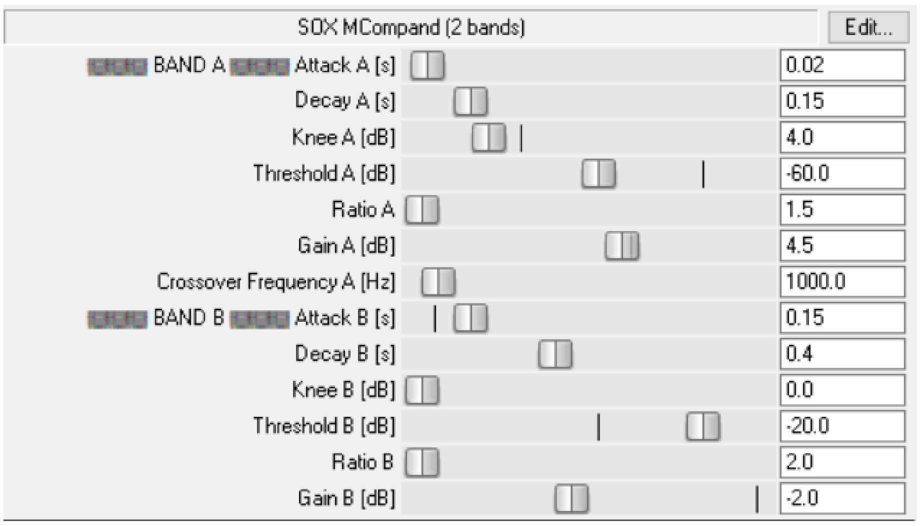
Figure 11: JSFX Panel for SOX Highpass

Parameter	Description	Unit
mode	selects between filter modes	highpass / lowpass
poles	selects between single and double pole filter	single/double
frequency	the 3dB point frequency of the filter	Hz
bandwidth	the bandwidth modulus of the filter	—
bandwidthKind	the bandwidth unit of the filter	frequency / octaves / quality / butterworth

By selecting the appropriate mode, this effect is either a high-pass or low-pass filter with a 3dB point frequency. Depending on poles the filter can be either single-pole or double-pole. The parameters bandWidth and bandwidthKind apply only to double-pole filters; a Butterworth response is given by butterworth selection or by a q of 0.707. The filters roll off at 6dB per pole per octave (20dB per pole per decade).

The double-pole filters are described in detail in [RBJFILT].

3.11 SOX MCompand



SOX MCompand (2 bands) Edit...

BAND A Attack A [s]	0.02
Decay A [s]	0.15
Knee A [dB]	4.0
Threshold A [dB]	-60.0
Ratio A	1.5
Gain A [dB]	4.5
Crossover Frequency A [Hz]	1000.0
BAND B Attack B [s]	0.15
Decay B [s]	0.4
Knee B [dB]	0.0
Threshold B [dB]	-20.0
Ratio B	2.0
Gain B [dB]	-2.0

Figure 12: JSFX Panel for SOX MCompand

Parameter	Description	Unit
for each band		
attack	the attack time of the compander band	s
decay	the decay time of the compander band	s
knee	the rounding of the corners in the transfer function	dB
threshold	the threshold of the compander band	dBFS
ratio	the compression factor of the compander band	—
gain	the compander band gain before processing	dB
topFrequency	the compander band top frequency (for all but the last band)	Hz

The multi-band compander is similar to the single-band compander but the audio is first divided into bands using Linkwitz-Riley cross-over filters and a separately specifiable compander run on each band (see the compand effect in 3.7 for the definition of its parameters).

Restriction: Only one overall pair of attack/decay parameters may be specified per band (where SOX allows one pair per channel).This is in principle supported by the effects engine of JSFX-SOX, but not supported in the current user interface.

Restriction: The original SOX allows an arbitrary multi-segmented transfer function. This is in principle supported by the effects engine of JSFX-SOX, but not supported in the current user interface.

Restriction: There is no delay parameter for delayed compansion.

Restriction: It is not possible to change the number of bands, but one has to select one of several multi-band companders with two to five bands.

3.12 SOX Overdrive

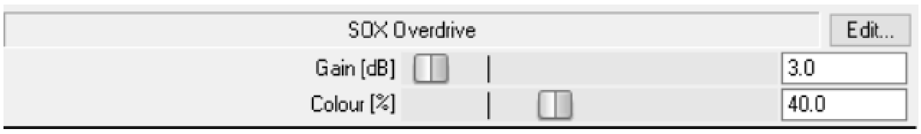

SOX Overdrive
Edit...
Gain [dB] 3.0
Colour [%] 40.0

Figure 13: JSFX Panel for SOX Overdrive

Parameter	Description	Unit
gain	the overdrive gain before processing	dB
colour	percentage for the amount of even harmonic content in output	—

This effect implements an tanh overdrive. gain gives the input gain in decibels, The parameter colour controls the amount of even harmonic content in the over-driven output.

3.13 SOX Phaser

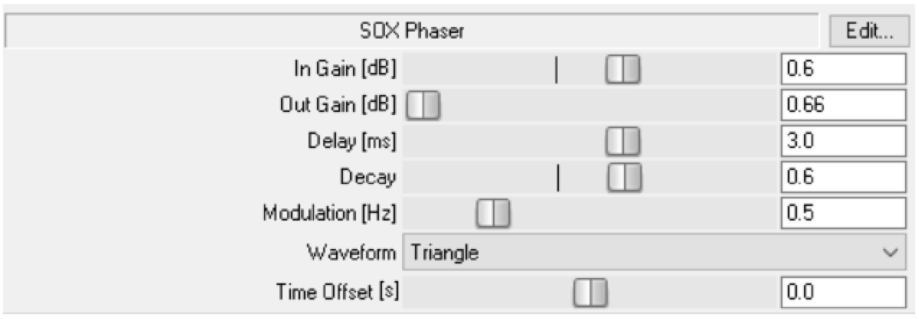

SOX Phaser
Edit...
In Gain [dB] 0.6
Out Gain [dB] 0.66
Delay [ms] 3.0
Decay 0.6
Modulation [Hz] 0.5
Waveform Triangle
Time Offset [s] 0.0

Figure 14: JSFX Panel for SOX Phaser
dB units for gains and seconds unit for time offset

Parameter	Description	Unit
inGain	the gain before processing	dB
outGain	the gain applied after processing	dB
delay	the predelay of the effect	ms
decay	the decay factor of the phaser	—
frequency	the phaser modulation frequency	Hz
waveform	the modulation waveform	sine / triangle
timeOffset	the point in project time where modulation is at phase 0° (see 3.16)	s

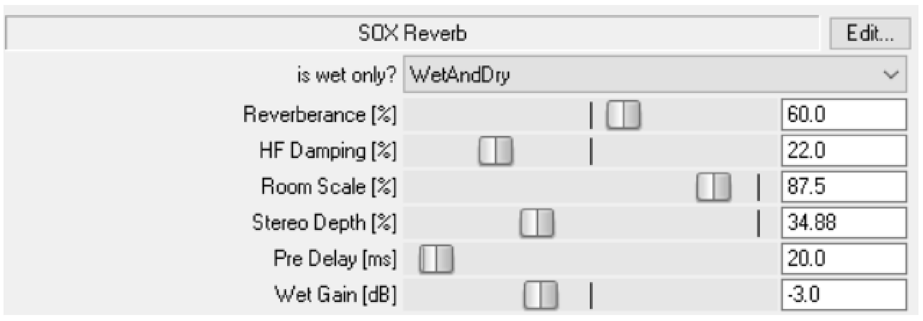
This effect implements a phaser effect to the audio.

inGain is the amplification factor for the input in decibels. delay gives the delay in milliseconds, decay a factor for the decay within the phaser and frequency gives the modulation frequency in Hz. The waveform of the modulation is either sinusoidal — preferable for multiple instruments — or triangular — gives single instruments a sharper phasing effect —. outGain is the volume of the output.

The decay should be less than 0.5 to avoid feedback, and usually no less than 0.1.

timeOffset shows that this effect is time-locked. For details refer to section 3.16.

3.14 SOX Reverb



SOX Reverb — Edit...

is wet only? WetAndDry
Reverberance [%] 60.0
HF Damping [%] 22.0
Room Scale [%] 87.5
Stereo Depth [%] 34.88
Pre Delay [ms] 20.0
Wet Gain [dB] -3.0

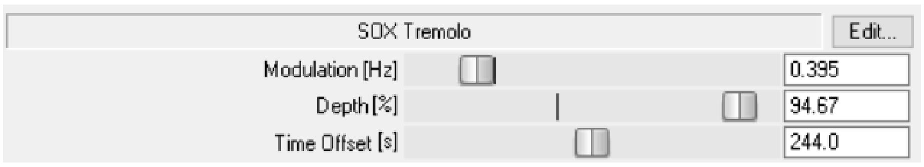
Figure 15: JSFX Panel for SOX Reverb

Parameter	Description	Unit
isWetOnly	tells whether only the wet signal should by produced	Boolean
reverberance	percentage for reverb density	—
hfDamping	percentage amount of damping of high frequencies for each reflection relative to low frequencies	—
roomScale	percentage for size of the room (more precisely the reflectivity of the room)	—
stereoDepth	percentage amount of stereo effect	—
predelay	time offset until first reverb occurs	ms
wetGain	gain of wet signal relative to dry signal	dB

This effect implements reverberation of audio using the "freeverb" algorithm, which uses eight parallel Schröder-Moorer filtered-feedback comb-filters followed by four Schröder allpasses in series.

Details on this algorithm can be found in [FREEVERB].

3.15 SOX Tremolo



SOX Tremolo — Edit...

Modulation [Hz] 0.395
Depth [%] 94.67
Time Offset [s] 244.0

Figure 16: JSFX Panel for SOX Tremolo

Parameter	Description	Unit
frequency	the modulation frequency of the tremolo	Hz
depth	percentage value for the intensity of modulation	—
timeOffset	the point in project time where modulation is at phase 0° (see 3.16)	s

This effect implements a tremolo effect. This tremolo is done by signal multiplication; hence it is a low frequency double sideband suppressed carrier modulation. Parameter frequency gives the tremolo frequency, depth gives the intensity as a percentage.

timeOffset shows that this effect is time-locked. For details refer to section 3.16.

3.16 Timelocking

There are effects that behave differently in time, technically they are *time-variant*. A filter does not care *when* a signal arrives, but a modulated effect like e.g. a phaser produces a different sound for different start times because the modulation is normally in another phase.

Hence when looking at the behaviour at a specific point in time, those time-variant effects would behave differently when the effect start time is varied.

E.g., assume a phaser with a 0.25Hz modulation (one cycle every 4s): when you start the effect 1s later, its modulation is now off by 90°. This is not helpful when the effect now depends on start time or loop positioning.

To circumvent this problem, all time variant effects from above (phaser and tremolo) are *time-locked* i.e. they check the current play position and always behave the same at some specific point in time regardless of the playback start time.

Additionally those effects have a parameter called timeOffset. This parameter tells at what time the effect has a phase of zero in its modulation. The default is 0s, but it may be adapted accordingly.

Take the phaser above and assume you want to make sure that its modulation is exactly at 0° at position 155s within your song[1]. Then you just set timeOffset to "155". Because the period of the modulation is 4s it is also okay to use $155 + 4k, k \in \mathbb{Z}$ as offset (e.g. "3"), but the above saves you from some calculation for complicated modulation frequencies.

By this method even time-variant effects can be synced with externally generated audio material.

4 Regression Test

To test that the plugins of JSFX-SOX really are bit-identical to sox, a little test suite has been set up.

The suite assumes that sox is installed in the Windows search path.

[1]This is a little lie, because the inital phaser modulation phase is 90°, but the argument is still valid.

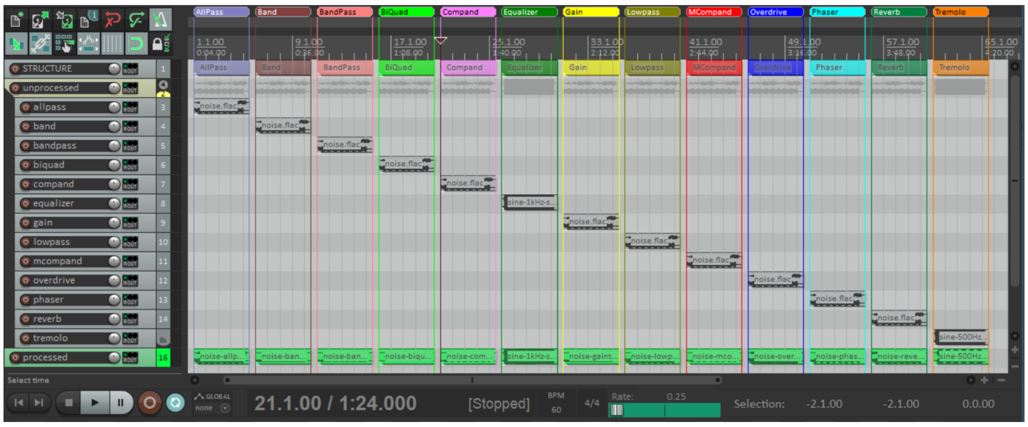


Figure 17: Regression Test Setup in Reaper

If so, a simple batch script sets up raw audio test files and — externally via the command line — applies sox effects to them producing audio result files. The parameters used are a bit exotic to ensure that algorithmic differences between sox and JSFX-SOX will show up. The batch script can be found in the `test` subdirectory and is called `makeTestFiles.bat`

Additionally there is a Reaper project referencing those audio test files and result files in autonomous tracks (see figure 17). JSFX-SOX plugins are configured with the exactly the same parameters as given in the batch file and are correspondingly applied to the raw audio test files.

When subtracting the rendered audio in Reaper and the externally rendered audio from sox, they (almost) cancel out. This can be checked by a spectrum analyser in the master channel, which is shown in figure 18. It shows a noise floor of typically less than -140dB. This comes from the 24 bit sample depth used in the FLAC files of the test suite; increasing that sample depth would even lead to less residual noise.

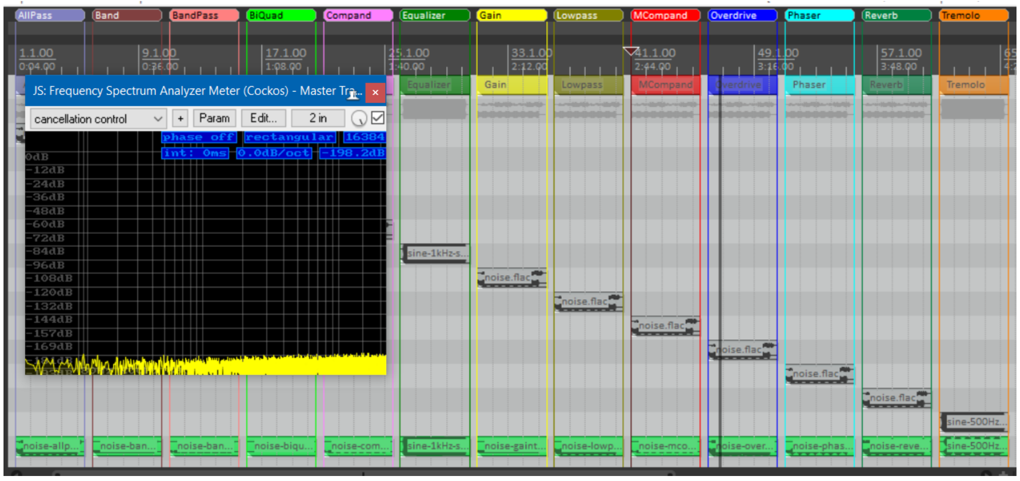


Figure 18: Example Noise Floor for Regression Test in Reaper

References

[DBIQFILT]	Wikipedia. *Digital Biquad Filter.* http://en.wikipedia.org/wiki/Digital_biquad_filter
[FREEVERB]	J. O. Smith. *Physical Audio Signal Processing.* W3K Publishing, 2010, ISBN 978-0-9745607-2-4. https://ccrma.stanford.edu/~jos/pasp/Freeverb.html
[JSFXSOX]	Dr. Thomas Tensi. *SOX Plugins for Reaper DAW in JSFX Language.* https://github.com/prof-spock/JSFX-SOX-Plugins
[REAPACK]	Christian Fillion. *ReaPack: Package manager for REAPER.* https://reapack.com
[REAPER]	Cockos Incorporated. *Reaper Digital Audio Workstation.* https://www.reaper.fm
[REAPLUGS]	Cockos Incorporated. *ReaPlugs VST FX Suite.* https://www.reaper.fm/reaplugs
[RBJFILT]	R. Bristow-Johnson. *Cookbook formulae for audio EQ biquad filter coefficients.* https://www.w3.org/2011/audio/audio-eq-cookbook.html
[SOXDOC]	Chris Bagwell, Lance Norskog et al. *SoX - Sound eXchange - Documentation.* http://sox.sourceforge.net/Docs/Documentation