Wah-Wah Plugin

Helmut Keller

This paper describes the source-code of a Wah-Wah software-plugin in JSFX format. JSFX-plugins are simple text files which can be loaded into the ReaJS plugin by Cockos Inc.. ReaJS is a VST2 plugin and part of the popular (and free) ReaPlugs suite. A second-order digital state-variable filter as described in the authors previous GITEC contribution [1] is used to implement the Wah-Wah effect. The default filter parameters fit the famous Vox V847 Wah. However, the user may customize the filter parameters and arrive at his or her very personal Wah-Wah sound. The plugin can also operate as a volume pedal.

1. Introduction

Writing one's very own effect plugin for a VST2 host seems to be all but impossible for most of us. However, applying a simple user interface turns this exercise into a rather doable affair. The JSFX programming as described at https://www.reaper.fm/sdk/js/js.php is very straightforward, and it requires only basic programming skills. The resulting source code is a simple text file which is compiled automatically when loaded into REAPER or the ReaJS plugin. The ReaJS plugin is available as part of the popular free ReaPlugs suite from Cockos Inc. at https://www.reaper.fm/reaplugs/. As a VST2 plugin, it runs under any VST2 host. The Wah-Wah plugin described in the present paper is a good starting point for writing one's own plugins. The basis for the below elaborations is found in the GITEC paper [1] where the theoretical background of the digital state-variable filters used is extensively discussed.

2. Vox V847 and Jim Dunlop Crybaby GCB95 Wah-Wahs

The first Vox Wah-Wah pedals were developed in 1966 by the Thomas Organ Company and appeared on the market in early 1967 as the "Clyde McCoy" version. Clyde McCoy was a trumpet player and famous for using the well-known purely acoustical Wah-Wah effect on his trumpet. At the time the Vox management did not believe that guitarists would be the main customers for this kind of effect. The Thomas Organ Company nevertheless released a demo record promoting their Wah-Wah pedal for the electric guitar later in 1967. When Cream and the Jimi Hendrix Experience released Wah-laden tracks ("Tales of Brave Ulysses" and "Voodoo Child (slight return)" being particularly poignant examples) in 1967 and 1968, everybody became aware that this pedal is most suitable for the electric guitar. The Vox V847 and the Jim Dunlop Crybaby GCB95 are later versions that prove to be still very popular. Published frequency responses in [2] and [3] show that both these later versions are in fact very similar. They are also very similar to the "old" Vox Wah-Wah which is analyzed in [4], and they can all be described as second-order bandpass filters with a varying resonance frequency f_0. The range of f_0 is typically from a bit more than 400 Hz to a bit less than 2.5 kHz.

The quality factor Q of the bandpass depends strongly on the resonance frequency f_0 as follows:

$$Q = Q_{1\text{kHz}} \left(\frac{f_0}{\text{kHz}} \right)^{e_Q} \tag{1}$$

The quality factor $Q_{1\text{kHz}}$ at $f_0 = 1$ kHz is approximately 4. The exponent e_Q is approximately -1.1.

The gain b_1 at $f = f_0$ depends slightly on f_0 as follows:

$$b_1 = b_{1\_1\text{kHz}} \left(\frac{f_0}{\text{kHz}} \right)^{e_M} \tag{2}$$

The gain $b_{1\_1\text{kHz}}$ at $f = f_0 = 1$ kHz is approximately 10. The exponent e_M is approximately -0.1.

Note that $e_Q - e_M = -1$. This ensures that the frequency response below 100 Hz does not depend much on f_0. This behavior is typical for Wah-Wah pedals with fixed inductors, such as the ones we have analyzed so far.

The universal Wah-Wah effect

The coefficients b_0 and b_2 of a second-order state variable filter are zero for a bandpass filter. However, we allow non zero values for both coefficients, as well. An exponent for both coefficients even allows varying them with changing f_0:

$$b_0 = b_{0\_1\text{kHz}} \left(\frac{f_0}{\text{kHz}} \right)^{e_T} \tag{3}$$

$$b_2 = b_{2\_1\text{kHz}} \left(\frac{f_0}{\text{kHz}} \right)^{e_B} \tag{4}$$

The high-pass portion (and thus the treble level) is controlled by b_0. The bandpass portion (and thus the middle level) is controlled by b_1, while b_2 is in charge of the lowpass portion and thus the bass level. Given this set of parameters we can vary every filter parameter with changing $f_{0,}$ and create an endless number of interesting Wah-Wah like sounds. The default value for $b_{0\_1\text{kHz}}$ and $b_{2\_1\text{Khz}}$ is zero. The default value for e_T and e_B is one.

With $b_{0\_1\text{kHz}}$ and $b_{2\_1\text{Khz}}$ set to values around 0.5 we can blend in some treble and bass. With $b_{0\_1\text{kHz}}$ and $b_{1\_1\text{Khz}}$ set to zero and $b_{2\_1\text{Khz}}$ set to one, a tunable lowpass filter is obtained. Setting $b_{2\_1\text{kHz}}$ and $b_{1\_1\text{Khz}}$ to zero, and $b_{0\_1\text{Khz}}$ to one, results in a tunable highpass filter. Changing $Q_{1\text{Khz}}$ as well as the four exponents, will give endless variations of interesting Wah-Wah-like effects.

The tuning range of f_0 can be customized via f_{min} and f_{max}. The relation between the position p of a MIDI expression pedal and f_0 is:

$$f_0 = f_{min} \mathrm{e}^{p \ln\left(\frac{f_{max}}{f_{min}}\right)} \tag{5}$$

This exponential relation assures a constant musical interval for a constant increment of p. The position p is assumed to increase linearly between 0.0 and 1.0 from the heel- to the toe-position of the expression pedal – as it is the case for most MIDI expression pedals. In typical Wah-Wah pedals the relation between f_0 and p depends strongly on the type of the potentiometer used. Since equation (5) seems to be the most desirable relation, we do not opt for a customization.

Fig.1 shows the frequency responses of the Wah-Wah effect with the default settings of the filter parameters, with the resonance frequencies set to 400 Hz, 1 kHz and 2.5 kHz. The red curves show the analog prototype filters. The blue curves show the derived digital filters with prewarping of f_0 and Q, and with a sampling rate of 48 kHz. An attenuation of the digital filters at frequencies above 5 kHz is apparent; it results from the bilinear transformation. Electric guitars and the associated loudspeakers do not transmit much energy in this frequency range. Thus it is not purposeful to make an effort compensating for this attenuation.

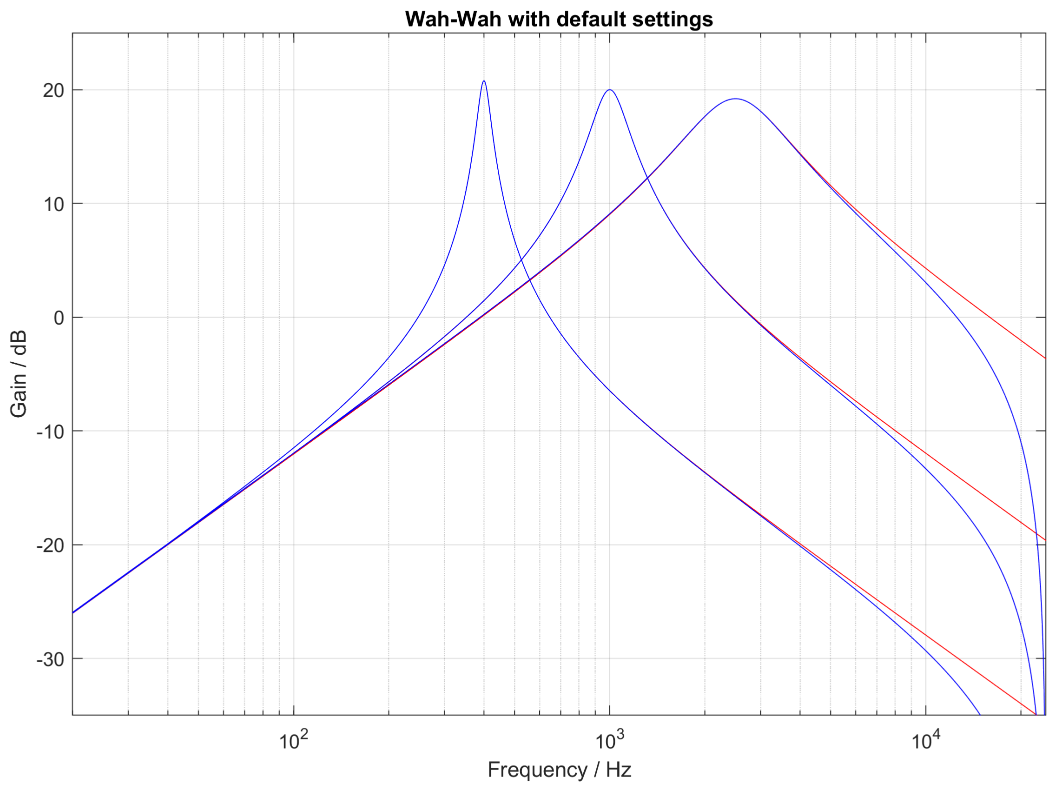


Fig. 1. Frequency responses of the Wah-Wah effect with the default settings. The resonance frequencies are 400 Hz, 1 kHz and 2.5 kHz. Red curves show the analog prototype filters. Blue curves show the derived digital filters at 48 kHz sampling rate with applied prewarping.

3. The volume pedal

The experience of the author is that a MIDI expression pedal in a guitar setup is mainly used for two purposes, namely controlling a volume-related plugin, or controlling a Wah-Wah plugin.

In many cases both uses are required alternatively at the same point in the signal chain. It thus makes sense to combine both pedal-uses into a single plugin. In this case a toe-switch of the MIDI expression pedal (if present) may select the mode of the plugin (Volume or Wah-Wah), and the position of the pedal will control the volume, or the resonance frequency of the Wah-Wah effect. If both these effects are implemented in two different plugins, both plugins need to be connected in series, and they have to be bypassed alternatively by one single switch, with the position parameter necessarily assigned to both plugins. This kind of assignment is, however, not possible in every VST host, and if it is, then the corresponding settings are typically rather tricky. Thus a combined volume and Wah-Wah plugin can make life much easier.

The gain *g* of the volume pedal depends on the *p* as follows:

$$g = p^{e_V} \tag{4}$$

The exponent e_V may be set from 1.0 (linear) to 4.0 (double quadratic). The default value is 2.0 (quadratic) which approximately emulates common “logarithmic” volume potentiometers.

A mode parameter allows the choice between the “Volume” and “Wah-Wah” modes.

4. The JSFX source code

So far we have defined the desired behavior and the parameters of our plugin. Now it is time to write the source code. We advance step by step thru the sections of the source code:

Some comments and meta data

```
// (C) 2023, Helmut Keller

// NO WARRANTY IS GRANTED. THIS PLUG-IN IS PROVIDED ON AN "AS IS" BASIS, WITHOUT
// WARRANTY OF ANY KIND. NO LIABILITY IS GRANTED, INCLUDING, BUT NOT LIMITED TO,
// ANY DIRECT OR INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGE ARISING
// OUT OF THE USE OR INABILITY TO USE THIS PLUG-IN, COMPUTER FAILTURE OF
// MALFUNCTION INCLUDED. THE USE OF THE SOURCE CODE, EITHER PARTIALLY OR IN
// TOTAL, IS ONLY GRANTED, IF USED IN THE SENSE OF THE AUTHOR'S INTENTION, AND
// USED WITH ACKNOWLEDGEMENT OF THE AUTHOR. LAST BUT NOT LEAST, BY USING THIS
// PLUG-IN YOU RELINQUISH YOUR CLAIM TO SUE IT'S AUTHOR, AS WELL AS THE CLAIM TO
// ENTRUST SOMEBODY ELSE WITH DOING SO.

desc:Vol-Wah
tags: Volume Wah-Wah
author: Helumut Keller
```

Here are some comment lines which begin with //

desc: the name which appears on the GUI of the plugin

tags: tags are space delimited and can be used in Reaper to search for specific plugins

author: the authors name

The sliders

slider1:position=	0.5 <0, 1, 0.001>	Position
slider2:mode=	0 <0, 1, 1 {Volume, Wah-Wah}>	Mode
slider5:eV=	2.0 <1.0, 4.00, 0.1>	Volume Exponent
slider7:fmin=	400 <200, 800, 5>	Min.Frequency[Hz]
slider8:fmax=	2500 <1250, 5000, 50>	Max.Frequency[Hz]
slider10:Q_1k=	4 <0.5, 20, 0.1>	Q
slider11:LB=	-60 <-60, 30, 0.5>	Bass [dB]
slider12:LM=	20 <-60, 30, 0.5>	Mid [dB]
slider13:LT=	-60 <-60, 30,0.5>	Treble [dB]
slider15:eQ=	-1.1 <-2, 2, 0.05>	Q Exponent
slider16:eB=	0 <-2, 2, 0.05>	Bass Exponent
slider17:eM=	-0.1 <-2, 2, 0.05>	Mid Exponent
slider18:eT=	0 <-2, 2, 0.05>	Treble Exponent

The sliders are the parameters of the plugin. They are shown in the GUI and can be used to control the plugin by the VST host. We will probably assign “position” and “mode” to a MIDI expression pedal with a toe switch.

Silders like “mode” with a name list in curly brackets appear in the GUI as a drop-down selector.

The text between **sliderx:** and **=** defines the JSFX variable name of the slider

The first value before **<** is the default value of the slider.

The values between **<** and **>** define the range and resolution of the slider

The text after **>** is shown in the GUI as the name of the slider.

The sliders are sorted by their numbers. If a number is missing between two sliders there will be a small gap in the GUI between both sliders.

The initialization of variables and constants

```
@init

// Time constant of the parameter smoothing:
tau = 0.010;

// Coefficient of the parameter smoothing fliters:
ksmooth = 1 / (tau*srate);

// Frequency scaling factor:
pit = 4*atan(1) / srate;

 // All smoothed coefficients, their updates and their states:

volume = 1.0;
volume_u = 1.0;
volume_s = 1.0;
k = 0.1;
k_u = 0.1;
k_s = 0.1;
kdiv = 1/ 1.11;
kdiv_u = 1 / 1.11;
kdiv_s = 1 / 1.11;
kf = 1.1;
kf_u = 1.1;
kf_s = 1.1;
b0 = 0.0;
b0_u = 0.00;
b0_s = 0.0;
kb1 = 10.0;
kb1_u = 10.0;
kb1_s = 10.0;
b2 = 0.0;
b2_u = 0.00;
b2_s = 0.0;

// Filter states of the state-variable filters (left and right channel):

s1l = 0.0;
s2l = 0.0;
s1r = 0.0;
s2r = 0.0;
```

The code after **@init** is called up when the plugin is loaded and when important system variables like **srate** (the sampling rate of the plugin in Hz) change. The constant **tau** is the time constant (in seconds) of the parameter smoothing filters. The variable **volume** is the gain of the volume pedal. The other filter coefficients, their update values and the filters states are explained in [1].

The slider updates

```
@slider

mode == 0 ?
(
        volume_u = position^eV;
):
(
        // Parameters of the analog prototype filter:

        f = fmin * exp(position*log(fmax/fmin));
        f_kHz = f /1000;
        Q = Q_1k * f_khz ^ eQ;
        LT == -60 ? b0_1k = 0 : b0_1k = 10^(LT*0.05);
        LM == -60 ? b1_1k = 0 : b1_1k = 10^(LM*0.05);
        LB == -60 ? b2_1k = 0 : b2_1k = 10^(LB*0.05);
        b0_u = b0_1k * f_kHz ^ eT;
        b1 = b1_1k * f_kHz ^ eM;
        b2_u = b2_1k * f_kHz ^ eB;

        // Prewarping of f and Q:

        fw = f*tan(pit*f)/(pit*f);
        aux= pit *f /sin(2*pit*f) * log( (sqrt(1+4*Q*Q)+1) / (sqrt(1+4*Q*Q)-1)
        kqw = exp(aux)-exp(-aux);

        // Parameters of the digital state-variable filter:
        k_u = pit * fw;
        kdiv_u = 1 / ( 1 + k_u * ( k_u + kqw));
        kf_u = kqw + k_u;
        kb1_u = b1 * kqw;
```

The code after **@slider** is called up when one of the slider values has changed. We distinguish between the two modes of the plugin, and calculate the update values for all filter coefficients of the actual mode.

We enter the bass, mid and treble levels in dB. We therefore have to compute the derived linear parameters b0_1k, b1_1k and b2_1k. A level of -60 dB sets the derived parameter to zero (-inf. dB).

The sample updates

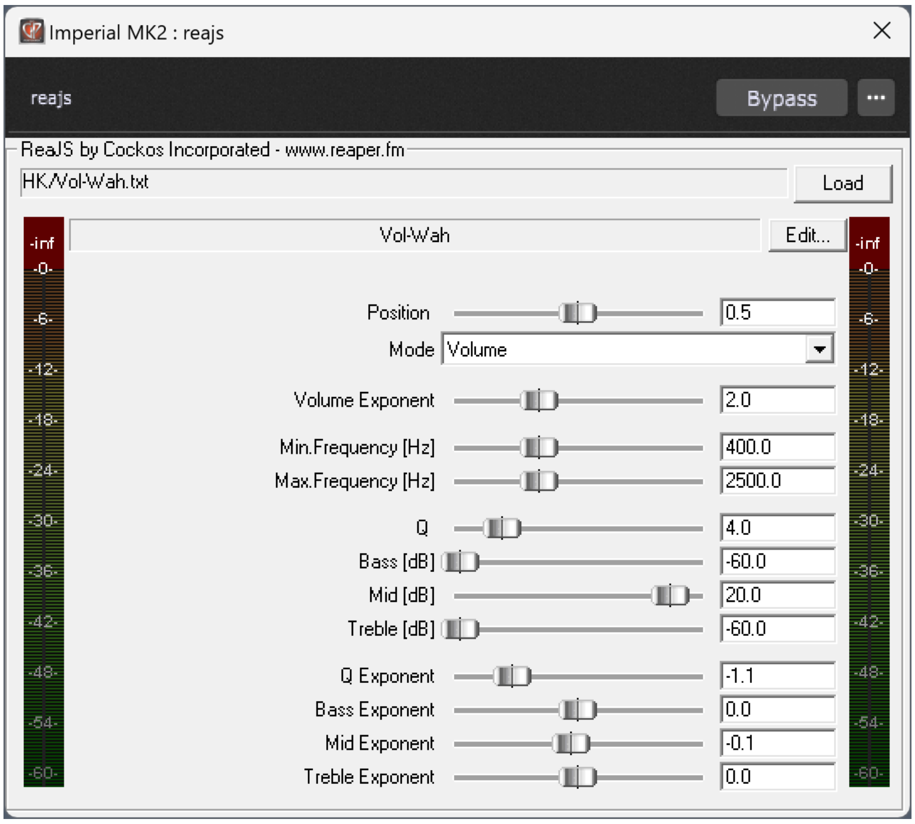
```
@sample

mode == 0 ?
(
        // The parameter smoothing filter:

        volume = ksmooth * (volume_u - volume_s) + volume_s;
        volume_s = volume;

        // The volume control:

        spl0 = volume * spl0;
        spl1 = volume * spl1;
):
(
        // The parameter smoothing filters:

        k = ksmooth * (k_u - k_s) + k_s;
        k_s = k;
        kdiv = ksmooth * (kdiv_u - kdiv_s) + kdiv_s;
        kdiv_s = kdiv;
        kf = ksmooth * (kf_u - kf_s) + kf_s;
        kf_s = kf;
        b0 = ksmooth * (b0_u - b0_s) + b0_s;
        b0_s = b0;
        kb1 = ksmooth * (kb1_u - kb1_s) + kb1_s;
        kb1_s = kb1;
        b2 = ksmooth * (b2_u - b2_s) + b2_s;
        b2_s = b2;

        // The state-variable filters:

        hpl = kdiv * (spl0 - kf * s1l - s2l);
        aux = k * hpl;
        bpl = aux + s1l;
        s1l = aux + bpl;
        aux = k * bpl;
        lpl = aux + s2l;
        s2l = aux + lpl;
        spl0 = b0 * hpl + kb1 * bpl + b2 * lpl;

        hpr = kdiv * (spl1 - kf * s1r - s2r);
        aux = k * hpr;
        bpr = aux + s1r;
        s1r = aux + bpr;
        aux = k * bpr;
        lpr = aux + s2r;
        s2r = aux + lpr;
        spl1 = b0 * hpr + kb1 * bpr + b2 * lpr;
);
```

The code after **@sample** is called up for each sample. The system variables **spl0** and **spl1** are the input samples for the left and right channels of the plugin. They can be overwritten by the plugin code and are used as the output samples of the plugin, as well. We distinguish between the two modes of the plugin. In a first step, we calculate the actual coefficients with the parameter smoothing filters. In a second step, we multiply the input samples with the actual gain or calculate the actual digital state variable filter of both channels. See [1] for the details of the filter calculations.

5. The usage of the plugin

You need to execute the following tasks before you can use our Wah-Wah plugin.

- Install the ReaPlugs suite on your computer.
- Copy the file “Vol-Wah.txt” into the directory where ReaJS expects JSFX plugins. Note that “Vol-Wah.txt” and this document are made available in one and the same ZIP-file.
- Load the ReaJS plugin into your VST Host.
- Load “Vol.Wah.txt” into the ReaJS plugin.



HK/Vol-Wah.txt

Position	0.5
Mode	Volume
Volume Exponent	2.0
Min.Frequency [Hz]	400.0
Max.Frequency [Hz]	2500.0
Q	4.0
Bass [dB]	-60.0
Mid [dB]	20.0
Treble [dB]	-60.0
Q Exponent	-1.1
Bass Exponent	0.0
Mid Exponent	-0.1
Treble Exponent	0.0

Fig. 2. The GUI of the plugin.

Fig. 2 shows the GUI of the plugin. The “Load” button is used to load the plugin file. The name of the loaded file and its sub-directory are shown left of the button. The next line gives the name of the plugin. The “Edit” button may be used to edit the plugin file. The two bars at the left and right side are the input and output level meters of the plugin. In the center we see all the parameter sliders we have defined.

You can use the plugin now – but you have to use the GUI controls of the plugin to change the position and mode parameter. If you have a MIDI expression pedal and a MIDI footswitch you probably want to control the position parameter with the pedal and the mode parameter with the switch. To do so, you need to assign both plugin parameters to the corresponding MIDI control change number in your VST host. See the manual of your VST host to find out how to do that.

In REAPER you can do the assignment directly in the “Param” menu of the plugin window.

In Gig Performer ® you need to assign a GUI widget like a knob or switch to the plugin parameter in a first step. In a second step, you assign the MIDI control change number of your physical control element to the GUI widget of the host. The author highly recommends Gig Performer ® for those who seek to implement a live setup entirely controlled by a computer.

6. Summary

A plugin that emulates a Volume and a Wah-Wah pedal has been developed. It is demonstrated that the JSFX plugin format is a good choice to write one’s own plugins using a simple user interface that can be handled by users with only very basic programming skills. The necessary digital signal processing theory was already been described in [1].

7. Literature

[1] H. Keller, “Digital State-Variable Filters”, GITEC, 2023, https://gitec-forum-eng.de/filter-design-digital-state-variable-filters/

[2] https://www.electrosmash.com/vox-v847-analysis

[3] https://www.electrosmash.com/crybaby-gcb-95

[4] M. Zollner, "Physik der Elektrogitarre", https://www.gitarrenphysik.de