Low Frequency & Function Generator

VCarrara

Definition

Some guitar effects depend on devices that generate an envelope on the input signal to make it louder or weaker over time. The envelopes may have an attack time (rising time), followed by a steady or flat time and ending with a release time. They are employed in several effects, like Tremolo, Phaser and Chorus. They can increase or decrease the output volume in Tremolo, for instance, or change the pitch in Delay, as in Phaser, Chorus and Vibrato. The envelopes are generated by a low-frequency oscillator (LFO) circuit on analog effects, but this can be easily done digitally. In fact, while analog circuits can only produce static envelopes, like sine or square oscillations, the digital procedures are able to generate several profiles, some of then too difficult to make with analog circuits. That explains why the model of the LFO shown here extends its domain to aboard also function generation and external profiles. Modulation of the audio signal can be done on amplitude, like in Tremolo, for instance, or frequency (pitch) in Chorus and Vibrato, or even in tone, like in Wah-wha.

The functions of the Low Frequency Function Generator (LFFG) can be generically grouped in 3 categories:

- Function Generation with Low-Frequency Oscillator
- Dynamics
- External profile

Function Generator

The Function Generator (FG) of the Guitar Sound Processing (GSP) can produce several periodic profiles selected by user. Frequency or period can be adjusted, as well as the duty cycle of square waves. The wave profiles are

1) Sine Wave. FG produces positive sine wave, with adjustable frequency. The output amplitude is achieved by

$$y_{lfo} = \frac{1 + \sin\varphi}{2}$$

where the phase angle φ is computed as function of the wave frequency ω, by

$$\varphi = \omega t - 2\pi \text{int}(\omega t / 2\pi),$$

and, therefore, $0 \leq \varphi < 2\pi$. This kind of wave produces a pulsating output. However, the output power is too weak when the amplitude y_{lfo} is small, due to a linear instead of a logarithmic wave power. The amplitude generated by the Sine Wave is shown in Figure 1.

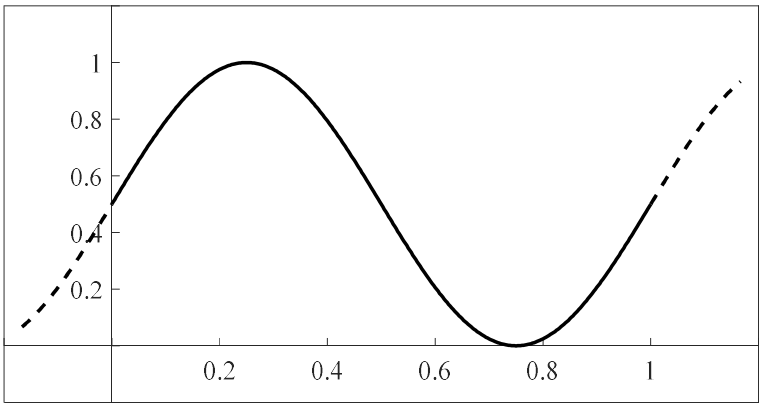

Fig. 1 – Sine wave profile.

2) Half Sine wave, with adjustable frequency ω. The wave function is given by

$$y_{lfo} = \sin(\varphi / 2) .$$

The Half Sine wave shows low amplitude for short time, as can be seen in Figure 2, and this help to correct the power problem found in the Sine Wave.

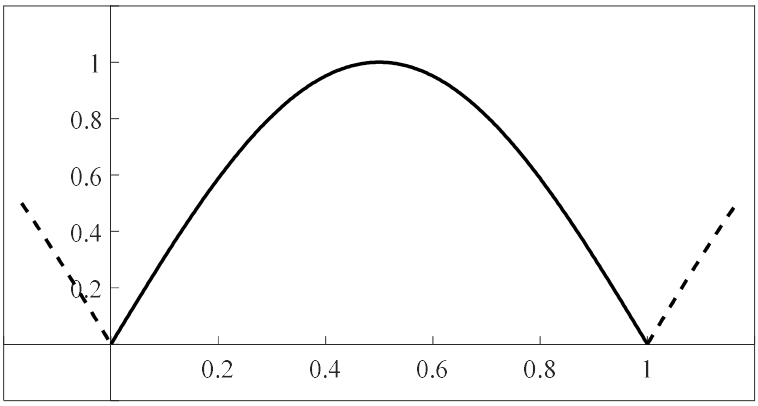

Fig. 2 – Half sine wave profile.

3) Ramp Wave with adjustable frequency ω. The wave function is given by

$$y_{lfo} = \frac{\varphi}{2\pi} .$$

which produces a saw teeth wave, as shown in Figure 3. This kind of wave can be used to simulate time inverted sounds.

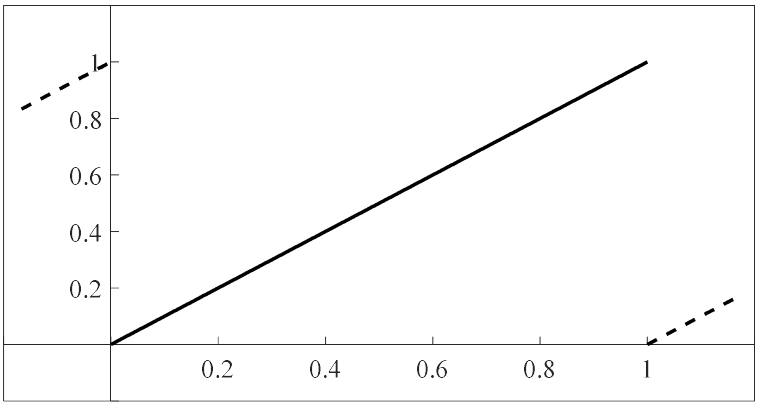

Fig. 3 – Ramp wave profile.

4) Saw Wave with adjustable frequency ω. The wave function is given by

$$y_{lfo} = 1 - \frac{\varphi}{2\pi}.$$

which produces a saw teeth wave, as shown in Figure 4.

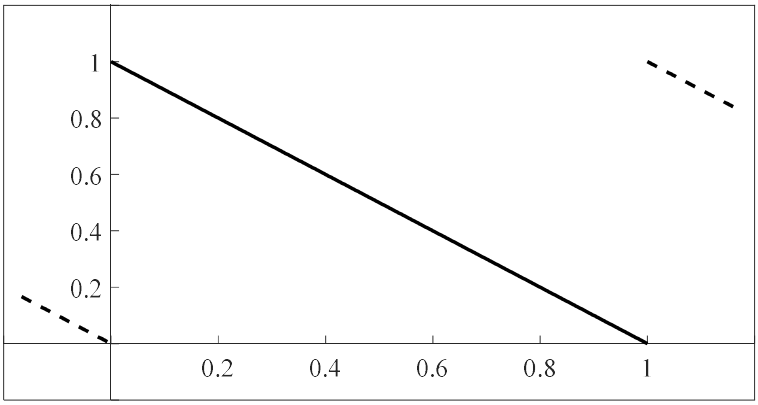

Fig. 4 – Saw tooth wave profile.

5) Triangle Wave with adjustable frequency ω. The algorithm for the wave function is given by

$$y_{lfo} = \begin{cases} \frac{\varphi}{\pi} \text{ if } \varphi \le \pi \\ 2 - \frac{\varphi}{\pi} \text{ if } \pi < \varphi \le 2\pi \end{cases}$$

whose output is shown in Figure 5.

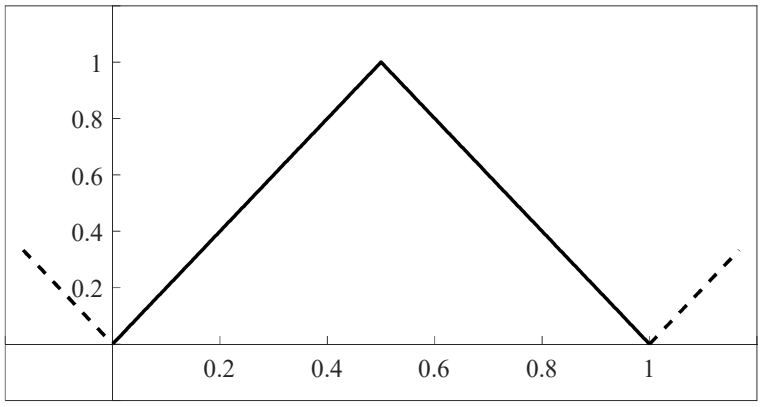

Fig. 5 – Triangle wave profile.

6) Square Wave with adjustable frequency ω and duty cycle δ. The algorithm for the wave function is given by

$$y_{lfo} = \begin{cases} 1 \text{ if } \varphi \le \delta \\ 0 \text{ if } \delta < \varphi \le 2\pi \end{cases}$$

whose output is shown in Figure 6.

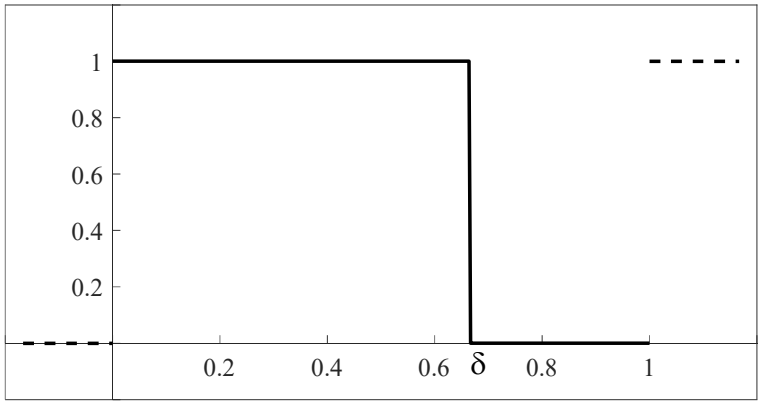

Fig. 6 – Square wave profile.

7) Exponential Decay with adjustable frequency ω and decay rate δ. The algorithm for the wave function is given by

$$y_{lfo} = \left(1 - e^{-g\frac{\varphi}{2\pi}}\right) e^{-\delta\frac{\varphi}{2\pi}}$$

where g is a variable related to the attack gain, and was fixed in $g = 256$. The decay rate δ changes the release time of this LFO, as can be seen in the Figure 7, for δ equal to 1 (blue), 4 (black) and 16 (red). The exponential decay mimics the envelope of a damped vibrating string.

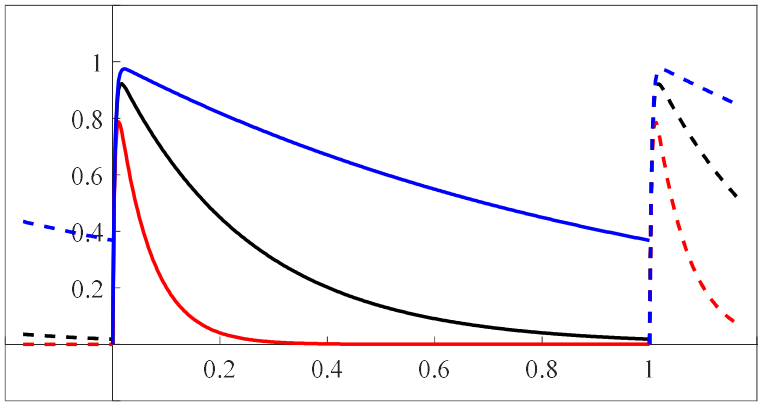

Fig. 7 – Exponential decay wave profile.

8) Exponential Wave with adjustable frequency ω and slope rate δ. This envelope is the reverse of the Exponential Decay. The algorithm for the wave function is given by

$$y_{lfo} = \left(1 - e^{-g\left(1-\frac{\varphi}{2\pi}\right)}\right) e^{-\delta\left(1-\frac{\varphi}{2\pi}\right)}.$$

The Exponential Wave simulates the time reverse envelope of a vibrating string. The slope rate δ changes the attack time, as can be seen in the Figure 8, for δ equal to 1 (blue), 4 (black) and 16 (red).

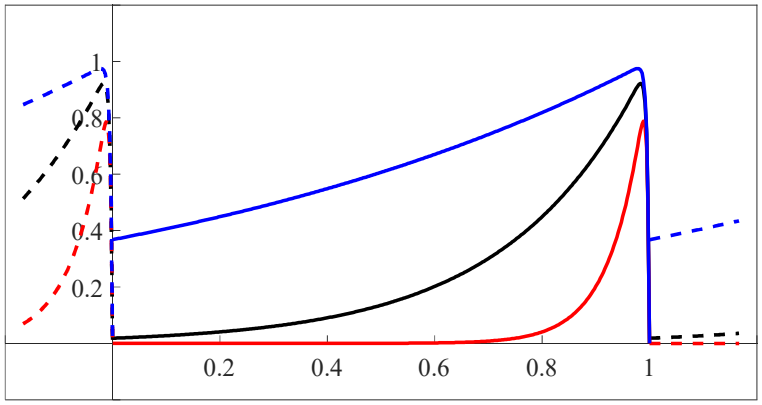

Fig. 8 – Exponential wave profile.

Dynamics

Some effects depend on the input envelope or the input power. For example, some sustain pedals increase the output gain whenever the input level decreases, therefore maintaining the strength of the guitar sound. Compressors and expanders also make use of the input power to adjust the output level. As the input power changes within long time intervals, it

can be used to envelope any input signal that uses LFO, such as Chorus, Phaser, Tremolo, etc, and even an automatic Wah-wah. Two kinds of dynamics are available: the Level Detector and the Reverse Level Detector.

1) Level Detector. The procedure computes the input power exactly in the same way as the Compressor effect does (see Compressor documentation for more details). However, instead using the Compressor to generate the LFO envelope, the same algorithm of level detection of Compressor was employed. This procedure runs separately from any effect and before processing any input audio signal.

   If $x(k)$ is the amplitude of the input audio at instant k, then the Level Detector is computed by

   $$x_L(k) = |x(k)|$$
   $$y_1(k) = \max\{x_L(k), \alpha_R\, y_1(k-1) + (1-\alpha_R)\, x_L(k)\}$$

   and then a first-order low-pass filter is applied to the input level, resulting

   $$y_{lfo}(k) = \alpha_A\, y_{lfo}(k-1) + (1-\alpha_A)\, y_1(k)$$

   where α_A and α_R are the attack and release coefficients, respectively, computed by

   $$\alpha_A = e^{-\frac{1}{\tau_A f_s}}$$

   and

   $$\alpha_R = e^{-\frac{1}{\tau_R f_s}}$$

   for a discrete filter, and such that f_s is the sampling rate, and τ_A and τ_R are the attack and release time constants.

2) Reverse Level Detector. Together with the Level Detector, GSP also provides the Reverse Level Detector, which is simply given by

   $$\overline{y}_{lfo}[k] = M_{mx} - y_{lfo}[k],$$

   where M_{mx} is the maximum level that an input signal can reach, and y_{lfo} is the Level Detector output. The Reverse Level Detector can be used to model a sustain effect.

External profile

Whenever the preceding functions fail, the External Profile is the final solution. The envelope level is provided by an external source through a digital serial line, at 100 Hz rate. Several channels are available to provide up to 8 input envelopes. Each one can be obtained by a simple Expression Pedal, or by customized algorithms. More information is available in Section 3.4 (Expression Pedal Linkage) of the GSP Manual. The main use of

this kind of envelope is to control the output level with a Volume pedal, or to provide the modulation for a Wah-Wah. However, any other effect that requires the LFO can be attached to an External Profile, like Chorus, Phaser, Vibrato and Tremolo, among others.

LFFG class

The LFFG class can be included as member in any effect class. Its methods are:

- void Init(uint32_t sampling_rate);
- void SetProfile(uint8_t prof);
- void SetPeriod(uint32_t prd);
- void SetFrequency(float freq);
- void SetDutyCycle(float duty_cycle);
- void SetGain(uint32_t gain);
- uint32_t GetAmplitude();
- uint32_t GetValue();

and the enumerator

```
enum  lfo_wave
        {
                LFO_SIN,              (Sine wave)
                LFO_HALFSINE,         (Half sine wave)
                LFO_RAMP,             (Ramp wave)
                LFO_SAW,              (Saw wave)
                LFO_TRIANGLE,         (Triangle wave)
                LFO_SQUARE,           (Square wave)
                LFO_EXPDECREASE,      (Exponential decay)
                LFO_EXPINCREASE,      (Exponential wave)
                LFO_EXTERNAL,         (External profile - potentiometer)
                LFO_LEVEL,            (Level detector)
                LFO_REVERSE_LEVEL,    (Reverse level detector)
                LFO_LAST              (None)
        }
```

LFFG class need to be initialized with Init() method before using it, where sampling_rate is the selected sampling rate of the audio signal. All LFFG profiles use lookup tables (arrays) to previously store the modulation amplitude, with 512 time intervals in each cycle period. No interpolation is made on retrieving the output level, so discrete changes may occur. In the current version this value can only be modified by program recompilation. The default values after initialization of this class are LFO_HALFSINE profile, period T_{LFO} equal to 500 milliseconds, and duty cycle of 50%. The class properties are configurable with the methods:

- SetProfile(uint8_t profile)
  Function to select the desired LFFG profile from the available function generators. profile can be any of the lfo_wave enumerator values.
- SetPeriod(uint32_t prd)

To set the LFFG period $T_{LFO} = 2\pi/\omega$ in milliseconds (between 100 and 5000) if applicable

- SetFrequency(float freq)
  To set the LFFG frequency ω in Hz (between 0.2 to 10 Hz) if applicable
- SetDutyCycle(float duty_cycle)
  To set the LFFG duty cycle in percentage of the LFFG period (between 0 and 100%), if applicable.

Additionally, if LFO_EXTERNAL profile is selected, LFFG counts with a function to set the output amplitude after reading the potentiometer level:

- SetGain(uint32_t gain)
  To set the LFFG output in GetValue() method. gain must be scaled to the 0 to 65532 interval.

LFFG has only two methods to retrieve information:

- uint32_t GetAmplitude()
  This function gets the maximum amplitude of the output modulation level, presently equal to 65532.
- uint32_t GetValue()
  This function returns with the modulation level, according with the selected profile, in the range 0 to 65532, at the time it is called. Note, however, that GetValue() must be called at the same frequency as the audio sampling rate, since it is time dependant, and any timer function can be cumbersome in some way, both for processing time or timer overflow. Therefore it was considered preferable to increment the phase angle φ at each GetValue() call by ω/f_s, but this requires calling this function at f_s rate. Of course the effects that require LFFG provide this action.