Enhanced Quality and Variety for Chorus/Flange Units

P. Fernández-Cid, F.J. Casajús-Quirós
SSR, Escuela Técnica Superior de Ingenieros de Telecomunicación, UPM
Ciudad Universitaria s/n; 28040-Madrid, SPAIN
pablo@gaps.ssr.upm.es

Abstract

Chorus and flanger are two classical effects that can be found in any current digital audio FX unit. Available chorus/flanger just translate into digital implementations the behaviour and parameters typical of cost-sensitive analogue realizations. But they can be greatly improved on a digital implementation with simple additions to the software. After an introduction to chorus/flange and their digital realizations, we address two such additions (randomized modulation law, and application of energy envelope for noise control), and then discuss further improvements easily achiveable by digital means (multiscale implementations).

1 Introduction

Typical modulation law in chorus/flanger units is a sinus or triangle (sometimes other periodical signals), and this causes a percetually boring result, once the ear detects the regular repetition pattern. A noisy modulation can be used for less of a regular pattern with pleasant results, though some considerations should be made about how to generate the noise.

The fact that chorus/flanger sum up modified versions of the original signal (particularly for feedbacked flanger and multivoice chorus) increases the noise in the final signal. Also the fact that the final signal has a 'timbral modulation' makes the original noise (steady, with a fixed spectral envelope, hence easy to 'forget' or mask) a dominant component of the signal during low level passages (noise is no longer static but ever-changing). This can be improved on a 'multieffect' unit, by using the envelope follower part of a compressor to drive the dry/wet mix of the chorus/flanger. This is an example on how side chains in multiFX units can, at a very low computational cost, provide enhanced results in relation to the simple series conection through FX algorithms.

Finally, benefits of a multiscale approach to chorus/flange are addressed.

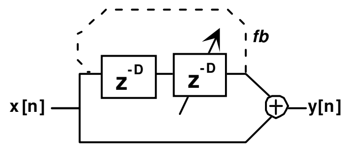
2 Chorus and Flanger

Chorus and flanger are two applications of a more general 'modulated delay' device. The difference between them is just the adjustment of some of the parameters, but leads to very different results.

Chorus tries to recreate the illusion of more than one instrument from the signal of one player. Two musicians never play in perfect unison (both time and pitch wise). In order to build up the proper illusion, one can sum the original signal with a slightly delayed and detuned version of itself. Instead of a constant pitch deviation, more natural results are to be expected from a varying pitch deviation (two players never keep constant their relative pitch distance).

The flanger started its life as a mechanical realization: two identical tapes were run in parallel while a human operator randomly controlled the speed of each unit, making minor variations up and down the nominal tape speed. Mixing the sound from both tapes, the signals sometimes aligned in phase, while other times aligned in counter phase, resulting in a time-varying filtering that has been named 'flanger'. The structure of the flanger is then that of the mix of two randomly delayed copies of a signal.

The chorus effect can in fact be obtained with the very same structure: the variable 'detune' can be imposed by means of a variable 'delay' (variable delay reproduces the original waveform at a non-constant speed that makes minor changes to pitch). A general 'chorus/flanger' can then be built like:



(delay has been split in a fixed and a -smaller- variable part, and a feedback path has been included, as can be usually found in flangers to further pronounce the effect).

3 Comb filter

The structure shown in previous figure, can be seen as that of a comb filter, or can also be thought of (nulling the feedback) as a two-ray propagation system: The original signal is summed with a (somewhat modified) close echo of itself. According to the period of the components of the signal and the delay between both rays, some of these components will be enhanced (both rays sum in phase if the delay is a multiple of the component's period), while others will cancel (counterphase). The maxima and minima of this 'filtering' are harmonically related, and change (always keeping its harmonic distribution) if the delay between rays changes. A mathematical description of comb filters can be found in [Moore].

This is the kind of modification to be expected from a flanger: harmonically related enhancement and cancellations in the spectrum changing their positions up and down.

Feedback enahnces difference between peaks and valleys and is most often used in flangers (it is a user adjustable parameter), up to the point that the harmonic structure of peaks and valleys imposes a tonal character of its own on the original signal (the feedback path can even be driven into self-oscillation, transforming the original sound into a 'siren' like complex tone). This tonal character of the spectral (timbral) modification gives the flanger its somewhat 'metallic' colour.

Typically the delay time on flangers is changed very slowly, to better appreciate the 'sweeping' movement of resonances, and the mean delay is quite short (few milliseconds, 0-10) so that the resonances fall in the audio region. Mean delay controls the main 'tone' or colour, that smoothly changes through time according to the delay's modulation intensity and speed.

Main delay and feedback are the parameters that make the chorus sound so different to a flanger. The short delay in the flanger means that we can not hear both instances of the sound appart from each other, they mix and end up -perceptually- as a single, dinamically filtered sound. On the other hand, the main delay on the chorus is larger (typically from 10 to 40 ms), so that both rays do no longer join a a single sound, but can give the illusion of two players (no feedback is applied on chorus, as the larger delay would make feedbacked rays sound as a repetitive echo, and would break the more-than-one player illusion: accumulated delay would be too long to be creditable as unison playing).

4 Digital modulated delay

Let *x(t)* be a sound signal, correctly sampled at *fs* Hz. to give the equivalent sequence *x[n]*. A delayed version of *x[n]* is $x_d[n]=x[n-d]$. If d is an integer number of sampling periods, $x_d[n]$ is one of the past samples of *x[n]*. If delay magnitude is fixed, $y[n] = x[n]+x_d[n]$ is a simple filter (an LTI system), whose main characteristics had been studied in the previous section. If the delay is not integer, we can no longer rely on past samples to get the *y[n]* signal.

The fact that delay is not fixed but moving in both chorus and flanger, means that sometimes the delay will not be integer in sample periods, but fractional. A possible approach would be to sample at a non fixed sample rate, but it is much more simple (hardware wise) to keep the fixed rate sampling and use interpolation to calculate the value of the signal in between available samples.

Suppose by now that the delay we want is fixed (though fractional). Let it be *d* composed of a *di* integer part and a *df* purely fractional part. The integer part is easy to obtain (by changing the position of the read pointer in the input buffer), so we can just face the problem of how to obtain a purely fractional delay. Linear interpolation can be used (sometimes it is used in commercial units), but of course is not the best interpolator available. Error in the interpolation would be heard as a 'noise' in the processed sound (further enhanced if feedback increasingly degrades signal to noise ratio on each pass though the loop).

In order to evaluate how much computation is needed to obtain a good quality interpolator, we can think of the best one and then introduce compromises. The best (perfect reconstruction) interpolator is the sinc shape. The following equality can be easily derived for signals sampled at *fs*:

$$x[n-d] = x[n] * sinc_d[n]$$

where the * operator stands for convolution, $sinc_d[n]$ stands for a 'delayed' sinc shape (*sinc[n-d]*), the *sinc[n]* function is defined as $sin(\pi n)/\pi n$. This convolution (or the perfect interpolator) can not be exactly computed due to the infinite length of the sinc function:

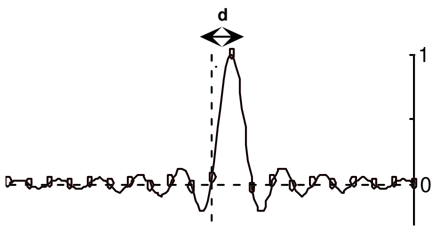


Figure. Continous and sampled delayed sinc

but we can approximate it arbitrarily by using us much coefficients of the sinc function as needed (in order to make the effect of discarding the tails negligible).

Say that we use *N* coefficients each side of the origin from the $sinc_d[n]$. We can evaluate signal to noise ratio due to this approximation to the ideal fractional delay. The error, not only depends on *N*, but also on the magnitude of *d* (a one half sample delay is more difficult to obtain than a 0.001 sample delay).

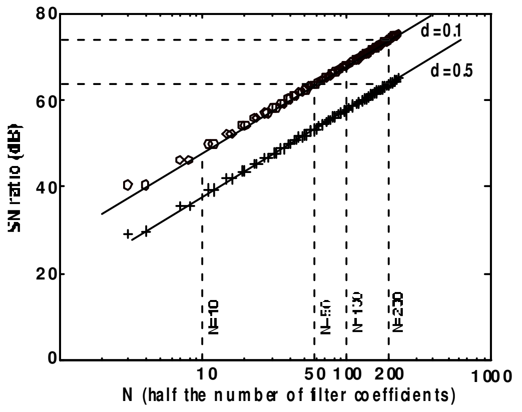


Figure. Signal to noise ratio achiveable on the windowed sinc approach to fractional delay for 0.5 and 0.1 sample delay, and various N.

Clearly, the amount of computation needed for a CD-quality (16 bit, or 96 dB S/N) fractional delay unit is really large. Though we usually tend to think of reverb as the highest computing power demandant FX, also chorus and flanger demand high computation to keep far away enough from noise.

Also notably, the large number of coefficients for a high-quality implementation, imposes restrictions on the minimum value for the delay (in order to keep the system causal), compromising the realization of (low-delay) flangers.

Taking this into account, a higher than CD-DA sampling rate is the only real world method to lower the inherent noise of these units to CD figures.

Chorus and flanger use a varying delay, so we can no longer talk about an LTI system, though the delay is usually changed at low (subaudio) speeds. It is then posible to apply the concepts of LTI systems, though on a short-time extent. The action of these modulated delays systems is still that of a (slowly moving) filter.

5 Laws for delay modulation

Traditionally, chorus and flanger have used simple periodic modulation to change the delay through time. Hearing the results of this approach ends up being perceptually boring. Better results are to be expected by using less predictible laws, particularly if some noisy (heavy uncorrelated) modulation source is used.

The sampling rate of the audio signal is 'fs', most probably 44.1 or 48 KHz. The noisy signal is a modulation signal that takes the role of the subaudio sinusoid for modulation, so it is expected not to vary very fast. Clearly the fs sampling frequency is large too fast for such a slow signal, though we still need to know the detailed evolution of the noise at the fs speed (otherwise stair modulation would be produced -useful sometimes, not the everyday needs-).

Ideally we would like to control the perceptual 'speed' (central frequency) of the modulation and the degree of 'randomness' (bandwidth), in order to be able to blend from the usual 'sinusoidal' behaviour to a 'random' one, keeping a sensation of 'faster' or 'slower' modulation in spite of the random nature.

One approach would be to use a white noise source sampled at fs (like a random number generator), and to use an adjustable bandpass filter. Unfortunately, the bandwidth of the filter would be typically of few Hz over a ~20000 Hz wide noise process. An enormously high Q filter is needed: a large number of coefficients (and computation) for a FIR realization, or an IIR with close to unit-circle pole distribution (with phase, stability and coefficient quantization problems). Neither way seems attractive.

We can also generate a noisy process at a low (say 40 Hz) sampling rate, and then interpolate these samples to generate a low speed random process at the fs rate. This time it is the interpolating filter that needs a complex realization (with the same problems).

By means of FM synthesis it is easy to generate wide band white-like noise, but is still highly computing demandant to create a very thin bandwidth noise.

The method we have finally implemented, generates a random process at a (user adjusted) low sampling frequency, and then uses a raised cosine law to 'interpolate' available points to the fs sampling rate. The raised cosine warranties an infinitely derivable shape (avoids clicks), and also is free from the large uncontrolled excursions that may appear with low-order polynomial interpolation. This produces a DC centered noisy signal (with a non flat, thin bandwidth spectrum), that can then be centered on the frequency of interest by a point to point multiplication with a cosine law of the desired central frequency. Lets call fsm (sampling frequency for the noisy modulation signal) the frequency at which new random values are calculated. Our implementation uses an (adjustable) integer submultiple of fs (for example we can calculate one new random value each 4410 samples to get 10 new 'noise' values per second at fs 44.1 KHz, which -more or less- corresponds to a 5 Hz wide noise process). This simplifies computing requirements, but keeps good enough resolution (the noise 'period' is always very large in fs samples, and so we have high resolution to define it). The flat shape of the raised cosine extremes, makes it possible to dynamically change the noise generation speed (fsm), while keeping an endlessly derivable final noise shape, so that the 'randomness' can be changed through time.

The exact shape of the noisy control signal spectrum with this approach is easy to obtain, because the whole process is equivalent to the interpolation of the fsm noise to an fs sampling rate by means of convolution with a cosine shaped (or Hanning) window. This final spectral shape of the noise is not flat nor strictly concentrated around the central frequency of interest, but the result is still pleasing because it avoids discontinuities in any derivative and also uncontrolled excursions in between original sample values, keeping the benefit of using a non-periodic waveform. Implementation demands a pseudo-random generator at fsm rate (random generators are already available by hardware in many processors) and a cosine function (called at many times: for noisy control signal interpolation, for modulation of DC noise to the frequency of interest, and for the generation of the proper sinc values for the modulated delay -if they are not stored in ROM, as usual-), at a cost that is well within the current processors reach.

6 Hiding unwanted noise

As said before, and particularly for feedbacked flanger, the fact that the signal is summed with (delayed) versions of itself, greatly increases the noise in the final signal. This added noise is not static but changing so it becomes much more perceptually relevant.

To keep this undesirable noise down is a must. The multi-FX units of today make it possible to hide this noise at a low cost.

Current multi-FX just chain (in series or parallel) different FXs to create a more complex sound modification. There is room for improvement here, as some of the parameters calculated at one of the boxes can be used at another one with benefits at no cost.
Dynamic margin modifiers (like compressors, expanders, etc.) can be easily implemented in a multi-FX, as their computational load is very low. Appart of their own interest, they extract the energy envelope of the input signal: a measure of the level of the signal at each moment.

In a typical Chorus/Flanger then Expander/Gate (serial) configuration, the expander (or gate) decreases (nulls) the level of low energy fragments of the Chorus/Flange output, in order to hide the noise when there is no signal to mask it. Note that input signal for the Gate is the output of the Chorus (not really the original signal). Various well known artifacts do appear (like 'noise pumping').
A best approach is to use the envelope of the original input signal to control the ratio between original signal and delayed signal in the Chorus/Flanger. Here, the original signal is always to be found at the output (with no 'gating') for a completely natural sound (the user still has the option to add the gate if he wants to), and it is just the processed paths of the Chorus/Flanger that are controlled by the energy of the signal.

7 Multiscale vs. multivoice chorus

To increase the chorus (multi-player) sensation, a common approach is to use various chorus in parallel. Simpler units share a single LFO for the control of the varying delay (for example applying the same control signal with just a sign change). More complex units provide independent complete chorus units in parallel, each with its own LFO, for a more complex so-called multivoice chorus.

Risk of noise enhancement (as it happens in the feedbacked flanger) is increased, due to the larger number of processed voices that are added.

We have tried a different approach without having to sum up so many outputs (with their associated noise). The idea is to use a filterbank to split the original signal (and the noise) into bands, and then to apply a different chorus to each band. By adding the outputs of these choruses, a signal results wich is (basically) the result of the sum of the original signal and just one (though multiband) delayed version of itself. The effect is less obvious than the classical single voice/single channel chorus (where modulation law is the same for all components of the signal), but is not so prone to noise as a complete multi-voice chorus. The current implementation uses a simple filterbank (a diadic structure with octave splitting) and we call the resultant chorus 'multiscale' to differentiate it from the 'multivoice' approach.

For still better results, we have tested a combination of the 'multiscale' chorus with per-scale envelope tracking to automate the original/processed mix ratio at each channel of the filterbank. This approach enables frequency selective action of the chorus, so that bands with little energy are heard unprocessed, while bands with more energy are heard with a combination of original and processed sound.

When the multiscale chorus structure was used to generate flanger like process, we found that the 'metallic' character typical of flanger (due to harmonically related peaks and valleys of the notch filter), was easily changed into a less 'tonal' one, not unlike the difference between the flanger and the phaser.

The phaser is similar to flanger, but instead of changing the delay time (a fixed delay for all frequencies) moves the phase of the signals (equivalent to a frequency dependent delay). The phase response of the phaser is usually such that by mixing the original signal with the processed one, many nulls appear on the spectrum, but those nulls do not fall into a harmonic serie (like in flanger), but in a log-spaced set of frequencies. This spacing generates a sound that is not so ‘tonal’ (or ‘metallic’) as that of the flanger.

This non-tonal character is also easy to achieve in the multiscale structure, with the benefit that the behaviour can be constantly changed from that of a flanger-like effect to a phaser-like sound. If each scale uses its own independent chorus, non-metallic flanging results. On the other hand, if all scales share a single modulation law, ‘metallic’ character arises.

8 Applications of envelope signal

Changing the fsm value according to the envelope, the chorus/flanger can automatically increase its own randomness (a more complex sound for loud signals). Envelope driven feedback control in the flanger gives a more metallic sound during loud parts.

Envelope signal, applied to main delay (not to the variable delay part) can automatically change the color or tonal center of the flanger, so that changes in intensity are correlated with changes in tonal color of the processed sound, more intimately linking the effect results to the signal.

References

[1] Moore. 1990. *Elements of Computer Music*. Prentice Hall.

[2] Casajús-Quirós and Fernández-Cid. Notes for the course *Técnicas de Audio Digital*. ETSIT-UPM.