Sifft VST Manual

Sample Inverse Fast Fourier Transform

http://www.xoxos.net

Sifft VST is an open source FFT based wav player that allows separate control over time and pitch. The method of timestretching uses the phase difference between blocks to reproduce accurate pitch at the expense of creating diffusion. This result is not so good for minor adjustments but reproduces accurate pitch even with extreme modification (eg. 1024 times timestretching or much more is possible). The sample may also be scanned as if it were "frozen". With the inclusion of frequency shifting, Sifft is well suited for experimental sounds.

Why does it only load short samples?

The entire sample is stored in memory as a single array of FFT analyzed blocks. Using lower sample rates (eg. 22.05kHz) allows for longer samples to be used.

Why does it sound diffused?

FFT analysis converts a time signal into a frequency signal consisting of magnitude and phase data. High quality FFT timestretching is based on the phase vocoder algorithm (which I haven't done yet).

Using simpler techniques, you can timestretch with FFT by interpolating between the magnitude and phase of source blocks, but this will strongly pitch the result based on the length of the analysis block. The technique used here determines the difference in phase between subsequent blocks so that each frequency "rolls along" at its original pitch, however when the signal is reconstructed from the magnitude and phase, the frequency bins are misaligned. Each frequency in the original signal is still present, but an effect like a sparse reverb or dense chorus is created by the misalignment.

You'll note that when the sample is triggered from the beginning at the original pitch and time, it is reconstructed perfectly. If either is adjusted slightly, the diffusion is very noticeable. The results are very good with extreme stretching or with noisy signals like drums because the original frequencies are retained. I figured that most companies wouldn't release a product using this method and that it had some value in sound design.

How do I use Sifft? (How to set the buffer)

The best results use the shortest buffer length suited to the spectrum of the signal, eg. a buffer length of 512 samples is suitable for samples with frequencies as low as 43Hz in a host running at 44100 samples per second. To determine this, divide Nyquist, or half the host rate, by the buffer length (22050 / 512 = 43.0664). Frequencies in the sample lower than this will result in aliasing, which is why you are allowed to use a buffer length of 2 samples if you like that sort of thing.

It is also necessary to use a buffer length shorter than the total length of the sample.

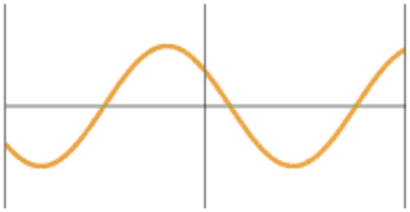
Assign controllers to the time and rotate controls, and fiddle around while it squeals and squawks.

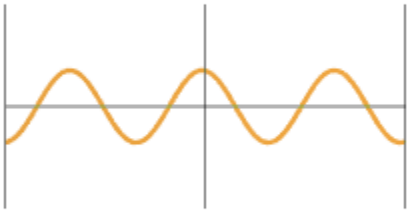
Why should I use Sifft?

Use this plugin if you can't afford anything else. Use it when you want to stretch a snare drum out into a ten minute drone. Do not use it if you want to make a spoken sample slightly longer... even a non-FFT time domain windowed process will give better results then. Sifft should be useful if you want to create new percussion samples. The frequency shifting ("rotate") is continuous and smooth (unless you have elected "block" mode).

What is Inverse FFT?

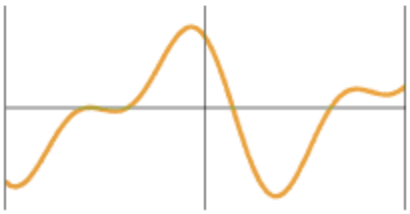
Inverse FFT is performed when an FFT analyzed signal is reconstructed and is a part of any FFT process (apart from pure analysis like a spectrum analyzer).

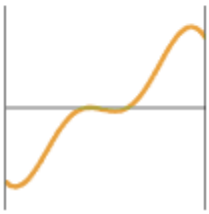


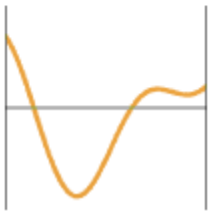


This "complex signal" is the sum of the two sines above.

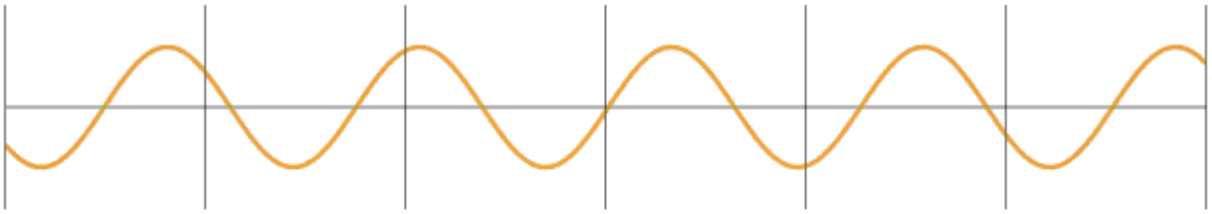
FFT analysis divides the signal into two "frames".

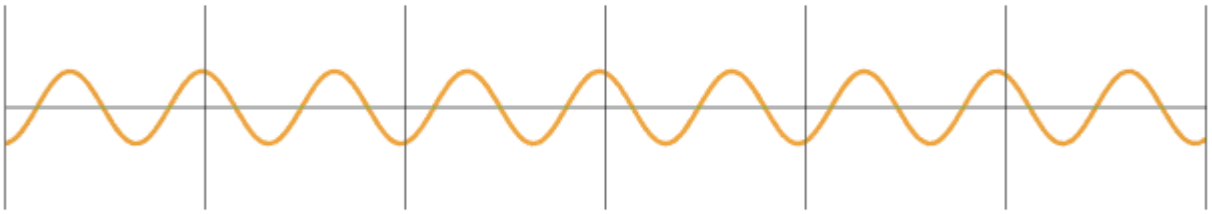


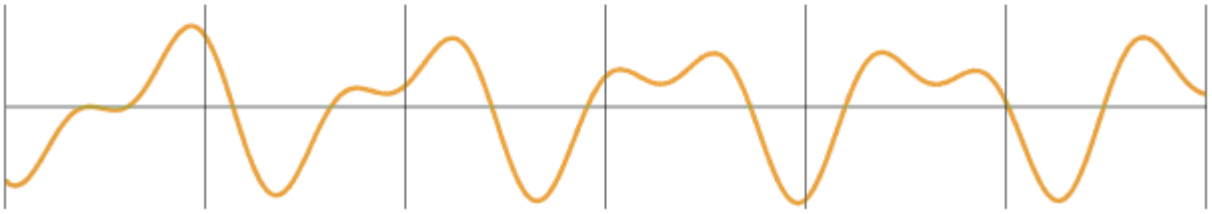




If we timestretched by interpolating between the two frames, the result would be "coherent" in that the relation between the phase of each frequency component would be retained, but the output would be strongly pitched by the length of the FFT analysis frame. You've probably heard this in some FFT effects.







Using this very simple example, if we wanted to stretch the two frames out to six frames, the phase relation between each frequency in the last frame (and the created intermediate frames) would become disparate, resulting in a diffused, or "reverby," smeared sound - but the pitch of each frequency component is accurately retained.

This example is overly simplified by the signal components in the end frame having the same frequency and amplitude as the first frame.. try and ignore that ;)

Rotation - Sifft and spin

The rotate function sounds keen. That's really all you need to know in order to use it.

When a signal is pitch shifted, the relationship between frequency components is retained, eg. a complex signal composed of 100Hz and 300Hz raised by an octave results in 200Hz and 600Hz.

Frequency shifting adds the same constant to each frequency. 100Hz and 300Hz shifted by 50Hz results in 150Hz and 350Hz. The closest recogniseable sound to this is poor radio reception.

Fourier analysis works by comparing the signal to sines equally spaced up to nyquist, or half the host sample rate. If we use a 4 bin analysis (as would happen with a buffer length of 4) then the signal is analysed at 1/4th nyquist, 2/4ths nyquist, 3/4ths nyquist and 4/4ths nyquist. Using interpolation, rotation 'spins' through the spectrum. The rotate parameter is continuous and can be modulated in either direction and will wrap completely around.

The usual stuff

Filters

There are three filter algorithms.. the famed Stilson/Smith Moog ladder, Andrew Simper's (Cytomic) open source trapezoidal SVF (which is nice because it attenuates around nyquist), and the usual Bristow-Johnson cookbook biquads. Each algorithm has low, band, high, cut and peak modes. Note that some of the modes are forcefully wrung out of the Moog emulation (eg. the cut filter forms a peak next to the cut with high resonance).

To conserve cpu, you can indicate how often you would like the filter coefficients to update. The filter stage includes emulated bitcrushing (this is performed on the line signal, not the sample).

LFOs

LFOs include sine, saw, ramp, triangle, pulse, saw-triangle, sample & hold, and random contours. The last 4 contours can be modulated. To create a square wave, mod must be at 50% using the pulse contour.

LFOs can be synced to the host at:

64m, 32m, 16m, 8m, 4m, 2m, 1m, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64,
1/48, 1/24, 1/12, 1/9, 1/6, 3/16, 5/16, 1/3, 3/8, 7/16, 2/3, 5/8, 3/4,
7/8, 5/4, 4/3, 3/2, 5/2, 3m, 7/2, 5m, 6m, 7m, 10m, 12m, 20m, 24m, 48m

They can also be synced to the phase position when a key is pressed.

Polyphonic or monophonic performance can be selected.

Envelopes

The knobs immediately under envelope sliders trim velocity responsivity. Higher velocity is configured to produce shorter attacks and longer decay/release.

The second row of knobs shape the contour of the attack and decay/release stages. The envelope is linear in the center position. To the left, the contour is bowed downwards, to the right the contour is bowed upwards. A classic analog envelope is bowed upwards in the attack stage and downwards in the decay/release stages.