CDP REVERB Functions

(with Command Line Usage)

Functions to REVERBERATE soundfiles
by R W Dobson

FASTCONV
Multi-channel FFT-based convolution
REVERB
Multi-channel reverberation (classic Schroeder)
ROOMRESP
Create early reflections data file for REVERB, ROOMVERB and TAPDELAY
ROOMVERB
Multi-channel reverberation with room simulation
TAPDELAY
Stereo multi-tapped delay line with feedback

Overview
Factors involved in reverberation

ALSO SEE:
ECHOES
Repeat a sound with timing and level adjustments between repeats
MODIFY REVECHO
Add reverberation or echo to the sound
EXTEND ITERATE
Repeat sound with subtle variations

Overview

Artifical reverberation seeks to recreate the acoustical properties of a real or imagined space, using a variety of delay techniques.

Reverberation is typically divided into two primary elements - the **early reflections**, consisting of the first relatively distinct echoes, and the **dense reverberation**, consisting of many thousands of diffused echoes.

The time-lapse between the direct signal and the first reflection is referred to as **'pre-delay'**, and is a significant parameter affecting not only the impression of the size of the space (echoes in a large space will naturally arrive later than those in a small space), but also the distance from the source to the listener.

Other important factors contributing to the realism or otherwise of the reverberation are the progressive loss of high frequencies with time, caused by **absorption** by the air and by surfaces, the relative strength of the direct and reverberated sound (suggesting distance from the source), and of course the overall **reverberation time** – which will be low for small rooms with soft furnishings, and very long for a large stone building such as a church or cathedral.

These factors interact in complex ways. It is widely acknowledged that the design of a good reverberator is not only a complex task, but also one requiring as much art as science. The reverberators provided here represent simple if classic designs, prividing enough control to the composer to enable a wide range of effects to be created, without making the process excessively complicated.

One of the most important factors in determining the overall character and realism is the **early reflections** – merely by changing these, it is possible to simulate both natural acoustic environments (with smooth random echoes) and artificial ones unrelated to any plausible physical space. For the emulation of physical spaces, the program ROOMRESP can be used; alternatively, it is possible to create early reflections by hand in a simple text format.

It can be said that is it very easy to create a 'bad' reverberator. However, similar techniques are widely used to create sounds such as cymbal crashes, and even early reverberation systems such as springs and plates. These typically arise when early reflections are almost synchronous, imparting a strong harmonic colouration to the sound.

Both Reverb programs include a set of simple **conditioning filters** for the input. Whereas the optional Highcut and Lowcut filters apply directly to the source, the *lpfreq* parameter is applied only to the reverberator input – not to the direct signal passed through. A lowpass filter is typically used when the source is required to sound far away. The Lowcut filter can be sued to remove excessive low frequency energy in the source (e.g the 'proximity effect' in a vocal recording).

Both REVERB and ROOMVERB can create a **multi-channel soundfile** of up to 16 channels. The default output is a stereo file, and inputs can be mono or stereo. In the case of multi-channel output, the direct signal is added to the first two channels only; the remaining channels contain only the reverb signal. Note that many of the parameters are identical in the two programs – the description of REVERB thus embraces both to a great extent.

FASTCONV – Multi-channel FFT-based convolution

Usage

fastconv [**-a***X*] [**-f**] *infile impulsefile outfile* [*dry*]

Example command line to create a convolved soundfile:

> **`fastconv -a0.7 count.wav sprrevm.wav countspr.wav 0.5`**
> (Soundfile extensions are mandatory. The *impulsefile* sprrevm.wav is a reverberated spring sound, Stereo converted to Mono.)

Parameters

-a*X* – scale output amplitude by *X* amount; values below 1 reduce the amplitude, and values above 1 increase it
-f – write the output as floats (no clipping). Play these with PVPLAY with the gain (level) reduced by MODIFY LOUDNESS, Mode **1**.
infile – input soundfile to be processed. It must be Mono, or have the same number of channels as the *impulsefile*.
impulsefile – soundfile or text file containing an impulse response, e.g., reverb or FIR filter. If text file, it must have the extension **.txt**. Supported channel combinations:

a. mono infile, N-channel impulsefile
b. channels are the same
c. mono impulsefile, N-channel infile

Note: some recorded reverb impulses typically include the direct signal. In such cases *dry* need not be used. Where *impulsefile* is a FIR filter, the optimum length is power-of-two minus 1. (See below)

outfile – output soundfile
dry – option to set a dry/wet mix, e.g., for reverb. Range: 0.0 to 1.0 (Default = 0.0): closer to 0 for dry, closer to 1 for wet. This uses the sin/cos law for constant power mix. (See below)

Understanding the FASTCONV Process

Convolution is, according to Wikipedia, "a mathematical operation on two functions, producing a third function that is typically viewed as a modified version of the original function." In this case, we are using an *impulsefile* (impulse response) to act upon an input soundfile.

With FASTCONV we have a specialised way to create reverberation. The key to its use lies in the *impulsefile*, which contains the reverberation information to be applied to the input soundfile. It was written to provide a fast and efficient way to carry out this process. The unexpected can happen when a soundfile is convolved using itself as the *impulsefile*.

Channel options

- When the *infile* is Mono, the *impulsefile* can be multi-channel, and the *outfile* has the channel count of the *impulsefile*.
- Where the *impulsefile* is Mono, data is duplicated for all input channels.
- When the *infile* is multi-channel, the *impulsefile* must be Mono, or have the same number of channels.
- The AMB file format (for Ambisonic B-Format audio) is supported, also *Logic Pro* SDIR files for reading (for the latter, change the file extension to **.aifc**).

It will be usual to supply a non-zero value for *dry*, e.g., 0.5. Note however that some recorded or synthetic impulse responses may already include a 'direct' component – meaning the sound of the source file without any additional reverberation. In such cases, a *dry* value may not be needed.

The program employs a power-averaging (rms) gain scaling algorithm which attempts to ensure all outputs are approximately at the same level as the input. In normal use (e.g., a naturally decaying reverb impulse response), the **-a** flag (scale output amplitude) should not be needed. When the **-f** flag is used (write output as floats), output is forced to floats, with no clipping. The program reports the output level together with a suggested corrective gain factor. This will be of particular relevance to more experimental procedures, such as convolving a soundfile with itself or with some other arbitrary source. In such extreme cases, automatic amplitude management becomes approximate at best!

Musical Applications

The primary application of FASTCONV is convolution-based reverberation using a sampled impulse response of a building or other responsive space. The term 'fast' refers to the use of the Fast Fourier Transform (FFT) to perform the convolution. The program can also be used more experimentally, as the impulse response input can be any Mono or Multi-channel file (see details of available channel combinations above). A file can also be convolved with itself. The sample rates of the two soundfiles must be the same.

The program uses double-precision processing throughout, and files of considerable length can be processed cleanly. Note however that the FFT of the whole impulse response is stored in main memory, so very large files may raise memory demands to critical levels. The input impulse response file is padded if necessary with silence to bring its size up to the next largest power-of-two size (required for the FFT process). This can result in a much larger memory footprint (in the worst case, twice as much) than the size of the file itself.

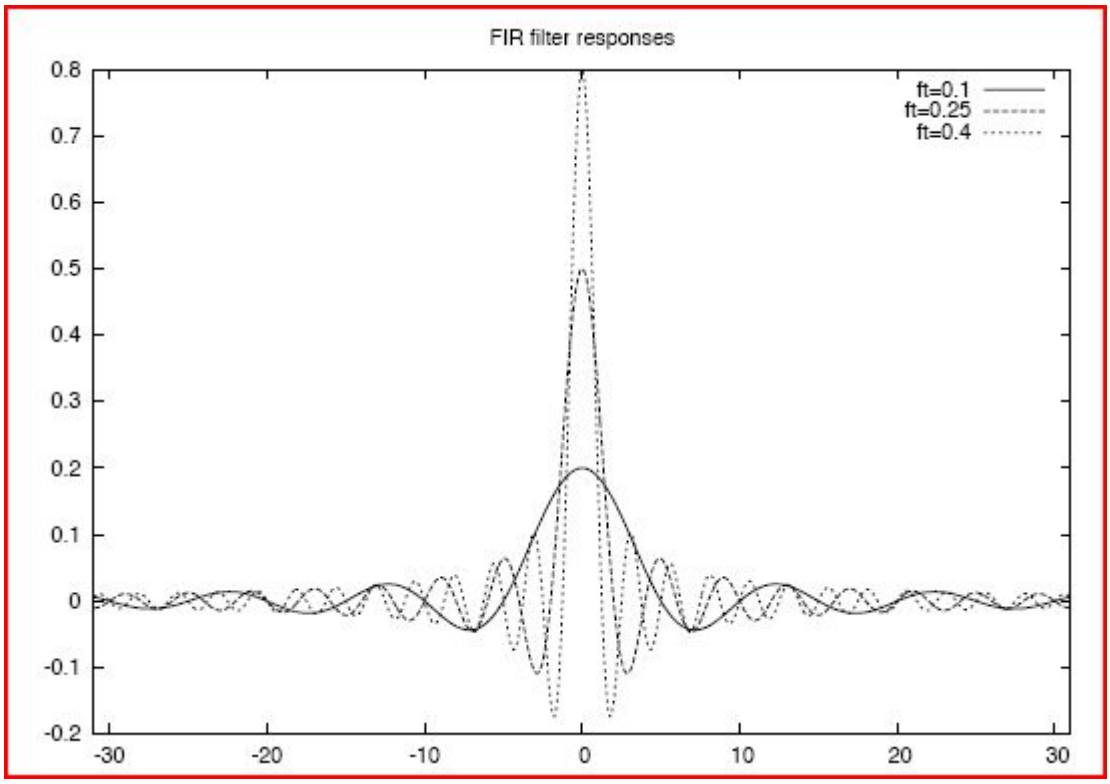
A future version of the program may implement 'partitioned convolution', which will not require this substantial memory overhead.

Note for advanced users – use for FIR linear-phase filtering.

Convolution implements a filtering process - equivalent to multiplication of the spectra of the two inputs. The most common filter in audio work is recursive – it recycles output samples back into the input. The output continues in principle forever, hence the term **infinite impulse response** (IIR). The advantage of this technique (as employed for example in CDP's **FILTER** group) is that even a low-order filter (i.e., using a small number of delayed inputs and outputs) can be very powerful in its effects.

A disadvantage in some applications can be that an IIR filter changes the relative phase of components in the input sometimes in a very non-linear way (i.e., frequency components are delayed by different amounts). This means, among other things, that the waveform shape of the input is not preserved, something which will be noticed especially with respect to sources with strong transients such as drums. Thus the timbre of the sound (even in regions not directly boosted or attenuated by the filter) will be changed. In common parlance, such a filter 'colours' the signal.

The alternative is a **linear phase filter**, which preserves all phase relationships in the source. All frequency components are delayed equally. The result is spectrally 'neutral'. To achieve this the impulse response must have a symmetrical shape (see illustration). The response decays identically either side of the central peak.



Finite Impulse Response (FIR) with all frequency components delayed equally

In this filter no outputs are re-injected into the filter; only delayed inputs are computed. Such a filter has a **Finite Impulse Response** (FIR). The impulse response data now comprises literally the complete response of the filter to a single input sample (impulse). An impulse response of 31 samples means that the filter generates and adds together 31 delayed copies of the input, to create each output sample.

While IIR filters may involve processing only two delayed samples (a 'second-order' filter), FIR filters need to employ many more samples to achieve similar effects. It would not be unusual to use a 511 th order FIR filter. This also means that the overall delay ('latency') of a FIR filter is much longer than that of an IIR filter.

A FIR filter cannot resonate as an IIR filter can. By computing only delayed inputs, it is unconditionally 'stable' – whereas a badly designed IIR filter can 'blow up' with output values rapidly exceeding the sample limit.

FASTCONV supports the use of FIR coefficient files either in the form of either a short soundfile, or a plain text file containing (in a single column) the list of sample values ('coefficients') as floating-point numbers within the 'normalised' range -1.0 to 1.0. Note that a text file offers no information on sampling rate – it is therefore the responsibility of the user to ensure that a given text file is appropriate to the source soundfile being processed. For orthodox filtering purposes a Mono soundfile should be used, to process all channels identically. Only Mono text files (one column) are supported. Such FIR coefficient text files are generated by many engineering-oriented filter design and digital signal processing tools. Users may be tempted to write response files by hand; this can be done, but the results will be virtually impossible to predict or control.

As noted above, for maximum efficiency such files should ideally have a size that is a power-of-two less one: e.g. 255, 511, 1023, etc. An odd-length impulse response causes all input samples to be delayed by an exact number of samples, facilitating any later adjustments to compensate for the net delay through the filter.

For more information about FIR filter design, see:
http://www.labbookpages.co.uk/audio/firWindowing.html.

For a detailed mathematical discussion of convolution, see the **Wikipedia article** on it. It is a fascinating and non-trivial subject. The link to cross-correlation is particularly interesting.

RWD: August 23 2010

End of FASTCONV

REVERB – Multi-Channel reverberation

Usage

reverb [flags] *infile outfile egain mix rvbtime absorb lpfreq trailertime* [*times.txt*]

Parameters

flags – any or all of the following:

-c*N* – create *outfile* with *N* channels (Range: 1 <= *N* < 16; Default = 2)
-e*FILE* – read early reflections from breakpoint file *FILE*
-f – write output as floating point (Default: format of *infile*)
-H*N* – apply Highcut filter to *infile* with cutoff frequency *N*Hz (6dB per octave)
-L*N* – apply Lowcut filter to *infile* with cutoff frequency *N*Hz (12dB per octave)
-p*N* – set reverb pre-delay to *N* msecs (shifts early reflections)

infile – soundfile to process
outfile – reverberated output
egain – set level of early reflections (Range: 0.0 to 1.0)
mix – dry/wet balance of direct and reverb signal
Range: 1.0 (weighted towards direct/'dry' signal) to 0.0 (weighted towards reverb/'wet' signal)
rvbtime – reverb decay time (to -60dB) in seconds
absorb – degree of high-freq. damping to suggest room size (Range: 0.0 to 1.0)
lpfreq – lowpass filter cutoff frequency in Hz applied at input to reverb
trailertime – time in seconds added to *outfile* for reverb tail
times.txt – list of delay times in milliseconds for 6 comb and 4 allpass filters

Understanding the REVERB Process

REVERB implements the now classic Schroeder/Moorer model, consisting of six comb filters in parallel, followed by four allpass filters in series. The comb filters generate the dense reverberation, and the allpass filters (with much shorter delay times) apply further smearing of the echoes to minimize the spectral colouration of the comb filters. A further allpass is applied to each output channel, each with a different randomly-chosen delay time.

Each comb filter contains a simple low-pass filter to simulate high-frequency absorption - this also affects the overall reverberation time. A preset set of early reflections as defined by Moorer is incorporated; this can be replaced by a user-defined set either hand-written or created using ROOMRESP. The delay times for the filters are preset to suit a 'medium room' model, as suggested by Moorer. In most situations the user will not want to change these; however the option is provided to change these times by means of a simple text file.

Further discussion of the parameters:
-ppredelay – Whether using the preset early reflections, or a custom set, setting a value for predelay adjusts all the early reflections earlier or later. This may be useful especially when simulating large spaces, but where the listener is presumed to be close to the source - the time between the direct sound and the first echo signifies the time for the direct sound to reach the listener.

egain – the relative level of the early reflections has a marked impact on the intensity and colour of the reverb. The setting for this parameter depends somewhat on the general level of the early reflections as a whole.

- Using those preset into REVERB, setting *egain* between 0.2 and 0.5 will produce a natural-sounding range between 'warm' and 'bright'.
- If the level is set very low, the reverb will consist mostly of the dense reverberation, which will also tend to fade more quickly, as it is receiving less energy from the early reflections.
- On the other hand, setting *egain* high will impart not only increased energy into the reverb (hence louder and longer) but also increased colouration.
- Adjusting *egain* will be specially important when importing custom early reflections data, whether created by hand or using **ROOMRESP**.

mix – For a typical 'small hall' simulation, mix should be high – betweeen 0.6 and 0.8. This corresponds to a listener position relatively close to the source. If it is required to suggest a distant sound in a large space, where the reverb sound may be almost as strong as the direct signal, a value of 0.5 or less can be used. It is also possible to remove the direct sound entirely if necessary, for example, to enable other processing of the reverb sound.

rvbtime – Reverberation time is usually defined as the time taken for the reverb to decay by 60dB. The program attempts to calculate this taking into account the amount of high-frequency damping used (absorb). Reverb time is dependent not only on room size, but also on the 'liveliness' of the space. Walls of brick or stone reflect a high proportion of the sound, thus extending reverberation time, whereas the presence of soft furnishings or a large audience will reduce it significantly. It will sound strange to set a long reverb time combined with a high degreee of HF absorption. In this program a true 'infinite reverb' cannot be set, though it can be very long!

According to the value of *rbvtime*, one of three preset sets of early reflections is used, representing small, medium and large rooms. Information about this is printed to the console when the program is run. Note that both the early reflections and pre-delay can be changed. The roomsize thresholds are:

- SMALL: 0.0 up to 0.6
- MEDIUM: above 0.6 up to 1.3
- LARGE: above 1.3

absorb – this is always required for realistic reverberation, a typical value will be around 0.7 (soft furnishings) to 0.3 (hard surfaces); *absorb* can be bypassed by setting a value of 0.

lpfreq: – reverberation typically loses the high-frequency components of a sound. Much of this is controlled by *absorb*, but it can be useful to be able to limit high frequencies in the source at the input to the reverberator, especially if the source is synthetic in origin. Also, as high frequencies tend to be lost with increasing distance, setting *lpfreq* (together with use of the **-H** flag – use a higher frequency here than for *lpfreq*) can assist in suggesting a distant source (e.g., in large open-air spaces such as canyons). Typical values are range between 5000 and 10000 Hz (depending on the sample rate); *lpfreq* can be bypassed by setting a value of 0.

trailertime – extra time added to the outfile for the reverb tail. Use a non-zero value if the source sound continues to the end of the file, otherwise the reverb tail may be cut off. Depending on the loudness of the input, *trailertime* can typically be equal to or slightly less than *rvbtime*.

times.txt – for experimenters! The internally preset times used are (milliseconds):

```
50 56 61 68 72 78 20 14 9 6
```

Note that the order of the first six is not relevant, as the comb filters are arranged in parallel. The four allpass filters are connected in series, so that in this case the relative ordering of the delays is very significant. Internally the delay times are converted into prime-number integral sample delays – this helps to avoid coincidence of echoes, which would lead to a highly coloured and uneven response.

End of REVERB

ROOMRESP – Create early reflections data file for REVERB, ROOMVERB and TAPDELAY

Usage

roomresp [**-a***maxamp*] [**-r***res*] *txtout.dat liveness nrefs*
roomL roomW roomH
sourceL sourceW sourceH
listenerL listenerW listenerH

*[**Note – in the CDP distribution, the program name may be "rmresp", not "roomresp". In that case, either change the program name or use "rmresp" in the command line.**]*

Parameters

-a*maxamp* – peak amplitude for data (Range: 0.0 to 1.0; Default: 1.0)
-r*res* – time resolution in milliseconds for reflections (Range: 0.1 to <= 2; Default: 0.1)
txtout.dat – text output file containing early reflections in breakpoint format suitable for input to **ROOMVERB**, **REVERB**,or **TAPDELAY**
liveness – degree of reflection from each surface (Range: 0 to 1; typical value: 0.95)
nrefs – number of reflections from each surface (> 0; typical value: 2 to 5). **Warning:** high values will create extremely long data files!
ROOM PARAMETERS:
roomL roomW roomH – room size (Length, Width, Height)
sourceL sourceW sourceH – position of sound source (as above)
listenerL listenerW listenerH – position of listener (as above

- All dimensions are in metres.
- The first output time is non-zero.

Understanding the ROOMRESP function

This program uses a simple ray-tracing model as widely used in computer graphics, based on the dimensions of a rectangular room.

Reverberation arises when a sound, as well as reaching listeners directly, also reaches them after bouncing several times off one or more surfaces (floor, walls, ceiling). Depending on the degree of reflectivity or *liveness* of these surfaces (e.g. stone walls reflect almost all the sound, whereas soft furnishings reflect much less) many or few reflections will reach the listeners, who will perceive discrete echoes or a smooth ambience, and a longer or shorter reverberation time.

Since the room model used is a simple rectangular shape, with parallel walls, it is possible for reflections to form regular patterns. For smooth reverberation this needs to be minimised; this can be achieved by placing the listener (and possibly the source too) assymetrically in the room – i.e., not equidistant between two walls. On the other hand, it is easy to generate the sorts of highly coloured reflections experienced in bathrooms, tanks and similar spaces.

Thus a typical arrangement for a medium room might be (all dimensions in metres):

ROOM:	length = 20	width = 11	height = 3.5
SOURCE:	length = 4.5	width = 5	height = 2.5
LISTENER:	length = 17	width = 6	height = 2

The number of reflections generated depends on *nrefs*. Useful values are between 2 and 5; the latter will generate approximately 876 taps. In theory there should be even more, but many echoes arrive virtually at the same time, and these are averaged together to reduce the overall data dize. The*liveness*parameter does not affect the number of reflections, but their amplitude. When a lowish level for *liveness* is set, many of the generated taps will have very low amplitudes (they would be submerged in the dense reverberation), and it will be reasonable to delete these to save processing time in the reverb programs.

The amplitude of the output taps is internally scaled ('normalized') to 1.0. The data can be evaluated by applying it to TAPDELAY, and once a satisfactory level is found (using the *tapgain* parameter in TAPDELAY), this can be used with the **-a** flag to generate a new data file suitable to apply to the reverb programs. The *egain* parameter in REVERB and ROOMVERB can be used to the same end. As ever, experimentation is the rule!

End of ROOMRESP

ROOMVERB – Multi-channel reverb with room simulation

Usage

roomverb [flags] *infile outfile rmsize egain mix fback absorb lpfreq trtime*

*[**Note – in the CDP distribution, the program name may be "rmverb", not "roomverb". In that case, either change the program name or use "rmverb" in the command line.**]*

Parameters

flags – any or all of the following:

-p*predelay* – override early reflections delay in milliseconds
-c*N* – create *N*-channel *outfile* (Default:2)
-e*early.brk* – load early reflections data from breakpoint file
-f – write output as floating point (Default: format of *infile*)
-L*N* – apply Lowcut filter to *infile* with cutoff frequency *N*Hz (12dB per octave)
-H*N* – apply Highcut filter to *infile* with cutoff frequency *N*Hz(6dB per octave)

infile – input soundfile to process
outfile – reverberated output
rmsize – **1** (small), **2** (medium), or **3** (large)
egain – set level of early reflections (Range: 0.0 to 1.0)
mix – dry/wet balance of direct and reverb signal
Range: 1.0 (weighted towards direct/'dry' signal) to 0.0 (weighted towards reverb/'wet' signal)
fback – reverb feedback level: controls decaytime (Range: 0.0 to 1.0)
absorb – degree of high-freq. damping to simulate air absorption (Range: 0.0 to 1.0)
lpfreq – lowpass filter cutoff frequency in Hz applied at input to reverb

Use the value **0** to disable either *absorb* or *lpfreq*.

trailertime – time in seconds added to *outfile* for reverb tail

Understanding the ROOMVERB Process

Although many of the parameters are the same as those for **REVERB**, the program operates very differently. Rather than use comb filters in parallel, ROOMVERB uses a variable network of 'nested' allpass filters inside an overall feedback loop. This has the effect of increasing the density of the reverberation over time, as is characteristic of most 'real' acoustic spaces. It is therefore dedicated more specifically to that task than is REVERB. Nevertheless, as the roomsize and feedback parameters are independently controllable, some unusal effects can still be created.

For authentic reverb, it is important that the feedback level is not set too high, otherwise audible pulsations can be heard at the start of the reverberation. These can be mitigated to a degree by running **REVERB** several times in parallel, with slightly different reverb times, and mixing the results, or combining into a multi-channel file. Conversely, it is also possible to create an 'infinite reverb' effect by setting the feedback level to 1.0.

The preset early reflections for a given room size are the same as those in **REVERB**.

End of ROOMVERB

TAPDELAY – stereo multi-echo generator with feedback

Usage

tapdelay [**-f**] *infile outfile tapgain feedback mix taps.txt [trailtime]*

Parameters

-f – write floating-point *outfile* (Default: *outfile* has same format as *infile*)
infile – input soundfile to process
outfile – reverberated output
tapgain – gain factor applied to output from delay line (Range: > 0.0; typical value: 0.25)
feedback – delayed signal fed back into the input (Range: -1.0 to 1.0)
mix – proportion of source mixed with delay output (Range: 0.0 to < 1.0)
taps.txt – text file consisting of a list of taps in the format: *time amp* [*pan*]

- All values must be floating-point, one tap per line.
- Times (in seconds) must be increasing. Duplicate times are ignored.
- A zero time (no delay) overrides the mix parameter, and determines the level and pan of the (mono) input.
- Amplitude values must be in the range 0.0 to 1.0.
- Empty lines are permitted, as are lines starting with a semi-colon, which may be used for comments.
- If a *pan* value is used in any line, the *outfile* will be stereo.
- *Pan* values are nominally in the range -1 to +1, with 0 as centre (mono), within which range constant-power panning is used.
- Values beyond these limits result in attenuation according to the inverse-square law, to suggest distance beyond the speakers.

trailtime – extra time in seconds (beyond *infile* duration) for delays to play out

Understanding the TAPDELAY Process

The principle behind a tapped delay line is that whereas in a standard delay line (e.g., 1 second long) the ouput is taken at the end, in a tapped delay line extra outputs are also taken at intermediate points, such as 0.1 secs, 0.3 secs and 0.75 secs.

The *taps.txt* file corresonding to these delays would be as follows:

```
0.1   0.2
0.3   0.9
0.75  0.05
1.0   1.0
```

In each line the second number sets the amplitude (with respect to the range 0.0 to 1.0) for the tap at the given time.

Without the use of feedback, this would apply a simple rhythm (with varying emphases per repeat) to the input sound. When feedback is applied, each of these tapped signals is returned to the input, so that each then generates further delays in the same pattern.

It is also possible to direct a tap to a position in the stereo field, using the same 'equal-poser' method used in PAN (see **MODIFY SPACE**). A third (optional) parameter is used in this case, for example, in the range -1 (full left) to 1 (full right).

The following illustrates a stereo four-stage tap. The final tap is at Centre:

```
;stereo 4-stage tap - final tap at centre
0.1             0.2         0.75
0.3             0.9        -0.5
0.75            0.05        0.1
1.0             1.0
```

This example also illustrates the use of comment lines, which require a semi-colon (';') at the beginning of the line.

So long as one position parameter is used in a line, the *outfile* will be stereo.

When *feedback* is applied, the stereo position for each tap is preserved. One musical consideration here is that the two channels may decay at different rates, depending on the total amplitude set for each side.

The *tapgain* parameter provides addition control over the output from the delay line. Although some internal gain adjustments are carried out, they cannot anticipate all the circumstances which a user may impose. The output of ROOMRESP can contain varying proportions of high and low-amplitude taps, for example, and the internal rescaling may result in an inappropriate signal level.

In the usual case, the first tap time will be non-zero, and *mix* controls the balance of direct and delayed signal. Stereo files are internally mixed to mono before entering the delay line. It is however possible to set a tap time of zero (for the first line only!). In this case, the position parameter is used to place the direct signal in an arbitrary position in the stereo field. This overrides *mix* which is ignored. This facility also enables mono tapfiles (only) to be edited in the CDP graphic breakpoint editor, *BrkEdit*. Normal breakpoint files require a first time of zero), though it must be borne in mind that **a tapfile is not a breakpoint file** – no intermediate times and positions are calculated.

Musical Applications

The natural application of TAPDELAY is with short impulsive sounds such as plucked strings and percussion. A variety of regular and irregular mono or stereo multiple echoes can be generated. A single tap can be used, to throw an echo to a position in the stereo field. There is no limit on the number of taps that can be specified, or on maximum delay, though large numbers will naturally demand more processing time, and large delay times may exceed available memory.

Short delay times combined with feedback can create a variety of quasi-reverberant effects (see also **ROOMRESP**). Longer tap times with feedback can generate chaotic (and possibly over-range) outputs quite quickly. Unless you have an extremely fast computer, it will be inadvisable to attempt to use TAPDELAY to create reverb, as this is effectively a process of convolution, which with many hundreds or even thousands or calculations per sample, will take a very long time! For this task, **ROOMVERB** or **REVERB** is much more suitable; in these progams the same tapped delay line system is used, for early reflections.

ALSO SEE: ECHOES, which repeats a sound with timing and level adjustments between repeats.

End of TAPDELAY

Last Updated 11 July 2015
Documentation: Richard Dobson, edited Archer Endrich; revised R. Fraser