CDP ENVEL Functions

(with Command Line Usage)

Functions to ENVELOPE soundfiles

(Names in brackets mean that these are separate programs. The others are sub-modules of ENVEL.)

ATTACK
Emphasize the attack of a sound
BRKTOENV
Convert a (text) breakpoint envelope to a binary envelope file
CREATE
Create an envelope
CURTAIL
Curtail a soundfile by fading to zero at some time within it
CYCLIC
Create a sequence of repeated envelopes, in a binary envelope file
DBTOENV
Convert a (text) breakpoint file with dB values to a binary envelope file
DBTOGAIN
Convert a (text) breakpoint file with dB values to gain values (0 to 1)
DOVETAIL
Dovetail a soundfile by enveloping its beginning and end
ENVTOBRK
Convert a binary envelope file to a (text) breakpoint envelope file
ENVTODB
Convert a binary envelope file to a (text) breakpoint envelope file with dB values
EXTRACT
Extract the amplitude envelope from an input soundfile
GAINTODB
Convert a (text) breakpoint file with gain (0 to 1) values to dB values
IMPOSE
Impose an envelope on an input soundfile
PLUCK
Pluck the start of a sound (Mono files only)
REPLACE
Replace the existing envelope of a soundfile with a different envelope
RESHAPE
Warp the envelope in a binary envelope file
REPLOT
Warp the envelope in a (text) breakpoint envelope file
SCALED
Impose an envelope on an input soundfile, scaling it timewise to the sound's duration
SWELL
Cause sound to fade in to and out from a peak moment
TIMEGRID
Partition a soundfile into a sequence of 'windows' separated by silence

[TOPANTAIL2]
Fade beginning & end of a sound
TREMOLO
Tremolo a sound
[TREMOLO]
Width-controlled tremolo
WARP
Warp the envelope of a soundfile

ALSO SEE:
[FLUTTER]
Add multi-channel-distributed tremolo to a multi-channel file
ENVNU EXPDECAY
Produce a true exponential decay to zero on a sound
ENVNU PEAKCHOP
Isolate peaks in a source and either play back at a specified tempo (Mode **1**) or output a peak-isolating envelope (Mode **2**)

ENVEL ATTACK – Emphasize the attack of a sound

Usage

envel attack 1 *infile outfile gate gain onset decay* [**-t***envtype*]
envel attack 2 *infile outfile time gain onset decay* [**-t***envtype*]
envel attack 3 *infile outfile time gain onset decay* [**-t***envtype*]
envel attack 4 *infile outfile gain onset decay* [**-t***envtype*]

Modes

1 Set attack point where sound level first exceeds gate level
2 Attack point at maximum level at your 'approx-time' (+/- 200 ms)
3 Attack point at your 'exact-time'
4 Attack point at maximum level in the soundfile

Parameters

gain – floating point multiplier by which to amplify the signal at the attack point (< 1 reduces the signal; > 1 amplifies the signal)
gate – level (threshold) to be reached for the attack point to be recognised (Range: 0 to 1)
time – time (approximate or exact) of the attack point, in seconds
onset – attack onset duration in milliseconds, i.e., the 'sharpness' of the attack (Range:5ms to 32767ms)
decay – attack decay duration in milliseconds (Range: 5 to < duration of *infile*)
-t*envtype* – type of envelope: **0** linear, **1** exponential (Default)

Understanding the ENVEL ATTACK Process

ENVEL ATTACK amplifies the beginning of a sound by applying an attack/decay envelope, starting at a point at which the amplitude of the *infile* exceeds a threshold value set by *gate*. The peak level of the attack is set by *gain*. This is a multiplier (as used in GAIN, for example); therefore the *infile* (in the attack region) must be at substantially less than maximum amplitude for the program to work as intended. The parameters *onset*, *decay* and *envtype* provide independent control over the length and sharpness of the attack and decay.

gate – To use this program effectively, you need to examine the soundfile at the sample level with a soundfile viewer such as VIEWSF. The first step is to find where the signal level starts to rise to a significant level. This is often not at the very beginning of the soundfile, which means that putting the attack right at the beginning would have no effect. Suppose that the level reaches 1000 by 0.02 sec. (20ms) and then continues upwards. The signal before this time is too low for the attack operation, so one sets a *gate* of 1000; now the attack will begin when the amplitude reaches 1000. *Gate* is used in Mode **1** only.

gain – This is the multiplier by which the original level of the sound in the attack region is increased to produce the attack. You need to observe the amplitude level of the signal just after the point where the attack will begin to see how much amplification it can sustain before overloading. This information is used to set a sensible *gain* level.

SNDINFO MAXSAMP will also give an indication of how much room for manoeuvre there is with the signal level. For example, if the maximum sample is shown to be 10560 at 1.2 seconds, a *gain* of at least 3 will be safe, and probably considerably more if you can hear that the sound begins with a soft attack.

ENVEL ATTACK checks the levels and will not proceed it it reckons that there will be an overload. In this case, you need to reduce the *gain* level.

The attack can be placed anywhere in the sound, so this function can be used, for example, to 'pepper' a sound with surprise impulses. The effectiveness will be determined by how much scope for level increase you have at a particular point in the sound.

onset – This sets the time between the moment when the attack starts and the end of the *decay*. It can therefore be referred to as the duration of the whole attack cycle. Given an attack peak at a certain time, and a *decay* of a certain duration, the *onset* is the time remaining for the build-up to the attack peak. A longer *onset* will simply cause the build-up to begin sooner. If this start point is mathematically before the beginning of the soundfile, it will begin at the start of the soundfile. The maximum range is 32,767 seconds, so it will be possible to create a very long attack if you wish, as long as the attack peak is some way into the soundfile. The value for *onset* does not affect the time placement of the attack peak.

decay – The decay portion of the attack is also set in milliseconds. Note that you could make the decay very long. This is the 'D' part of the 'ADSR' found on some synthesisers and is a very important parameter. Making it shorter than *onset* helps bring out the plosive nature of an attack. Making it as long or longer than *onset* smooths the transition into the rest of the sound.

envtype – Two types of envelope are provided: straight line (linear) or curved (exponential).

I (Editor) always seem to pause slightly when converting seconds to milliseconds and *v.vs.*, even though it's a simple matter of moving the decimal point three places to the right or left: 0.1 sec = 100 ms. So for my own reference, and perhaps yours, here's a little chart of conversions. The third column is to help imagine the numbers as parts of a beat @ crotchet (quarter note) = 60.

Seconds & Milliseconds Reference Chart

Seconds	Milliseconds	Number per second
0.001	1	1000
0.010	10	100
0.100	100	10
0.025	25	40
0.040	40	25
0.050	50	20
0.25	250	4
0.500	500	2
1.000	1000	1

Musical Applications

To:

- put extra 'bite' into the start of a sound
- emulate the attack transient of a particular musical instrument
- introduce a surprise pulsation into a sound
- 'pepper' a sound with impulses
- make longish exponential shapes at the start of or within a sound

Note that ENVEL ATTACK can be used to introduce a plosive quality to the beginning of a sound, while retaining its timbral characteristics. This differs from ENVEL PLUCK, which introduces (pronounced) noise elements into the attack.

Also see: **ENVEL CREATE**, **ENVEL WARP**, and **ENVEL PLUCK**.

End of ENVEL ATTACK

ENVEL BRKTOENV – Convert a (text) breakpoint envelope to a binary envelope file

Usage

envel brktoenv *inbrkfile outenvfile wsize*

Parameters

inbrkfile – input textfile containing *time value* breakpoint pairs
outenvfile – output binary envelope file
wsize – enveloping window: time resolution in milliseconds over which to average amplitude data (Range: 5 to length of file)

Understanding the ENVEL BRKTOENV Function

The text format is an editable *time value* text breakpoint file. The binary format is used by some processes of the system and cannot be edited.

Musical Applications

This is a utility to convert text format to binary format.

Also see: **ENVEL ENVTOBRK**.

End of ENVEL BRKTOENV

ENVEL CREATE – Create an envelope

Usage

envel create 1 *envfile createfile wsize*
envel create 2 *brkfile createfile*

Modes

1 the output is a binary envelope file
2 the output is a breakpoint text file

Parameters

envfile – output binary envelope file in Mode **1**

If you specify a start time > 0, the values from 0 to the start time hold your start level.

brkfile – output text breakpoint envelope file in Mode **2**

The file starts at the time you specify.

createfile – input text envelope data file in the following format:

- *time* [**e**]*level*
  time [**e**]*level* ....
- *time level* pairs can be repeated as often as desired
- *time* is in seconds, must increase throughout the file, and must be separated by at least 0.010 sec (10ms).
- *level* is a number between 0.0 and 1.0, or a dB value between -96dB and 0dB (you must write 'dB' if you want dB).
- If preceded by an **e**, the envelope rises /falls **exponentially** to that level.
- Default: **linear** rise/fall.

wsize – enveloping window: time resolution in milliseconds over which to average emplitude data (Range: 5 to length of file)

Understanding the ENVEL CREATE Function

ENVEL CREATE provides a facility for hand-crafting an envelope shape. There can be as many 'points' as you like, and specific segments may be given an exponential curve, rather than a straight line shape, by preceding the value with an **e**.

It may be useful to study the existing envelope of a soundfile in a soundfile viewer before creating your own envelope file, because it will need to relate to the existing values in order to produce predictable results.

Virtually simultaneous jumps can be accomplished by making the next time increment the minimum 0.010 sec allowed (10ms).

Windows users may also use the graphic **BRKEDIT** program.

Mode **2** may at first seem redundant because both the input and the output are text breakpoint files. The reason it is here is that the *createfile* may contain **-e**'s for exponential curves. These are appropriately converted in the output textfile.

Musical Applications

Here are just a few ideas among the myriad possibilities:

- create very long, gradual changes in amplitude
- create many ragged, jagged changes in amplitude
- introduce regular, slow amplitude pulsations
- pull down the amplitude after the attack transient
- tailor the attack and/or decay transients

End of ENVEL CREATE

ENVEL CURTAIL – Curtail a soundfile by fading to zero at some time within it

Usage

envel curtail 1 *insndfile outfile fadestart fadeend envtype* [**-t***times*]
envel curtail 2 *insndfile outfile fadestart fade-dur envtype* [**-t***times*]
envel curtail 3 *insndfile outfile fadestart envtype* [**-t***times*]
envel curtail 4 *insndfile outfile fadestart fadeend* [**-t***times*]
envel curtail 5 *insndfile outfile fadestart fade-dur* [**-t***times*]
envel curtail 6 *insndfile outfile fadestart* [**-t***times*]

Modes

1 specify the precise times at which the file is to begin and end
2 specify the duration of the fade and when it begins
3 specify when the fade begins and fade to the existing end of *insndfile*
4 specify the precise times at which the file is to begin and end, and apply a doubly exponential (steeper) editing slope
5 specify the duration of the fade and when it begins, and apply a doubly exponential (steeper) editing slope
6 specify when the fade begins and fade to the existing end of *insndfile*, applying a doubly exponential (steeper) editing slope

Parameters

insndfile – input soundfile to curtail
outfile – output (curtailed) soundfile
fadestart – time at which to begin the fade
fadeend – time at which to complete the fade (In Modes **3** and **6** the end-time is presumed to be the end of *insndfile*).
envtype – type of envelope shape for the fade

envtype = **0**: **linear** fade
envtype = **1**: **exponential** fade (the Default)
In Modes **4-6** the slope is automatically doubly exponential.

-t*times* – identifies the type of time unit being used:

time = **0**: times are in seconds (the Default)
time = **1**: times are in samples
time = **2**: times are in grouped-samples; e.g., in stereo, each sample-pair counts as 1 item

Understanding the ENVEL CURTAIL Process

There may be times when it is useful to make a soundfile a certain length. If it is all right to lose the latter part of the sound, then ENVEL CURTAIL can be used to specify the time at which the sound is to end (*fadeend*), and to customise the fade to this end point: where the fade is to begin (*fadestart*) and how long it is to last (*fadeendr* in Mode **1** and *fade-dur*) in Mode **2**).

The linear *envtype* makes the fade a little less gradual.

Musical Applications

Here we have a handy utility to cut soundfiles to a given length while at the same time customising the manner in which they end. It gives more control over shaping the final part of the soundfile than is possible with **SFEDIT CUT**.

End of ENVEL CURTAIL

ENVEL CYCLIC – Create a sequence of repeated envelopes, in a binary envelope file

Usage

envel cyclic 1-3 *outenvfile.evl wsize total-dur cell-dur phase trough env-exp*
envel cyclic 4 *outenvfile.evl userenv.brk wsize total-dur cell-dur phase*

Modes

1 Rising
2 Falling
3 Troughed (falls then rises)
4 User-defined (in a text breakpoint file)

Parameters

outenvfile.evl – name of the output binary envelope file created by the program in Modes **1-3**, shown here using the CDP extension for binary envelope files (**.evl**).
wsize – envelope window size in milliseconds, i.e., the resolution of the envelope generated
total-dur – duration of the output soundfile file
cell-dur – duration of (repeating) envelope-cell
phase – where in the cell-envelope to begin the output: **0** = start, **1** = end of cell
trough – the lowest point of the envelope (Range: 0 to 1)
env-exp – shapes are linear (0) or exponential (< 1 = start slow and speed up: steep at top; > 1 = start quickly and slow down: steep at bottom)
userenv – a user-defined textfile of *time value* pairs defining the envelope, in which the time units are arbitrary: the envelope is stretched to fit into each of the envelope cells.

Cell-dur and *trough* may vary over time.

Understanding the ENVEL CYCLIC Function

ENVEL CYCLIC creates a binary envelope file that cyclically repeats the shape that you give to it, at whatever repetition rate you want. It is therefore a way to rhythmicise a continuous sound by imposing a regularly pulsating envelope.

The program is very flexible, allowing you to set the output duration, the duration of the repeating contour shapes (time-variable), phase (start point in the shape), type and depth of shape.

The window-size (*wsize*) determines how accurately the output envelope is defined:

- a small window-size gives a very precisely defined envelope
- a larger window-size gives a less well-defined envelope

Mode **4** accepts a (special) text breakpoint file with which you can define an envelope shape. What is special about this user-defined breakpoint file is that its time-units are relative. You the user define the pulse shape between *times* 0 and 1, or between *times* 0 and 1000, but the actual duration will be scaled up or down to fit the duration of (each) *cell-dur*. *Cell-dur* may be a constant, or it may itself be

defined in a time-varying breakpoint file. The shape will then be stretched or compressed to to fit each (changing) *cell-dur*. In this way, you can create a pulsing envelope that accelerates, gets slower, or is randomised.

Musical Applications

ENVEL CYCLIC can be used to create an amplitude tremolo with a contour shape designed specifically for the task in hand.

End of ENVEL CYCLIC

ENVEL DBTOENV – Convert a (text) breakpoint file with dB values to a binary envelope file

Usage

envel dbtoenv *db_brkfile outenvfile wsize*

Parameters

db_brkfile – input breakpoint file with dB values
outenvfile – equivalent data written in binary form
wsize – enveloping window: time resolution in milliseconds over which to average amplitude data (Range: 5 to length of file)

Understanding the ENVEL DBTOENV Function

This is a utility to convert breakpoint data in dB format to envelope data in binary format. The input breakpoint file contains *time dB* values and can be accessed with a text editor. The binary envelope file can only be handled by the software.

The dB format is amplitude expressed along the logarithmic decibel scale. See the reference chart which relates these to gain and the 0 to 32767 scale.

Musical Applications

This is a utility to convert between dB and binary formats.

Also see: **ENVEL ENVTODB**

End of ENVEL DBTOENV

ENVEL DBTOGAIN – Convert a (text) breakpoint file with dB values to gain values (0 to 1)

Usage

envel dbtogain *dbbrkfile outbrkfile*

Parameters

dbbrkfile – input breakpoint file with *time dB* values
outbrkfile – output breakpoint file with *time gain* values

Understanding the ENVEL DBTOGAIN Function

Both the input and the output files are breakpoint files, i.e., with time points and values for each of those time points. The times will remain the same, and the amplitude values will be converted.

Both files can be accessed with a text editor.

Gain is a multiplier operating between 0 and 1. See the reference chart which relates gain to dB and to the 0 to 32767 scale.

Musical Applications

This is a utility to convert between amplitude value formats.

Also see: **ENVEL GAINTODB**

End of ENVEL DBTOGAIN

ENVEL DOVETAIL – Dovetail a soundfile by enveloping its beginning and end

Usage

envel dovetail 1 *infile outfile infadedur outfadedur intype outtype* **-t***times*
envel dovetail 2 *infile outfile infadedur outfadedur* **-t***times*

Modes

1 a choice of linear or exponential slopes at the beginning and the end
2 a doubly exponential slope (steeper) is applied to the beginning and the end

Parameters

infile – input soundfile to dovetail
outfile – output (dovetailed) soundfile
infadedur – duration of the fade-in at the start of the file
outfadedur – duration of the fade-out at the end of the file

infadedur and *outfadedur* must not overlap each other

intype – envelope shape for the fade-in at the start of the file

intype = **0**: **linear** fade
intype = **1**: **exponential** fade (the Default)

outtype – envelope shape for the fade-out, at the end of the file

outtype = **0**: **linear** fade
outtype = **1**: **exponential** fade (the Default)

-t*times* – identifies the type of time unit being used:

times = **0**: times are in seconds (the Default)
times = **1**: times are in samples
times = **2**: times are in grouped-samples; e.g., in stereo, each sample-pair counts as 1 item

Understanding the ENVEL DOVETAIL Process

This function provides a way to smooth the beginning and end of a soundfile without having to create an envelope breakpoint file. The duration of the fade-in and fade-out can be short or long, with the restriction that they must not overlap: i.e., their sum must not be greater than the length of the soundfile.

The linear shape creates a steady rise and fall in amplitude, while the (default) exponential shape is curved: slower and then speeding up on the rise, faster and then slowing down on the fall.

Musical Applications

ENVEL DOVETAIL can be used to make a soundfile begin and end more gradually if it is too abrupt or even has clicks at its edges.

On the other hand, you may want it slowly to emerge out of silence and return to silence in a similar fashion. This is easily done with ENVEL DOVETAIL because the durations of the fades can be very long.

End of ENVEL DOVETAIL

ENVEL ENVTOBRK – Convert a binary envelope file to a (text) breakpoint envelope file

Usage

envel envtobrk *inenvfile outbrkfile* [**-d***datareduce*]

Parameters

inenvfile – input binary envelope file
outbrkfile – output text breakpoint envelope file
-d*datareduce* – weights in favour of the quantity or the accuracy of the data written to the breakpoint file (Range: 0 [quantity] to 1 [accuracy])

Understanding the ENVEL ENVTOBRK Function

Converts from binary envelope to text breakpoint envelope data.

The *datareduce* parameter enables you to reduce the amount of data contained in the file. A low value, such as 0.2 will keep most of the data, while a high value such as 0.8 will lose most of the data – while trying to retain the basic shape of the envelope.

Musical Applications

ENVEL ENVTOBRK makes it possible to gain access to binary envelope data so that it can be edited, printed, or used with another function which uses a breakpoint file as input (such as pitch transposition).

Also see: **ENVEL BRKTOENV**

End of ENVEL ENVTOBRK

ENVEL ENVTODB – Convert a binary envelope file to a (text) breakpoint envelope file with dB values

Usage

envel envtodb *inenvfile outbrkfile* [**-d***datareduce]*

Parameters

inenvfile – input binary envelope file
outbrkfile – output text breakpoint file, with values expressed in dB
-d*datareduce* – weights in favour of the quantity or the accuracy of the data written to the breakpoint file (Range: 0 [quantity] to 1 [accuracy])

Understanding the ENVEL ENVTODB Function

This function converts a binary envelope file to a text breakpoint envelope file with its values expressed in dB. Bear in mind that 0.0dB represents full volume, so the numbers in the dB file ought to be negative. A positive value indicates a boosting of the original level (care!).

See the Gain - dB Chart for orientation regarding the aural significance of dB values.

The *datareduce* parameter enables you to reduce the amount of data contained in the file. A low value, such as 0.2 will keep most of the data, while a high value such as 0.8 will lose most of the data – while trying to retain the basic shape of the envelope.

Musical Applications

If you prefer to enter mixfile amplitude data in dB, using ENVEL ENVTODB can provide experience in relating dB values to what you hear in various sounds.

Also see: **ENVEL DBTOENV**

End of ENVEL ENVTODB

ENVEL EXTRACT – Extract the amplitude envelope from an input soundfile

Usage

envel extract 1 *infile outenvfile wsize*
envel extract 2 *infile outbrkfile wsize* [**-d***datareduce*]

Modes

1 extract envelope, creating a binary envelope file
2 extract envelope, creating a (text) breakpoint file

Parameters

infile – input soundfile
outenvfile – output binary envelope file created in Mode **1**
outbrkfile – output (text) breakpoint envelope file containing *time gain* values created in Mode **2**
wsize – enveloping window: time resolution in milliseconds over which to average amplitude data (Range: 5 to length of *infile*)
-d*datareduce* – weights in favour of the quantity or the accuracy of the data written to the breakpoint file (Range: 0 [quantity] to 1 [accuracy])

Understanding the ENVEL EXTRACT Function

The envelope of a soundfile charts its rise and fall in amplitude over time. It is the overall, usually fairly coarse, shape we see in the standard time-amplitude waveform display of a soundfile. Only at the very smallest scale to we see the signal continually rising above and falling below the zero line, showing instantaneous change in the amplitude of the signal and defining its waveshape. The total span of this rise and fall from the peak above zero to the trough below zero tells us something about the loudness of the signal at that moment.

If we chart how this total span changes over time, we get an idea of how the loudness of the sound varies over time. When we see a graphic display of a waveform at a coarser scale, the instantaneous up and down motion of the amplitude cannot be seen. What we do see is a kind of block representation of the signal, whose variation in width (top to bottom) gives an indication of the changing loudness, or envelope, of the sound.

The fineness or degree of resolution of the amplitude envelope extracted from the sound or as defined in a breakpoint file is determined by the *windowsize* parameter. This is the amount of time between measurements of the waveform's amplitude. The minimum value here is 5 milliseconds, i.e., every 200th of a second. Thus, at a sample rate of 44100 per second, an amplitude value is registered every 220.5 samples – and whatever changes in amplitude happen between these points *are not registered*.

A resolution of 5 milliseconds is considered to be rather 'fine' and even coarser values can be adequate. It depends on the nature of the sound, and this is what you must consider when you set the *windowsize*. If you have a sound that tremolos through a crescendo, for example, a small *windowsize* will track the undulations of the tremolo as well as the crescendo. A larger *windowsize* will see the crescendo but ignore the tremolo fluctuations. Which *windowsize* you choose will depend on what musical information you want to extract from the source sound.

The *datareduce* parameter enables you to reduce the amount of data contained in the file. A low value, such as 0.2 will keep most of the data, while a high value such as 0.8 will lose most of the data – while trying to retain the basic shape of the envelope.

Musical Applications

Extracting the envelope is a preliminary step for several operations, such as imposing it on a different soundfile (**ENVEL IMPOSE**) or using it as a breakpoint input for the parameter of some other function altogether. In this case, you may find you have too much data, so you may need to extract the envelope again, using a *datareduce* value – or go straight to BRKEDIT (if you have a Windows 95 system), where there is a 'Reduce' option.

BRKEDIT can also be useful to alter the vertical range of the breakpoint file to suit a given parameter, or to adjust the overall duration of the breakpoint file to match the length of a different sound.

Re-using envelope data in a variety of contexts is one of the key tools for creating unity by means of aural cross-references, sometimes overt and sometimes only very subtly present to the ear.

End of ENVEL EXTRACT

ENVEL GAINDODB – Convert a (text) breakpoint file with gain (0 to 1) values to dB values

Usage

envel gaintodb *brkfile out_db_brkfile*

Parameters

brkfile – input *time gain* envelope text breakpoint file
out_db_brkfile – output *time dB* envelope text breakpoint file

Understanding the ENVEL GAINTODB Function

ENVEL GAINTODB makes the conversion between these two text data formats, from gain to dB.

Musical Applications

The conversion may be made for educational purposes, or to provide values relevant for expressing amplitude in dB in mixfiles.

Also see: **ENVEL DBTOGAIN**

End of ENVEL GAINTODB

ENVEL IMPOSE – Impose an envelope on an input soundfile

Usage

envel impose 1 *input_sndfile imposed_sndfile outsndfile wsize*
envel impose 2 *input_sndfile imposed_envfile outsndfile*

In Modes **1** and **2** the whole soundfile is enveloped.

envel impose 3 *input_sndfile imposed_brkfile outsndfile*
envel impose 4 *input_sndfile imposed_brkfile_dB outsndfile*

In Modes **3** and **4** the breakpoint data may begin (and end) at any point in the file, effectively editing it – but there must be at least 2 breakpoint pairs in the file.

Modes

1 Extract an envelope from another soundfile and impose it on *input_sndfile*
2 Impose an envelope from a binary envelope file
3 Impose an envelope from a (text) breakpoint file
4 Impose an envelope from a (text) breakpoint file with dB values

Parameters

input_sndfile – input soundfile on which to impose the envelope
imposed_sndfile – soundfile from which to extract the amplitude envelope
imposed_envfile – envelope to impose is contained in a binary envelope file
imposed_brkfile – envelope to impose is contained in a (text) *time value* breakpoint file (Range of *value* is 0 to 1)
imposed_brkfile_dB – envelope to impose is contained in a (text) *time dB_value* breakpoint file (Range of *dB_value* is -96 to 0)

> **NB:** Put only the dB value in the file, without the letters 'dB' – e.g., '-6', not '-6dB'.

outsndfile – output soundfile with new amplitude envelope
wsize – enveloping window: time resolution in milliseconds over which to average amplitude data (Range: 5 to length of *imposed_sndfile*)

Understanding the ENVEL IMPOSE Process

This is the matching function for ENVEL EXTRACT, which extracts an envelope from a soundfile, saving it as a binary (Mode 1) or breakpoint (Mode 2) file.

For convenience, ENVEL IMPOSE can extract an envelope directly from a soundfile (Mode 1) as well as imposing envelopes from existing binary or breakpoint files.

Musical Applications

Mode **3** can be very useful when you would like to fine-tune the envelope of a soundfile with a breakpoint envelope file you have written yourself, e.g., to soften or sharpen an attack, or some other portion of the soundfile. Sometimes you may need a customised result which the other more generalised envelope shaping functions (such as DOVETAIL) do not provide.

Another key application is to impose a changing envelope onto a soundfile which has a fairly steady-state envelope. In this way, for example, a bubbling stream of water can be made to pulse to the rhythm of spoken words. Similarly, a given sound can acquire the amplitude shape of some other specific sound, thus blending features of the two sounds. This is all part of creating relationships among sonic data.

Note that this deals with amplitude only. It is quite a different matter to extract and impose timbral envelopes, because then the tonal characteristics are moved. See the Formants Group in the spectral dimension.

Also see: **ENVEL EXTRACT**, **ENVEL REPLACE**, **ENVEL REPLOT**, **ENVEL RESHAPE** and **ENVEL WARP**.

End of ENVEL IMPOSE

ENVEL PLUCK – Pluck the start of a sound (Mono files only)

Usage

envel pluck *infile outfile startsamp wavelen* [**-a***atkcycles*] [**-d***decayrate*]

Parameters

infile – input soundfile to pluck
outfile – output soundfile with pluck effect
startsamp – the sample in *infile* at which the pluck is to end: **it must be at a zero crossing**; this is also where *wavelen* begins. Zero crossings can be seen in VIEWSF.
wavelen – the number of (absolute) samples in the wavelength of the pluck effect; its value is the number of samples in a waveform starting at *startsamp* in the *infile* – see below! (Range: 4 to 2205)
-a*atkcycles* – the number of wavecycles in the pluck attack (Range: 2 to 32767; the Default is 32)
-d*decayrate*– rate of decay of the pluck effect (Range: 1 [slow] to 64 [fast]; the Default is 48)

Understanding the ENVEL PLUCK Process

Although similar to ENVEL ATTACK, ENVEL PLUCK differs in that the attack imposed on the soundfile also has the noise and decay characteristics associated with plucked sounds.

To pluck the sound, ENVEL PLUCK creates a 'pluck' constructed from a number of wavecycles and splices it into the beginning of the sound. To achieve this, it needs a goal wavecycle to pluck towards: this is located at *startsamp*.

We can get an idea of what is going on here by reviewing how the Karplus-Strong pluck algorithm used in *Csound* works. It achieves a pluck effect by iteratively filtering a block of noise until it becomes pitched. It does this because noise components are an essential ingredient of a plucked sound.

ENVEL PLUCK achieves the same effect by working backwards – backwards because it is starting with an existing sound. The process is to repeat *wavelen* backwards a number of times towards the beginning of the soundfile, becoming noisier as it goes.

- *startsamp* is where the pluck effect links into the sound. It will usually be a little after the beginning of the sound, not at the very beginning, because we want the pluck to link smoothly into stable signal. So we look for a fairly regular waveform close to the beginning of the sound, from the first positive sample after a zero crossing to the last negative sample before the next zero crossing. The sample count of the 'first positive sample' is *startsamp*.

- *wavelen* is the number of samples between *startsamp* and the 'last negative sample', i.e., the number of samples in the 'fairly regular waveform' you've identified.

- *atkcycles* is the number of cycles of *wavelen* to use for the pluck. The Default is 32, but the max is very high: 32767. Be careful about values much beyond 128: you may introduce a prolonged sustain into the sound.

- *decayrate* specifies the rate at which the pluck effect will decay. The default is a fairly high 48, but can be made slower (< 48 – the pluck-noise lasts longer) or quicker (> 48 – the pluck-noise fades away very quickly).

The order of events looks like this:

> Start of sound ... *atkcycles* of pluck ending at ... *startsamp*, which is also where a 'fairly regular waveform' begins, extending for *wavelen* samples. The pluck effect is built out of this specified waveform.

If your waveform display program does not show both time and sample count, you can convert between them with SNDINFO SMPTIME and SNDINFO TIMESMP.

If the *infile* has a gradual attack transient, it is possible that the added pluck noise component may fade before the amplitude of the sound reaches a matching level, thus leaving a hole. This can perhaps be mitigated by having a modest wavelength (less than 1000), not too many attack cycles (e.g., 100 or less) and a slow decay rate (e.g., 2) – depending on what works for that sound. Additional strategies may be to cut out the new pluck section and then join it to the rest of the sound with MORPH BRIDGE or MORPH MORPH.

Musical Applications

The pluck effect relates to a specific technique used with stringed instruments, whether standard 'picking' on guitar/mandolin type instruments, or 'pizzicato' on orchestral strings. The noise component is an essential part of this type of sound, making it different from a sharp attack, although this can also introduce timbral components, especially on brass instruments.

The attack part of a sound is one of the key means of identifying the source of a sound, so altering the attack implies playing with the perception of the source, as well as simply intensifying the onset of the sound.

I seemed to get the best results with a rather long *wavelen* (several wavecycles of *infile*, a modest number of repeats (*atkcycles*), and the slowest *decayrate* (1). Even so, the effect was not like a stringed instrument, but rather like a tap on a tambourine: there seems to be more noise than tuning in the pluck portion of the new sound. If you want to achieve the sound of a plucked string instrument, I would recommend that you use **pluck** in *Csound*.[Ed.]

Also see HILITE PLUCK in the spectral dimension.

End of ENVEL PLUCK

ENVEL REPLACE – Replace the existing envelope of a soundfile with a different envelope

Usage

envel replace 1 *input_sndfile replacing_sndfile outsndfile wsize*
envel replace 2 *input_sndfile replacing_envfile outsndfile*
envel replace 3 *input_sndfile replacing_brkfile outsndfile wsize*
envel replace 4 *input_sndfile replacing_brkfile_dB outsndfile wsize*

Modes

1 Replaces envelope with a new one extracted from another soundfile
2 Replaces envelope with a new one from a binary envelope file
3 Replaces envelope with a new one from a (text) breakpoint file
4 Replaces envelope with a new one from a (text) breakpoint file with dB values

In all cases the entire soundfile is enveloped.

Parameters

input_sndfile – input soundfile on which to impose the envelope
replacing_sndfile – soundfile from which to extract the amplitude envelope
replacing_envfile – envelope to impose is contained in a binary envelope file
replacing_brkfile – envelope to impose is contained in a (text) *time value* breakpoint file (Range of *value* is 0 to 1)
replacing_brkfile_dB – envelope to impose is contained in a (text) *time dB_value* breakpoint file (Range of *dB_value* is -96 to 0)
outsndfile – output soundfile with new amplitude envelope
wsize – enveloping window: time resolution in milliseconds over which to average amplitude data (Range: 5 to length of *replacing_sndfile*)

Understanding the ENVEL REPLACE Process

With ENVEL IMPOSE, the imposed envelope is imposed **on top of** any existing envelope. The result is the combination of the imposed envelope and the existing envelope.

In contrast, ENVEL REPLACE attempts to **replace** the existing envelope of the sound with the newly sampled envelope. (To do this it must start by flattening out the original envelope of the sound.)

Care! If there are (near) zero amplitudes in the original sound, it will not be possible to replace these meaningfully.

Musical Applications

The author observes that Mode **1** is useful for 'restoring the amplitude contour of a sound after filtering with time-varying Q'. In this case, the amplitude envelope of the *input_sndfile* is extracted before applying time-varying Q. It will then be 'in storage', available, in this case, to be reapplied to the filtered sound in order to restore the original amplitude contour.

End of ENVEL REPLACE

ENVEL REPLOT – Warp the envelope in a (text) breakpoint envelope file

Modes

Usage

envel replot 1–12 *brkfile outbrkfile wsize various_params* [**-d***reduce*]
envel replot 13 *brkfile outbrkfile rampfile wsize various_params* [**-d***reduce*]
envel replot 14–15 *brkfile outbrkfile wsize various_params* [**-d***reduce*]

wsize –: enveloping window: time resolution in milliseconds over which to average amplitude data
-d*reduce* – forces interpolation of data in the breakpoint output file to reduce unnecessary data output. Range: 0 to 1.
NB: TRIGGER works best with ENV RESHAPE or ENV WARP.

ENVEL RESHAPE – Warp the envelope in a binary envelope file

Modes

Usage

envel reshape 1–12 *envfile outenvfile various_params*
envel reshape 13 *envfile outenvfile rampfile various_params*
envel reshape 14–15 *envfile outenvfile various_params*

ENVEL WARP – Warp the envelope of a soundfile

Modes

Usage

envel warp 1–12 *sndfile outsndfile wsize various_params*
envel warp 13 *sndfile outsndfile rampfile wsize various_params*
envel warp 14–15 *sndfile outsndfile wsize various_params*

wisize – the duration of the enveloping window, in milliseconds

Modes

The 15 Modes of ENVEL RESHAPE, REPLOT & WARP

Modes 1 – 4	Modes 5 – 8	Modes 9 – 12	Modes 13 – 15
1 NORMALISE	**5** LIFT	**9** INVERT	**13** TRIGGER
2 REVERSE	**6** TIMESTRETCH	**10** LIMIT	**14** CEILING
3 EXAGGERATE	**7** FLATTEN	**11** CORRUGATE	**15** DUCKED
4 ATTENUATE	**8** GATE	**12** EXPAND	

The Modes and their Parameters

1 NORMALISE

Expand envelope so that the highest envelope point is at the maximum possible amplitude value.

No other extra parameters are needed.

2 REVERSE

Reverse the envelope in time.

No other extra parameters are needed.

3 EXAGGERATE

Exaggerate the envelope contour. One other parameter:

exaggerate – Range: > 0.0
< 1, low values are boosted
= 1, no change
> 1, high values are boosted
exaggerate may vary over time

4 ATTENUATE

Reduce the amplitude of an envelope. One other parameter:

attenuation – multiplier by which to scale down the envelope. Range: 0 to 1
attenuation may vary over time

5 LIFT

Raise the envelope by a fixed amount. One other parameter:

lift – amount to add to each amplitude value. Range: 0 to 1
lift may vary over time

6 TIMESTRETCH

Stretch or compress an envelope in time. One other parameter:

timestretch – multiplier by which to stretch the envelope. Range: > 0.0
< 1.0 compresses the envelope in time
= 1.0 no change
> 1.0 stretches the envelope in time

7 FLATTEN

Even out the envelope contour. One other parameter:

flatten – number of envelope values over which to average. Range: 1 to 5000.
flatten may vary over time
Note that there is an envelope value for every segment of sound, and that this may be considerably more widely spaced than, for example, the control points specified by *ksmps* in a Csound orchestra (often set to 15 samples).

8 GATE

Zero lower portion of envelope. Two other parameters:

gate – levels less than *gate* are set to 0. Range: 0 to 1.
smoothing – excises segments with an amplitude less than *smoothing*. Range: 0 to 32767.
smoothing < 0 turns off the smoothing effect.
gate may vary over time

9 INVERT

Turn envelope upside down: negative values become positive, and positive values become negative. Two other parameters:

gate – levels less than *gate* are set to 0. Range: 0 to < *mirror*.
mirror – reflection point: all values other than those below *gate*, both above and below *mirror*, are inverted to the other side of *mirror*. Range: > *gate* to 1.
gate and *mirror* may vary over time

10 LIMIT

Squeeze down amplitude into a specified range. Two other parameters:

limit – upper limit of range; amplitude values which were above *limit* are scaled down such that 'maxamp' becomes *limit*. Range: > *threshold* to 1.
threshold – lower limit of range; only values above *threshold* are affected. Range: 0 to *limit*.
limit and *threshold* may vary over time

11 CORRUGATE

Tighten the peaks into close-set ridges by taking troughs (downward dips) in the envelope to zero. Two other parameters:

trofdel – number of windows to set to zero per trough. Range: 1 to < *peak_separation*.
peak_separation – minimum number of windows per peak. Range: 2 to 32767.
trofdel and *peak_separation* may vary over time

12 EXPAND

Inflate the envelope. Three other parameters:

gate – levels below *gate* are set to 0. Range: 0 to *threshold*.
threshold – lower limit of range: amplitude values that were above *gate* and below *threshold* are scaled up such that the minimum level becomes *threshold*. Range: *gate* to 1.
smoothing – excises low-level segments which are less than *smoothing* windows in length. Range: 0 to 32767. 0 turns off the smoothing effect.
gate and *threshold* may vary over time

13 TRIGGER

Create a new envelope of sudden **on** bursts, triggered by the rate of rise of the current envelope. Four other parameters:

rampfile – your new breakpoint file for the (triggered) bursts
gate – the average level must be above *gate* before triggering can occur. Range: 0 to 1.
rise – minimum loudness-step before triggering can occur. Range: 0 to 1.
dur – maximum duration over which the *rise* can take place before triggering can occur. Range: must be >= the envelope window duration.
gate may vary over time

14 CEILING

Force envelope up to its maximum level, **everywhere**. No other extra parameters needed.

15 DUCKED

Create 'ducking' envelope: envelope kept below a set level. Two other parameters are needed:

gate – highest level allowed for the envelope. Range: 0 to 1.
threshold – amplitude level the envelope must reach before 'ducking' is applied. Range: 0 to 1.
gate and *threshold* may vary over time

- **With envelope Warping:** When the input envelope level exceeds *threshold*, the output envelope is reduced to *gate* level. Elsewhere it is unchanged.
- **With envelope Reshaping or Replotting:** When the input envelope level exceeds *threshold*, the output envelope takes *gate* level. Elsewhere, it gives unity gain (no change).
- This function can be used to create an envelope to apply to another sound which can then be mixed with the sound from which the original envelope was extracted.

Understanding the ENVEL RESHAPE, REPLOT & WARP Processes

The envelope shaping routines for these three functions are identical. Only the output to which the changes are applied differs.

Movement in time creates shapes. Amplitude change plays a key role in performance articulation and in the identification of sounds. When we hear the amplitude 'grow' as a soprano sings a glorious long tone, we are aware of its amplitude 'envelope'. When we hear the sharp 'report' of a snare drum rimshot, we are aware of the 'attack transient' of the sound: i.e., its very rapid rise in amplitude. Voices rise and fall in both pitch and amplitude.

ENVEL WARP is a way of working with the shape produced by the envelope over the course of its whole duration. If we think of it as made of rubber, we can imagine it being pulled and stretched and squeezed.

Musical Applications

Amplitude envelope is an essential part of musical 'articulation' and has a great bearing on emotional content. These functions can be used to increase contrasts and make the music 'vivid', or to reduce the contrasts and 'tone down' the emotional effect.

On the other hand, the amplitude shape can be handled as an abstract form which can be extracted from a sound, remoulded in some way, and then applied to another sound or to some other musical data. This introduces a point of connection between different parts of the musical material. For example, we might hear an amplitude shape in the form of a series of vocal bursts. Then this same/similar shape may become audible in the form of a pitch transient (pitch glissando). The musical principle is 'sameness and difference': creating unity amidst disparate material by introducing elements of sameness.

End of ENVEL RESHAPE, REPLOT & WARP

ENVEL SCALED – Impose an envelope on an input soundfile, scaling it timewise to the sound's duration

Usage

envel scaled *input_soundfile imposed_brkfile outsoundfile*

Parameters

input_soundfile – input soundfile to process
imposed_brkfile – breakpoint file describing the envelope to impose: *time amplitude_value* pairs. **Note that this is a text, not a binary envelope file.** *Times* can be over any duration: e.g., 0 to 1, 0 to 10000 etc. *Amplitude_values* can go from 0 to more than 1. Thus parts of the input sound can be amplified, but take care: too much amplification may lead to distortion.
outsoundfile – the output is a reshaped input

Must have at least 2 breakpoint pairs

Understanding the ENVEL SCALED Process

The main point here is that the times in the breakpoint file do not need to match the length of the input soundfile. The breakpoint file is therefore generic: whatever its times, this function will scale it to fit the length of the input sound.

Musical Applications

Because the *imposed_breakpoint* file needs to be a text file, you can use ENVTOBRK to convert a binary envelope file extracted from a sound to a text breakpoint file.

This function can be convenient when you want to apply existing envelope files to a given sound. Sometimes it is useful to apply a process to many sounds, slightly modifying that process to suit the different durations of the source files. This is especially useful when using BULK PROCESSING on the *Sound Loom*, so that new data files do not need to be written for every single input source.

For example, a simple fade in, fade-out envelope could be applied to hundreds of sounds of slightly different duration, at a single BULK PROCESSING pass, using this process.

End of ENVEL SCALED

ENVEL SWELL – Cause sound to fade in to and out from a peak moment

Usage

envel swell *infile outfile peaktime peaktype*

Parameters

infile – input soundfile
outfile – output soundfile
peaktime – time in *infile* at which the peak moment is located
peaktype – type of swell shape: **0** (linear) or **1** (exponential – the Default)

Understanding the ENVEL SWELL Process

This is a generic envelope type which can be applied very easily by specifying the time of the peak moment and the shape to be applied.

The peak moment (*peaktime*) can be the location of the maximum amplitude in the soundfile, which can be found with SNDINFO MAXSAMP). It could also be any other place in the file that you define. For example, you can take a completely flat sound (in terms of loudness) and make it swell to a peak (at any point you want) and die away again.

The linear *peaktype* produces a less gradual fade.

Musical Applications

This function makes it easy to create an audible peak anywhere in a soundfile, and implement a swell to and from it without having to create a breakpoint file.

End of ENVEL SWELL

ENVEL TIMEGRID – Partition a soundfile into a sequence of 'windows' separated by silence

Usage

envel timegrid *input_soundfile generic_outsoundfile_name gridcount gridwidth splicelength*

Parameters

input_soundfile – soundfile from which to create the grid
generic_outsoundfile_name – root name for the several output soundfiles, numbered 0 to *gridcount* - 1
gridcount – number of grids (and hence output files)
gridwidth – duration of grid windows, in seconds
splicelen – splice length, in milliseconds

gridwidth and *splicelength* may vary over time

Understanding the ENVEL TIMEGRID Process

Imagine you have a regular fence composed of uprights separated by gaps. If you look through the fence, the view will be broken up into vertical slices, with the bits inbetween hidden by the fence posts. If you move slightly to your right you will see a different set of vertical slices of the view beyond the fence. Move a little more, and you will see the original view again, as the N+1th fencepost blocks the same bits of the view as was previously blocked by the Nth fencepost.

ENVEL TIMEGRID therefore creates a series of 'views' of a sound as if seen through a fence (or mask). Where the fence-posts block our 'view', there is only silence.

To illustrate this we can set ENVEL TIMEGRID to use 2 grids (*gridcount* = 2) and a *gridwidth* of 0.5 seconds:

At Time	Grid number 1 has	Grid number 2 has
0.0 - 0.5	see the SIGNAL	signal replaced by silence
0.5 - 1.0	signal replaced by silence	see the SIGNAL
1.0 - 1.5	see the SIGNAL	signal replaced by silence
1.5 - 2.0	signal replaced by silence	see the SIGNAL
ETC		

If we mix these two grids, we reconstitute the original sound.

with 3 grids (*gridcount* = 3) and a *gridwidth* of 0.5 seconds, we have:

At Time	Grid 1 has	Grid 2 has	Grid 3 has
0.0 - 0.5	see SIGNAL	silence	silence
0.5 - 1.0	silence	see SIGNAL	silence
1.0 - 1.5	silence	silence	see SIGNAL
1.5 - 2.0	see SIGNAL	silence	silence
1.5 - 2.0	silence	see SIGNAL	silence
ETC			

If we mix these three grids, we reconstitute the original sound.

The movement in and out of silence is controlled by the length of the splice, so it could be abrupt or, if long, quite gradual.

Musical Applications

This function, therefore, partitions a sound into a number of grids. Each grid is like the sound seen through a 'fence' with regular gaps. Were we to mix these output sounds together, all starting at time 0, the original sound would be reconstituted. We could also mix them together in a different order. And if we made several of these grids from different sounds (with the same grid distance in time, i.e., *gridwidth*), they could be be mixed together to form hybrid sounds.

ENVEL TIMEGRID is a new take on the idea of segmentation. What is original is that separate soundfiles are formed. The length of the segments (*gridwidth*) are user-defined, but you probably get the most interesting results when they are very short. With a mixing operation, with SUBMIX MIX (MIXFILES) or with an audio sequencer, the segments from the same **or different** sounds can be reassembled, creating strange hybrids, or collage effects if the segments are fairly long.

Note that there are various parameters involved in exploring and fine-tuning the results:

- you can use just one soundfile and create staccato or stuttering effects
- you can repeat ENVEL TIMEGRID with more soundfiles, creating grids from several soundfiles that can be mixed, and the order of these files in the mix can varied
- *gridcount* – the number of output soundfiles from each input soundfile that you can put into the mix
- *gridwidth* – the temporal width of the grid, which can be different for each input sound that you use, or could also vary over time(!)
- *spliclen* – which can be sharper or more gradual, adjusted in an appropriate way to *gridwidth* or also vary over time
- *starttime* – the soundfile start times in the *mixfile*; they may be synchronous or offset by varying amounts

As with many CDP programs, therefore, there are many possibilities to explore.

Finally, bear in mind that SUBMIX DUMMY can help you to create a *mixfile* from the outputs of ENVEL TIMEGRID very quickly. You just submit your list of soundfiles to SUBMIX DUMMY and it creates a basic *mixfile* that you can edit. In *Sound Loom* the process can be even quicker, because you can select all the files you want in the mix in CHOSEN FILES mode, go straight to SUBMIX DUMMY and create the *mixfile*.

End of ENVEL TIMEGRID

TOPANTAIL2 TOPANTAIL – Fade both the beginning and end of a sound

Usage

topantail2 topantail *insndfile outsndfile startgate endgate* [**-s***splicelen*] [**-b***backtrack*]

Example command line:

```
topantail2 topantail insnd.wav outsnd.wav 0.035 0.035 -s25 -b200
```

Parameters

insndfile – input mono soundfile
outsndfile – output soundfile
startgate – level at start before which sound is to be removed. (Range: 0 to 1, Default: 0.001)
endgate – level at end after which sound is to be removed. (Range: 0 to 1, Default: 0.001)
-s*splicelen* – splice length in milliseconds
-b*backtrack* – backtrack in time from the initial gate point(found by the program) to an earlier splice point, in milliseconds

Understanding the TOPANTAIL2 Process

TOPANTAIL2 provides another approach to extracting part of a soundfile by using gate levels. In this case it trims away signal before and after the extracted portion. The 'backtracking' function moves the initial cut point earlier in the sound, thus retaining some of the sound which may be below the *startgate*'s level.

TOPANTAIL2 is designed both to deal with low-level noise and to aid the synchronisation of sounds in a mixfile. As such it is both an extension of ENVEL DOVETAIL and a replacement for HOUSEKEEP EXTRACT. The latter cuts out soundfile before and after good signal. (It also has a backtracking facility, but also a more complex command line.)

When low-level noise is present, a useful value to try for *startgate* and *endgate* is 0.1. The aim here is to remove level below this level while capturing the beginning of strong signal at the start of the sound. The fade-in is present so that some soundfile remains before this first strong signal, soundfile into which one can backtrack in time. One result of this is that the start of the sound, e.g., a vocal sound, will be more natural. Some sounds begin very quietly before rising to an acceptable signal level, as set by *startgate*, but removing this may make the beginning of the sound more abrupt than expected – the ears are very sensitive instruments! The backtracking enables you to keep this very quiet portion of the sound.

If you then set *backtrack* to e.g., 250ms, the sound will be cut 250ms **before** the good signal starts, not **when** it starts. This enables you to set the start points of two sounds in a mixfile both to 0.0 while enabling them to synchronise their attacks. If there is a latency in the beginning of the other sound, your backtracked sound will become audible at the same time while preserving its natural attack transient.

Musical Applications

The original purpose of TOPANTAIL2 was to enable the processes described above to become automated once the appropriate research into the behaviour of the sounds being used or that could be used had been completed.

End of TOPANTAIL2

ENVEL TREMOLO – Tremolo a sound

Usage

envel tremolo mode *infile outfile freq depth gain*

Modes

1 Interpolate linearly between frequencies in any frequency breaktable (Default Mode)
2 Interpolate logarithmically (like pitch) – take care with zero frequencies

Parameters

infile – input soundfile
outfile – output soundfile
freq – frequency of the tremolo (a low frequency oscillation: Range: 0 to 500)
depth – frequency displacement of the oscillation (Range: 0 to 1; the Default is 0.250)
gain – the overall signal gain, or envelope (Range 0 to 1; the Default is 1)

freq, *depth* and *gain* may vary over time.

Understanding the ENVEL TREMOLO Process

'Tremolo' is a pulsation in the amplitude of a sound. The frequency of the tremolo is the rate of the amplitude pulsation. ('Vibrato' is a frequency displacement.)

Musical Applications

This technique is familiar to us as a feature of various singing styles, and also as a control available on synthesisers.

It can be used to enliven or enrich a sound, or to create more marked flutter effects, such as the 'flutter-tongue' technique on a wind instrument.

Also see: **TREMOLO TREMOLO**, **MODIFY SHUDDER**, **REPITCH VIBRATO** and **STRANGE WAVER**.

End of ENVEL TREMOLO

TREMOLO TREMOLO – Apply width-controlled tremolo to a soundfile

Usage

tremolo tremolo mode *insndfile outsndfile frq depth gain fineness*

Example command line to create TREMOLO:

```
tremolo tremolo 1 insnd.wav outsnd.wav 4 0.6 1 40
```

Modes

1 Interpolate linearly between frequencies in any frequency breaktable. This is the Default.
2 Interpolate logarithmically (like pitch). Take care with any zero frequencies.

Parameters

frq – frequency of the tremolo. Range: 0 to 500
depth – depth of the tremolo. Range: 0 to 1. Default 1
gain – overall signal gain, or envelope. Range: 0 to 1. Default 1
fineness – squeeze the width of the tremolo: an integer value >= 1

frq, *depth* and *gain* may vary over time.

Understanding the TREMOLO TREMOLO Process

This Release 7 TREMOLO function complements **ENVEL TREMOLO**. It applies tremolo to a soundfile with the added ability to narrow ('squeeze') the width of the tremolo segments.

End of TREMOLO TREMOLO

Last Updated 19 Sep. 2016 -- HTML5 version
Documentation: Archer Endrich, revised R.Fraser