CDP ENVNU Functions

(with Command Line Usage)

Specialised Enveloping Operations

EXPDECAY
Produce a true exponential decay to zero on a sound

PEAKCHOP
Isolate peaks in a source and either play back at a specified tempo (Mode **1**) or output a peak-isolating envelope (Mode **2**)

ENVNU EXPDECAY – Produce a true exponential decay to zero on a sound

Usage

envnu expdecay *infile outfile starttime endtime*

Example command line to create an exponential decay:

```
envnu expdecay horn hornxdec 4.6 5.386
```

Parameters

infile – input soundfile
outfile – output soundfile
starttime – time at which the decay begins
endtime – time at which the decay reaches zero

Understanding the ENVNU EXPDECAY Process

The existing CDP 'exponential' and 'double exponential' envelopes are actually linear approximations to a true exponential curve. The approximations were written at a time when processor speed was much lower, but for all musical purposes they are fine – you can't hear the difference. However, a true exponential decay might be needed for certain technical applications, and this is provided by ENVNU EXPDECAY.

An exponential curve involves a steady increase in the exponent of a number, its 'power': $2^1 = 2$, $2^2 = 4$, $2^3 = 8$ – i.e., a progressive doubling, as opposed to a linear curve, which adds a constant value each time: 2 - 4 - 6 - 9 - 10 etc. In the case of an amplitude envelope, 1 is divided by these progressively increasing numbers: $1 \div 2 = 0.5$, $1 \div 4 = 0.25$, $1 \div 8 = 0.125$ etc. This means that the rate of decay will be large to start with and then get progressively less (halving each time). The effect is that the decay slope starts very steeply and gradually becomes less steep as it proceeds to zero. It reaches zero (-100 dB) very gradually so that the tail-off is very smooth – and much of the last part of the soundfile is likely to be less than -50dB.

Note that an *endtime* needs to be specified. On the Command Line, you can actually set an *endtime* beyond the end of the soundfile (so you don't have to look it up) and the program will then use the correct end of the soundfile. *Sound Loom* automatically sets the end-of-file as the default (you can see this by sliding the parameter bar to the far right), but if you enter an *endtime* after the end of the file, it will create a decay right up to the end of the file.

As an aside, finding the duration of a soundfile is often needed, for example when you are setting times in a breakpoint file. This information is available in various ways in the CDP system:

- In *Sound Loom* there is a button named 'Src Props' on the Parameter Page when you go to PROCESS a sound. When you click on this, you see all the properties of the sound, including its duration.
- In *Soundshaper*, after you open a soundfile, its length is displayed in the central "play" panel.

- Running SNDINFO LEN will display the duration of a soundfile. This function can be accessed in either GUI or via the Command Line.
- Running DIRSF on the command line will list all soundfiles (or enter `DIRSF soundfilename.wav/.aif` for just the one soundfile) and display its duration.

However, be aware that some displays could be rounded, e.g., to three decimal places. This means that the final number could in fact be lower than the one displayed, generating an 'out of range' error message regarding *endtime*: e.g., .3867 is rounded to .387. Reducing the value of the last displayed digit by 1 will cure the problem.

End of ENVNU EXPDECAY

ENVNU PEAKCHOP – Isolate peaks in a source and either play back at a specified tempo (Mode 1) or output a peak-isolating envelope

Usage

envnu peakchop 1 *insndfile outsndfile wsize pkwidth risetime tempo gain* [**-g***gate* **-q***skew* **-s***scatter* **-norm -r***repeat* **-m***miss*]
envnu peakchop 2 *insndfile outsndfile wsize pkwidth risetime* [**-g***gate* **-q***skew*]

Example command line to double the tempo of peaks (other parameters are the defaults):

```
envnu peakchop 1 count countchop 50 20 10 120 1
```

Modes

1 – Isolate peaks and play back at a specified tempo
2 – Output a peak-isolating envelope

Parameters

insndfile – input soundfile
outsndfile – output soundfile
wsize – windowsize (in milliseconds) for extracting the envelope (Range: 1 to 64, Default: 50)
pkwidth – width in milliseconds of retained peaks (Range: 0 to 1000, Default: 20ms)
risetime – risetime in milliseconds from zero to peak (Default: 10ms)
tempo – tempo of resulting output (Metronome Mark = events per minute: as is customary, 60 means 1 per second, 120 means 2 per second etc.) (Range: 20 to 3000)
gain – lower this if rapid tempo causes peaks to overlap (The program should produce a warning message should clipping result from the settings.) (Range: 0 to 1, Default 1)
Optional Parameters:
-g*gate* – level (relative to max) below which peaks are ignored (Range: 0 to 1, Default: 0)
-q*skew* – envelope centring on peak (Range: 0 to 1, Default: 0.25)

- 0.5 = centred
- 0 = peak at envelope start
- 1 = peak at envelope end

-s*scatter* – randomisation of output times (Range: 0 to 1, Default: 0)
-norm – force peakevent levels towards that of the loudest peak (Range: 0 to 1, Default: 0)
-r*repeat* – number of times peakevents are repeated (Range: 0 to 256, Default: 0)
-m*miss* – use peakevent, then skip the next *miss* peakevents (Range: 0 to 64, Default: 0)

All parameters may vary in time, except *wsize*, *gate* and *skew*.
NB: Times in breakpoint files for this program are **times in the output**.

Understanding the ENVNU PEAKCHOP Process

This a delightful program. Let us first get an overview of Mode **1**. The example isolates the peaks in the spoken count from 1 to 10 and then halves the distance between each peak. Because the *wsize* and *pkwidth* are relatively small, one hears only a brief 'dot' of sound for each peak. Increasing these parameters, especially *pkwidth* means that more of the original sound around each peak is used. The maximum value for *pkwidth* (1000) is 1 second, so with higher values for *tempo*, portions of soundfile will overlap and some repetitions will be heard. Note that the upper limit for *tempo* is 3000! As with most CDP programs, there is much opportunity to explore and play here.

More about the *pkwidth* parameter in ENVNU PEAKCHOP

The *pkwidth* parameter of PEAKCHOP functions slightly differently in Modes **1** & **2**.

Pkwidth in Mode 1

- ENVNU PEAKCHOP first finds the peaks in the source.
- It then defines an area surrounding/following the peak, whose duration is determined by *pkwidth*.
- It then cuts chunks from the source at these places, cutting each chunk *independently*. This means that the first chunk can extend beyond the beginning of the next chunk(s) to be cut.
- These chunks are then played back in turn, at the specified *tempo*.

Imagine a source sound with regular peaks every 0.1 seconds. If you define the (output) *pkwidth* to be 0.05 secs (i.e., shorter than the peaks in the source)...

- Each source peaks would be trimmed shorter in the output.
- The *tempo* determines where these peak-areas will be placed, in sequence, in the output sound.

At a fast tempo the output peak-areas might overlap one another, and at a slow tempo they might be separated by silence.

However, if you defined the output peakwidth as 0.3 secs (larger than the actual peaks in the source), the chunks you cut out will span an area ***larger*** than the original peaks. In our example the peak-area will in fact span 3 of the peaks in the source (0.3 = 3 x 0.1), so the output sequence of events would be:

- The peak-area around A, in fact spanning peaks ABC
- The peak-area around B, spanning peaks BCD
- The peak-area around C, spanning peaks CDE
- etc

Thus the output sequence of events would be ABC...BCD...CDE...

In a typical source sound, of course, not all the peaks are of the same length, but the same reasoning applies. If the *pkwidth* parameter ***exceeds*** the width of any source peak (A), then a little bit (or a lot) of the succeeding peak(s) may be grabbed into the peak-area associated with peak A.

In our example, if the peaks in the source are at
0.1 : 0.2 : 0.3 : 0.4 : 0.5 : 0.6 : 0.7 : 0.8 : 0.9 : and 1.0 seconds
and the *pkwidth* is defined as 0.3 secs, the source, will be cut into *overlapping* chunks at, say,

- 0 – 0.3 secs
- 0.1 – 0.4 secs
- 0.2 – 0.5 secs
- etc.

which are then reassembled into the output at the specified tempo.

Pkwidth **in Mode 2**

In Mode **2** we are creating an envelope over the source *without changing its tempo*. In this case it is not possible to have a *pkwidth* which is larger than the peakwidths in the source because, to achieve that, the envelope would have to back-track on itself.

The envelope is a sequence of time-level pairs, with times ***in ascending order***, which is to be applied to the original source sound. Each segment of the envelope (corresponding to a single peak) has 4 values: **0 1 1 0** going from silent, to full level, where it remains for the peakwidth, then descends to silence.

To simplify the discussion, let's suppose that the envelope rise and decay durations are zero, so we can think of just the *two* times corresponding to where the enveloping peak starts and ends. If the *peakwidth* were specified as ***longer*** than the peaks in the input, the envelope times (for our example sound) would have to be at:

- 0.0 and 0.3 (for the start and end of peak 1)
- 0.1 and 0.4 (for the start and end of peak 2)
- 0.2 and 0.5 (for the start and end of peak 3)
- etc.

so the time sequence in the envelope would have to be:
0 : 0.3 : 0.1 : 0.4 : 0.2 : 0.5. But these times are ***not*** in ascending order, so the envelope data is not valid.

It is clear from this that the *pkwidth* in Mode **2** has to be less than (or equal to) the width of the input peaks in the source material, or the enveloping will fail.

- Where *pkwidth* has a fixed value, it must therefore be less than (or equal to) the ***minimum*** peak width in the input.
- Where the *pkwidth* parameter varies over time, at the time of each input peak it must less than (or equal to) the peak width in the input at that time.

The input peakwidth can be judged by looking at the sound waveform, or guessed and the *pkwidth* parameter subsequently adjusted until it is accepted as a valid value by Mode **2**.

Mode **2** finds the peaks and produces generates an envelope which is written to the outfile as a text file of *time value* pairs. This envelope can be applied to the original soundfile to isolate those peaks *while preserving their original tempo*. This can be done with ENVEL IMPOSE, Mode **3** (takes a breakpoint file as input).

Musical Applications

Mode **1** can be used to restructure a sound into (regularly spaced) staccato bursts at varying tempi, and given the maximum value for *tempo* can be sped up considerably. They can also be slowed down (spread out) to well less than 1 per second. These will normally produce a regularly spaced series of pulses at the specified tempo.

However, a very wide *pkwidth* (100+) together with a fast *tempo* (300+) will produce overlapping which may result in rhythms that may appear to be irregular. On the other hand, a narrow *pkwidth* (e.g., 30) and a fast *tempo* (e.g., 700) seems to produce sub-pulses between the main peaks – hard to describe. The peaks are spread out nicely when the *tempo* is very slow (e.g., 30).

Bear in mind that *tempo* is a time-varying parameter!

The amount of original signal coming through can be increased by using higher values for *pkwidth*. The combination of a high *tempo* and a wide *pkwidth* will result in overlaps and repetitions – indeed, some rearrangement is likely to take place, with peaks later in the source appearing earlier in the output.

Mode **2** is useful for rhythmicising musical materials which have definable peaks by aurally isolating those peaks. Speech or melodies on instruments with attack-decay type events, such as guitar, provide good source material for this process.

End of ENVNU PEAKCHOP

Last Updated 28 Mar 2016 -- HTML5 version
Documentation: Archer Endrich, revised Robert Fraser