Ircam
documentation

- *Research reports*
- *Musical works*
- ***Software***

OpenMusic

OM-AS Library

First English Edition, November 1998

IRCAM Centre Georges Pompidou

This manual was written by Hans Tutschku, and was produced under the editorial responsibility of Marc Battier, Marketing Office, Ircam.

OpenMusic was conceived and programmed by
Gérard Assayag and Carlos Agon.

The OM-AS library was conceived and programmed by Hans Tutschku.

First edition of the documentation, November 1998.

This documentation corresponds to version 1.0 of the OM-AS library, and to version 2.0 or higher of OpenMusic.

Ircam
1, place Igor-Stravinsky
F-75004 Paris
Tel. 01 44 78 49 62
Fax 01 44 78 15 40
E-mail ircam-doc@ircam.fr

IRCAM Users' group

The use of this software and its documentation is restricted to members of the Ircam software users' group. For any supplementary information, contact:

Département de la Valorisation
Ircam
Place Stravinsky, F-75004 Paris

Tel. 01 44 78 49 62
Fax 01 44 78 15 40
E-mail: bousac@ircam.fr

Send comments or suggestions to the editor:

E-mail: bam@ircam.fr
Mail: Marc Battier,
Ircam, Département de la Valorisation
Place Stravinsky, F-75004 Paris

http://www.ircam.fr/forumnet

Contenu

Introduction 1
Overview 2
Communication between OpenMusic and AudioSculpt 3
Use of the command line and handling of parameter files in AudioSculpt 3
Practical example 5

1 time stretch 6
1.1 stretch-dyn-random 7
1.2 stretch-dyn-exact 9
1.3 marker-stretch 11

2 transposition 13
2.2 rand-trans-steps 15
2.3 trans-melody 17
2.4 vibrato 19

3 fbande 21
3.2 fund-zero-filter 25
3.3 fbande-parallel 26
3.4 fbande-not-parallel 28
3.5 fbande-melody 30

4 fbreakpt 32
4.1 rhythm-fbreakpt 33
4.2 partials-fbreakpt 35
4.3 partials-fbreakpt-bpf 40
4.4 partials-amp-filter 42
4.5 fbreakpt-rand 44

5 formantfilter 46
5.2 seq-to-fifof-bpf 48
5.3 fifof-rand 50
5.4 fundamental-fof 52
5.5 fund-fifof-rand 54

6 cross 56
6.2 cross-rand-float 58
6.3 cross-rand-int-rhythm 59
6.4 cross-rand-float-rhythm 61
6.5 cross-rand-combinations 63
6.6 cross-rand-comb-rhythm 65
6.7 cross-bpflib 67

7 util 69

8 Index 70

Introduction

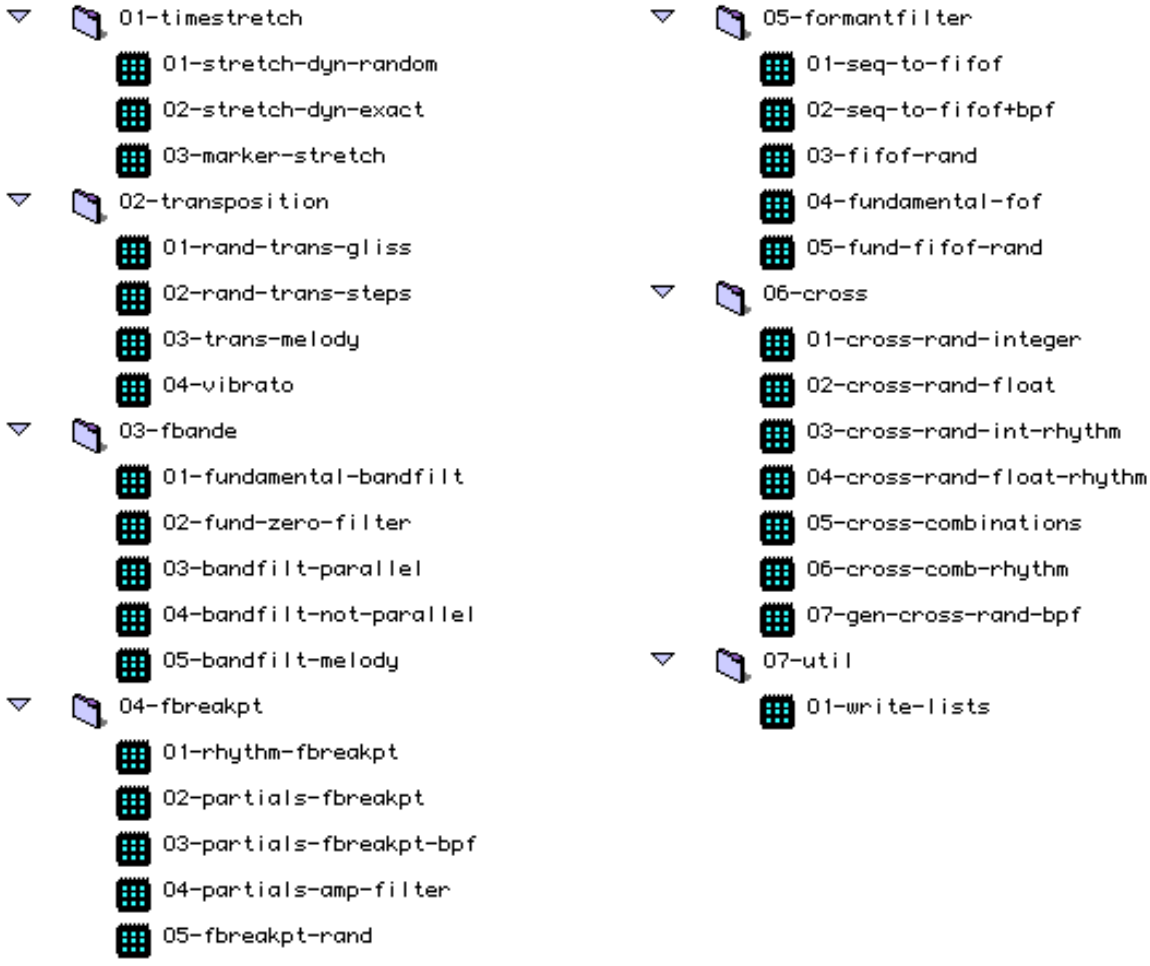
The library OM-AudioSculpt is a collections of functions to generate parameter files for several sound transformations in AudioSculpt . Many of them are a "compiled" version of Patchwork-patches I used during recent years for my own compositions.

Some functions take analysis-text files from AudioSculpt and transform them into parameter files.
I want to thank Mikhail Malt, Gerard Assayag and Carlos Agon Amado for their help and critics.

Hans Tutschku, November 1998

Overview

Here are all functions of the OM-AS-library.



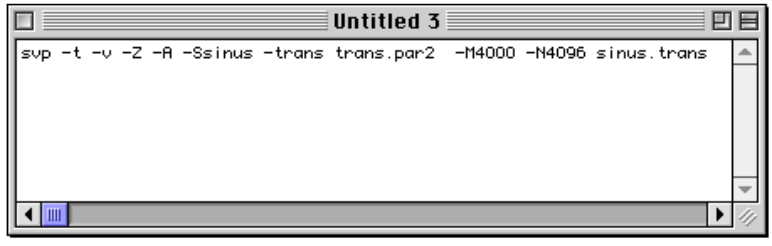
There is a demo-patch for every function with explanations and examples for the use of the SVP-command line in AudioSculpt . The functions are grouped by type of transformation. All transformations use FFT-size 4096 or higher, to insure having good frequency resolution during analysis/resynthesis.

Communication between OpenMusic and AudioSculpt

The transfer of data between both programs is done through text files. Analysis processes in AudioSculpt such as "partial-tracking", "place-markers" and "fundamental-analysis" allow one to save/export to a text file. These files then are imported into a "textwin-function" in OpenMusic.
All functions of the OM-AS library save text files as output. These files will become parameter-files for several transformations in AudioSculpt .

Use of the command line and handling of parameter files in AudioSculpt

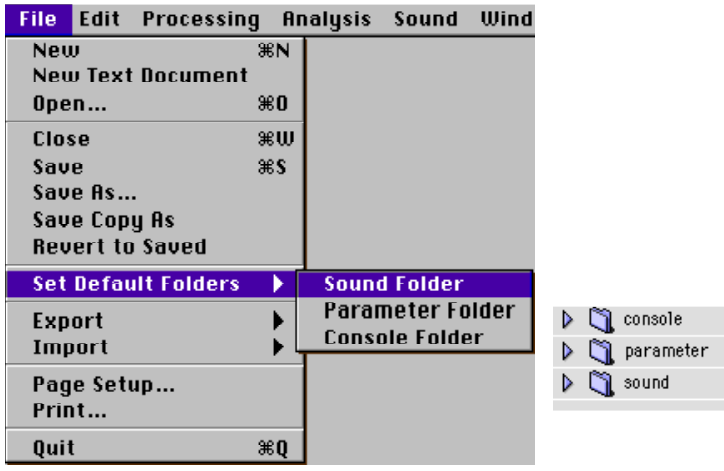
Using command lines one can communicate directly with the SVP synthesis-engine of AudioSculpt , without using the menus.
These command lines can be typed or copied into a textwindow in AudioSculpt .



By hitting the <enter> key (not <return>) the command line will be executed.

A lot of transformations in AudioSculpt are dynamic. This means the parameters can change over time. (A filter curve can evolve, a time stretch-factor can change etc.)

To describe these changes, AudioSculpt uses parameter-files. The library OM-AS is generating such parameter files for different types of transformations . We suggest that you save these parameter files in a folder, called "parameters", that you copy your sounds into a folder "sound" and that you create a third folder called "console".



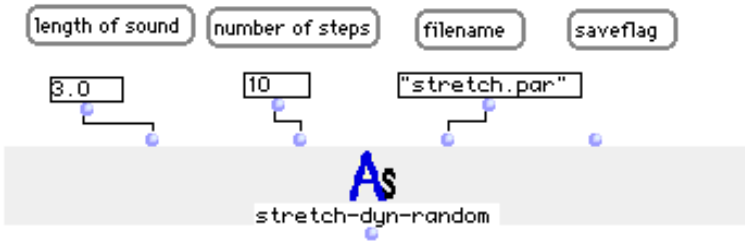
In the menu "Set Default Folders" you point AudioSculpt at these three folders.
A command line in AudioSculpt is a collection of "flags", which pass parameters to the SVP synthesis-engine.
The most important flags used in OM-AS library are:

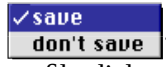
-S	name of inputsound (without a space directly after the flag ex: -Snoise.aiff)
-s	in case of cross-synthesis, the name of second sound
-trans	invokes a transposition - the flag has to be followed by the name of the parameter file; ex: -trans trans.par
-Fbande	invokes a bandfilter - the flag has to be followed by the name of the parameter file; ex: -Fbande fbande.par
-Ffifof	invokes a formantfilter - the flag has to be followed by the name of the parameterfile ex: -Ffifof fifof.par
-Ffof	invokes an interpolating formant filter - the flag has to be followed by the name of the parameter file; ex: -Ffof fof.par
-D	invokes a time stretch - the flag has to be followed (without space!) by the name of the parameter file; ex: -Dtrans.par
-M	Windowsize
-N	FFT-size - the examples always use FFT-size 4096 or bigger

The last argument in the command line is always the name of the resulting sound.

Practical example

Open the patch "01-stretch-dyn-random" in the examples-folder.



Specify the length of your sound (in seconds), give the number of random-steps and a filename (between quotes). Choose from the save-flag-menu 

.
Option-clic, to evaluate the function. The save-file-dialogue opens and you will be asked to save the file. Locate the parameter folder and save the file.

Open AudioSculpt , open a new text document. Type the following line: (you have to replace "mysound" with the name of your own sound)

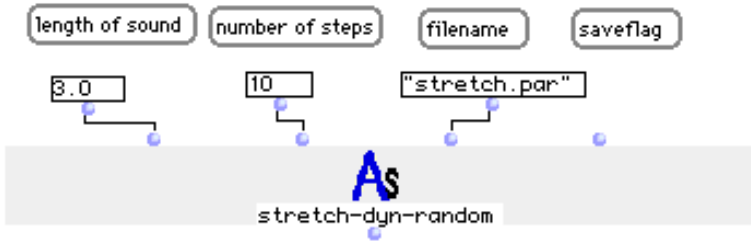
```
svp -t -v -Z -A -Smysound -Dstretch.par  -M4000 -N4096 mysound.stretch
```

Copy your sound in the specified "sound"-folder and set the three folders in the menu "Set Default Folders". Hit the <enter> key. The calculation should start. If you get an error-message, verify that the "Set Default Folders" is pointing to your three folders and that the name of your sound file and the created parameter file are correct.
After calculation you will find the resulting sound in the specified "sound"-folder.

Section 1 - time stretch

This section is made up of a group of functions for doing time stretching

1.1 stretch-dyn-random



Syntax

(AS::stretch-dyn-random *soundlength steps filename saveflag*)

Inputs

soundlength	floating point number to specify the length of the original sound in seconds
steps	whole number for random steps
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

text file

Description

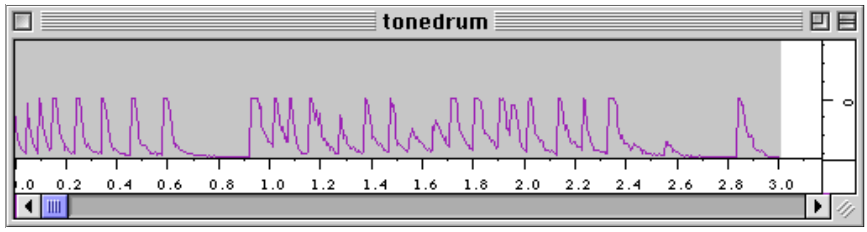
This function is creates a parameter file for doing an AudioSculpt time stretch.
One <step> is generating a triplet of two different contractions and one stretch.

Example for two steps

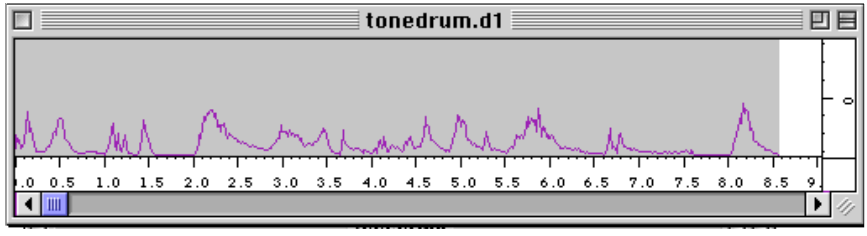
```
time        stretch factor
0.0         0.48      (contraction)
0.022       0.77      (contraction)
0.17        8.3       (stretch)
0.221       0.49      (contraction)
0.368       0.79      (contraction)
0.37        4.0       (stretch)
```

Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -Dstretch.par  -M4000 -N4096 mysound.stretch
```

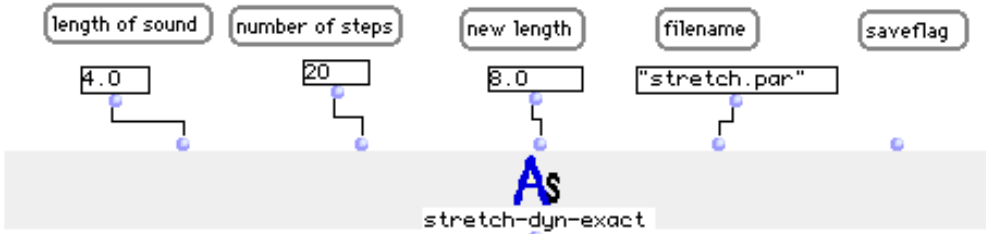


Original sound



Result

1.2 stretch-dyn-exact



Syntax

(AS::stretch-dyn-exact *soundlength steps newlength filename save-flag*)

Inputs

soundlength	floating point number to specify the length of the original sound file in seconds
steps	whole number for random steps
newlength	floating point number to specify the approximate length of the new sound in seconds
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

text file

Description

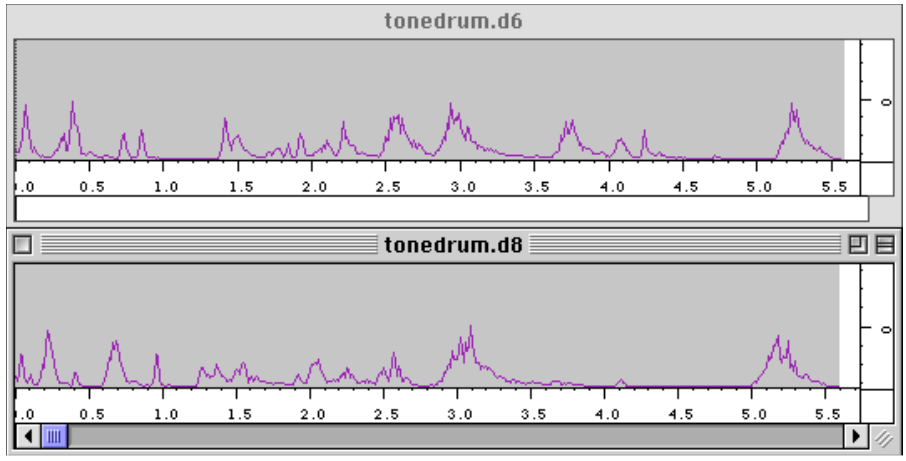
This function creates a parameter file for doing an AudioSculpt time stretch.
One <step> is generating a triplet of two different contractions and one stretch.
This version allows one to determine approximately the length of the new sound after the stretch/compression has taken place.

Example for two steps

```
time        stretch factor
0.0         0.48      (contraction)
0.022       0.77      (contraction)
0.17        8.3       (stretch)
0.221       0.49      (contraction)
0.368       0.79      (contraction)
0.37        4.0       (stretch)
```

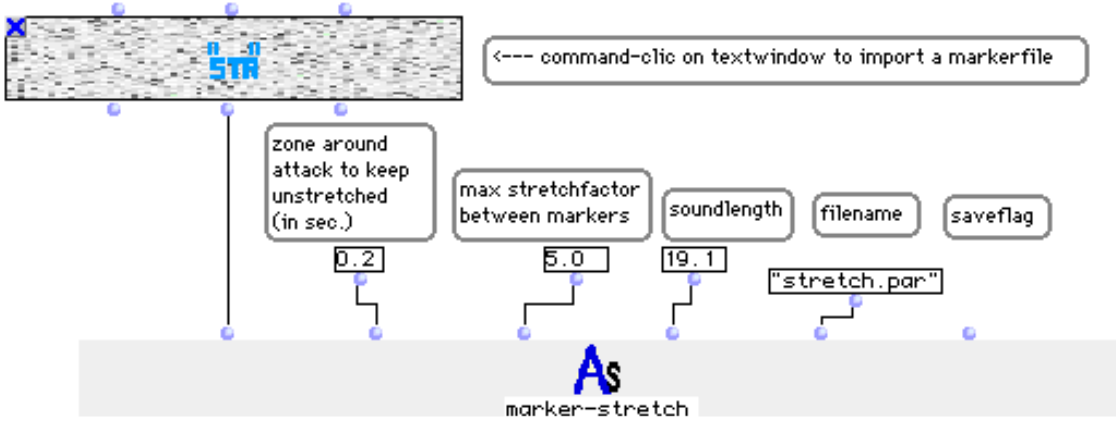
Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -Dstretch.par  -M4000 -N4096 mysound.stretch
```



Two resulting sounds with different stretch-parameters. Their resulting length is the same.

1.3 marker-stretch



Syntax

(AS::marker-stretch *markers zone factor soundlength filename save-flag*)

Inputs

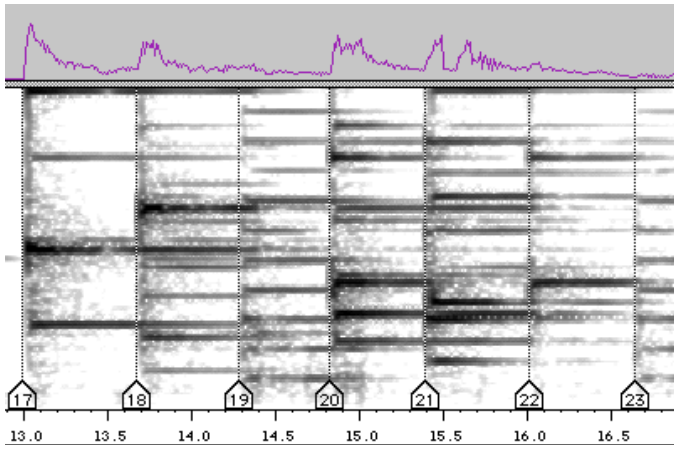
markers	marker file from AudioSculpt in the following format (MARKERS 5 0.126 0.74 1.41 2.031 3.414)
zone	floating point number to specify the zone not to stretch in seconds around the marker
factor	floating point number for maximum stretch factor
soundlength	floating point number to specify the length of the original sound in seconds
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

text file

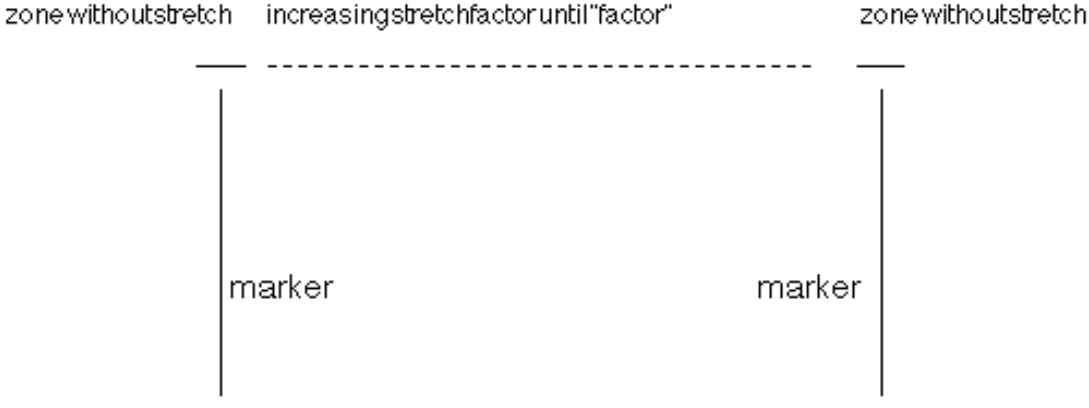
Description

Takes a marker file from AudioSculpt and creates a parameter file for time stretch. Around each marker one can define a zone not to stretch, to keep this portion of sound untouched. This works very well with percussive sounds, where the attacks will not be stretched, just the resonance.



Markers in AudioSculpt placed on every rapid change in the spectrum

The time points of a marker are taken to calculate a zone around each marker, where no time stretch takes place (stretch factor = 1). After each marker the stretch factor increases up to the "factor" (the maximal stretch factor), to jump back to value 1 just before the next marker.



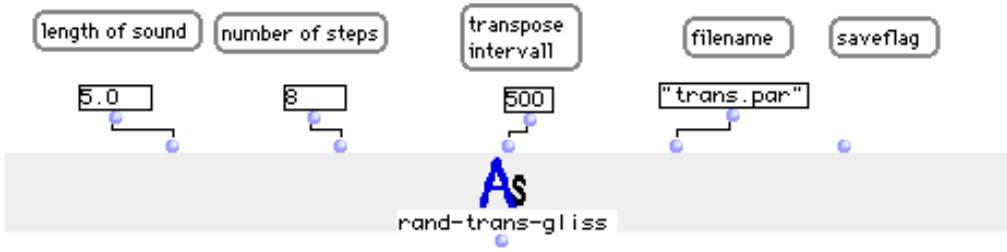
Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -Dstretch.par  -M4000 -N4096 mysound.stretch
```

Section 2 - transposition

This section is made up of a group of functions for doing transposition.

2.1 rand-trans-gliss



Syntax

(AS::rand-trans-gliss *soundlength steps randinterval filename saveflag*)

Inputs

soundlength	floating point number to specify the length of the original sound in seconds
steps	whole number for random steps
randinterval	floating point number to specify the transposition interval around the normal pitch
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

text file

Description

This function creates a parameter file for AudioSculpt transposition. The transposition interval is given in cents around the original pitch. Since AudioSculpt interpolates the transpositionvalues between time points, you get glissandi.

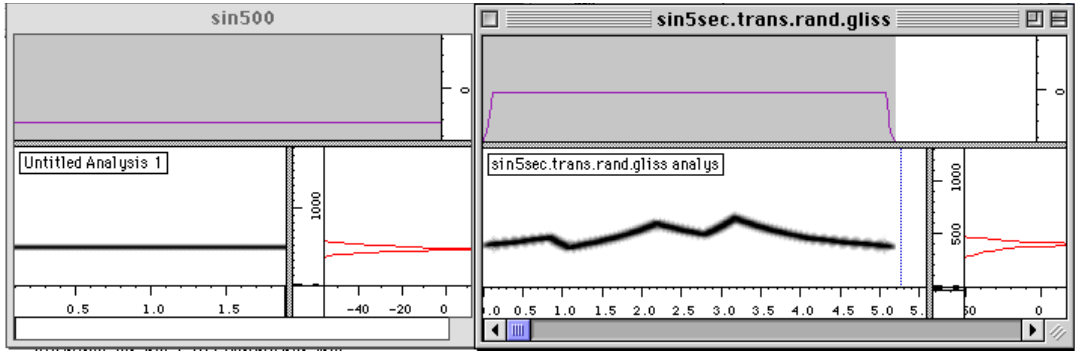
Example

```
time        transposition factor
```

```
0.0         149
0.954       -161
1.285       27
2.257       -240
3.471       -131
4.196       241
4.764       -118
5.0         -44
```

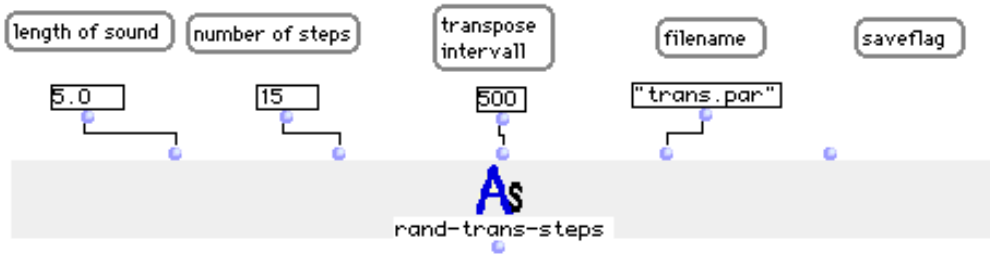
Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -trans trans.par  -M4000 -N4096 mysound.trans
```



original sinewave and resulting sound

2.2 rand-trans-steps



Syntax

(AS::rand-trans-steps *soundlength steps randinterval filename saveflag*)

Inputs

soundlength	floating point number to specify the length of the original sound in seconds
steps	whole number for random steps
randinterval	floating point number to specify the transposition interval around the normal pitch
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

text file

Description

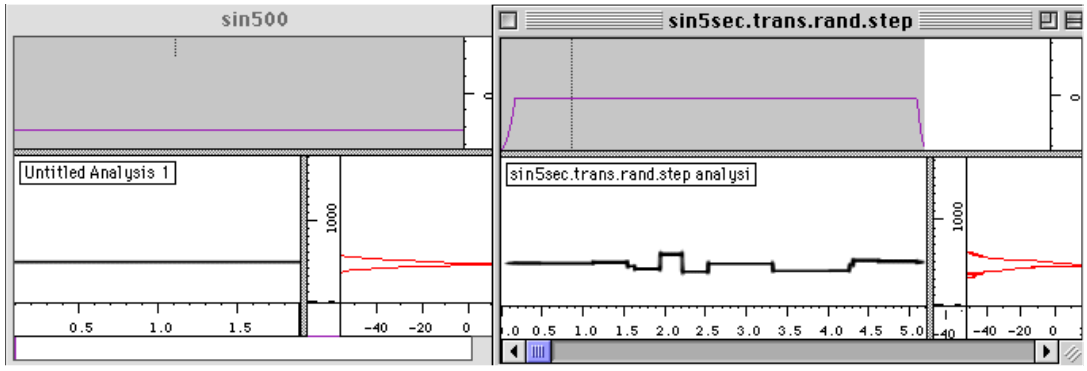
This function creates a parameter file for AudioSculpt transposition.
The transposition interval is in cents to determine the random range around the original pitch.
This function holds the transposition value between two timepoints - no glissandi.

Example

```
time        transposition factor
0.0         -201
0.35        -201
0.351       426
0.848       426
0.849       8
1.42        8
1.421       -292
1.64        -292
```

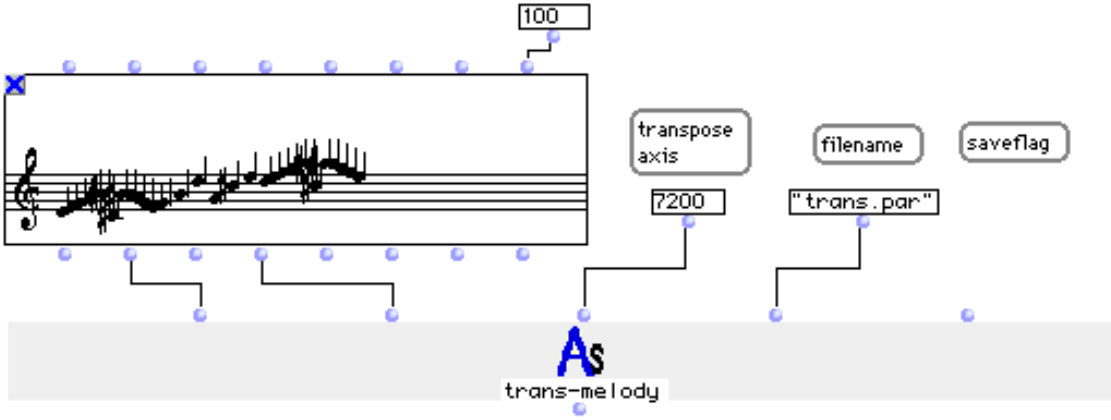
Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -trans trans.par  -M4000 -N4096 mysound.trans
```



original sinewave and resulting sound

2.3 trans-melody



Syntax

(AS::trans-melody *LMIDIC LDUR mirror filename saveflag*)

Inputs

LMIDIC	list of Midicent values
LDUR	list of duration's in milliseconds
mirror	whole number to specify the transposition axis - all notes higher will create a transposition upwards, all notes lower will create a transposition downwards
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

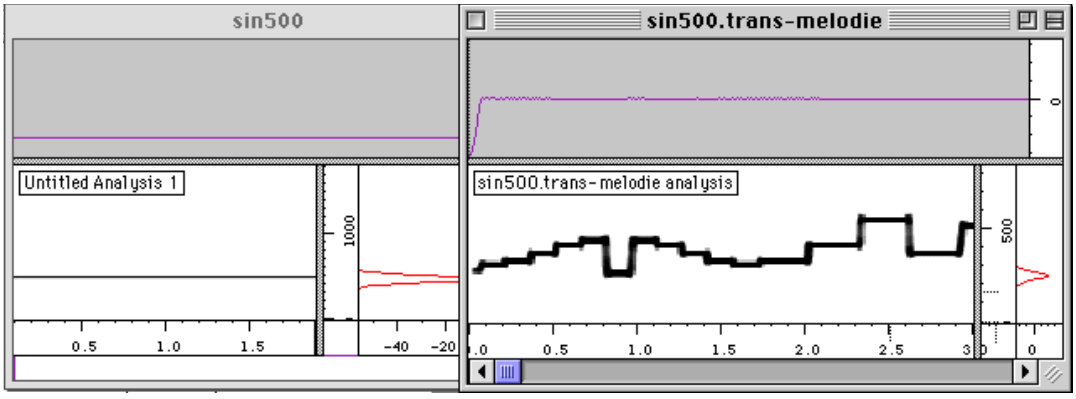
text file

Description

This function creates a parameter file for AudioSculpt transposition from two lists:
Midicents and durations. (which can come from a cordseq)

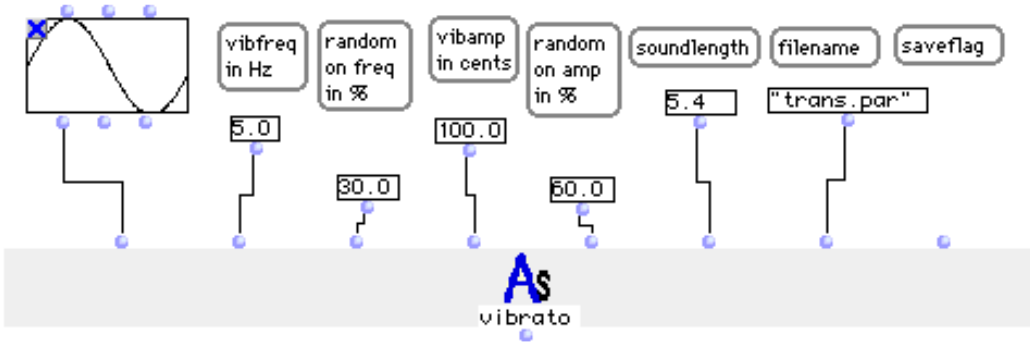
Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -trans trans.par  -M4000 -N4096 mysound.trans
```



original sinewave and resulting sound

2.4 vibrato



Syntax

(AS::trans-melodie *vibfunc vibfreq freqrand vibamp amprand soundlength filename saveflag*)

Inputs

vibfunc	bpf-function for vibrato (waveform)
vibfreq	floating point number - frequency of vibrato
freqrand	whole or floating point number between 0 and 100- % of random on frequency
vibamp	whole number - amplitude of vibrato in Midicents around the original pitch
amprand	whole or floating point number between 0 and 100- % of random on amplitude
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

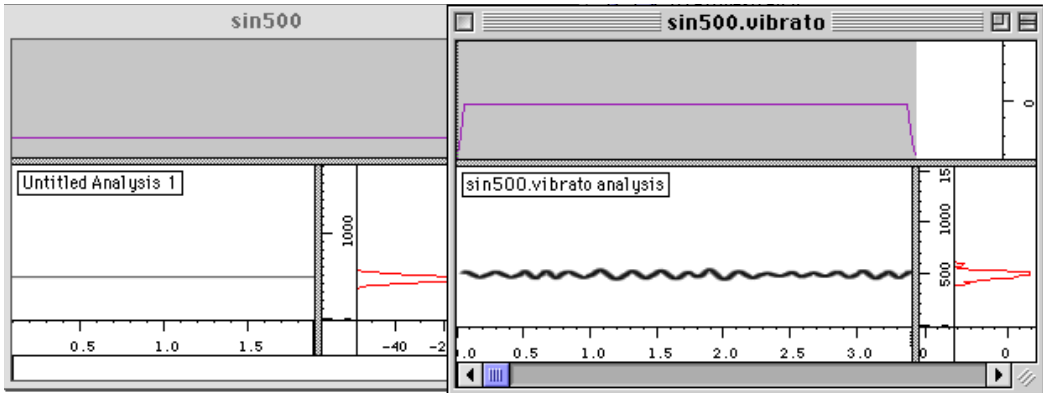
text file

Description

This function creates a parameter file for doing an AudioSculpt transposition. You can specify a vibrato function, the frequency and amplitude for the vibrato. For vibfreq and vibamp you can also specify an amount of random in %.

Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -trans trans.par  -M4000 -N4096 mysound.trans
```

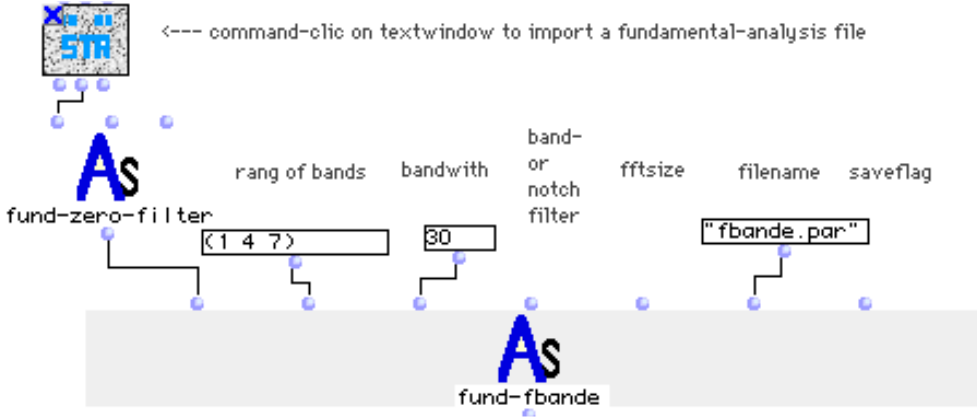


original sinewave and resulting sound

Section 3 - fbande

This section is made up of a group of functions for using a bandfilter "fbande" in AudioSculpt .

3.1 fund-fbande



Syntax

(AS::fund-fbande *fundamental rang bw bandswitch fftsize filename saveflag*)

Inputs

fundamental	list in form of text file „fundamental analysis“ from AudioSculpt
rang	list - which frequency bands as multiples of fundamental frequency
bandwidth	floating point number - bandwidth in Hz between 14 Hz and (minimal f0 - 14 Hz)
bandswitch	string / menu to specify whether to keep or to reject bands
fftsize	integer / menu - has to be equal to the value you will give in the commandline (-N).
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

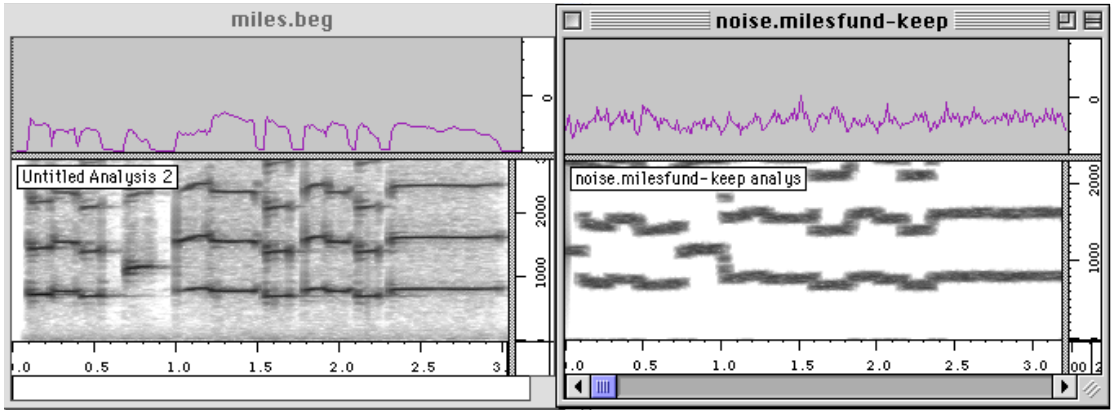
text file

Description

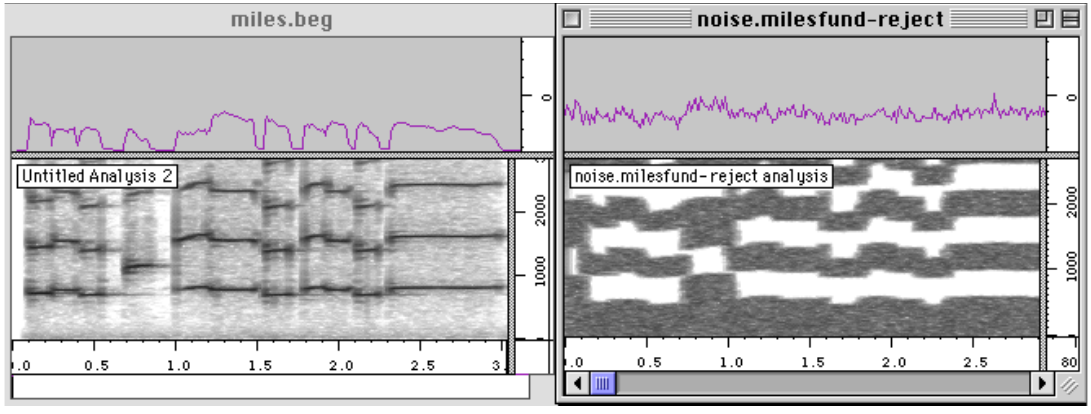
This function creates a parameter file for AudioSculpt Fbande-filter. The frequencies come from a fundamental analysis. <rang> specifieswhich multiples of fundamental frequency are calculated.
fftsize has to be equal to the value you will give in the commandline (-N).
"fund-zero-filter" is eliminating errors in the fundamental analysis file (see 3.2)

Example for the SVP-command line

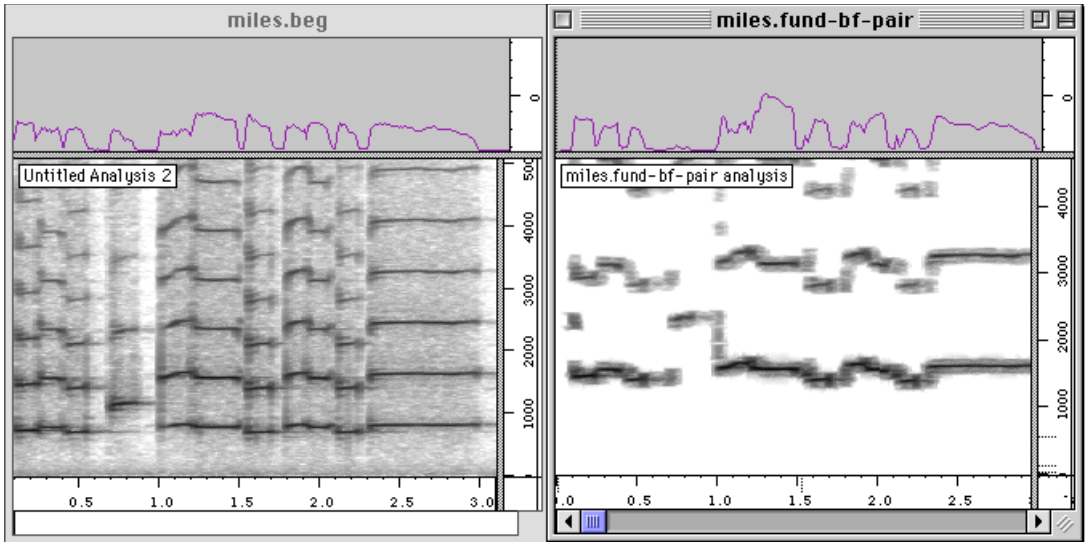
```
svp -t -v -Z -A -Snoise  -Fbande fbande.par  -M4000 -N4096 noise.fb
```



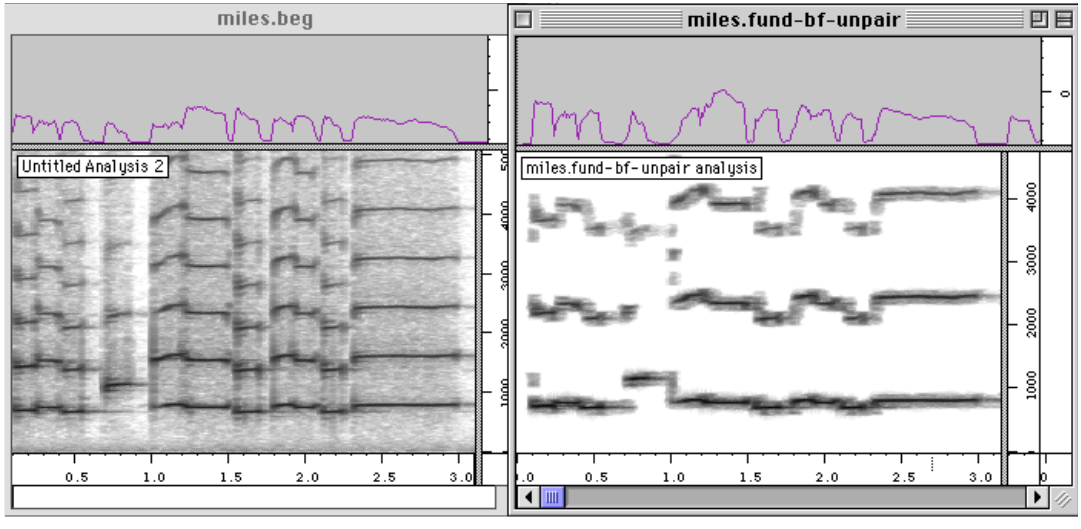
left: original sound on which the fundamental analysis was taken ; right: white noise filtered by the resulting parameter file (option keep bands)



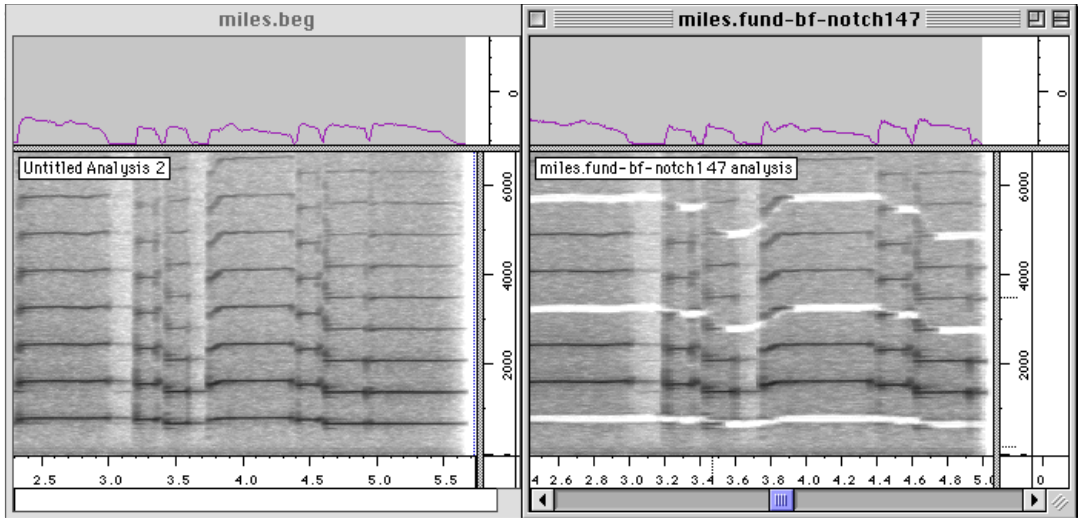
left: original sound on which was taken the fundamental analysis; right: white noise filtered by the resulting parameter file (option reject bands)



left: original sound on which was taken the fundamental analysis; right: same sound filtered (option keep bands, rang: 2 4 6 8 etc., which keeps just the pair harmonics)

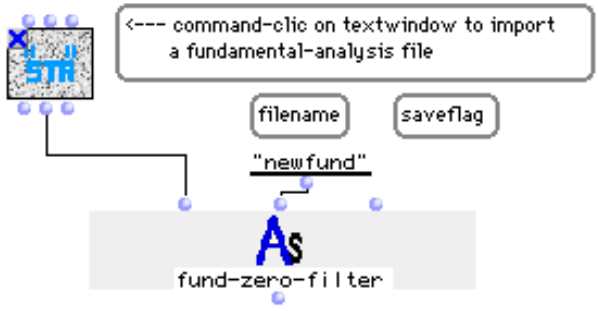


left: original sound on which was taken the fundamental analysis; right: same sound filtered (option keep bands, rang: 1 3 5 7 etc., which keeps just the odd harmonics)



left: original sound on which the fundamental analysis was taken; right: same sound filtered (option reject bands, rang: 1 4 7, rejects first, fourth and sevens partial)

3.2 fund-zero-filter



Syntax

(AS::fund-zero-filter *fundamental filename saveflag*)

Inputs

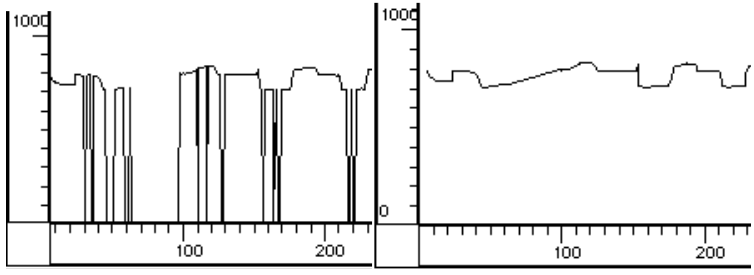
fundamental	list in form of text file „fundamental analysis“ from AudioSculpt
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

text file

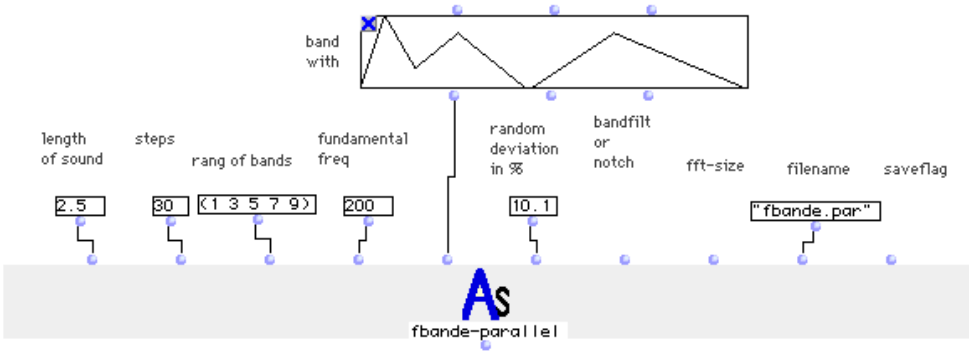
Description

If AudioSculpt can't find an appropriate value for the fundamental frequency at a certain timepoint, it gives 0 Hz. This function eliminates these values.



fundamental without zero-filter and fundamental after zero-filter

3.3 fbande-parallel



Syntax

(AS:: fbande-parallel *soundlength steps rang fundamental bandwidth rand bandswitch fftsize filename saveflag*)

Inputs

soundlength	floating point number to specify the length of the original sound in seconds
steps	whole number for random steps
rang	list - which frequency-bands as multiples of a fundamental frequency
fundamental	whole or floating point number - fundamental frequency
bandwidth	BPF-function - bandwidth in Hz between 14 Hz and (f0 - 14 Hz)
rand	whole number between 0 and 100 - random on fundamental frequency
bandswitch	string / menu to specify whether to keep or reject the bands
fftsize	integer / menu - has to be equal to the value you will give in the commandline (-N).
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

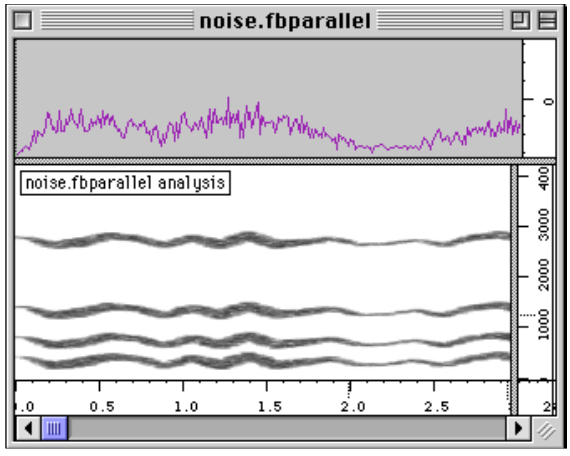
text file

Description

This function creates a parameter file for AudioSculpt Fbande-filter. Starting with a fundamental frequency a certain number of bands centred around the harmonics will be created. <rang> specifies which harmonic bands on the fundamental are calculated.
You have to choose a number of random steps to get variations over time - this variation is parallel for all frequencies
The bandwidth can be drawn in a BPF (will be clipped - depending on fftsize - to avoid overlapping bands).

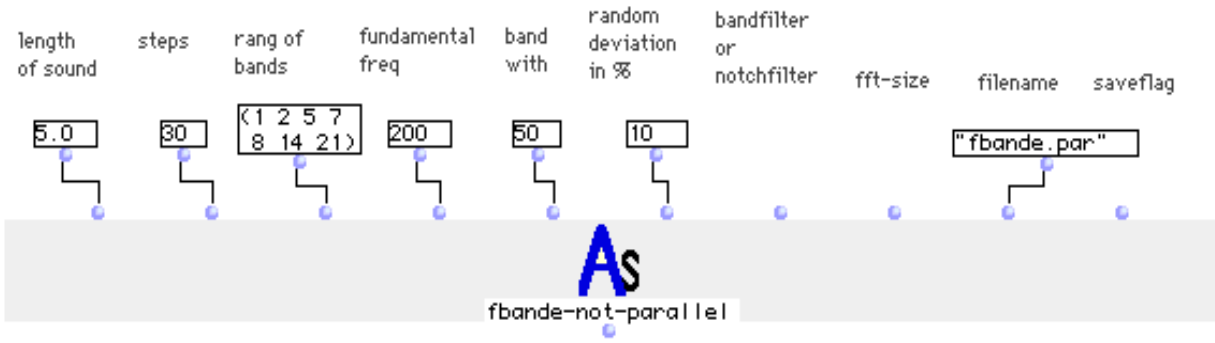
Example for the SVP-command line

```
svp -t -v -Z -A -Snoise  -Fbande fbande.par  -M4000 -N4096 noise.fb
```



parallel random bandfilter with changing bandwith (rang 2 4 7 11)

3.4 fbande-not-parallel



Syntax

(AS:: fbande-not-parallel *soundlength steps rang fundamental bandwidth rand bandswitch fftsize filename saveflag*)

Inputs

soundlength	floating point number to specify the length of the original sound in seconds
steps	whole number for random steps
rang	list - which frequency-bands as multiples of fundamental frequency
fundamental	whole or floating point number - fundamental frequency
bandwidth	floating point number - bandwidth in Hz between 14 Hz and (f0 - 14 Hz)
rand	whole number between 0 and 100 - random on fundamental frequency
bandswitch	string / menu to specify whether to keep or reject the bands
fftsize	integer / menu - has to be equal to the value you will give in the commandline (-N).
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

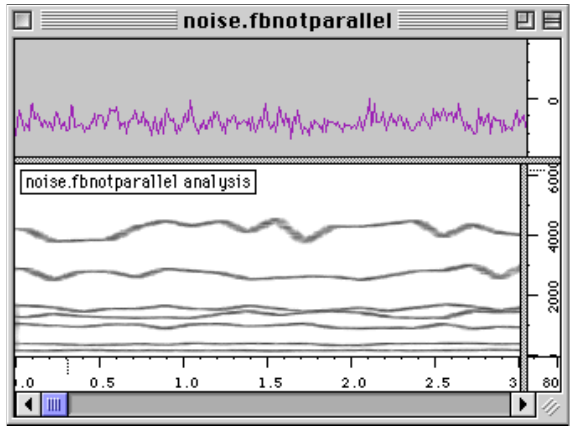
text file

Description

Creates a parameter file for AudioSculpt Fbande-filter. Starting from a fundamental frequency a certain number of bands centred around the harmonics will be created. <rang> specifies, which harmonic bands on the fundamental are calculated.
If random is > 0 you have to choose a number of random steps to get variations over time.
The random-movement is not parallel for all bands.

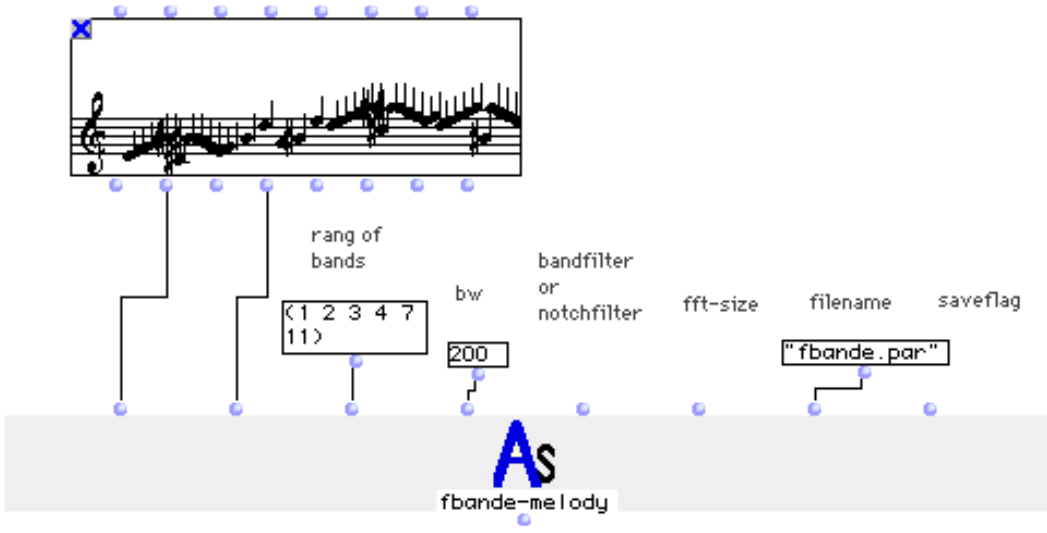
Example for the SVP-command line

```
svp -t -v -Z -A -Snoise  -Fbande fbande.par  -M4000 -N4096 noise.fb
```



the random evolution for all bands is not parallel

3.5 fbande-melody



Syntax

(AS:: fbande-melody *LMIDIC LDUR rang bw bandswitch fft-size filename saveflag*)

Inputs

LMIDIC	list of Midicent values
LDUR	list of duration's in milliseconds
rang	list - which frequency-bands as multiples of fundamental frequency
bandwidth	whole number - bandwidth in Hz between 14 Hz and (f0 - 14 Hz)
bandswitch	string / menu to specify whether to keep or reject the bands
fftsize	integer / menu - has to be equal to the value you will give in the commandline (-N).
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

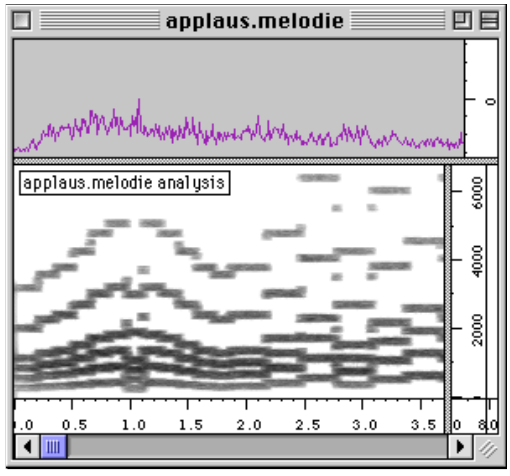
text file

Description

This function converts a melody with Midicents < LMIDIC > and durations < LDUR > into a band filter file. A certain number of bands centred around the harmonics will be created. <rang> specifies, which harmonic bands on the fundamental are calculated.
fftsize has to be equal to the value you will give in the commandline (-N).

Example for the SVP-command line

```
svp -t -v -Z -A -Snoise  -Fbande fbande.par  -M4000 -N4096 noise.fb
```

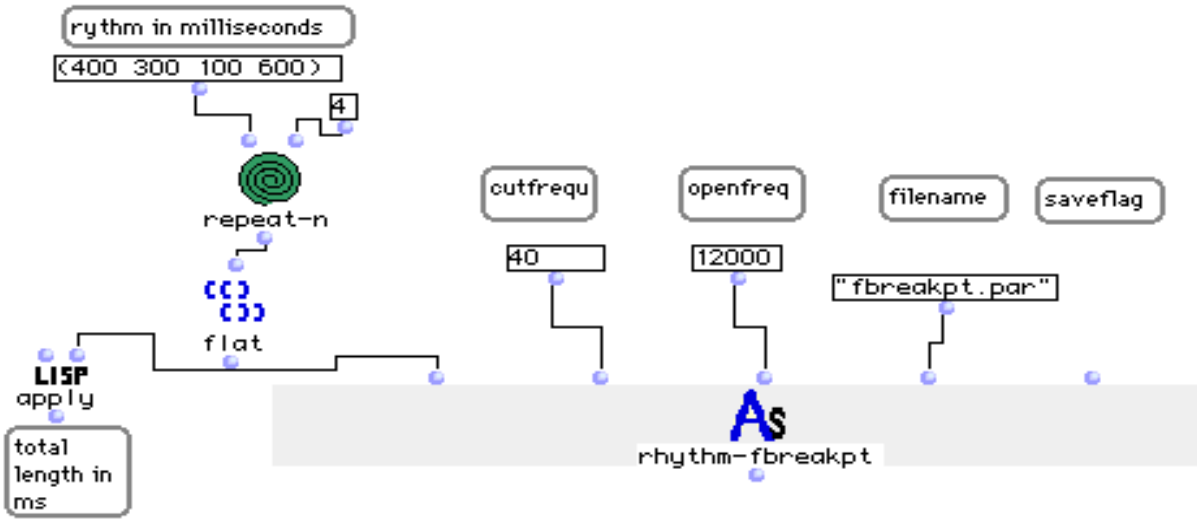


noise filtered by melodie

Section 4 - fbreakpt

This sections is made up of a group of functions for doing a break point filter "fbreakpt" in AudioSculpt .

4.1 rhythm-fbreakpt



Syntax

(AS:: rhythm-fbreakpt *LDUR cutfreq openfreq filename saveflag*)

Inputs

LDUR	list of duration's in milliseconds
cutfreq	whole number - frequency for closed filter
openfreq	whole number - frequency for open filter
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

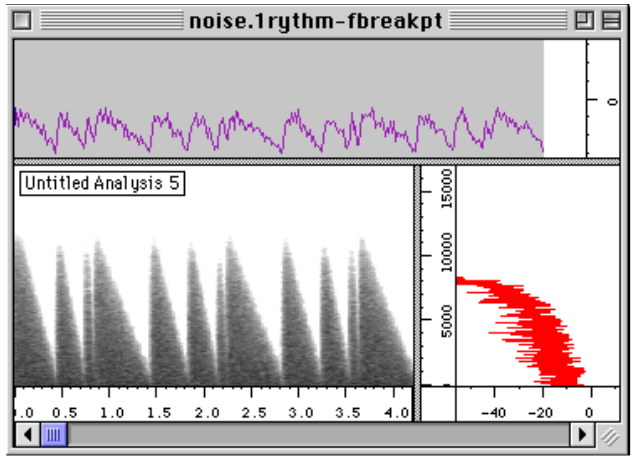
text file

Description

This function converts a rhythm into a sequence for opening and closing a break point filter.

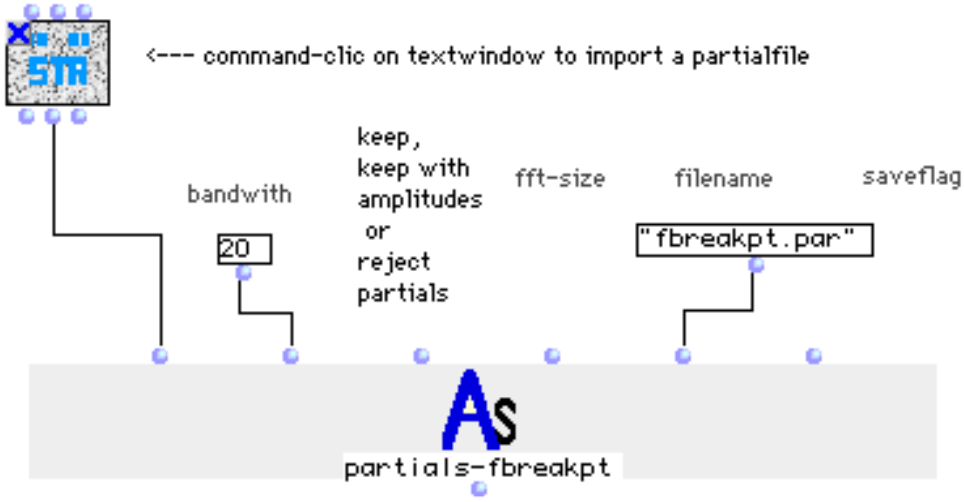
Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -Fbreakpt fbreakpt.par  -M4000 -N4096 my-
    sound.fbrkpt
```



On every onset of the given rhythm, the filter opens up to the "openfreq", then closes over time continously up to the "closefreq".

4.2 partials-fbreakpt



Syntax

(AS:: partials-fbreakpt *partials bw bandswitch fftsize filename saveflag*)

Inputs

partials	list - text file "partials" from AudioSculpt "partial tracking"
bw	whole number - bandwidth in Hz between 14 Hz and (f0 - 14 Hz)
bandswitch	string / menu to specify whether to keep or reject the partials
fftsize	integer / menu - has to be equal to the value you will give in the commandline (-N).
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

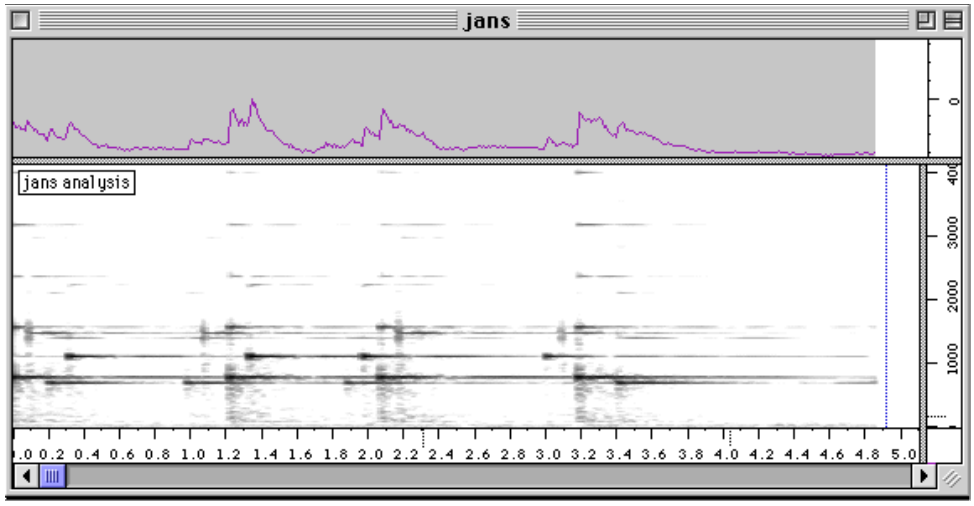
text file

Description

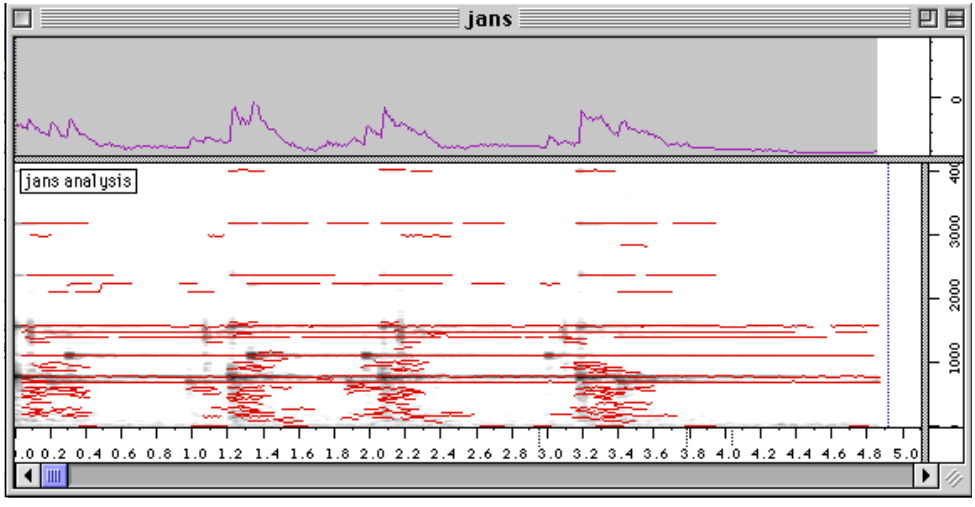
Takes a partial file from AudioSculpt and creates a parameter file for break point filter.
The frequency areas of all traced partials are converted into filter bands and can be used to keep or reject these areas.The option "keep with amp" takes also the amplitudes of the traced partials in account.

Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -Fbreakpt fbreakpt.par  -M4000 -N4096 mysound.fbrkpt
```

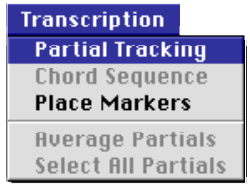


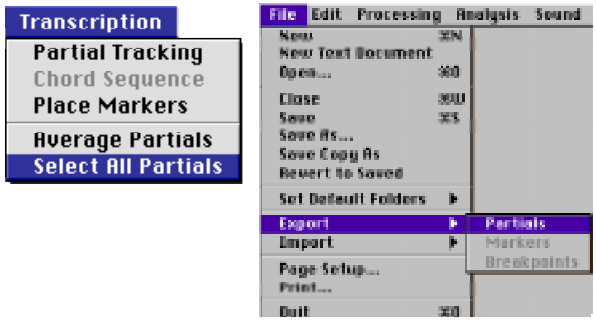
sonogramm of original sound



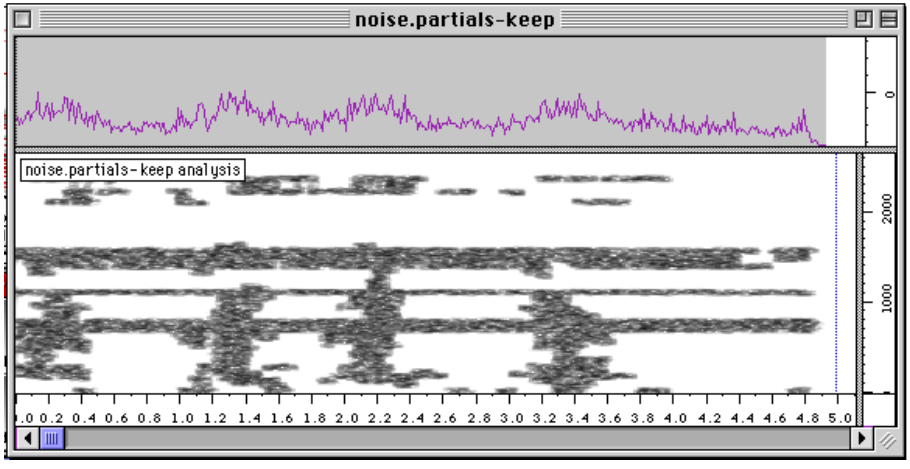
partial tracking in AudioSculpt (with default values)

Select all partials, don't average partials, and export them into a text file.

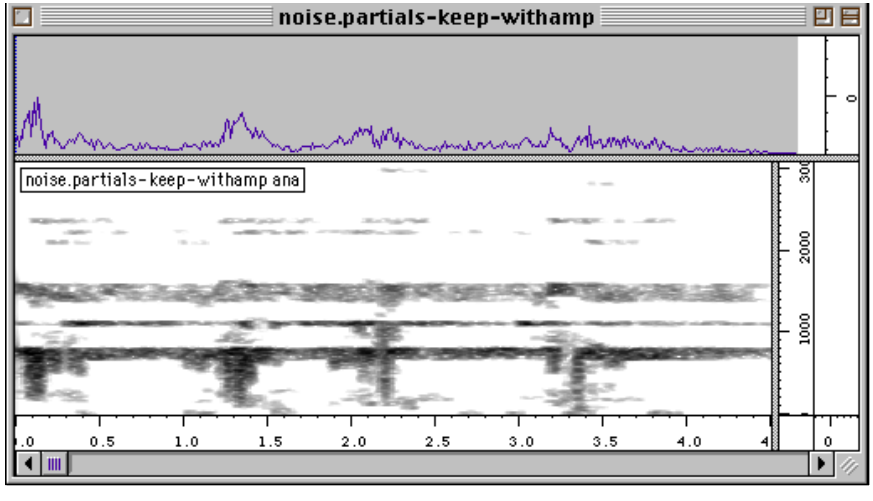




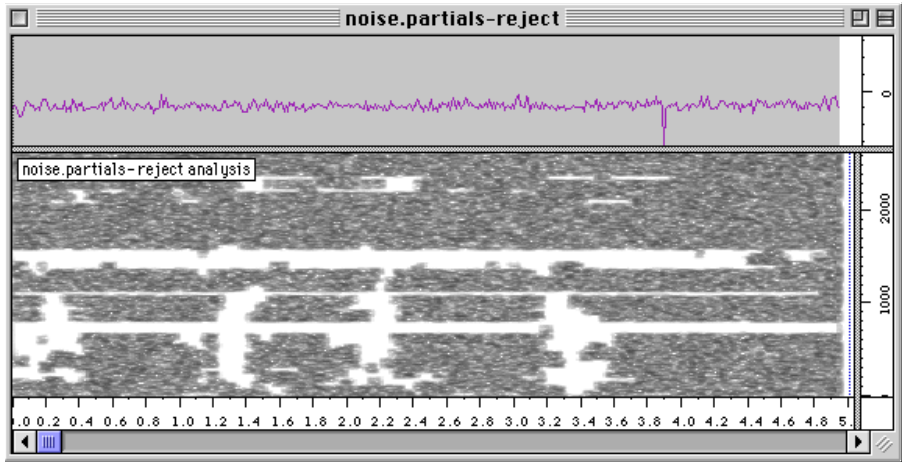
some results of filtering:



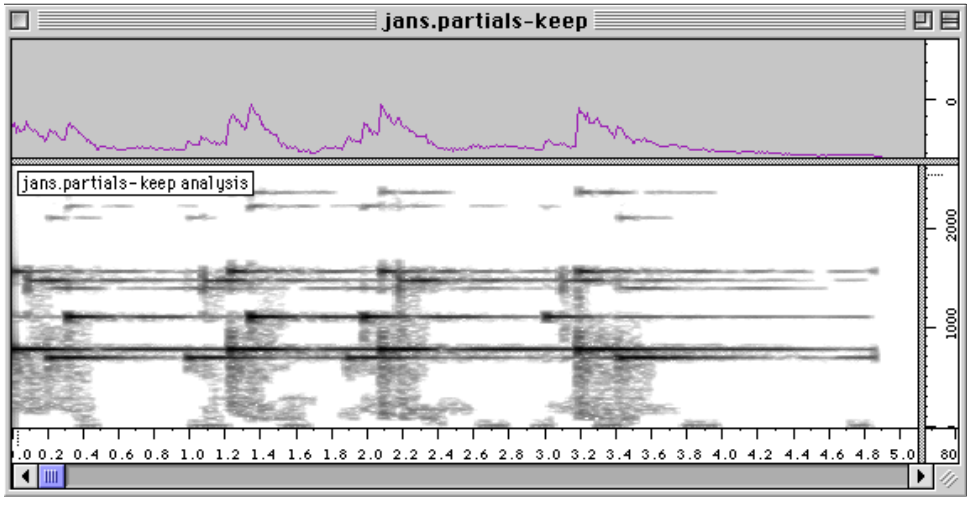
noise filtered (option: keep bands)



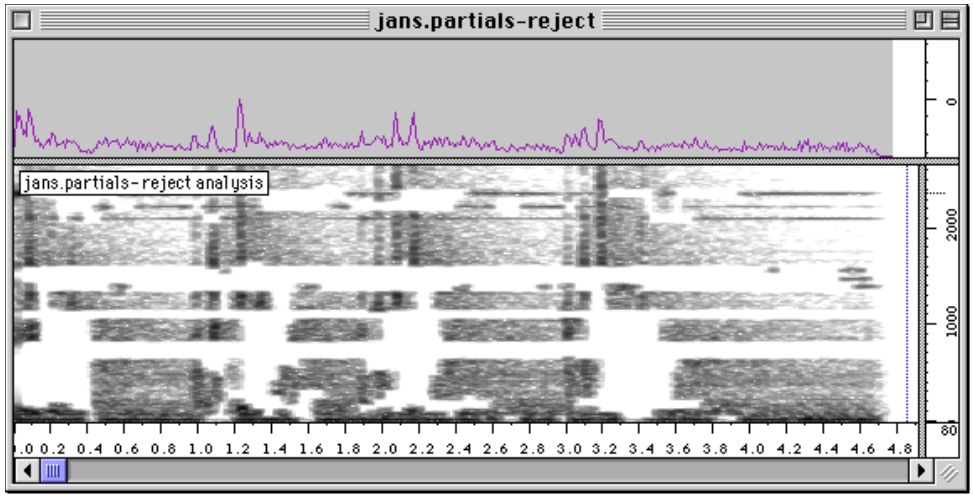
noise filtered (option: keep with amplitudes)



noise filtered (option: reject bands)

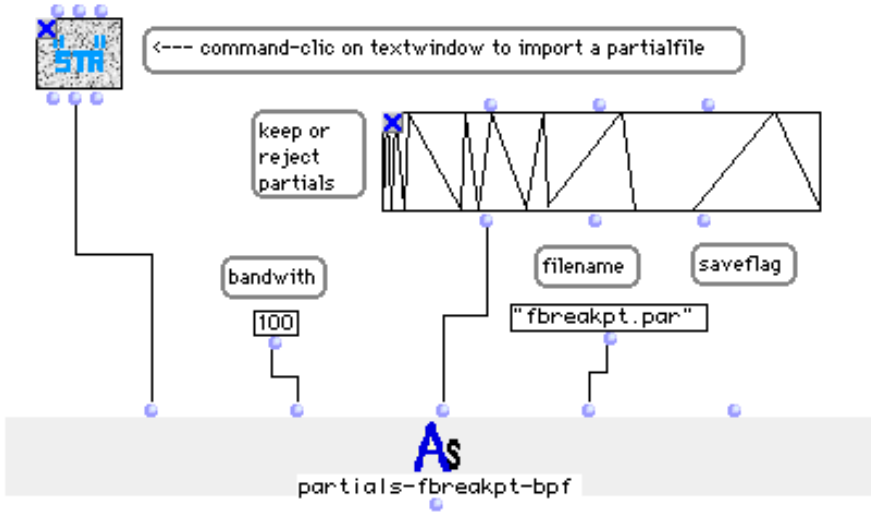


original sound filtered (option: keep bands)



original sound filtered (option: reject bands)

4.3 partials-fbreakpt-bpf



Syntax

(AS:: partials-fbreakpt-bpf *partials bw bandswitch filename save-flag*)

Inputs

partials	list - text file "partials" from AudioSculpt "partial tracking"
bw	whole number - bandwidth in Hz between 14 Hz and (f0 - 14 Hz)
bandswitch	BPF-function - continuous change between keeping and rejecting partials
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

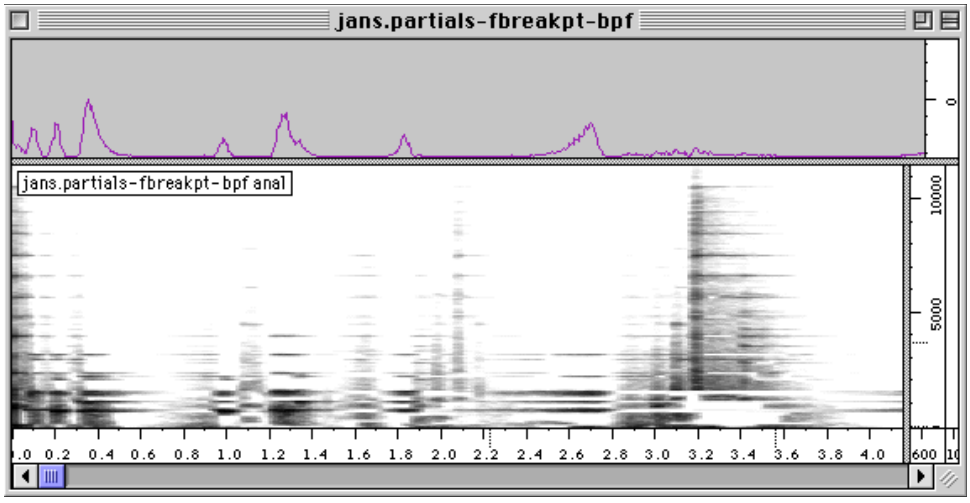
text file

Description

Takes a partial file from AudioSculpt and creates a parameter file for doing a break point filter. The frequency areas of all traced partials are converted into filterbands and can be used to keep or reject these areas.
The bpf assigns changes between keeping and rejecting.

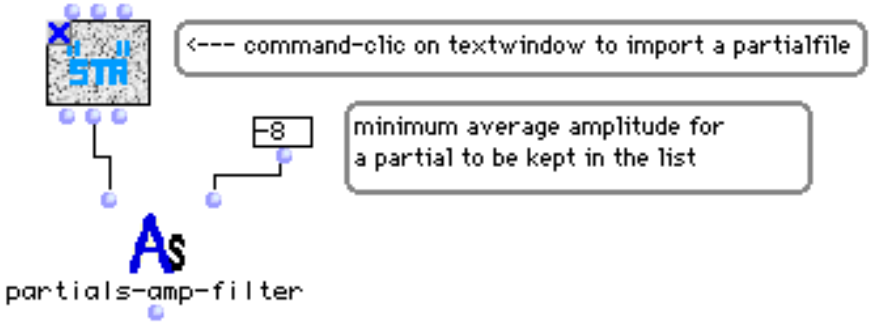
Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -Fbreakpt fbreakpt.par  -M4000 -N4096 my-sound.fbrkpt
```



the BPF-function is used to make continuous changes between keeping and rejecting bands.

4.4 partials-amp-filter



Syntax

(AS:: partials-amp-filter *partials* *ampmin*)

Inputs

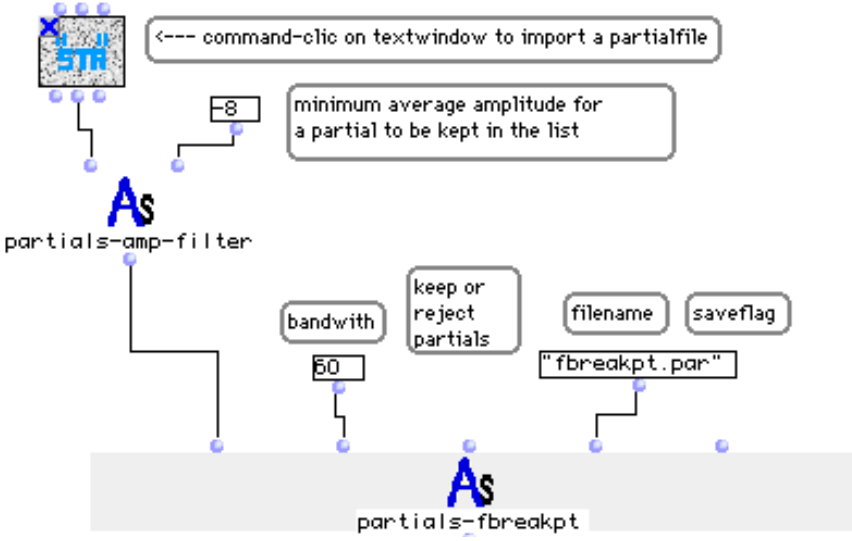
partials	list - text file "partials" from AudioSculpt "partial tracking"
ampmin	floating point number - minimum average amplitude for a partial

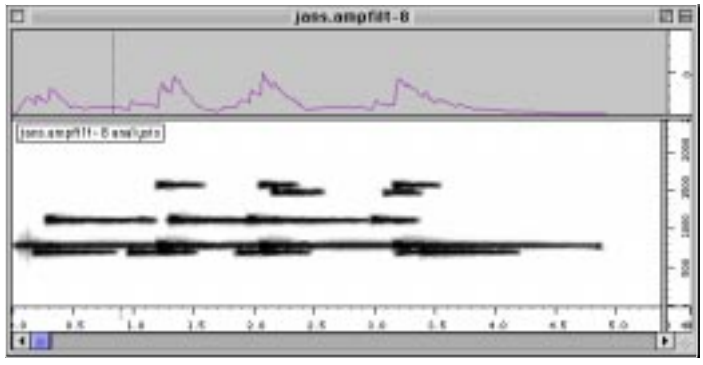
Output

list

Description

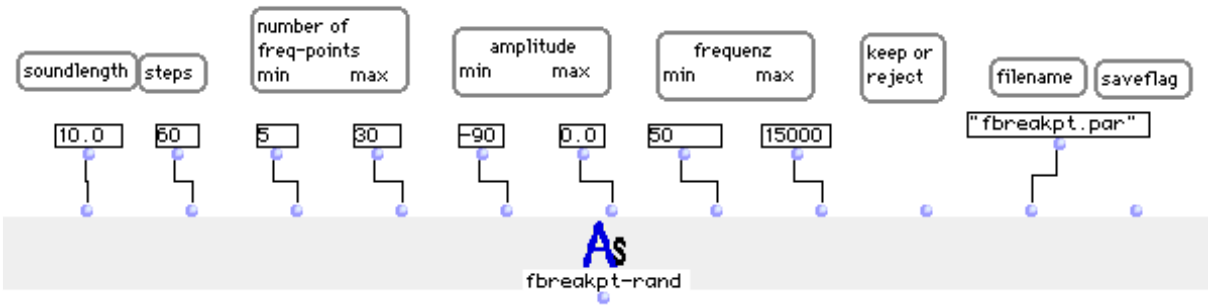
Takes a partial file from AudioSculpt . All partials with a average amplitude lower then <ampmin> will be filtered out. This function is useful in combination with partials-fbreakpt





only the partials stronger than "ampmin" are used to define the filter

4.5 fbreakpt-rand



Syntax

(AS:: fbreakpt-rand *soundlength steps minpoints maxpoints minamp maxamp minfreq maxfreq bandswitch filename saveflag*)

Inputs

soundlength	floating point number to specify the length of the original sound in seconds
steps	whole number for random steps
minpoints	whole number - minimum number of frequency-points at one step
maxpoints	whole number - maximum number of frequency-points at one step
minamp	floating point number - minimum amplitude in dB
maxamp	floating point number - maximum amplitude in dB
minfreq	floating point number - minimum frequency
maxfreq	floating point number - maximum frequency
bandswitch	string / menu to specify whether to keep or reject the bands
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

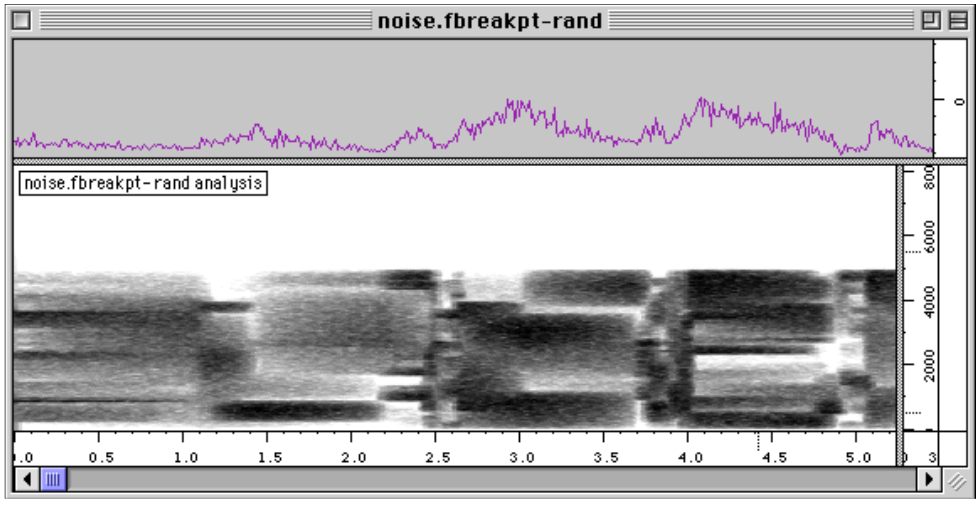
text file

Description

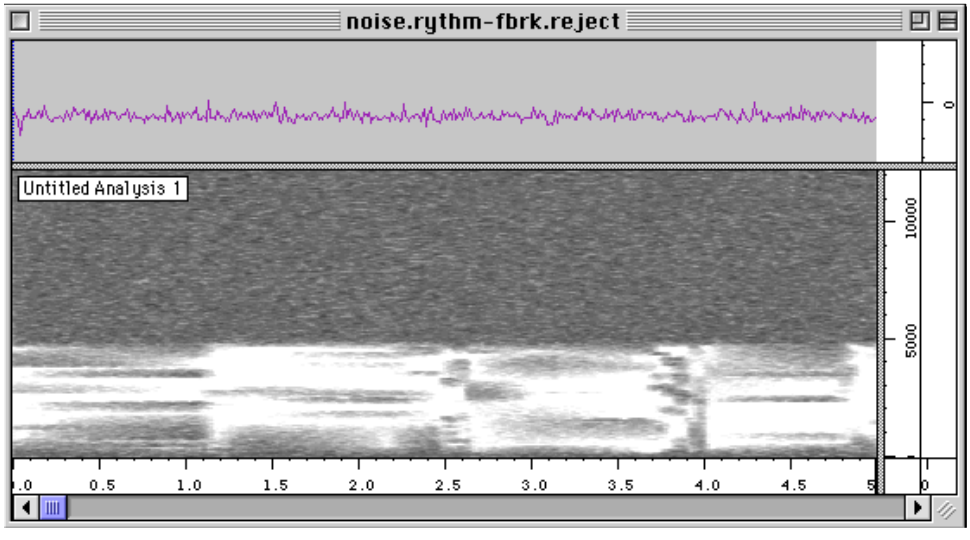
Calculates a parameter file for doing a break point filter within a frequency-range.
At every step a certain number (between min. and max. freq-points) of freq-points is chosen and for each point a random amplitude (between min and max. amp) is calculated.

Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -Fbreakpt fbreakpt.par  -M4000 -N4096 mysound.fbrkpt
```



noise filtered (option: keep bands)

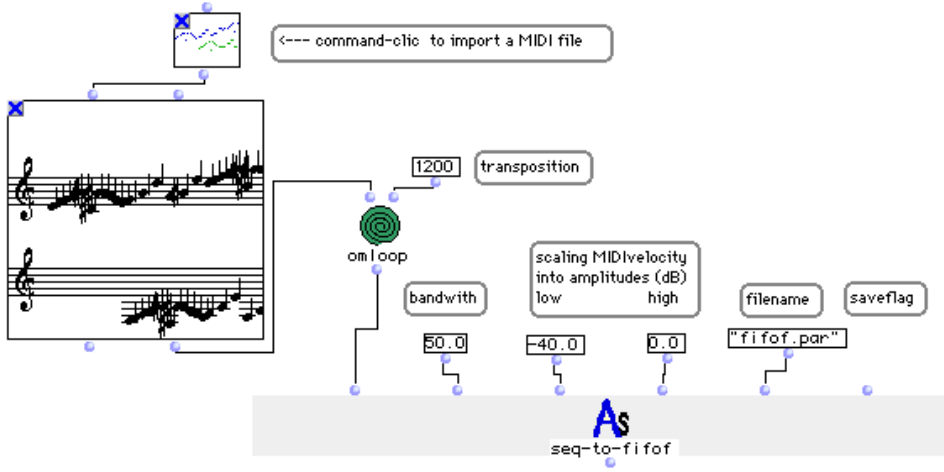


noise filtered (option: reject bands)

Section 5 - formantfilter

This section is made up of a group of functions for doing a formant filter "fifof" and "fof" in AudioSculpt .

5.1 seq-to-fifof



Syntax

(AS:: seq-to-fifof *list bandwidth amplow amphigh filename saveflag*)

Inputs

list	list of triplets (Midicent onsettime duration)
bandwidth	whole ore floating point number
amplow	floating point number - minimum amplitude in dB for scaling of MIDI-velocity
amphigh	floating point number - maximum amplitude in dB for scaling of MIDI-velocity
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

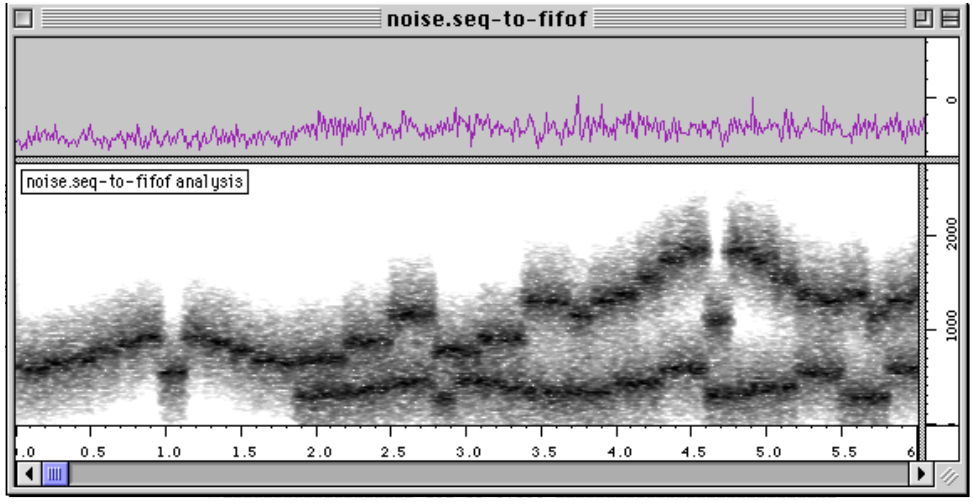
text file

Description

Converts a note-sequence into a filter file for formant filtering.

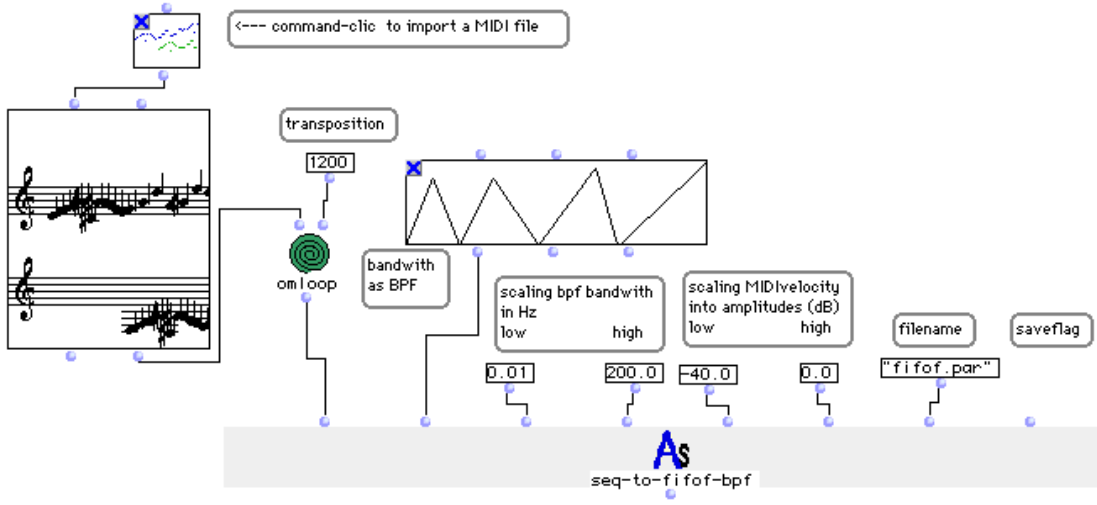
Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -Ffifof fifof.par  -M4000 -N4096 mysound.fifof
```



noise filtered by a polyphonic midifile

5.2 seq-to-fifof-bpf



Syntax

(AS:: seq-to-fifof-bpf *list bandwidth amplow amphigh filename save-flag*)

Inputs

list	list of triplets (Midicent onsettime duration)
bandwidth	bpf-function
amplow	floating point number - minimum amplitude in dB for scaling of MIDI-velocity
amphigh	floating point number - maximum amplitude in dB for scaling of MIDI-velocity
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

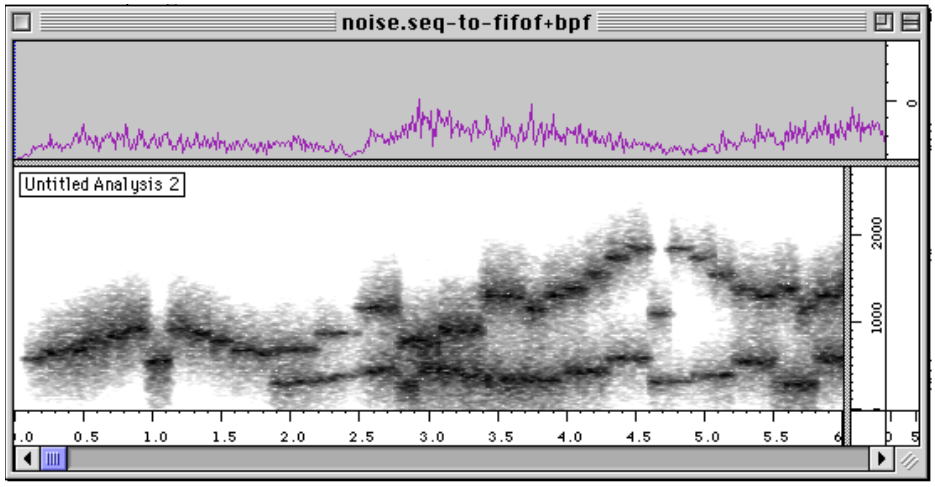
text file

Description

Converts a note-sequence into a filter file for formant filtering. In this version the bandwidth can be drawn in a BPF.

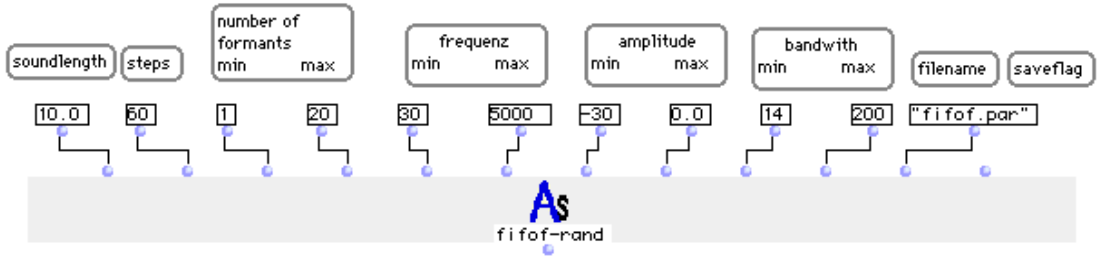
Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -Ffifof fifof.par  -M4000 -N4096 mysound.fifof
```



noise filtered by a polyphonic midifile (changement of bandwith drawn in BPF)

5.3 fifof-rand



Syntax

(AS:: fifof-rand *soundlength steps minpoints maxpoints minfreq maxfreq minamp maxamp minbw maxbw filename saveflag*)

Inputs

soundlength	floating point number to specify the length of the original sound in seconds
steps	whole number for random steps
minpoints	whole number - minimum number of frequency-points at one step
maxpoints	whole number - maximum number of frequency-points at one step
minfreq	floating point number - minimum frequency
maxfreq	floating point number - maximum frequency
minamp	floating point number - minimum amplitude in dB
maxamp	floating point number - maximum amplitude in dB
minbw	floating point number - minimum bandwidth in Hz
maxbw	floating point number - minimum bandwidth in Hz
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

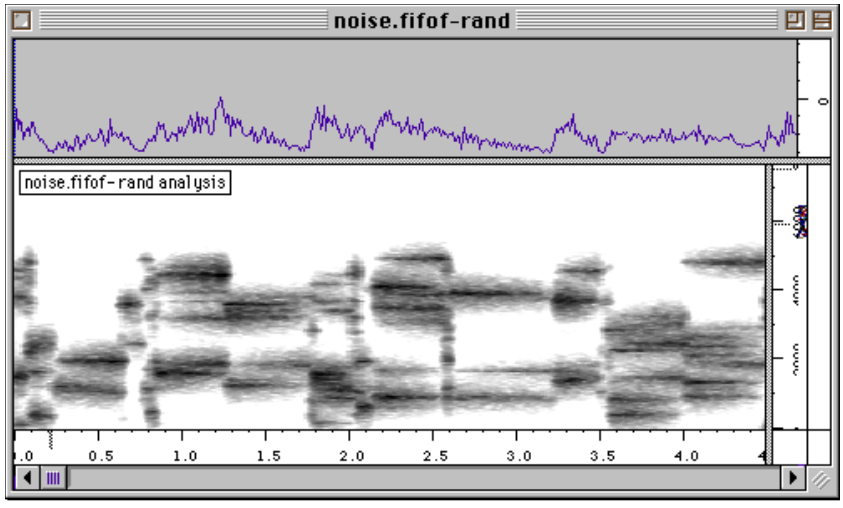
text file

Description

Calculates a parameter file for doing a formant filter within a frequency-range.
At every step a certain number (between min and max. freq-points) of formants are chosen and for each formant a random amplitude (between min and max. amp) and a random bandwidth (between min and max. bw) is calculated.

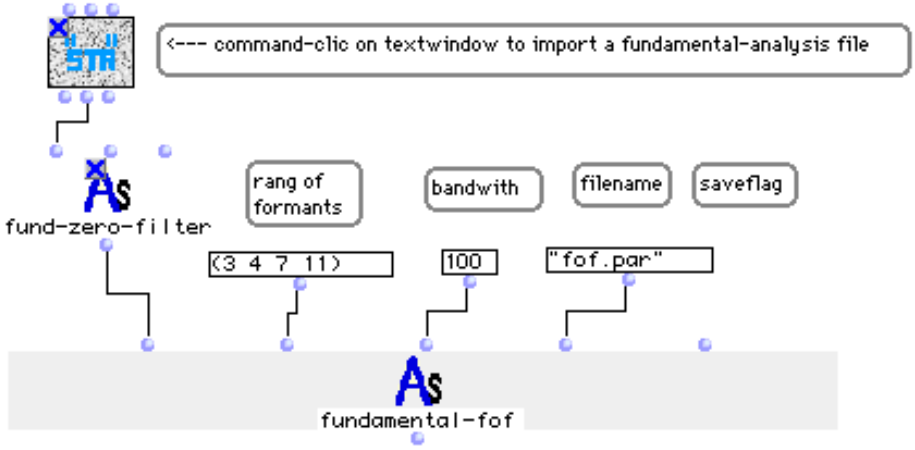
Example for the SVP-command line

```
svp -t -v -Z -A -Smysound  -Ffifof fifof.par  -M4000 -N4096 mysound.fifof
```



noise filtered

5.4 fundamental-fof



Syntax

(AS::fundamental-fof *fundamental rang bw filename saveflag*)

Inputs

fundamental	list in form of text file „fundamental analysis“ from AudioSculpt
rang	list - which frequency-bands as multiples of fundamental frequency
bandwidth	floating point number - bandwidth in Hz
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

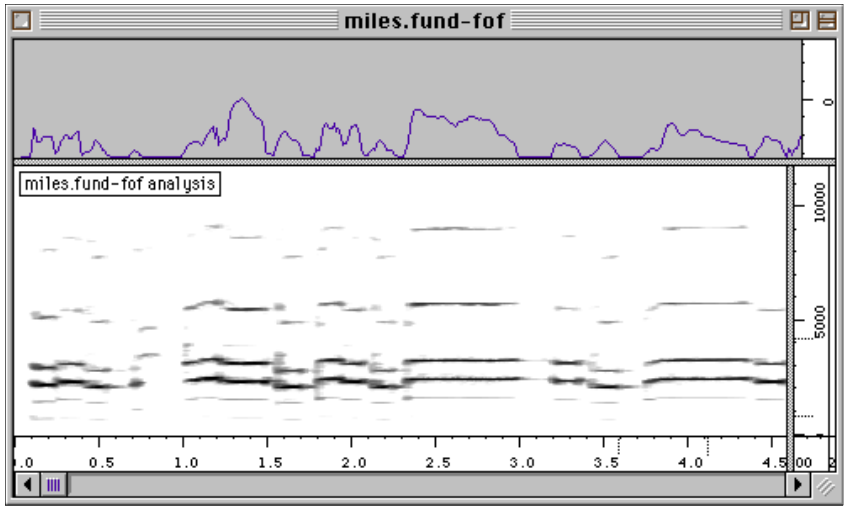
text file

Description

Createsa parameter file for the AudioSculpt fof-filter.
The frequencies come from a fundamental analysis. <rang> specifies which harmonic multiples of the fundamental are calculated.
"fund-zero-filter" eliminates errors in the fundamental analysis file.

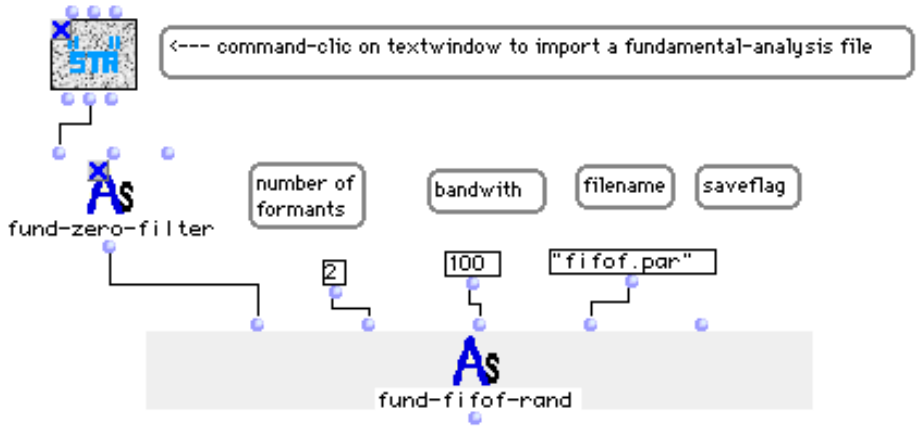
Example for the SVP-command line

```
svp -t -v -Z -A -Snoise  -Ffof fof.par  -M4000 -N4096 noise.fof
```



<rang> specifies, on which harmonics the formants will be placed

5.5 fund-fifof-rand



Syntax

(AS::fund-fifof-rand *fundamental number bw filename saveflag*)

Inputs

fundamental	list in form of text file „fundamental analysis“ from AudioSculpt
number	integer - number of formants
bandwidth	floating point number - bandwidth in Hz
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

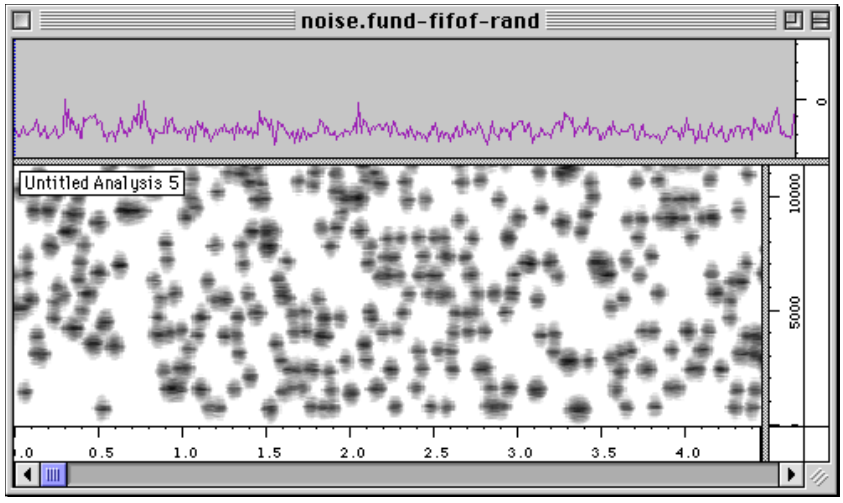
text file

Description

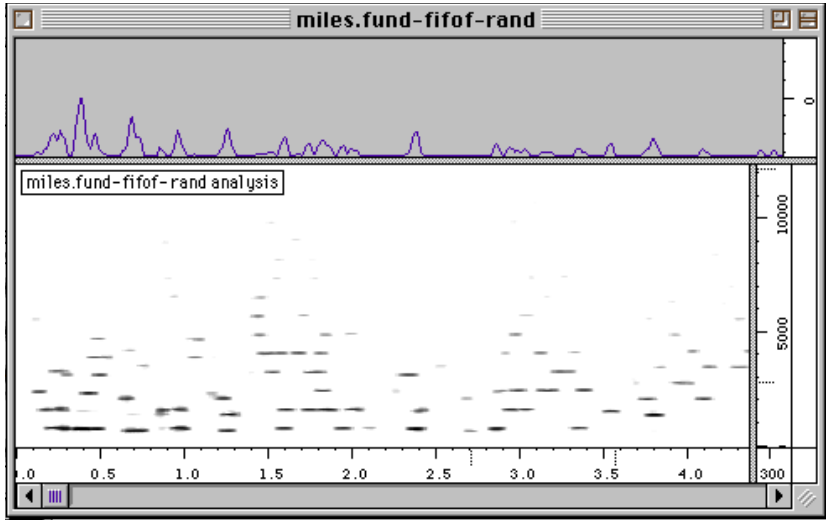
This patch creates a parameter file for an AudioSculpt fifof-filter.
The frequencies come from a fundamental analysis. Randomly, a specified number of formants on the harmonic multiples on the fundamental are calculated.
"fund-zero-filter" is eliminating errors in the fundamental analysis file.

Example for the SVP-command line

```
svp -t -v -Z -A -Snoise  -Ffifof fifof.par  -M4000 -N4096 noise.fof
```



noise filtered with random formants, based on a fundamental analysis

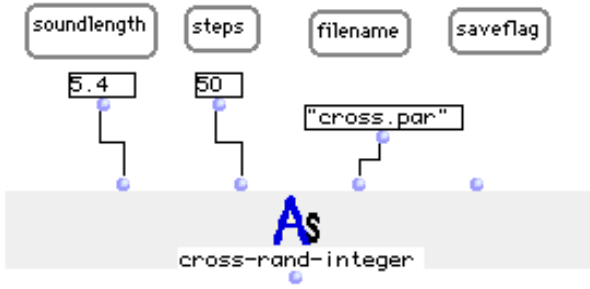


original sound filtered with random formants, based on a fundamental analysis

Section 6 - cross

This section is made up of a group of functions for generalised cross-synthesis in AudioSculpt .

6.1 cross-rand-integer



Syntax

(AS:: cross-rand-integer *soundlength steps filename saveflag*)

Inputs

soundlength	floating point number to specify the length of the original sound in seconds
steps	whole number for random steps
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

text file

Description

Calculates a parameter file for general cross synthesis which does a “random walk” between both sounds. The values for X x Y y are chosen randomly. This version gives just ore 0 or 1 (no intermediate values). The factor q is always 0.
As just integer values for X x Y y are allowed, you get combinations as
<amplitudes of first sound with frequencies of first sound>
<amplitudes of first sound with frequencies of second sound>
<amplitudes of second sound with frequencies of second sound>

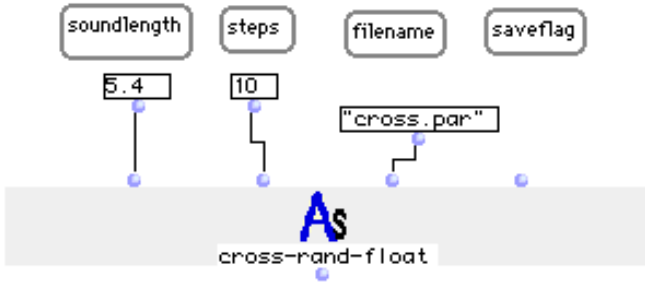
<amplitudes of second sound with frequencies of first sound>

time	X	x	Y	y	q
0.0	0	1	1	0	0
0.011	1	0	0	1	0
4.879	0	1	0	1	0
4.893	1	0	0	1	0
5.4	1	0	1	0	0

Example for the SVP-command line

```
svp -v -t -a -A -Z -Sfirstsound -ssecondsound -Gcross cross.par -Jhanning
    -N4096 -M4096 -Whanning -whanning -m4096 -n4096  cross.result
```

6.2 cross-rand-float



Syntax

(AS:: cross-rand-float *soundlength steps filename saveflag*)

Inputs

soundlength	floating point number to specify the length of the original sound in seconds
steps	whole number for random steps
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

text file

Description

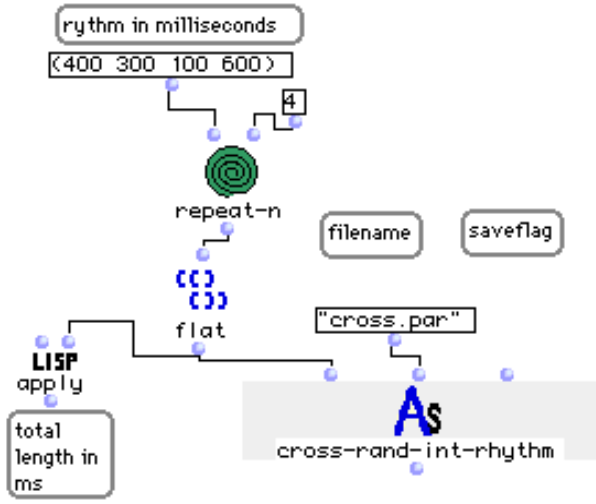
Calculates a parameter file for general cross synthesis which does a “random walk” between both sounds. The values for X x Y y are chosen randomly.
The factor q is always 0.

time	X	x	Y	y	q
0.0	0.1	0.9	0.3	0.7	0.0
0.003	0.0	1.0	0.4	0.6	0.0
3.452	0.6	0.4	0.1	0.9	0.0
3.992	0.7	0.3	0.8	0.2	0.0
4.4	0.8	0.2	0.6	0.4	0.0
4.889	0.6	0.4	0.8	0.2	0.0
4.99	0.3	0.7	0.0	1.0	0.0
5.365	0.6	0.4	0.6	0.4	0.0
5.371	0.5	0.5	0.3	0.7	0.0
5.4	0.6	0.4	0.8	0.2	0.0

Example for the SVP-command line

```
svp -v -t -a -A -Z -Sfirstsound -ssecondsound -Gcross cross.par -Jhanning
    -N4096 -M4096 -Whanning -whanning -m4096 -n4096  cross.result
```

6.3 cross-rand-int-rhythm



Syntax

(AS:: cross-rand-int-rhythm *LDUR filename saveflag*)

Inputs

LDUR	list of duration's in milliseconds
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

text file

Description

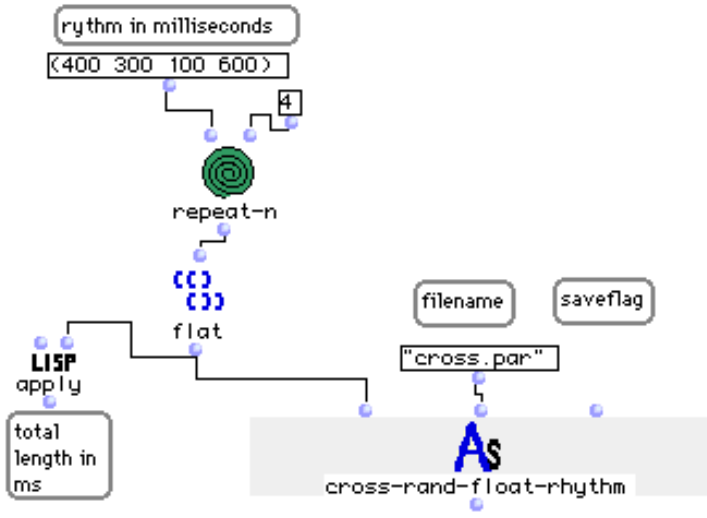
Calculates a parameter file for general cross synthesis which does a “random walk” between both sounds. The rhythm is used to calculate time points for new combinations of X x Y y. The values for X x Y y are chosen randomly on each onset of a new duration. This version gives just ore 0 or 1 (no intermediate values). The factor q is always 0.

time	X	x	Y	y	q
0.0	0	1	0	1	0
0.39	0	1	0	1	0
0.4	1	0	1	0	0
0.69	1	0	1	0	0
0.7	0	1	1	0	0
0.79	0	1	1	0	0

Example for the SVP-command line

```
svp -v -t -a -A -Z -Sfirstsound -ssecondsound -Gcross cross.par -Jhanning
    -N4096 -M4096 -Whanning -whanning -m4096 -n4096  cross.result
```

6.4 cross-rand-float-rhythm



Syntax

(AS:: cross-rand-float-rhythm *LDUR filename saveflag*)

Inputs

LDUR	list of duration's in milliseconds
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

text file

Description

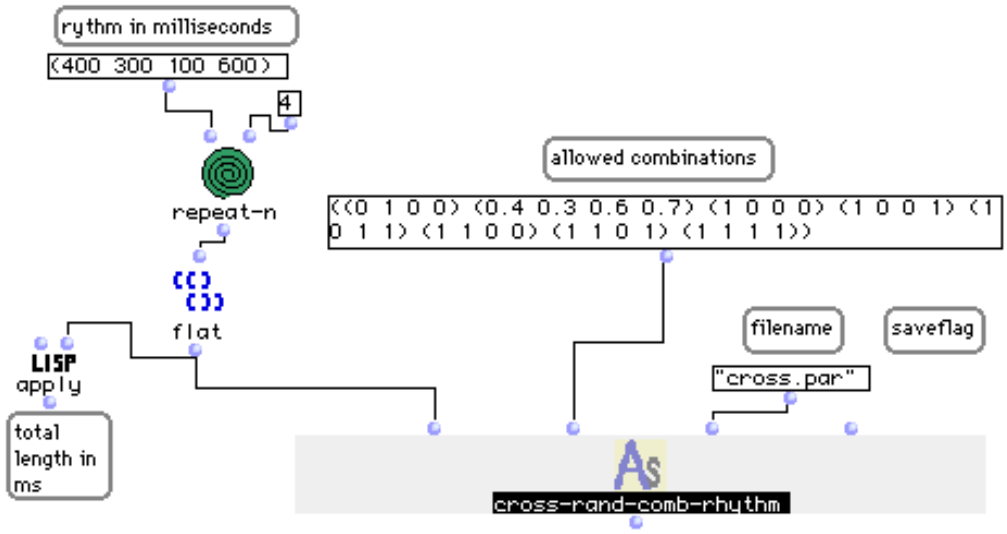
Calculates a parameter file for general cross synthesis which does a “random walk” between both sounds. The rhythm is used to calculate time points for new combinations of X x Y y. The values for X x Y y are chosen randomly on each onset of a new duration.
The factor q is always 0.

time	X	x	Y	y	q
0.0	0.1	0.1	0.9	0.6	0.0
0.39	0.1	0.1	0.9	0.6	0.0
0.4	0.2	0.2	0.3	0.2	0.0
0.69	0.2	0.2	0.3	0.2	0.0
0.7	0.4	0.1	0.2	0.8	0.0
0.79	0.4	0.1	0.2	0.8	0.0
0.8	0.8	1.0	0.6	0.8	0.0
1.39	0.8	1.0	0.6	0.8	0.0

Example for the SVP-command line

```
svp -v -t -a -A -Z -Sfirstsound -ssecondsound -Gcross cross.par -Jhanning
    -N4096 -M4096 -Whanning -whanning -m4096 -n4096  cross.result
```

6.5 cross-rand-combinations



Syntax

(AS:: cross-rand-combinations *soundlength steps combinations filename saveflag*)

Inputs

soundlength	floating point number to specify the length of the original sound in seconds
steps	whole number for random steps
combinations	list of allowed combinations for X x Y y
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

text file

Description

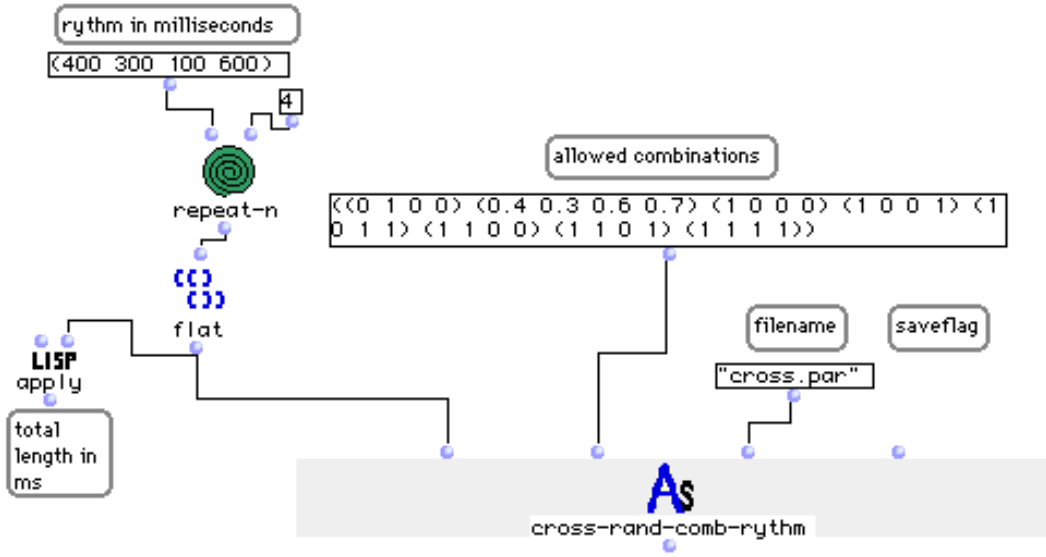
Calculates a parameter file for general cross synthesis. The function chooses randomly between combinations given by the user.

```
time        X         x     Y       y       q
0.0         0         1     0       0       0
0.131       1         1     0       0       0
0.132       1         1     0       1       0
0.133       0.5       0.2   0.5     0.8     0
0.135       0         1     0       0       0
0.136       1         1     1       1       0
0.691       0.5       0.2   0.5     0.8     0
0.699       1         0     1       1       0
```

Example for the SVP-command line

```
svp -v -t -a -A -Z -Sfirstsound -ssecondsound -Gcross cross.par -Jhanning
    -N4096 -M4096 -Whanning -whanning -m4096 -n4096  cross.result
```

6.6 cross-rand-comb-rhythm



Syntax

(AS:: cross-rand-comb-rhythm *LDUR combinations filename saveflag*)

Inputs

LDUR	list of duration's in milliseconds
combinations	list of allowed combinations for X x Y y
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

text file

Description

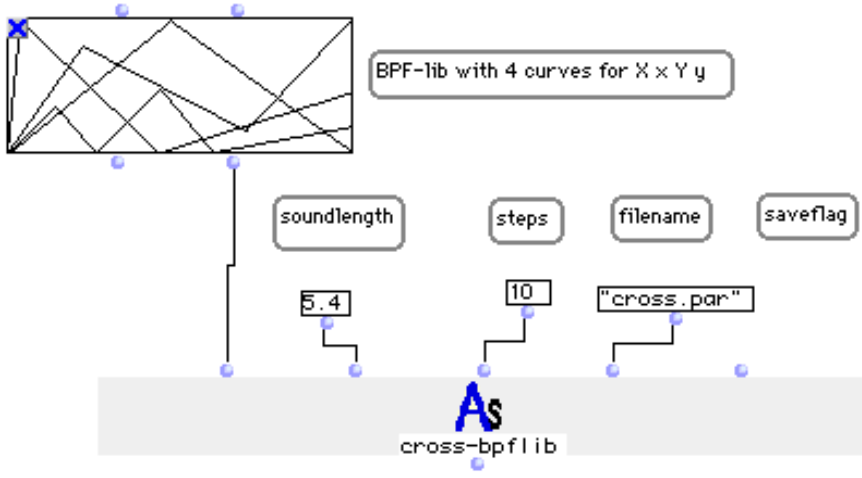
Calculates a parameter file for general cross synthesis which does a “random walk” between both sounds. The rhythm is used to calculate timepoints for new values of X x Y y. The values for X x Y y are chosen randomly from a list, given by the user on each onset of a new duration. The factor q is always 0.

```
time        X         x     Y       y       q
0.0         0         1     0       0       0
0.39        0         1     0       0       0
0.4         0.4       0.3   0.6     0.7     0
0.79        0.4       0.3   0.6     0.7     0
0.8         0         1     1       1       0
1.79        0         1     1       1       0
1.8         0.4       0.3   0.6     0.7     0
2.79        0.4       0.3   0.6     0.7     0
```

Example for the SVP-command line

```
svp -v -t -a -A -Z -Sfirstsound -ssecondsound -Gcross cross.par -Jhanning
    -N4096 -M4096 -Whanning -whanning -m4096 -n4096  cross.result
```

6.7 cross-bpflib



Syntax

(AS:: cross-bpflib *tab soundlength steps combinations filename saveflag*)

Inputs

tab	list of four BPF-functions (right outlet of BPF-LIB)
soundlength	floating point number to specify the length of the original sound in seconds
steps	whole number for random steps
combinations	list of allowed combinations for X x Y y
filename	string
saveflag	string / menu to specify whether to save the result in a text file or to print it in the listener

Output

text file

Description

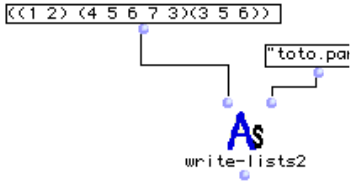
takes four bpf-curves to calculate a parameter file for generalised cross synthesis the order for the 4 bpfs is as follows:
bpf0 - amplitude first sound (X)
bpf1 - amplitude second sound (x)
bpf2 - frequency first sound (Y)
bpf3 - frequency second sound (y)
Example for the SVP-command line

```
svp -v -t -a -A -Z -Sfirstsound -ssecondsound -Gcross cross.par -Jhanning
    -N4096 -M4096 -Whanning -whanning -m4096 -n4096  cross.result
```

time	X	x	Y	y	q
0.0	0	1	0	0	0
0.39	0	1	0	0	0
0.4	0.4	0.3	0.6	0.7	0
0.79	0.4	0.3	0.6	0.7	0
0.8	0	1	1	1	0
1.79	0	1	1	1	0
1.8	0.4	0.3	0.6	0.7	0
2.79	0.4	0.3	0.6	0.7	0

Section 7 - util

7.1 write-lists2



Syntax

(AS:: write-lists2 *data filename*)

Inputs

data	list of lists
filename	string

Output

text file

Description

writes a text file to disc where every sublist becomes a new line (without the parenthesis)

Example :

```
((1 2) (4 5 6 7 3)(3 5 6))
```

becomes a text file with the following format:

```
1 2
4 5 6 7 3
3 5 6
```

Index

A

AudioSculpt 1

B

bandfilter 4, 21
break point filter 32

C

Carlos Agon Amado 1
commandlines 3
cross-bpflib 67
cross-rand-combinations 63
cross-rand-comb-rhythm 65
cross-rand-float 58
cross-rand-float-rhythm 61
cross-rand-integer 56
cross-rand-int-rhythm 59
cross-synthesis 56

E

error-message 5

F

fbande 21
fbande-melody 30
fbande-not-parallel 28
fbande-parallel 26
fbreakpt 32
fbreakpt-rand 44
FFT-size 2, 4
fifof-rand 50
flags 4
formantfilter 4, 46
fundamental-fof 52
fund-fbande 21
fund-fifof-rand 54
fund-zero-filter 25

G

generalised cross-synthesis 56
Gerard Assayag 1

I

inputsound 4

M

marker-stretch 11
Mikhail Malt 1

P

parameter-files 3
partials-amp-filter 42
partials-fbreakpt 35
partials-fbreakpt-bpf 40

R

rand-trans-gliss 13
rand-trans-steps 15
rhythm-fbreakpt 33

S

seq-to-fifof 46
seq-to-fifof-bpf 48
Set Default Folders 4
stretch-dyn-exact 9
stretch-dyn-random 7
SVP 3

T

text files 3
time stretch 4, 6
trans-melody 17
transposition 4, 13

V

vibrato 19

W

Windowsize 4
write-lists2 69