CDP Pitch Extraction and Handling

(with Command Line Usage)

Functions to REPITCH spectral pitch data

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

The REPITCH Group consists mostly of operations on pitch data files – binary (**.frq**) or breakpoint – extracted from spectral analysis files (**.ana**) by **GETPITCH**. To get back to sound from a binary pitch data file, it must be combined with a formant file (**.for**), extracted via **FORMANTS GET**, using the functions **COMBINE MAKE** or **MAKE2**.

Note also two key functions in the group which process analysis files (**.ana**): **TRANSPOSE** and **TRANSPOSEF**. These aim to transpose pitch without altering time.

ANALENV
Extract the window-loudness envelope of an analysis file

APPROX
Make an approximate copy of a binary pitch data file

[BRKTOPI]
Convert a breakpoint pitch data file to a binary pitch data file

COMBINE
Generate transposition data from 2 sets of pitch data,
or transpose pitch data with transposition data,
or combine 2 sets of transposition data to form new transposition data, producing a binary data file output

COMBINEB
Generate transposition data from 2 sets of pitch data,
OR transpose pitch data with transposition data,
OR combine 2 sets of transposition data to form new transposition data, producing a *time value* breakpoint file output

CUT
Cut out and keep a segment of a binary pitch data file

EXAG
Exaggerate the contour of a pitch data file

FIX
Massage pitch data in a pitch data file

GENERATE
Create a binary pitch data file from a textfile of *time midi* value pairs

GETPITCH
Extract pitch from spectrum to a pitch data file

INSERTSIL
Mark areas as silent in a binary pitch data file

INSERTZEROS
Mark areas as unpitched in a pitch data file

INTERP
Replace noise or silence by pitch interpolated from existing pitches
INVERT
Invert pitch contour of a pitch data file
NOISETOSIL
Replace unpitched windows by silence in a pitch data file
PCHSHIFT
Transpose pitches in a pitch data file by a constant number of semitones
PCHTOTEXT
Convert binary pitch data to text
PITCHTOSIL
Replace pitched windows by silence
QUANTISE
Quantise pitches in a pitch data file
RANDOMISE
Randomise pitch line in a pitch data file
SMOOTH
Smooth pitch contour in a pitch data file
SYNTH
Create the spectrum of a sound following the pitch contour in a pitch data file
TRANSPOSE
Transpose spectrum (spectral envelope also moves)
TRANSPOSEF
Transpose spectrum: but retain original spectral envelope
VIBRATO
Add vibrato to pitch in a pitch data file
VOWELS
Create spectrum of vowel sound(s) following pitch contour in pitch file

Technical Discussion
Extracting and Tracking Spectral Pitch Data

ALSO SEE:
COMBINE MAKE
Generate spectrum from pitch & formant data
COMBINE MAKE2
Generate spectrum from pitch, formant & envelope data only
PTOBRK
Convert pitch trace from binary file (.frq) to text breakpoint file (with zeros) for PSOW

REPITCH ANALENV – Extract the window-loudness envelope of an analysis file

Usage

repitch analenv *inanalysisfile outenvelopefile*

Parameters

inanalysisfile – analysis file from which to extract the window-loudness data
outenvelopefile – the output is a binary envelope file (**.evl**)

Understanding the REPITCH ANALENV Function

This function extracts the loudness envelope of an analysis file. It is part of a suite of processes to extract the pitch, formants or time-loudness envelope of spectral and then cross-combine them. The output is saved as a binary envelope file, for which **.evl** is the standard CDP extension. This file can be used in a number of other CDP processes, as described below.

Both *Soundshaper* and *Sound Loom* produce the **.evl** extension. If you find that *Sound Loom* is not doing this, you can change it in the SYSTEM STATE menu: SYSTEM STATE > OUTPUT FILETYPES & EXTENSIONS > FILENAME EXTENSIONS TO USE.

Musical Applications

The binary envelope file produced by this application can be used with CDP Time Domain processes, notably ENVEL IMPOSE, ENVEL REPLACE and ENVEL RESHAPE. You can use these to transform the (binary) envelope file before using it again.

Having the envelope data as a separate file also allows for more flexibility in the Spectral Domain. With FORMANTS GET, for example, when combining the **pitch contour** from one sound with the **formant data** from a 2nd, the resultant loudness contour will come from the formant envelope, and so will be from the 2nd sound. However, you may want the loudness envelope to be that of the 1st sound. Extracting the loudness envelope here would make that a possibility.

This possibility (and others) is realised by using COMBINE MAKE2. With this program you can generate a spectrum by combining a pitch trace (**.frq**), a formant structure (**.for**) and envelope data (**.evl**), with each extracted from a different sound. The envelope data overwrites the loudness contour in the formant file.

The binary envelope file can be used with or without additional transformations. It also enables you to continue your processing without having to revert back to the original signal.

End of REPITCH ANALENV

REPITCH APPROX – Make an approximate copy of a pitchfile

Usage

repitch approx mode *pitchfile outfile* [**-p***prange*] [**-t***trange*] [**-s***srange*]

Modes

1 Gives a pitchfile as output
2 Gives a transposition file as output

Parameters

pitchfile a binary pitch data file produced by REPITCH GETPITCH
-p*prange* the interval in semitones over which the pitch varies (upwards or downwards) from the original (Range: > 0.0)
-t*trange* time interval in milliseconds by which the pitch value can stray from the original time. It is applied to turning points in the pitch contour. (Range: from the duration of 1 analysis window to the duration of the entire file.
-s*srange* the time interval in milliseconds over which the pitch contour is scanned. A pitch shift by a semitone within *srange* indicates pitch rise or fall. (Range: from the duration of 8 analysis windows to (approximately) the duration of pitchfile.)

prange, *trange* and *srange* may vary over time

Understanding the REPITCH APPROX Function

This produces a deliberately approximate copy of the original pitch file.

Musical Applications

When two people sing 'in unison', they do not sing *exactly* the same pitches, with exactly the same vibrato, exactly synchronously. This function might be used to make such a 'natural' copy of a pitch line.

End of REPITCH APPROX

BRKTOPI – Convert a breakpoint pitch data file to a binary pitch data file

Usage

brktopi brktopi *intextpitchfile outbinarypitchfile*

Parameters

intextpitchfile – breakpoint pitch data file (**.txt**, **.brk** or **.pch**) extracted by REPITCH GETPITCH or created / manipulated as text
outbinarypitchfile – binary pitch data file (**.for**)

Understanding the BRKTOPI Function

BRKTOPI converts a breakpoint pitch data file (text) to a binary pitch data file (**.for**). The function is the opposite of REPITCH PCHTOTEXT.

Musical Applications

While a great variety of manipulation is possible with binary pitch data files (**.for**), there is unlimited scope for altering the pitch data in breakpoint (text) form. As only binary pitch data can be converted into sound (via **MAKE**), this function provides the necessary conversion to the binary format.

You need to be aware, however, that altering pitch data is not altering the formant data with which it will be combined in MAKE. The pitches you specify will sound only to the extent of the level that their equivalent frequencies (fundamental and harmonics) have in the formant file.

End of BRKTOPI

REPITCH COMBINE – Generate transposition data from 2 sets of pitch data, or transpose pitch data with transposition data, or combine 2 sets of transposition data to form new transposition data, producing a binary data file output

Usage

repitch combine 1 *pitchfile pitchfile2 outtransposfile*
repitch combine 2 *pitchfile transposfile outpitchfile*
repitch combine 3 *transposfile transposfile2 outtransposfile*

Modes

1 Generate transposition data from 2 sets of pitch data (**.frq** + **.frq** = **.trn**)
2 Transpose pitch data with transposition data (**.frq** + **.trn** = **.frq**)
3 Combine 2 sets of transposition data to form new transposition data, producing a binary data file as output (**.trn** + **.frq** = **.trn**)

Parameters

pitchfile binary pitch data file (.frq) OR *time pitch (frq-in-Hz)* .brk breakpoint file
transposfile binary transposition file (.trn) OR *time transposition-ratio* breakpoint file (.brk)
outpitchfile binary pitch data file (.frq)
outtransposfile binary transposition file (.trn)

NB: It's IMPOSSIBLE to generate a binary outfile from exclusively breakpoint infiles.

Understanding the REPITCH COMBINE Function

We need to look carefully at each of the modes in order to learn to navigate these functions.

NB: REPITCH COMBINE is the only way to produce a binary pitch transposition file (.trn) with the output of REPITCH PCHSHIFT (which is a .frq file). A .trn file is one of the required inputs for REPITCH TRANSPOSE/F Mode 4.

In **Mode 1** two files of time-varying pitch data are used as inputs. They can both be produced from input analysis files by (separate runs of) REPITCH GETPITCH. Both can be binary pitch data files (.frq), or one can be a *time pitch (frq-in-Hz)* breakpoint file – not both, because this process will not generate (the needed) binary output from breakpoint data only.

> These input pitch data files are used by REPITCH COMBINE in **Mode 1** to produce a new **binary** transposition file (.trn), or, using REPITCH COMBINEB (Text) Mode 1, a *time transposition-ratio* breakpoint file (.brk). This new transposition file contains the **difference** between the two input pitch data files **translated into the transposition ratios** needed to move from the pitch shape of the first file to the pitch shape of the second file. (The order of the two input pitch data files is therefore important!).
>
> You then go to REPITCH TRANSPOSE or REPITCH TRANSPOSEF, using Mode 4 (binary .trn file) to apply this transposition data to the original sound (as an analysis file) and create a new version reshaped with the pitch shape of the second file. REPITCH TRANSPOSE is used to apply this transposition data without preserving formants and REPITCH TRANSPOSEF is used to apply it while preserving formants.

Mode 2 puts together a binary pitch data file (.frq) and a transposition file: a binary .trn file made from two sound sources, or a (hand-written) breakpoint file containing *time transposition-ratio* pairs. The transposition file reshapes the pitch trace of the pitch file. The output produced is a new pitch shape in binary pitch data file format (.frq).

> A binary transposition file (.trn) made in Mode 1 involves two source .frq files and can be used in Mode 2 to reshape the first of these sources with the characteristics of the second. However, it may also be applied to the pitch data (.frq) from any other sound. Thus the difference between the two Mode 1 source files is applied to a new, third file (with results that will be difficult to predict). The COMBINE Mode 2 output is another time varying binary pitch data file (.frq). **NB:** This file **cannot** be used as an input to TRANSPOSE or TRANSPOSEF, which only use transposition files (.trn or .brk) as inputs.
>
> **The primary purpose of Mode 2** is to create binary pitch data (.frq) which can be given to various of the other REPITCH pitch shaping functions in order to further modify the data: see APPROX, CUT, EXAG, FIX, INVERT, QUANTISE, RANDOMISE, SMOOTH and VIBRATO in the listing at the top of this file.

In **Mode 3**, two sets of tranposition data (two binary .trn files – **NB:** this means 4 sources), or one binary .trn file and one *time transposition-ratio* breakpoint file – 3 sources) are the inputs combined to form a new binary .trn transposition file as the output. **This process will sum the transposition data in the two files.** At least one of these input files must be binary, because two input breakpoint files will NOT produce a binary output. The output transposition data can be cycled round again with Mode 2, or can be applied to a sound with TRANSPOSE or TRANSPOSEF, using Mode 4 (with a .trn binary input).

Summary of things to remember:

- COMBINE **Mode 1** inputs can be two binary pitch data files (.frq) from GETPITCH or one binary and one (possibly handwritten) *time frequency-in-Hz* breakpoint file. Both CANNOT be breakpoint. (The binary .frq file also contains *time frequency-in-Hz* data.)

- COMBINE Mode 1 output is a binary transposition file (.trn). COMBINEB (Text) Mode 1 output is a *time transposition-ratio* breakpoint file.

- TRANSPOSE and TRANSPOSEF apply a transposition to a soundfile. As COMBINE only produces a transposition file which is binary(.trn), Mode 4 (binary .trn file) of TRANSPOSE/F must be used when using data made by COMBINE. Mode 1 (ratios) is used when COMBINEB (Text) is used to create *time transposition-ratio* breakpoint output.

- Note that ENVELOPE EXTRACT in the Time Domain (the **ProcessSF** menu) can produce breakpoint output (this will be amplitude envelope data), which can be used as transposition data in TRANSPOSE/F because the envelope data is between 0 and 1. To load the .brk file, tick time-vary, click on edit and then go to File\Load breakpoint file. When you click OK to save, this file will then be present on the TRANSPOSE page.

- COMBINE **Mode 2** inputs use a pitch file (.frq or .brk) and a transposition file (.trn or .brk). Either of these can be binary or breakpoint, but both CANNOT be breakpoint.

- The main purpose of Mode 2 is to provide a binary pitch data file (.frq) for use as input by the other pitch shaping functions, but it can also be recycled in Mode 1 with a different second file.

- COMBINE **Mode 3** sums transposition data (two .trn inputs or one binary .trn and one *time transpose-ratio* .brk). Its binary transposition file output (.trn) can be recycled or used directly with TRANSPOSE or TRANSPOSEF (Mode 4).

- To save confusion, you are recommended to choose binary or breakpoint file types and use them consistently.

Table to summarise the origin & uses of the Pitch & Transposition files:

Pitch File	**Origin**	binary pitch data file (.frq) or *time pitch (frq-in-Hz)* breakpoint file as made by REPITCH GETPITCH
		binary pitch data file (.frq) made by COMBINE Mode 2 from a pitch (.frq) & a binary transposition file (.trn) (either – but not both – of which can be breakpoint)
		time pitch (frq-in-Hz) breakpoint file made by COMBINEB from a pitch (.frq) & a binary transposition file (.trn) (either – but not both – of which can be breakpoint)
		time pitch (frq-in-Hz) breakpoint file hand written with a text editor
	Use	inputs to Modes 1 or 2 of COMBINE or COMBINEB
		(.frq binary data file only) inputs to the other REPITCH pitch processing functions
Transposition File	**Origin**	binary made by COMBINE, Mode 1 from two pitch data files (.frq) (either – but not both – of which can be breakpoint)
		binary made by COMBINE, Mode 3 from two transposition files (.trn) (either – but not both – of which can be breakpoint)
		time transposition-ratio made by COMBINEB, Mode 1 from two pitch data files (.frq) (either – but not both – of which can be breakpoint)
		time transposition-ratio hand-written with a text editor
	Use	inputs to Modes 2 or 3 of COMBINE or COMBINEB
		if breakpoint (.brk), input for Modes 1–3 of TRANSPOSE or TRANSPOSEF
		if binary (.trn), input for Mode 4 of TRANSPOSE or TRANSPOSEF **NB: These require a transposition file proper, not just a binary (or breakpoint) pitch data file.**

Musical Applications

Shape, often visually imagined somehow, is one of the key factors in musical organisation. It can be used in an abstract manner by which it forms the material structurally without drawing too much attention to itself, or it can be used in a very direct, overt manner as a 'gesture'.

Extracting shapes from one sound and imposing them on another sound is one way of altering sounds to get interesting new effects, or of drawing together disparate data. In FORMANTS, this idea is used with spectral envelope data. Here in REPITCH, it is used with pitch data, i.e., the pitch curve a sound makes.

For example, sometimes spoken words can cover a remarkably wide pitch range. This could be used to shape non-pitch material, thus establishing some common ground between the two different kinds of sound. Conversely, a melodic passage could be used to put some 'song' into otherwise somewhat monotonous spoken material.

COMBINE is designed to be flexible. The inputs can mix binary and breakpoint, and the latter make it possible to design arbitrarily simple or complex shape-designs by writing the files by hand. Pitch data can be extracted from existing sounds with REPITCH GETPITCH in a binary or a breakpoint format. REPITCH COMBINEB can be used to produce breakpoint files in each of the three modes, thus making it possible to work and rework breakpoint data before producing the binary transposition files needed by TRANSPOSE and TRANSPOSEF, or the binary pitch data files needed by the other pitch shaping functions.

When planning manoeuvres with pitch data, it might be an idea to sketch out what you want to happen to the data using mind maps, and then write in the names of the functions (and any necessary intermediate steps) that will be needed at each stage.

End of REPITCH COMBINE

REPITCH COMBINEB – Generate transposition data from 2 sets of pitch data, or transpose pitch data with transposition data, or combine 2 sets of transposition data to form new transposition data, producing a *time value* breakpoint file output

Usage

repitch combineb 1 *pitchfile pitchfile2 outtbrkfile* [**-d***I*]
repitch combineb 2 *pitchfile transposfile outpbrkfile* [**-d***I*]
repitch combineb 3 *transposfile transposfile2 outtbrkfile* [**-d***I*]

Modes

1 Generate transposition data from 2 sets of pitch data (**.frq** OR *time frq-in-Hz* **.brk** + **.frq** OR *time frq-in-Hz* **.brk** = *time transposition-ratio* **.brk**)
2 Transpose pitch data with transposition data (**.frq** OR *time frq-in-Hz* **.brk** + **.trn** OR *time transposition-ratio* = *time frq-in-Hz* **.brk**)
3 Combine 2 sets of transposition data to form new transposition data (**.trn** OR *time transposition-ratio* **.brk** + **.trn** OR *time transposition-ratio* **.brk** = *time transposition-ratio* **.brk**)

Parameters

pitchfile – binary pitch data file (.frq) OR *time pitch (frq-in-Hz)* breakpoint file (.brk)
transposfile – binary transposition file (.trn) OR *time transposition-ratio* breakpoint file (.brk)
outpbrkfile – time pitch (frq-in-Hz) breakpoint file
outtransposfile – time transposition-ratio breakpoint file
-d*I* acceptable pitch error in breakpoint file data reduction

Range: *I* > 1.0. Default: eighth_tone = 0.250000

Understanding the REPITCH COMBINEB Function

The inputs to COMBINEB can be the same mix of binary and breakpoint as for COMBINE. However, all the ouputs are breakpoint files, whether of pitch or of transposition data. (Scroll up a bit to see the 'Table summarising the origin & uses of the Pitch & Transposition files').

Musical Applications

See REPITCH COMBINE above.

End of REPITCH COMBINEB

REPITCH CUT – Cut out and keep a segment of a binary pitch data file

Usage

repitch cut 1 *pitchfile outpitchfile starttime*
repitch cut 2 *pitchfile outpitchfile endtime*
repitch cut 3 *pitchfile outpitchfile starttime endtime*

Modes

1 Cut and retain from *starttime* to end of file
2 Cut and retain from start of file to *endtime*
3 Cut and retain portion between *startime* and *endtime*

Parameters

pitchfile the input is a binary pitch data file, as produced by GETPITCH or COMBINE, Mode 2
outpitchfile the output is a binary pitch data file
starttime time in seconds at which to begin the cut if 0, begins at start of the file *endtime* time in seconds at which to end the cut

Understanding the REPITCH CUT Function

This function makes it possible to use part of a pitch data file without having to return to the time domain. PITCHINFO SEE and PITCHINFO HEAR might be helpful in deciding upon the time cut points in the binary pitch data file. Or you could view/listen to the original sound in the time domain (the timings ought to match!).

Musical Applications

The cut segment might be used, for example, as a template for the pitch of another sound, using REPITCH COMBINE to generate the appropriate transposition data (Mode 1 or 2), and REPITCH TRANSPOSE or REPITCH TRANSPOSEF to change the pitch of the other file.

End of REPITCH CUT

REPITCH EXAG – Exaggerate pitch contour

Usage

repitch exag 12 *pitchfile outfile meanpch range*
repitch exag 34 *pitchfile outfile meanpch contour*
repitch exag 56 *pitchfile outfile meanpch range contour*

Modes

1,3,5 Give a pitchfile as output
2,4,6 Give a transposition file as output

Parameters

pitchfile the input is a binary pitch data file, as produced by GETPITCH or COMBINE, Mode 2
outfile the output is a binary pitch data file or a transposition file, depending on the Mode
meanpch the pitch as a MIDI note number around which intervals are stretched
range exaggeration of the pitch range: an interval expressed in semitones and used as a multiplier (Range: > 0)
contour exaggeration of the pitch contour (Range: values between 0 and 1)

meanpitch, *range* and *contour* may all vary over time

Understanding the REPITCH EXAG Function

Mean pitch – which may vary over time – is the pitch around which the stretchings or shrinkings of intervals takes place. With the *contour* exaggeration, the program measures how far (above or below) each pitch is from the specified mean, then pushes it up or down, depending on the value set for *contour*. This either stretches the intervals close to the mean by a lot, and those far from the mean by a little, or *vice versa*.

With the *interval* exaggeration, the program finds the interval between the mean and a note and multiplies it by the *range* variable. For example, a value of 0.5 will compress the intervals.

Musical Applications

Some form of alteration of the spectrum will result, but it is virtually impossible to tell what, as it is so highly dependent on the spectral content of the *infile*.

End of REPITCH EXAG

REPITCH FIX – Massage pitch data in a binary pitchfile

Usage

repitch fix *pitchfile outpitchfile* **-r***T1* **-x***T2* **-l***bf* **-h***tf* **-s***N* **-b***f1* **-e***f2* **-w -i**

Parameters

pitchfile the input is a binary pitch data file, as produced by GETPITCH or COMBINE, Mode 2

outpitchfile the output is a binary pitch data file

CUT
-r*t1* Starttime: remove pitch from time *t1* (Default: 0.0)(
-x*t2* Endtime: end pitch removal at time *t2* (Default: end of file)

FILTER
-l*bf* Bottom frequency: remove pitch below frequency *bf*
-h*tf* Top frequency: remove pitch above frequency *tf*

SMOOTH
-s*N* Smooth count: smooth onset errors & glitches in pitchdata, *N* times
-w removes 2-window glitches (Default: 1-window) (Use ONLY with the **-s** paramter)

MARK
-b*f1* Start frequency: force start frequency to be *f1*
-e*f2* End frequency: force end frequency to be *f2*

INTERPOLATE
-i interpolate through ALL non-pitch windows in the pitch data, producing pitch everywhere

Rules:
1. AT LEAST ONE flag must be used.
2. When pitches are removed, they are replaced by a 'no-pitch' indicator. These (and any already existing unpitched windows in the file) can be converted to pitch data (by interpolation between adjacent pitches) using the **-i** flag.
3. With multiple flags, the ORDER of operations within the program is:

    1. Remove time-wise (**-r/-x**)
    2. Remove frequency-wise (**-l/-h**)
    3. Smooth (**-s/-w**)
    4. Set end values (**-b/-e**)
    5. Interpolate (**-i**)

Understanding the REPITCH FIX Function

Formerly SPECPMOD, this function attempts to improve on (or smooth) the pitch contour produced by REPITCH GETPITCH. It is easiest to use if the pitch contour is first converted to a pseudo-soundfile (with **PITCHINFO SEE**) and viewed at the sample level with VIEWSF or another sound editor capable of zooming to sample accuracy. This should reveal the presence of any unwanted glitches.

Musical Applications

If the pitch output from REPITCH GETPITCH sounds awry, you can look at it with **PITCHINFO SEE**) to see what kind of glitches there might be. Then use the appropriate REPITCH FIX option to eliminate or reduce the glitches.

End of REPITCH FIX

REPITCH GENERATE – Create binary pitchdata from a textfile of *time midi* value pairs

Usage

repitch generate *outpitchdatafile midipitch-data srate*

Parameters

outpitchdatafile – binary pitchdata file produced by the program (these files have a **.frq** extension)
midipitch-data – a text file containing a list of paired *time* and *midi-note-values*:

- *times* must start at zero and increase
- *midi-note-values* may be:
  - numeric MIDI note values (MPV); e.g., Middle-C = 60. See the **Chart of Equivalent Pitch Notations** for a full listing.
  - OR: note names (A,B,C,D,E,F or G)
  - possibly followed by a '#' for a sharp or a 'b' for a flat
  - and (required) followed by an octave number between -5 and 5, where 0 corresponds to the octave starting at Middle C

Example file:

```
[time MPV]
0.0     60
1.0     Eb0
2.4     F#0
3.7     67.25
4.0     C1
```

srate – the sample rate of the soundfile that might later be generated from the binary pitch data, such as 44100.

Understanding the REPITCH GENERATE Function

REPITCH GENERATE allows you to generate a binary pitchdata file **.frq** by directly typing in **MIDI** or musical **note name** information. Note that this is the same type of file as that produced by **REPITCH GETPITCH**. The difference is that with REPITCH GENERATE you are designing your own pitch trace by specifying the (changing) pitch levels using musician-friendly data entry, and with REPITCH GETPITCH the pitch levels are extracted automatically from the input analysis file.

The advantages offered by REPITCH GENERATE are:

1. an easy, musical, form of data entry
2. the ability to use the resultant **.frq** file in combination with other files to produce complex transposition relationships. REPITCH COMBINEB can produce text breakpoint file output, but neither it nor REPITCH COMBINE accept a text breakpoing file as an input.

3. the ability to use your own **.frq** pitch data file crafted with the help of REPITCH GENERATE as an input to COMBINE MAKE2. With this program, you can create new sounds by using pitch **.frq**, formant **.for** and envelope data **.evl** from various sources.

See the **Transposition & Shifting Inputs and Outputs** Chart for a summary of the relevant PITCH and REPITCH operations.

Musical Applications

Two programs in the COMBINE group enable you to create a spectrum from inputs derived from different sources. MAKE accepts a binary pitch data file **.frq** and a formant file **.for**. You can make the binary pitch data file (**.frq**) with REPITCH GETPITCH, which does an automatic pitch-trace, or you can design it yourself in a musician-friendly way with REPITCH GENERATE. COMBINE MAKE outputs an analysis file **.ana**.

COMBINE MAKE2 is another variant of this, **also accepting envelope loudness data** in binary form (**.evl**). There are various ways to create an **.evl** file, including ANALENV in the REPITCH Group. It also produces an analysis file **.ana**.

Another option is simply to impose a specific sequence of pitches on a spectrum. You could do that by combining auto-pitch extraction with your own hand-crafted pitch contour:

1. extract the pitch of the analysis-file source (REPITCH GETPITCH, producing a binary pitch data file **.frq**),
2. combine (REPITCH COMBINE, Mode **1**) this with your new pitch data file (a **.frq** made with REPITCH GENERATE) to generate a transposition file **.trn** (**.frq** + **.frq** = **.trn**)
3. and then apply the transposition data (**.trn**) to the original analysis-file (REPITCH TRANSPOSE, Mode **4**).

Note that the **.trn** file is only produced by REPITCH COMBINE or COMBINEB. It is the required (binary) input for REPITCH TRANSPOSE. The binary pitch data file **.frq** is not a transposition file and cannot be used as an input to REPITCH TRANSPOSE.

Because of the complexity of these various program combinations, it is recommended that you prepare some **batch files** (**Instruments** in *Sound Loom*) for your favourite operations.

End of REPITCH GENERATE

REPITCH GETPITCH - Attempt to extract pitch from spectral data

Usage

repitch getpitch 1 *infile outfile pfil* [**-t***R*] [**-g***M*] [**-s***S*] [**-n***H*] [**-l***L*] [**-h***T*] [**-a**] [**-z**]
repitch getpitch 2 *infile outfile bfil* [**-t***R*] [**-g***M*] [**-s***S*] [**-n***H*] [**-l***L*] [**-h***T*] [**-d***i*] [**-a**]

Parameters

outfile (dummy) analysis file (.ana): when resynthesized, produces a tone at the detected (possibly varying) pitch
pfil binary output file (.frq) containing pitch information
bfil breakpoint (text) output file (.brk) of pitch information (as *time Hz* pairs)
Either of these may be reused in other pitch-manipulating options.
-t*R R* = Tuning range(semitones) within which harmonics are accepted as in tune (Default 1)
-g*M M* = minimum number of adjacent windows that must be pitched, for a pitch-value to be registered (Default 2)
-s*S S* = signal to noise ratio, in decibels (Default 80dB)
Windows which are more than *S*dB below maximum level in sound, are assumed to be noise, and any detected pitch value is assumed spurious.
-n*H H* = how many of the 8 loudest peaks in spectrum must be harmonics to confirm sound is pitched (Default 5)
-l*L L* = frequency of LOWEST acceptable pitch (Default minimum: 0Hz)
-h*T T* = frequency of TOPMOST acceptable pitch (Default maximum: Nyquist/8)
-d*i i* = acceptable pitch-ratio error in data reduction (semitones) (Default 1/8 tone = 0.250000)
-a Alternative pitch-finding algorithm (avoid N < 2)
-z Retain unpitched windows (set them to -1)
Default: Set pitch by interpolation between adjacent pitched windows

Understanding the REPITCH GETPITCH Function

This is the key program in the REPITCH Group. It extracts a pitch trace from analysis data, producing a binary pitch data file (**.frq**) in Mode **1**, which can then be massaged by the other REPITCH functions. A number of these functions handle noise or silent data in the the **.frq** file. An overview of these is given in the **Technical Discussion** section. The other key program is COMBINE/B, which can combine binary pitch data files (i.e., pitch traces) to produce transposition data etc.

You can resynthesise the output analysis file produced during the pitch extraction process in order to audition the results. This is a simple sine-wave sound based on the pitch extraction. See the **Technical Discussion** for an overview.

Handling noise elements is important. A complex sound with a varying degree of noise elements will need to be handled carefully to enable the process to find the pitched material. Pay special attention to the parameters relating to noise.

You can:

a. use **-z** to flag the retention of unpitched windows if you're going to use **SPEC BARE** because it helps SPEC BARE to identify non-harmonic windows; unpitched windows are marked with a '-1' to indicate that they have zero pitch. Otherwise, they are replaced with interpolated pitch values.

b. use **-g** to discard a specified range of windows to be treated as noise

c. use **-s** (signal to noise ratio) to set a dB level below which the signal is to be regarded as noise

On the other hand, notice how you can specify how many of the eight loudest partials must be harmonics. Here the source will be a clearly pitched tone, so you're expecting harmonics to be present, and you try to judge just how many the process should find at any given point in order to verify that a pitch is present. That is, when the criterion is not met, too much of a noise element may be present. In this way, the pitched material may be separated out from the noise more effectively. But if you overspecify for the source sound, you may throw away too much; it depends how clear it is in the first place.

When generating a binary pitch data file for use by another process, you can use the **-z** parameter to mark unpitched windows for retention. These are flagged in the data file (with a -1), and this information can be used by a subsequent process, for example, to help it ignore unpitched material. **SPEC BARE** is the primary (and possibly only) application for this.
NB: At the time of writing, it is my understanding (Ed.) that ***no REPITCH process will work if the binary pitch data file has been created with the flag to retain unpitched windows set***.

Musical Applications

The many musical applications are indicated by the following list of processes which use the pitch data file produced by PITCH as their input:

SPEC BARE (isolate the harmonic partials)
REPITCH APPROX (approximate copy of the file)
REPITCH COMBINE (generate pitch transposition data, producing a binary output)
REPITCH COMBINEB (generate pitch transposition data, producing a breakpoint output)
REPITCH CUT (cut out and retain part of the file)
REPITCH FIX ('correct' pitch data)
REPITCH QUANTISE (snap to a pitch grid)
REPITCH RANDOMISE (randomise pitch line)
PITCHINFO SEE (view data in the file)
REPITCH SMOOTH (smooth pitch contour)
REPITCH TRANSPOSE (transpose pitch values)
REPITCH VIBRATO (add vibrato)
PITCHINFO WRITE (convert a binary pitch data file to a *time frequency* breakpoint file; this breakpoint file can then be used as an input for any time-varying process)
PITCHINFO ZEROS (show if uninterpolated zeros are present)

End of REPITCH GETPITCH

REPITCH INSERTSIL – Mark areas as silent in a pitchdata file

Usage

repitch insertsil mode *inpitchdatafile outpitchdatafile silence-data*

Modes

1 the data is given as a list of times
2 the data is given as (grouped) sample counts: count samples in mono, pairs in stereo, etc.

Parameters

inpitchdatafile – input binary pitchdata file (**.frq**)
outpitchdatafile – output binary pitchdata file produced (**.frq**) by the program
silence-data – text file (**.txt**) specifying where to mask the unsatisfactory analysis data. See examples below.

Understanding the REPITCH INSERTSIL Function

This function enables you to replace questionable pitch information with silence.

When the binary pitch data file (the 'pitch trace') is created with REPITCH GETPITCH, the software is examining the analysis bins for definable pitch content. This is not infrequently rather ambiguous, leading to spurious artefacts in the output, such as noise components. By default, REPITCH GETPITCH smooths over these places with an interpolation function, but you can set an option retain the unpitched windows (**-z** flag), which makes it possible for REPITCH INSERTSIL to distinguish between pitch and noise. It then goes through the file looking for clearly recognisable pitch data, retains this and replaces the 'questionable' material with silence.

Suppose you had extracted the pitch trace (**.frq**) from some complex soundstream (A) containing pitch, noise and undecidable material, such as a sequence of avant-garde sounds from a string or brass instrument with noisy ormultiphonic effects. Then you also make a formant analysis (**.for**) – this could in fact be from another source (B). If you use COMBINE MAKE to combine the pitch (**.frq**) and formant data (**.for**) to make a new sound via the new analysis file produced, you will probably discover some rather unpleasant artefacts where the original signal (A) did not produce reliable pitch data. This applies even if the input is a perfectly acceptable harmonic sound but uses **more than one** pitch.

You can use this program to mask out the bits that are a bit dubious, i.e., you don't get the pitch or pitchiness you expect, in order to achieve a file of (say) pure pitch and silence. You can then recombine the pitch and formant data and the silent sections in the pitch file will be silent in the output. Thus only sound for which you have reliable data will be resynthesized in the output.

Musical Applications

REPITCH INSERTSIL therefore enables you to focus in on the most salient pitch features of a sound, discarding the rest. It is part of a group of programs designed especially for processing spoken material.

This process was used by Trevor Wishart, for example, in *The Division of Labour*. There he extracted the pitch of the speaker and used it to synthesise a chord that glides around, following the pitch of the original voice. This is a typical example of using a pitch trace as a contour with which to shape other material. But in this case, he also wanted to have silent sections within this movement. He was able to do this by adding silence to the pitch trace (binary pitch data file), with INSERTSIL.

To end up with a sound that contains real silence when shaping it with the contours of a pitch data file, it is not enough to use REPITCH COMBINE with the REPITCH INSERTSIL **.frq** output. Rather, a pitchfile containing silence will only lead to an output **sound** containing silence if you use it **directly** for synthesising that output, i.e., with REPITCH SYNTH.

It is useful to understand why this is so.

- Combining two pitch data files to make a transposition file (REPITCH COMBINE Mode **1**) must generate meaningful a transposition value **everywhere** in the file, or the transposition process will fail.
- A transposition file (**.trn**) does *transposition*, not changes of level.
- Thus, when combining two pitch data files, at times where **either** of the pitch input files has pitch-zeros or silence indicated, a transposition value of 1.0 is generated (i.e., no transposing takes place at those times).
- If you start off with one pitchfile, then insert silence to create a second, then combine the two to make a transposition file (REPITCH COMBINE: **.frq** + **.frq** = **.trn**), the places where you inserted the silence will output transposition values of 1.0: it *doesn't* create silence (nor does it transpose).

Creating the required *silence-data* file might be a bit haphazard unless you make use of the **Pitch Editor** in *Sound Loom*. This facility displays binary pitch data files, with both time and frequency labelling. If you have extracted the pitch trace with interpolation turned off ('retain the unpitched windows'), you will see these portions of the pitch trace marked out in grey.

Control-Alt-MouseClicks will then give you precise location data for the edges of these grey sections, so that you can find the precise time, or window or sample where they appear, for use in preparing your *silence-data* file. The **Pitch Editor** also enables you to smooth and transpose selected portions of the pitch trace. If you then display the **.frq** file with the silence inserted, you will see it marked out in red.

ALSO SEE: INSERTZEROS, **INTERP**, **NOISETOSIL** and **PITCHTOSIL**.

End of REPITCH INSERTSIL

REPITCH INSERTZEROS – Mark areas as unpitched in a pitchdata file

Usage

repitch insertzeros mode *inpitchdatafile outpitchdatafile zeros-data*

Modes

1 the data is given as a list of times
2 the data is given as (grouped) sample counts: count samples in mono, pairs in stereo, etc.

Parameters

inpitchdatafile – input binary pitchdata file
outpitchdatafile – output binary pitchdata file produced by the program
zeros-data – a list of pairs of times between which unpitched data is to be inserted. See examples below.

Understanding the REPITCH INSERTZEROS Function

REPITCH INSERTZEROS indicates where **no pitch** is detected in the binary pitch data file. No pitch means that there is actual noise or signal too complex to extract the pitch successfully.

Note that this process is different from **INSERTSIL** which indicates where there is **no signal** in the file.

Musical Applications

Pitch data can be extracted with various caveats. For example, what should happen at places where there is no sound, or no pitched sound (just noise)? On comparing the pitch output with the original, you may decide that a segment of pitch extracted by the process is so vaguely defined in the original that it would be better to treat it as unpitched. With this process you can do this.

End of REPITCH INSERTZEROS

REPITCH INTERP – Replace noise or silence by pitch interpolated from existing pitches

Usage

repitch interp *inpitchdatafile outpitchdatafile*

Modes

1 Glides from a previous valid pitch to the next valid pitch.
2 Sustains the previous valid pitch until the next valid pitch appears.

Parameters

inpitchdatafile – input binary pitchdata file (.frq)
outpitchdatafile – output binary pitchdata file (.frq) produced by the program

Understanding the REPITCH INTERP Function

When you use **REPITCH GETPITCH** to extract the pitch data from a source with interpolation turned off (option to 'Retain unpitched windows' / 'Keep pitch zeros'), the resulting binary pitch data file (**.frq**) will keep a record of any unpitched (noise) or silent sections.

- **unpitched** means noise or waveforms too complex to deduce pitch successfully
- **silent** means no significant signal is detected

If you decide not to keep this information (i.e. you prefer the pitch data to be continuously pitched, for some musical purpose, rather than to contain gaps) REPITCH INTERP will cause the pitch data to interpolate across the unpitched gaps.

Using one or the other Mode, you can do this in two ways:

- either you create pitch glissandi across the unpitched areas, moving ('gliding') from the pitch before the noise or silence to the pitch after it, or
- you sustain the pitch before the noise or silence until you reach the pitch after it.

There are also facilities within the *Sound Loom* **Pitch Editor** to smooth across areas of noise or silence manually.

Musical Applications

This is particularly important when working with two different files in which continuity of pitch is important. For example, **REPITCH COMBINE, Mode 1** enables you to combine two different pitch traces, creating a transposition file (**.trn**) that 'contains the difference between the two input pitch data files translated into the

transposition ratios needed **to move from the pitch shape of the first file to the pitch shape of the second file**'. In other words, you are imposing the pitch contour of file 2 onto file 1. If, for example, you have recorded two instruments one or both of which is playing staccato, there will be silent gaps in your pitch files. Without interpolation, these gaps will make it impossible for the pitch traces to interact properly, unless they were perfectly in sync.

ALSO SEE: REPITCH INSERTSIL for a discussion of how to end up with an output sound that *does* contain silence.

End of REPITCH INTERP

REPITCH INVERT – Invert pitch contour of a pitch data file

Usage

repitch invert mode *pitchfile outfile map* [**-m***meanpitch*] [**-b***bot*] [**-t***top*]

Modes

1 Gives a pitchfile as output
2 Gives a transposition file as output

Parameters

pitchfile the input is a binary pitch data file
outfile the output may be a binary pitch data file or a (binary) transposition file, depending on which Mode is set
map set *map* to 0 if no mapping is required. Otherwise, *map* is a text file of paired values showing how intervals expressed in (possibly fractional) semitones are to be mapped onto their inversions. (Range: -96 to 96)
-m*meanpitch* pitch expressed as a MIDI note value around which the pitch line is to be inverted (Range: MIDI equivalents of 9Hz to the Nyquist frequency (sample rate divided by 2))
-b*bot* bottom pitch (MIDI note value) permissible (Default: 0; Range: MIDI equivalents of 9Hz to Nyquist)
-t*top* top pitch (MIDI note value) permissible (Default: 127; Range: MIDI equivalents of 9Hz to Nyquist)

meanpitch may vary over time

Understanding the REPITCH INVERT Function

The no mapping option (*map* set to 0) will give a mirror inversion of the original contour, i.e., all intervals will be identical, but in the opposite direction. This is fine for various styles, such as atonal and serial music, but may not be what is wanted in a tonal context. Here the inversion often needs to remain within the same scale or 'key'. The mapping function makes it possible to specify what's needed here.

The Mode 1 output (binary pitch data file) can be fed back into another pitch modification process. The Mode 2 output (binary transposition file) can be given as input to **TRANSPOSE/TRANSPOSEF** Mode 4 to create an analysis file which can be resynthesised: i.e., to complete the process.

Musical Applications

Inversion is a very basic and universal musical technique. It is a way of extending material, introducing something new while retaining a good part of what is already familiar. Here in the spectral domain, there will also be timbral alterations as a result of the inversion.

End of REPITCH INVERT

REPITCH NOISETOSIL – Replace unpitched windows by silence

Usage

repitch noisetosil *inpitchdatafile outpitchdatafile*

Parameters

inpitchdatafile – input binary pitchdata file (.frq)
outpitchdatafile – output binary pitchdata file (.frq) produced by the program

Understanding the REPITCH NOISETOSIL Function

When you've made a pitch-extraction of a real sound, it will have moments of pitch, moments of silence, and moments when it can't decide what the pitch is but knows it does not have silence. It marks these indeterminate areas as 'noise'. They may be noise in the conventional sense, or complex tones where a single pitch cannot be determined.

If your original source contains pitched and unpitched (noise) elements, REPITCH NOISETOSIL allows you to make a 'pitch' file where all the unpitched (noise) data is eliminated.

The binary pitch data file (**.frq**) is made with **REPITCH GETPITCH**. Its default mode sets pitch extraction so that it interpolates the pitch over any unpitched (noise) material. In this way you get a continuously pitched readout from a sound that may not be continuously pitched. Therefore, to make use of REPITCH NOISETOSIL and the related functions in REPITCH (**INSERTSIL** and **INTERP**), you need to set the **-z** flag in REPITCH GETPITCH so that interpolation does NOT take place, and the noise components are retained.

REPITCH NOISETOSIL then replaces these noise components in a **.frq** file with silence, i.e., leaving pauses without breaking the pitch continuity of the sound.

Musical Applications

This contour with pauses where the noise components were is compositionally useful because it allows you transfer e.g., the pitch contour of speech onto another sound (e.g. a flute) without having to worry about the unpitched areas in the speech, and still obtain a continuous line. With NOISETOSIL (replace noise by silence), your flute output will have pauses where noise (e.g. sibilants) occur in the original speech.

This is quite a different result than interpolating in REPITCH GETPITCH when extracting the pitch trace, because then the pitch contour is continuous, without pauses where the noise components occurred.

Again, you may well want to skip these undecidable areas in reconstructing a sound, such as when combining with formants in **COMBINE MAKE** or in **COMBINE MAKE2**. REPITCH NOISETOSIL therefore enables you to mask out the noise. You can reconstruct sounds with the noise left in if you would like to, but REPITCH NOISETOSIL provides an option because at present, reconstruction of signals from noise and format data does not work ideally (especially for speech sibilants).

End of REPITCH NOISETOSIL

REPITCH PCHSHIFT – Transpose pitches in a binary pitch data file by a constant number of semitones

Usage

repitch pchshift *inpitchdatafile outpitchdatafile transposition*

Parameters

pitchfile the input is a binary pitch data file (.frq)
outpitchfile the output is a binary pitch data file (.frq)
transposition amount of (constant) transposition in (fractions of) semitones; negative numbers transpose downwards

To transpose pitch in a time-varying way, see the various PITCH functions, especially REPITCH TRANSPOSE or REPITCH TRANSPOSEF.

Understanding the REPITCH PCHSHIFT Function

This is a straightforward upward or downward shift by a constant amount, in this case applied directly to a binary pitch data file (.frq), as made by **REPITCH GETPITCH**, or even by **REPITCH COMBINE**, Mode 2.

Musical Applications

As part of REPITCH, the intention here is to provide a function to change the pitch of a binary pitch data file prior to further processing with another REPITCH function. The output of REPITCH PCHSHIFT is a binary pitch data file (.frq) with the requested transposition.

This .frq output must then be run through **REPITCH COMBINE** to create a transposition (.trn) file proper. It can be combined with the pitch trace (.frq) from the original analysis file (.ana) used with PCHSHIFT or with a pitch trace from some other analysis file (.ana). Note how a transposed pitch trace from one file can be applied to another one, an example of cross-file processing.

The sound is then made with **REPITCH TRANSPOSE** or **REPITCH TRANSPOSEF**, using the .trn output of REPITCH COMBINE as the input.

Though the use of all these different functions may seem cumbersome, the purpose is to provide opportunities for cross-file processing.

For straightforward transposition in the spectral domain – transposition without change of duration – you can use the other modes of REPITCH TRANSPOSE and REPITCH TRANSPOSEF, which allow the use of both transposition constants and time-varying files.

End of REPITCH PCHSHIFT

REPITCH PCHTOTEXT – Convert binary pitch data (.frq to breakpoint text file (.txt)

Usage

repitch pchtotext *inbinarypitchdatafile outtextpitchdatafile*

Parameters

inbinaryfile – binary pitch data file (**.frq**) generated by REPITCH GETPITCH or REPITCH GENERATE
outtextfile – data in breakpoint text format (**.txt**)

Understanding the REPITCH PCHTOTEXT Function

Converts binary pitch data **.frq** to text data **.txt** so that you can read it. This can be further converted to MIDI data on the *Sound Loom* **Table Editor** (see COLUMNS in *Soundshaper* and on the Command Line).

Note also two other functions which do much the same thing:

- **PITCHINFO CONVERT** and
- **PTOBRK WITHZEROS**, developed for the **PSOW** program group.

Musical Applications

The binary pitch data is encapsulating the pitch contour of a sound. In its binary format it can be manipulated by most of the REPITCH functions, including combining with other binary pitch data files so that the contour of one file is used to reshape that of another.

Once this data is in text format, it can not only be altered 'by hand' but also by the many numerical facilities provided in **COLUMNS**, such as adding a random value between min and max limits to each number in the file (i.e., column of numbers), thus randomising the pitch trace of a sound. The result can be reapplied to the original sound or to another sound. Given the range of operations in COLUMNS, the potential for reshaping pitch contours is enormous.

Related functions

PITCHINFO CONVERT, **PTOBRK**, **BRKTOPI**.

End of REPITCH PCHTOTEXT

REPITCH PITCHTOSIL – Replace pitched windows by silence

Usage

repitch pitchtosil *inpitchdatafile outpitchdatafile*

Parameters

inpitchdatafile – input binary pitchdata file (.frq)
outpitchdatafile – output binary pitchdata file (.frq) produced by the program

Understanding the REPITCH PITCHTOSIL Function

The opposite of **REPITCH NOISETOSIL**, REPITCH PITCHTOSIL may be useful if you want to do something different with the noise segments of the speech – because in this case the pitch components become pauses, leaving only the noise components.

Again, it is important that the binary pitch data file extracted via REPITCH GETPITCH retains the unpitched material so that REPITCH PITCHTOSIL can separate out the pitch from the noise. Note that retaining the unpitched material is the option (which would need to be selected), and interpolating across the unpitched material (keeping the pitch contour going) is the default when running REPITCH GETPITCH.

Musical Applications

The spoken word contains vowels and consonants. The vowels tend to be pitched to some degree, while the consonants contain noise. With REPITCH PITCHTOSIL the vowel content is removed, leaving the consonants.

The application comes when combining this consonants-only pitch data file (the essential components of speech) with a formant file to generate a new, speech-like sound. See COMBINE MAKE and COMBINE MAKE2. REPITCH COMBINE offers the possibility of other types of file combinations to make new spectra.

End of PITCHTOSIL

REPITCH QUANTISE – Quantise pitches in a pitch data file

Usage

repitch quantise mode *pitchfile outfile q_set* **-o**

Modes

1 Gives a pitchfile as output
2 Gives a transposition file as output

Parameters

pitchfile the input is a binary pitch data file
outfile the output may be a binary pitch data file or a (binary) transposition file, depending on which Mode is set
q_set a file of (possibly fractional) MIDI pitch values over which the pitch is to be quantised
-o duplicates *q_set* in all octave transpositions

Understanding the REPITCH QUANTISE Function

Operating on extracted pitch contour data saved as a binary pitch data file, this function 'snaps' the pitches of the contour to the intervallic grid specified in *q_set*. Note that this grid can be defined for a single octave and then duplicated in all the octaves. The pitch or transposition outfile can be used by other REPITCH functions as appropriate. The transposition outfile will be used mostly by REPITCH TRANSPOSE or REPITCH TRANSPOSEF.

If the **-o** flag is to be used, enter the grid as MIDI note values starting at the bottom of the range. This pattern will then be duplicated in all the higher octaves (within the MIDI pitch range).

The *q_set* file is written as a series of MIDI note values, either separated by spaces, or on separate lines.

The Mode 1 output (binary pitch data file) can be fed back into another pitch modification process. The Mode 2 output (binary transposition file) can be given as input to TRANSPOSE/TRANSPOSEF Mode 4 to create an analysis file which can be resynthesised: i.e., to complete the process.

Musical Applications

This process has to be regarded as experimental in nature. In most cases, it will probably be most useful to employ the **-o** flag to make the grid apply to more than one octave region. From my own use of this function so far, the results vary a great deal, generally producing very strange combinations of pitching and glissandi (Ed.).

End of REPITCH QUANTISE

REPITCH RANDOMISE – Randomise pitch line

Usage

repitch randomise mode *pitchfile outfile maxinterval timestep* **-s***slew*

Modes

1 Gives a pitchfile as output
2 Gives a transposition file as output

Parameters

pitchfile the input is a binary pitch data file
outfile the output may be a binary pitch data file or a (binary) transposition file, depending on which Mode is set
maxinterval number of semitones over which the pitches can plus or minus randomly vary (Range: 0 to 96; single value or a *time maxinterval* breakpoint file)
timestep maximum timestep in milliseconds between random pitch fluctuations. The timesteps will be random values less than *timestep*. Range: duration of 1 analysis window to the duration of the entire file.
-s*slew* e.g. 2: the range of the upward variation will be twice that of the downward variation
e.g. -3: the range of the downward variation will be 3 times that of the upward variation.
(Range: > 1 OR < -1)

maxinterval and *timestep* may vary over time

Understanding the REPITCH RANDOMISE Function

The provision of both *maxinterval* and *timestep* parameters provide ways to vary a process which is inherently unpredictable.

The Mode 1 output (binary pitch data file) can be fed back into another pitch modification process. The Mode 2 output (binary transposition file) can be given as input to TRANSPOSE/TRANSPOSEF Mode 4 to create an analysis file which can be resynthesised: i.e., to complete the process.

Musical Applications

With this process you can produce strange and unpredictable pitch movements/distortions. It can only be used on an experimental basis while searching for wierd effects.

End of REPITCH RANDOMISE

REPITCH SMOOTH – Smooth pitch contour in a pitch data file

Usage

repitch smooth mode *pitchfile outfile timeframe* [**-p***meanpitch*] [**-h**]

Modes

1 Gives a pitchfile as output
2 Gives a transposition file as output

Parameters

pitchfile the input is a binary pitch data file
outfile the output may be a binary pitch data file or a (binary) transposition file, depending on which Mode is set
timeframe number of milliseconds over which to interpolate pitch values (Range: duration of 1 analysis window to the duration of the entire file)
-p*meanpitch* interpolate between the **peak** value in each *timeframe* block of pitch values

meanpitch is the pitch from which the peaks are measured (and must be within the pitch range, in every timeframe block).
Peak is the maximum displacement from the mean (up or down)

-h At the end of the file, hold the last interpolated pitch value calculated (Default: interplate from that point to last actual value in input)

timeframe meanpitch may vary over time

Understanding the REPITCH SMOOTH Function

The smoothing operation is applied to the pitch contour. If the time frame is 8 windows, REPITCH SMOOTH takes the average pitch of those 8 windows, then interpolates to the average pitch of the next 8 windows. This is the PITCH equivalent of BLUR BLUR.

With the **-p** flag set, it takes the **peak** value (the highest above the specified mean, or the lowest below the specified mean), rather than the average pitch, in each window, and interpolates as before.

Musical Applications

The averaging/interpolating of pitch data process will produce results which will vary according to the spectral content of the *infile*, so the degree of pitch focus and differentiation in the source will greatly affect the result.

End of REPITCH SMOOTH

REPITCH SYNTH – Create the spectrum of a sound following the pitch contour in the pitch file

Usage

repitch synth *inpitchdatafile outanalysisfile harmonics-data*

Parameters

inpitchdatafile – input binary pitchdata file (.frq)
outanalysisfile – output analysis file (.ana) produced by the program
harmonics-data – a text file that just lists the amplitudes of each harmonic in sequence, from 1 upwards. These amplitude values must lie in the range 0 to 1. For example, the file below gives amplitudes for harmonics 1 to 5 in ascending order:

```
1.0
0.8
0.6
0.4
0.2
```

Understanding the REPITCH SYNTH Process

REPITCH SYNTH synthesises the spectrum of a sound while following the pitch trace in the input pitch data file. Once the pitch contour of a sound (e.g. a stream of speech, with a complex changing pitchline) has been extracted, new sounds can be generated which follow the same pitchline. Furthermore, the pitchline can be modified (inverted, quantised etc.) using the other pitch data file manipulation programs in REPITCH and then the new spectrum added to it.

Musical Applications

The general effect of this function is to create a buzzy tone. Its timbre is affected by the weighting of the amplitudes, i.e., the relative amplitude of the various harmonics. It would be worthwhile to write out the (lower part of) the harmonic series in musical notation as a reference sheet to help select harmonics with more understanding. The function tables given *Csound* provide another point of reference.

It is possible to adjust this further by skipping harmonics. This is done by putting a zero as the amplitude value. The example above used the first 5 harmonics. Suppose we wanted to use 1, 4, 6, 7, 8. Then the *harmonics-data* file would be:

```
1.0
0
0
0.8
0
0.6
0.4
0.2
```

Having extracted the pitch from such natural events as a dog barking, you can therefore listen to it with whatever spectrum you specify by using this process. It takes some experience, however, to get beyond simple buzzy tones.

End of REPITCH SYNTH

REPITCH TRANSPOSE – Transpose spectrum (spectral envelope also moves)

Usage

repitch transpose 1–3 *infile outfile transpos* [**-l***minfrq*] [**-h***maxfrq*] [**-x**]
repitch transpose 4 *infile transpos outfile* [**-l***minfrq*] [**-h***maxfrq*] [**-x**]

Modes

1 Transposition as a frequency ratio
2 Transposition in (fractions of) octaves
3 Transposition in (fractions of) semitones
4 Transposition as a binary data file (**.trn**)

Parameters

infile and *outfile* are analysis files
transpos for modes **1–3** a transposition factor or time varying breakpoint file in a format according with the mode being used, but note that Mode 4 must be used if the transposition data is in a binary format (.trn). The binary or breakpoint input files can be made by REPITCH COMBINE or REPITCH COMBINEB, and a breakpoint file can also be created 'by hand' in a text editor.
-l*minfrq* – minimum frequency, below which data is filtered out
-h*maxfrq* – maximum frequency, above which data is filtered out
-x – fuller spectrum

Frequency ratio, octave or semitone transpositions may vary over time.

Understanding the REPITCH TRANSPOSE Process

REPITCH TRANSPOSE does fixed or time varying transposition in the spectral domain. It accepts a variety of breakpoint file inputs as well as a binary data file. The spectral envelope moves with the transposition (as opposed to TRANSPOSEF, where it does not), meaning that the timbres associated with formants are going to change.

Musical Applications

The process of transposition can be approached in various ways:

- Transpose a file by a constant amount.
- Create a transposition breakpoint file by hand and apply it to any sound.
- Extract a pitch contour with REPITCH GETPITCH, saving as a breakpoint (.brk) file and use this output to combine with another pitch contour or transposition file in COMBINE/B to form another (time varying) transposition file to apply to an *infile*.
- Extract pitch contours from two files with REPITCH GETPITCH. Then create a transposition file with REPITCH COMBINE/COMBINEB Mode 1 which contains the information needed to change the pitch contour of the first file into that of the second. Then apply this transposition data to the first file here with TRANSPOSE in order to effect the transposition.
- Similarly the output of COMBINE/COMBINEB Mode 3 can be used as the transposition input to TRANSPOSE. Summing transposition data can be an interestingly serendipitous (unpredictable) exercise.
- Modify *pitchfile* data in some way, save as a transposition file and use as an input to TRANSPOSE/F. The following REPITCH pitchfile modification functions have a transposition file output mode: APPROX, EXAG, INVERT, QUANTISE, RANDOMISE, SMOOTH and VIBRATO.

The results of these transpositions can be used to alter timbre, especially with TRANSPOSE. Also, some use of transposition could form an important step when realising a morph: to draw one sound closer to that of another in frequency region or timbre, or to create a aurally transitional stage in the morphing process.

End of REPITCH TRANSPOSE

REPITCH TRANSPOSEF – Transpose spectrum: but retain original spectral envelope

Usage

repitch transposef 1–3 *infile outfile* **-f***N* | **-p***N* [**-i**] *transpos* [**-l***minfrq*] [**-h***maxfrq*] [**-x**]
repitch transposef 4 *infile transpos outfile* **-f***N* | **-p***N* [**-i**] [**-l***minfrq*] [**-h***maxfrq*] [**-x**]

Modes

1 Transposition as a frequency ratio
2 Transposition in (fractions of) octaves
3 Transposition in (fractions of) semitones
4 Transposition with a binary data file (.trn) as input

Parameters

infile and *outfile* are analysis files
transpos transposition factor or time varying breakpoint file, with data in a form according with one of the four modes. This can be made by REPITCH COMBINE or REPITCH COMBINEB, but note that Mode 4 must be used if the transposition data is in a binary format (.trn).
-f*N* extract formant envelope linear frequency-wise, using 1 point for every *N* equally spaced frequency channels
-p*N* extract formant envelope linear pitchwise, using *N* equally spaced pitch bands per octave

> There is more (and very important) information on extracting formants in the FORMANTS GET reference section.

-i quicksearch for formants (less accurate)
-l*minfrq* – minimum frequency, below which data is filtered out
-h*maxfrq* – maximum frequency, above which data is filtered out
-x – fuller spectrum

> Frequency ratio, octave or semitone transpositions may vary over time.
> *maxfrq* and *minfrq* may vary over time

Understanding the REPITCH TRANSPOSEF Process

Here, formants are preserved (see TRANSPOSE, where they are not). Thus, for example, the vowel characteristics of the original source should be preserved.

Musical Applications

This process may be used when one wants to retain the original sound as much as possible, though in a different register. On the other hand, it creates an interesting play between the unfamiliar (appearing in a different, and perhaps unnatural, register), and the familiar (original formants are preserved).

The process of transposition can be approached in various ways:

- Transpose a file by a constant amount.
- Create a transposition breakpoint file by hand and apply it to any sound.
- Extract pitch contours from two files with REPITCH GETPITCH. Then create a transposition file with REPITCH COMBINE/COMBINEB Mode 1 which contains the information needed to change the pitch contour of the first file into that of the second. Then apply this transposition data to the first file here with TRANSPOSEF in order to effect the transposition.
- Similarly the output of COMBINE/COMBINEB Mode 3 can be used as the transposition input to TRANSPOSEF. Summing transposition data can be an interestingly serendipitous (unpredictable) exercise.

End of REPITCH TRANSPOSEF

REPITCH VIBRATO – Add vibrato to pitch in a pitch data file

Usage

repitch vibrato mode *pitchfile outfile vibfreq vibrange*

Modes

1 Gives a pitchfile as output
2 Gives a transposition file as output

Parameters

pitchfile the input is a binary pitch data file
outfile the output may be a binary pitch data file or a (binary) transposition file, depending on which Mode is set
vibfreq frequency of vibrato itself in Hz (Range: values > 0)
vibrange maximum interval over which the vibrato moves away from the central pitch (in semitones) (Range: values > 0)

vibfreq and *vibrange* may vary over time

Understanding the REPITCH VIBRATO Function

This operates like a LFO (low frequency oscillator). *Vibfreq* is the speed of oscillation and *vibrange* is the depth (interval swing) of the oscillation.

Musical Applications

Perhaps the following chart maps out the territory. The numbers in parentheses are fairly extreme parameter values, used to illustrate the different types of result.

Table to outline VIBRATO options:

vibfreq	***vibrange***	***(approximate) result***
slow (1)	narrow (1)	gentle undulation
slow (1)	wide (24)	glissando see-sawing
fast (50)	narrow (0.5)	fluttering/shimmering
fast (50)	wide (24)	starts to reduce to burbling

To go directly to making a sound, use Mode 2 to produce a (binary) transposition file; then use this as an input to TRANSPOSE or TRANSPOSEF, resynthesise and play.

End of REPITCH VIBRATO

REPITCH VOWELS – Create spectrum of vowel sound(s) following pitch contour in pitch file

Usage

repitch vowels *inpitchdatafile outanalysisfile vowel-data halfwidth curve pk_range fweight foffset*

Parameters

inpitchdatafile – input binary pitchdata file (.frq)
outanalysisfile – output analysis file, ready to play or resynthesise
vowel-data – text file containg a vowel, OR a file of paired *time vowel* values, where vowel is one of the following strings:

Chart of Vowel Symbols ('V' = Vowel)

V	as in	V	as in	V	as in	V	as in
ee	*heat*	**i**	*hit*	**ai**	*hate*	**aii**	Scottish *e*
e	*pet*	**a**	*pat*	**ar**	*heart*	**o**	*hot*
or	*taught*	**u**	*hood*	**uu**	Scottish *you*	**ui**	Scottish *could*
x	neutral V in *herb* or *a*			**xx**	Southern English *hub*		
n	*mean, can, done*						

Vowel breakpoint file

```
[time   vowel]
     0.0     e
     1.0     a
     2.0     i
     3.0     x
     4.0     o
     5.0     u
     6.0     n
```

Simply enter the times and the vowels you would like to use at those times. They remain the same until the next vowel comes in, but there is no interpolation between the vowels as with numerical values.

halfwidth – the half-width of a formant, in Hz, as a fraction of the formant centre-frequency. Range: 0.01 to 10
curve – the steepness of the formant peak. Range: 0.1 to 10.0
pk_range – the ratio of the (maximum) amplitude range of the formant peaks to the (maximum) total range. Range: 0 to 1
fweight – the amplitude weighting of the fundamental. Range: 0 to 1
foffset – the range of scattering of the frequencies of the harmonics from their true value. Range: 0 to 1

Halfwidth, curve, pk_range, fweight and *foffset* may vary over time. In breakpoint files, times must start at zero, and increase.

Understanding the REPITCH VOWELS Process

This process allows a sequence of vowels to be synthesized over a pitch-contour extracted from a source sound.

Musical Applications

You can combine pitch data (**.frq**) from one speaker with formant data (**.for**) from another (COMBINE MAKE). In contrast, REPITCH VOWELS allows you to define the vowel (formant) data from scratch, by writing out the vowels in a breakpoint file, before combining them with the binary pitch data.

The overall effect is the semblance of speech added to a sound.

End of REPITCH VOWELS

Technical Discussion About Extracting and Tracking Spectral Pitch Data

The main program in the REPITCH Group is **GETPITCH**, which seeks to extract a pitch trace from an analysis file. Pitch tracking in the spectral domain is concerned with identifying the frequency content ('partials') in all the various analysis windows. This changes constantly throughout the course of the sound, more with some sounds than with others.

Identifying the frequency content on a channel-to-channel (frequency band) and window-to-window is a difficult operation, particularly when there is a good deal of inharmonic/noise content. This is why **REPITCH GETPITCH** produces a sine-wave based output file: when resynthesised, you can listen to it to see if sensible results have been achieved. Saving as a text breakpoint file makes it possible to hand-edit any anomalies which you would like to remove.

Release 5.0 introduced a number of programs to handle the pitch tracking data in the binary pitch data file (**.frq**) produced by REPITCH GETPITCH.

- First of all, REPITCH GETPITCH enables you to extract the pitch while interpolating pitch through any noise or silence anomalies. **This is the default.**

- REPITCH GETPITCH also enables you to preserve all details of the extraction process, i.e., where noise or silence as well as pitch is found. 'Noise' is actual noise or places where the waveform is too complex a signal for the extraction process to handle. It is represented as 'pitch zeros'. 'Silence' is where no significant **signal** is detected. **To make use of this option in REPITCH GETPITCH, you must turn off interpolation.** This is done by ticking the **-z** flag: 'retain unpitched windows'.
  (In *Sound Loom*, this appears as a tick-box below parameter (8): 'Keep Pitch Zeros'. In *Soundshaper*, it appears as a tick-box labelled 'retain unpitched windows'.)

- **REPITCH INSERTSIL** puts an indicator where **no signal** appears in the file. This makes it possible to make use of real silence in the file.

- **REPITCH INSERTZEROS** indicates where **no pitch** is detected. It therefore zeros the noise areas.

- **REPITCH NOISETOSIL** converts noise portions to silence portions of the file. If the source were speech, this would remove most of the consonants, leaving a weaving of (tuneful – somewhat pitched, anyway) vowels.

- **REPITCH PITCHTOSIL** converts areas where pitch has been successfully extracted to silence. This process will retain the noise data, such as the consonants of speech, which can then be used to shape other material.

- **REPITCH INTERP** enables to to interpolate across areas of noise or silence in two different ways: either **glide between** (different) pitches (Mode **1**, or **sustain** the previous pitch (Mode **2**). This provides more flexibility than REPITCH GETPITCH with interpolation on (the default), which always glides.

Transposition can be done in many different ways with the CDP System. Basic time-domain transposition moves the whole sound up and down, either by a constant amount, or in a time-varying way, which can produce glissandos. A similar operation can be achieved in the spectral domain, using ratios or semitones, constant or time-varying. Also, parts of the spectrum can be transposed or shifted inharmonically, tuned to chords, or replaced by the harmonics of specified pitches.

However, another approach involves imposing shapes taken from elsewhere onto a soundfile. This includes extracting an amplitude envelope from one soundfile and imposing it on another soundfile – a time-domain operation. In the spectral domain, one can extract a formant envelope (amplitude shape of the partials on a window-to-window basis) and impose it on another file.

The REPITCH set provides another specialised approach which focuses on the pitch traces extracted from analysis files. These can be used in various combinations to form transposition data which can be applied to the original or another sound. The point here is that one is working with the extracted pitch traces, which related in a detailed way to the *spectral* content of sound. Thus you are more likely to get (often somewhat unpredictable) internal modulations of the sound.

Various functions provide an option to retain or not retain the spectral envelope. This is the time-changing amplitude pattern of the time-changing partial content of the sound. When formants are present in a sound – steady state frequency regions – retaining the spectral envelope will tend to keep more of the original timbral qualities of the sound. The discussion on extracting formants can be a useful point of reference in this area.

End of Technical Discussion

Last Updated 20 March 2021
Documentation: Archer Endrich, revised R. Fraser