Ircam
documentation

- Research reports
- Musical works
- Software

AudioSculpt

Cross-Synthesis Handbook

by Mario Mary

Second edition, April 1996

IRCAM Centre Georges Pompidou

This manual was written by Mario Marcelo Mary under the supervision of Marie-Hélène Serra, and was produced under the editorial responsibility of Marc Battier, Département de la Valorisation, Ircam. English translation by Richard Dudas.

AudioSculpt

Design	Philippe Depalle, Chris Rogers
Program	Chris Rogers
Project supervision	Gerhard Eckel

SVP

Design	Philippe Depalle
Program	Philippe Depalle, Gilles Poirot, Chris Rogers, Jean Carrive

Analysis modules

Algorithms	Xavier Rodet, Philippe Depalle, Guillermo Garcia, Ernst Terhardt, Boris Doval

Documentation

Coordination	Marie-Hélène Serra
Manuals	Marc Battier, Jean Carrive, Peter Hanappe, Mario Mary, Brice Pauset, Marie-Hélène Serra

This documentation corresponds to version 1.0 or higher of AudioSculpt and to version 1.3A of SVP.

Apple Macintosh is a trademark of Apple Computer, Inc.
AudioSculpt is a trademark of Ircam.

First Edition, October 1995

Ircam
1 place Igor-Stravinsky
F-75004 Paris
Tel. 01) 44 78 12 33
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/musinfo

To see the table of contents of this manual, click on the Bookmark Button located in the Viewing section of the Adobe Acrobat Reader toolbar.

Contents

List of examples

Résumé

Ce manuel est une illustration d'une des possibilités les plus intéressantes offerte par le programme AudioSculpt : le synthèse croisée. A travers une série d'exemples sont expliqués différents aspects de ce traitement qui lie intimement deux sons.

Le manuel s'accompagne d'une série de fichiers de son utilisés par les exemples. Ils sont disponibles votre CD-Rom.

Comme ce manuel a une vocation pédagogique, il est conseillé au lecteur d'étudier les exemples l'un après l'autre.

1 General Information

AudioSculpt is an Ircam program which permits the visualization and processing of sounds; it runs on the Macintosh (either 68k or Power PC). Sounds can be visualized in the form of either an amplitude/time envelope or a sonogram. The processing options currently available (time-expansion /compression, transposition, cross-synthesis, etc…) are those of the program SVP, which is based on the principles of a two-channel phase vocoder. SVP was originally developed on the Unix platforms Dec (Mips and Alpha), SGI and NeXT. To run the SVP program in a Unix environment it is necessary to type a command line such as:

svp -v -t -T -A -Z -Scymbale -D2 -M1024 -N1024 cymbale-D2

The various options in the command line specify which treatment to apply to a sound and with which parameters.

In contrast, AudioSculpt was conceived from the beginning as a graphic interface to SVP in order to facilitate the use of the program.

In place of a command line, the user is able to select a processing command from a menu, and fill in the parameters of the various options within a standard Macintosh dialog box. It is always possible, however, to revert to the older method of controlling SVP by selecting **Use Command Line** from the menu. For each type of processing presented in this manual, both forms (dialog box and command line) will be given.

2 Introduction

Cross-synthesis requires the use of two sound files. If the two files are of different lengths, the resulting sound file will have the length of the longer file (before version 1.3 of SVP the shorter file's length was used). Be careful not to confuse the length of the sound file with the duration of the sound actually heard, which may be shorter than the actual size of the file due to silence which could be either at the beginning or at the end of the file.

There are three types of cross-synthesis available within SVP:

- cross-synthesis by mixing (add mode)
- source-filter cross-synthesis (mul mode)
- generalized cross synthesis (cross mode)

Currently, AudioSculpt only gives access to source-filter and generalized cross-synthesis from within the **Processing** menu. Cross-synthesis by mixing must be requested by using a command line.

Explained in the simplest and most general manner, cross-synthesis by mixing (add mode) is based on the mixing of two sounds, source-filter cross-synthesis (mul mode) on the transformation of one sound by another, and generalized cross-synthesis (cross mode) on the exchange of characteristics of and interpolation between two sounds. However, the user may go much deeper into the use of each of these three cross-synthesis modes in order to realize original and creative cross-synthesis by calculating different types of hybrid sounds whose sources may no longer be identifiable.

In order to realize a desired cross-synthesis, there are two basic considerations to take into account:

1. the choice of sounds to cross, and
2. the cross-synthesis mode, and the manner in which to use it.

In this tutorial, the user will find different examples geared to provide a guide for his or her personal work.

Note from the editor: because this handbook is really a tutorial which takes you gradually through all the steps involved in doing cross synthesis, you are advised to read the sections in the order they appear. For instance, explanation on how to adjust analysis parameters in introduced in the 3rd chapter devoted to Source-filter cross-synthesis. We hope that reading the handbook in that manner will help you fully understand the processes in-

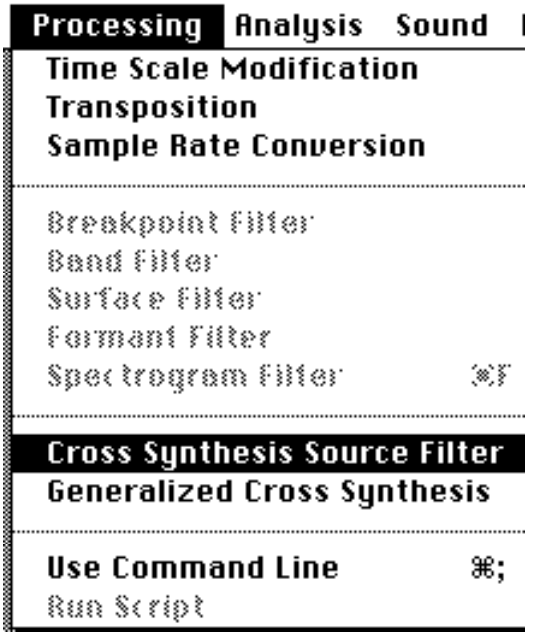
volved in achieving successful cross-synthesis. Do refer to the sound examples that come with this handbook[1]. The recorded examples have been compiled as examples of effects made possible by cross-synthesis with AudioSculpt.

1. The Sound examples are available on a DAT tape or on the Forum CD-ROM.

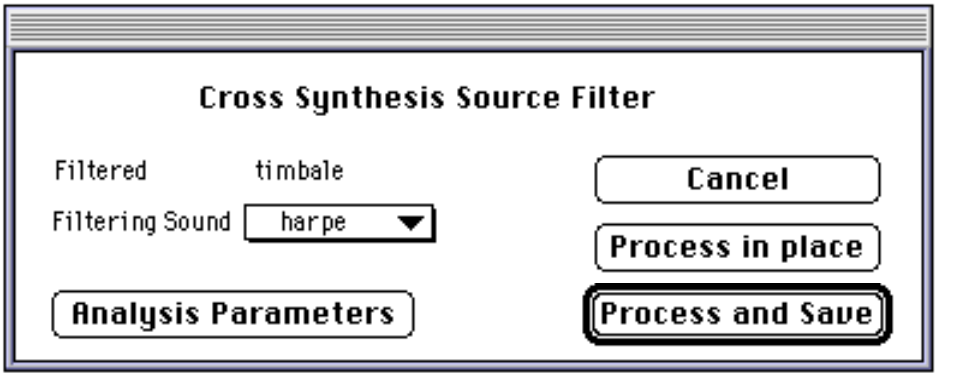
3 Source-Filter Cross-Synthesis

This type of cross-synthesis involves multiplying the short-term spectrum of one sound by the short-term spectral envelope of another sound, which causes a filtering of one sound by the other's spectral envelope. A sound's spectral envelope is calculated by using a linear prediction algorithm (LPC).

To begin processing, first open the sound files which you would like to cross by selecting **Open** from the **File** menu (or by typing command-O). Once the sound files are open, you may select the type of cross-synthesis in the **Processing** menu:

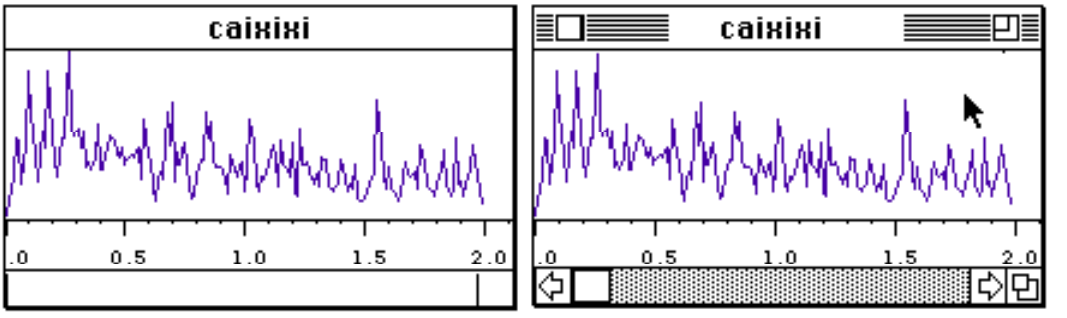


The following dialog box will appear:

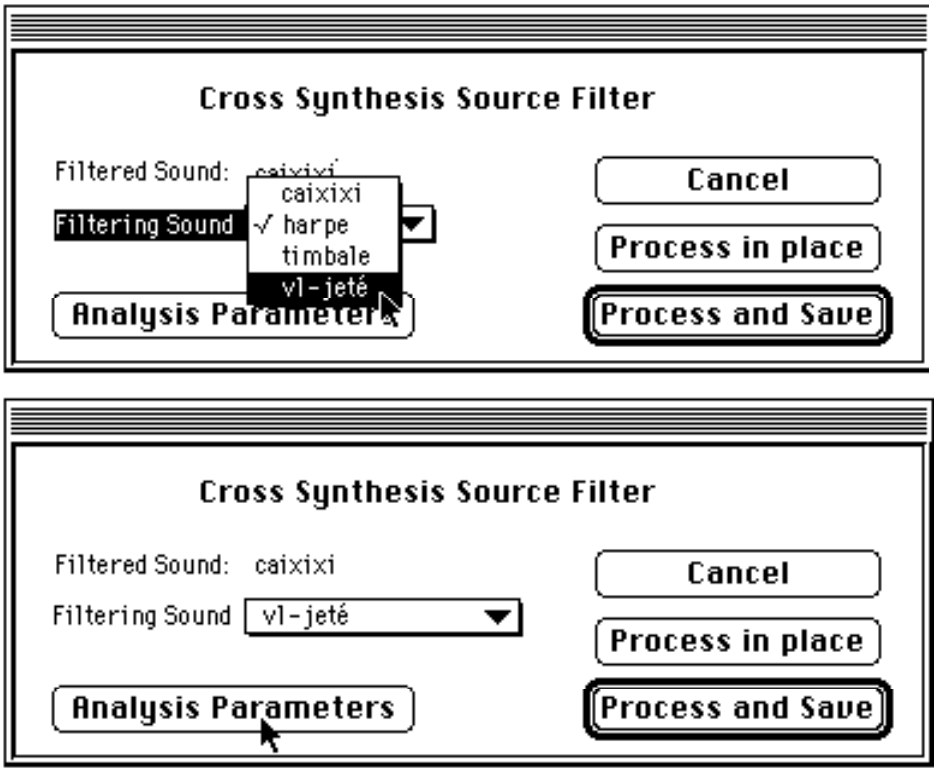


It is in this dialog box that you must specify the file to be filtered and the filtering file. The sound to be filtered is by default the sound file which is in the active window (the window whose borders are greyed). If you would like to select another file to be filtered, you must close the dialog box (**Cancel**), click on the desired file's window (to make it active) and re-select the type of cross synthesis desired.

non-active window active window



In order to select the filtering sound, you may choose between any of the currently open sound files.



Click on **Analysis Parameters** to open the window which allows you to set the spectral analysis parameters: window size, hop size, analysis window type, number of poles determined by LPC, size of the FFT, and synthesis window type.

Adjusting the Analysis Parameters

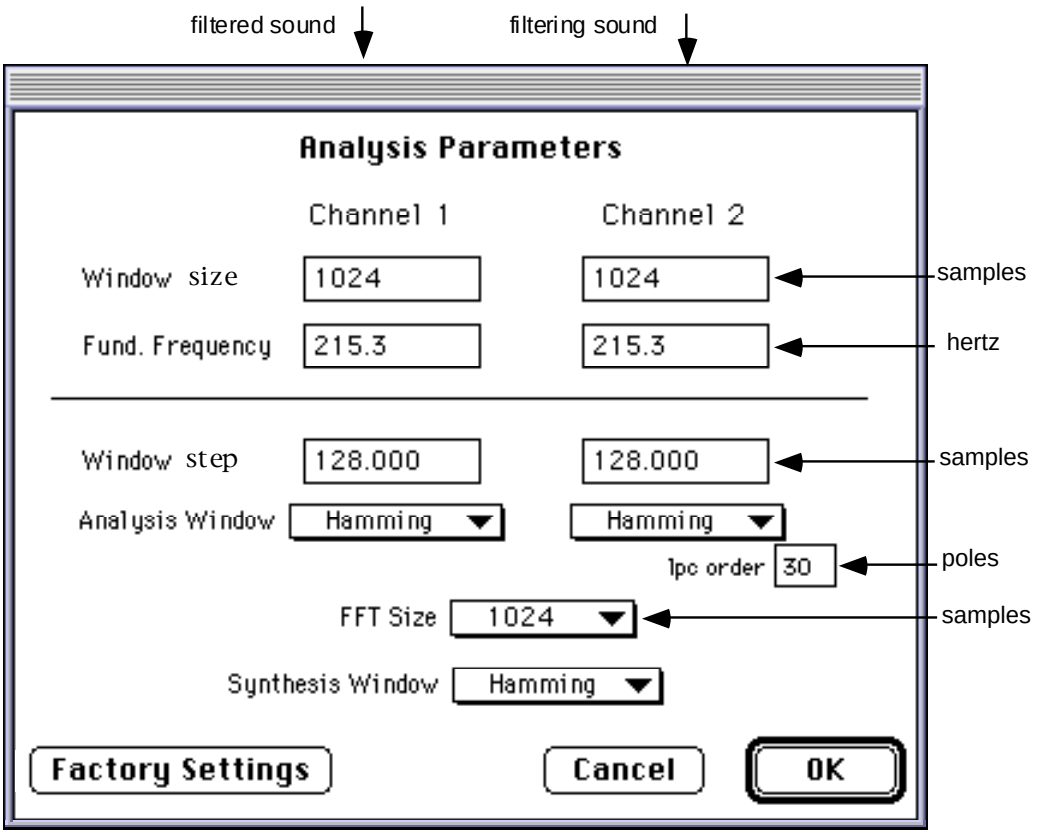
In theory, the analysis window size used on a sound should be equal to at least four times the period of the sound (for periodic sounds), or at least four times the period of the lowest partial contained in the sound (for non-periodic sounds). AudioSculpt automatically calculates the window size according to this rule when you type a frequency value in the **Fund. Frequency** box (see below). As such, the frequency you give should correspond to either the fundamental frequency of the sound (for periodic sounds) or the lowest frequency present in the sound (for non-periodic sounds). Generally, since the sounds to be crossed will more than likely not have the same characteristics, a different window size will be automatically choosen by the program for each sound file. The FFT size must be the same for both sounds since the two spectra calculated by the Fourier transform must have the same number of points. By default, AudioSculpt sets the FFT size to the first power of 2 greater than the larger of the two windows. The hop sizes of the two analysis windows are equal to 1/8 of their respective window sizes by default.

Assuming that the two analysis windows are of different sizes, AudioSculpt will, by default, create two different hop sizes. This implies that the short-term spectra of the two sounds will not be calculated at the same rate in each of the two sound files. When the resulting sound is synthesized, AudioSculpt will calculate the sound according to the temporal base of the spectra in the primary sound file, thereby changing the speed of advancement of the secondary sound in relation to the primary.

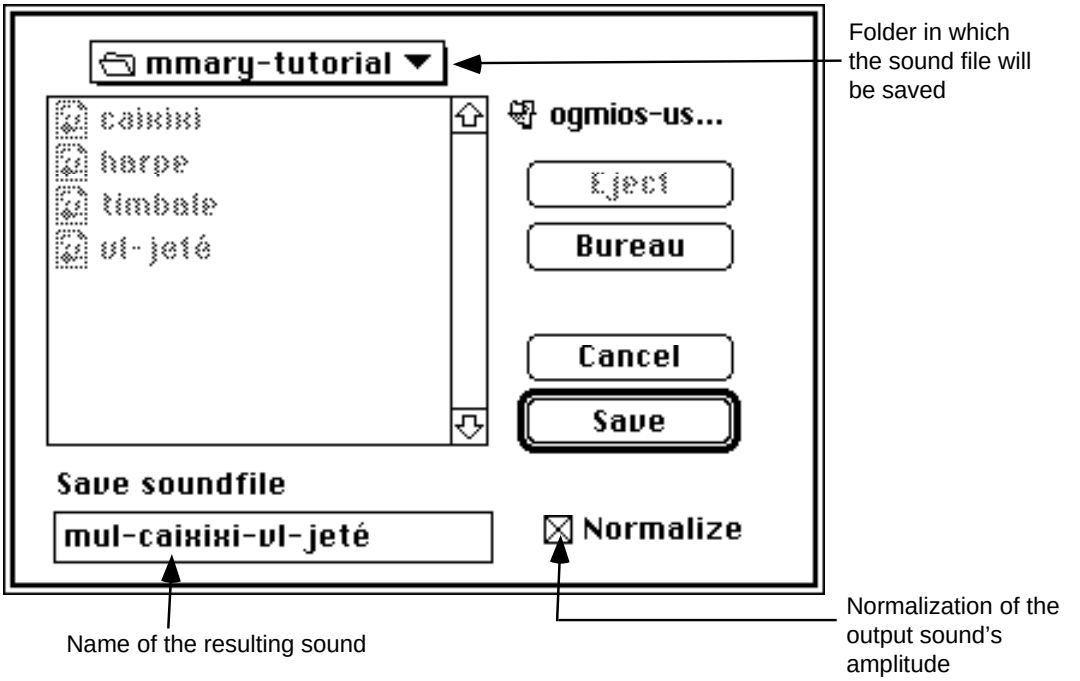
For example, if the hop size of the primary sound is less than that of the secondary (analysis rate is faster in the primary sound), the rhythm of spectral changes which the output sound inherits from the secondary sound will seem faster than in the original. Conversely, if the hop size of the primary sound is greater than that of the secondary (analysis rate is slower in the primary sound), the rhythm of spectral changes inherited from the secondary sound will seem slower than in the original. There is, therefore, a temporal compression or expansion inherent in the second sound when a cross synthesis is calculated using two different hop sizes. One should be aware of this phenomenon, and know that one simple way to avoid it is by using identical analysis window sizes. In general it will suffice to increase, within reasonable limits, the size of the smaller analysis window to be equal to the size of the larger, since the larger window gives a better frequential resolution. This phenomenon of temporal scale distortion may often yield interesting results, in which case two different hop sizes may be kept.

Occasionally, the theoretical window sizes may not be the best to realize a cross-synthesis. In certain cases, by reducing the size of the windows in respect to the theoretical minimum, the cross synthesis may gain an additional sonic quality. Indeed, by reducing the frequential resolution of the spectra, one could also obtain a spectral mix which might be particularly disturbing.

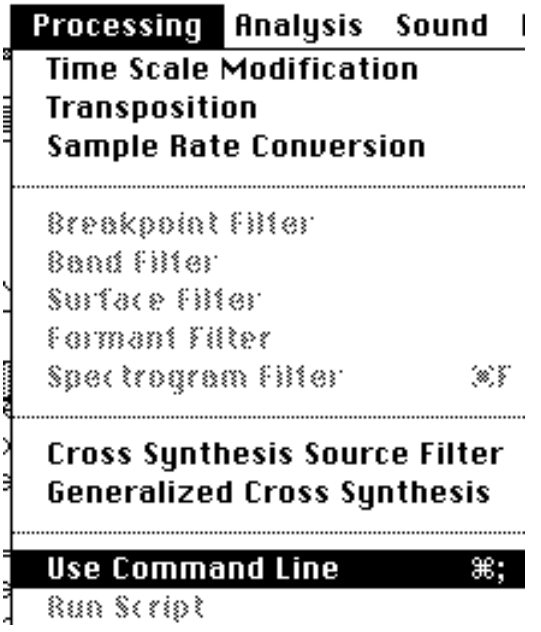
The button labeled **Factory Settings** will re-initialize all analysis parameters to their default values.



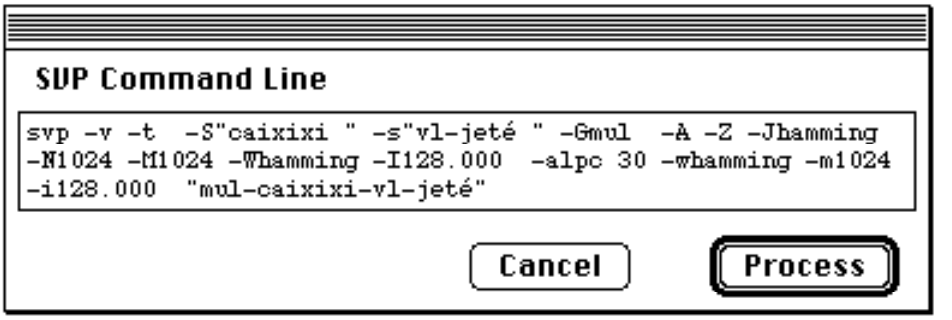
Once the desired values have been chosen and set within the **Analysis Parameters** dialog box, click on **OK**, and then on **Save** in the **Cross Synthesis Source Filter** dialog box. You will then be presented with yet another dialog box in which you can name the resulting sound file and select the folder in which it will be saved. You can click on the **Normalize** checkbox to normalize the output sound file, i.e. scale the amplitude of the sound to its maximum.



The same cross-synthesis may also be requested by using a command line:

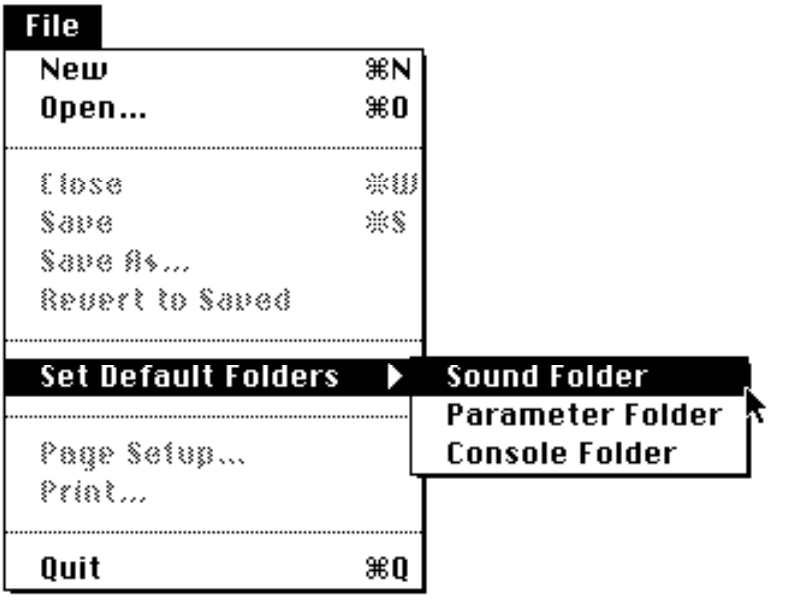


This is the command line equivalent toa source-filter cross-synthesis which is performed on the files **caixixi** and **vl-jeté**:

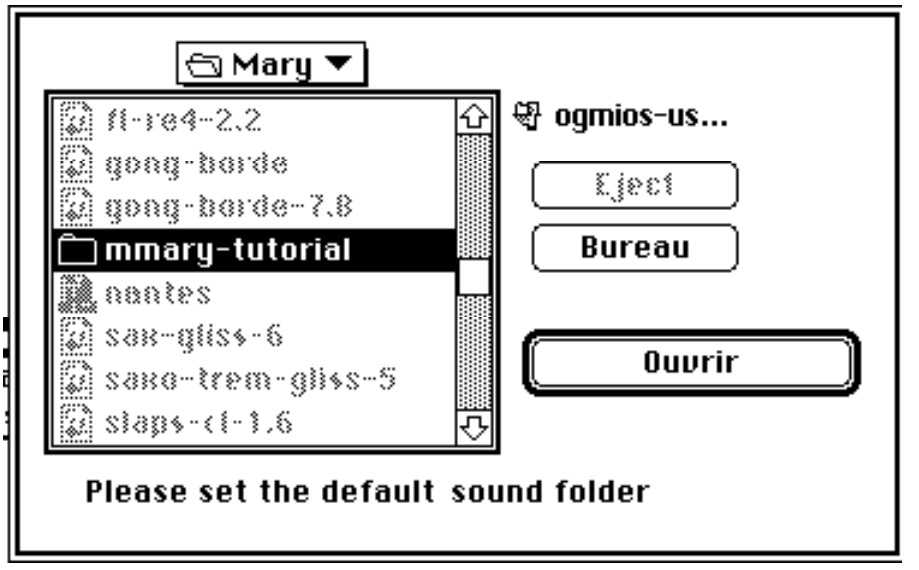


If you need to brush up your SVP options, please refer to the list of options given at the end of this handbook, or, better yet, see you AudioSculpt or your SVP manual.

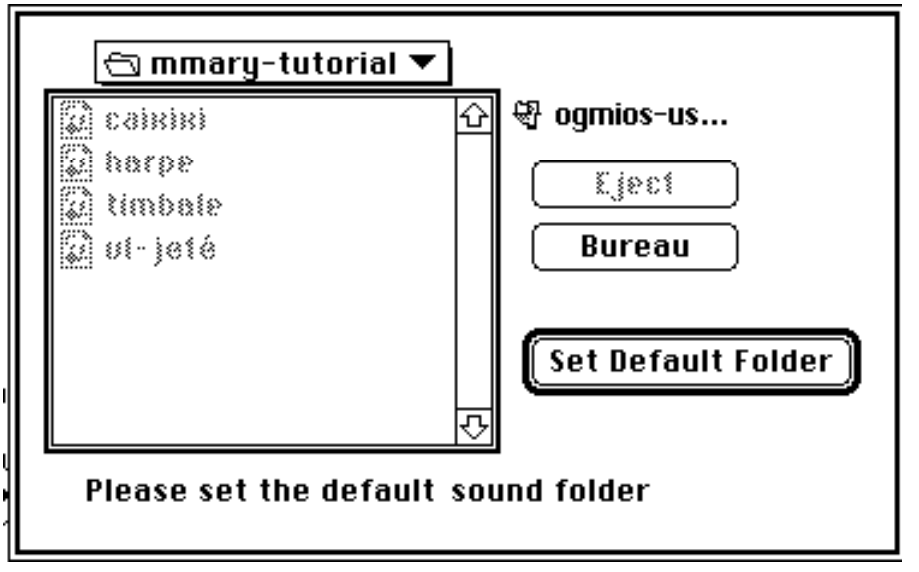
CAUTION: you may change the order of any of the arguments to SVP except the last, which is always the name of the resulting sound file. If you use a command line you should indicate the folder which contains the sound files which you want to cross, and the folder(s) where you would like to keep the parameter files and comment (console) files. Select **Set Default Folder>Sound Folder** from the **File** menu.



to open the following dialog box:



where you may search for the folder which contains the sound files.



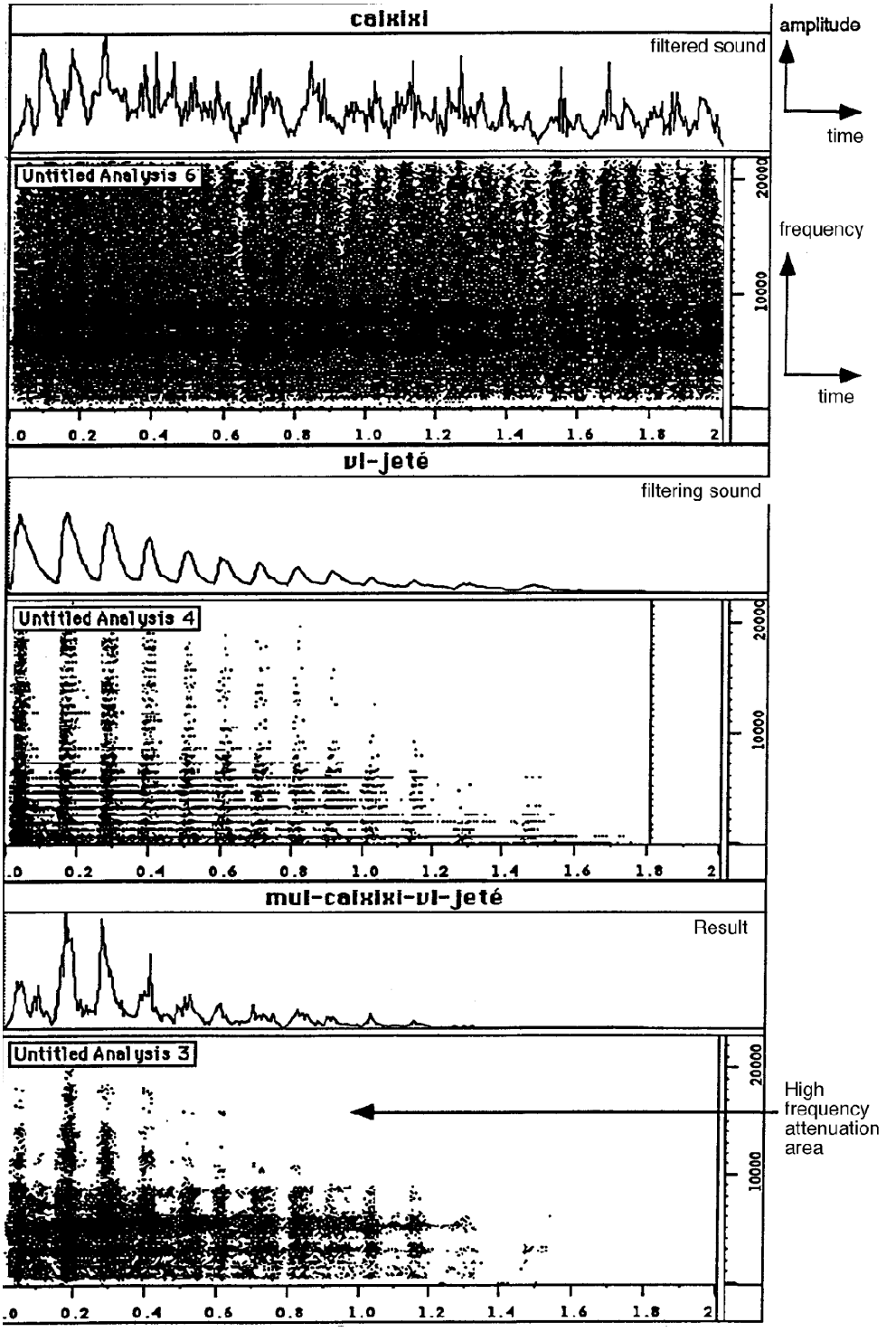
Once you have indicated the folder where the input sound files are to be found, AudioSculpt will always look there for them by default; you may, however, change the default directory at any time. To set the default folders where AudioSculpt will search for parameter files and save console files, select **Set Default Folder>Parameters Folder** and **Set Default Folder>Console Folder** (both in the **File** menu), respectively.

Example 1 index 1, 2 and 3

The graphic representation of sounds in AudioSculpt (amplitude/time envelope or sonogram) allow inspection of different aspects of the sound such as those calculated in this example of source-filter cross-synthesis. The spectral content of the instrument caixixi is quite close to white noise; its sonogram shows a very compact spectral disposition, contrary to that of the violin where the progression of harmonic partials (horizontal lines) is very clear.

The resulting crossed sound contains the spectral organization of the filtered sound, but with the dynamic profile of the filtering sound. Otherwise stated, one can recognize the sequence of attacks of the violin with jeté bowing, but with the sonority of the instrument caixixi. One could call this a caixixi played “jeté.”

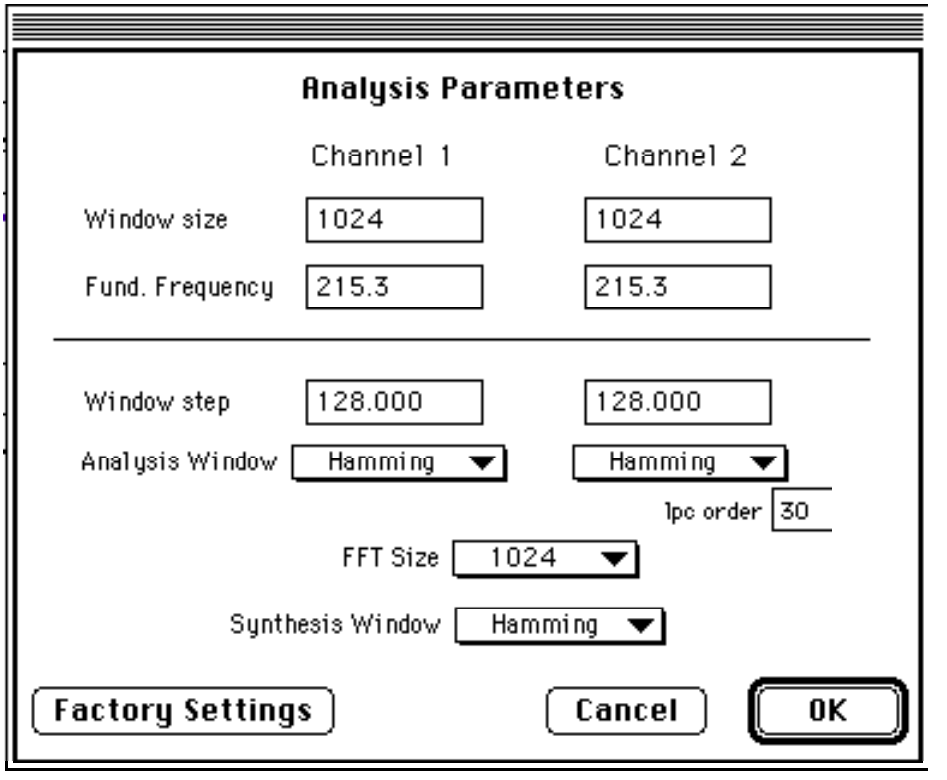
Please note that with this type of cross-synthesis, there can be a definite loss of high frequencies in the resulting sound. This is due to the spectral multiplication which reinforces the common amplitude zones, leading to an attenuation of zones containing weak amplitude.


caixixi
filtered sound
amplitude
time
Untitled Analysis 6
20000
10000
frequency
time
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2
vl-jeté
filtering sound
Untitled Analysis 4
20000
10000
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2
mul-caixixi-vl-jeté
Result
Untitled Analysis 3
20000
10000
High frequency attenuation area
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2

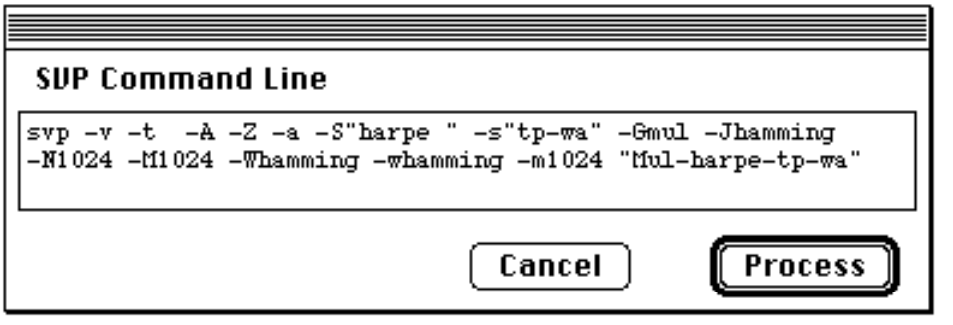
Example 2 . 1 index 4, 5 and 6

A harp arpeggio is filtered (in source-filter mode) by a trumpet using a wah-wah mute which opens and closes rapidly. The resulting sound has a hybrid sonority in which the filtered sound predominates.

Here are the analysis parameters used:

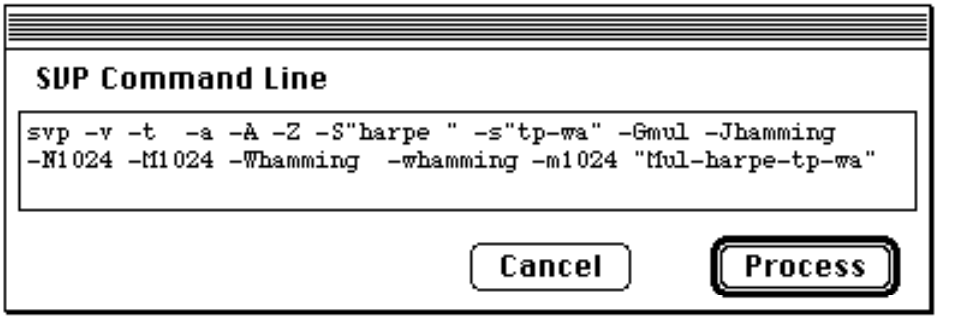


and the equivalent command line:



Example 2 . 2 index 7

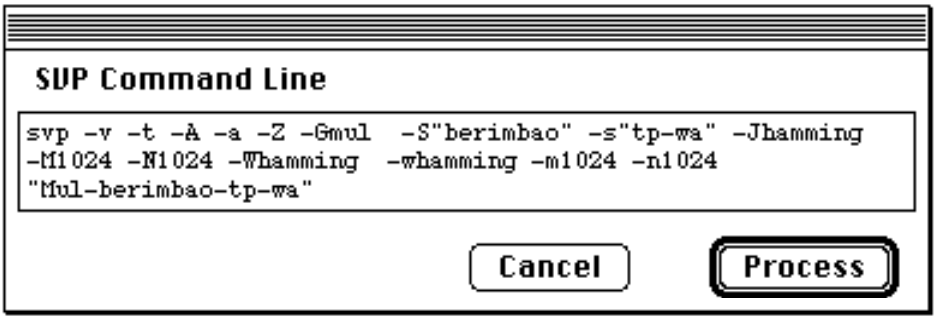
This is the inverse of the previous example; the trumpet is now filtered by the harp. All of the cross synthesis parameters remain the same, but the order of the sound files and the name of the resulting sound have been changed.



Example 3 . 1 index 8, 9

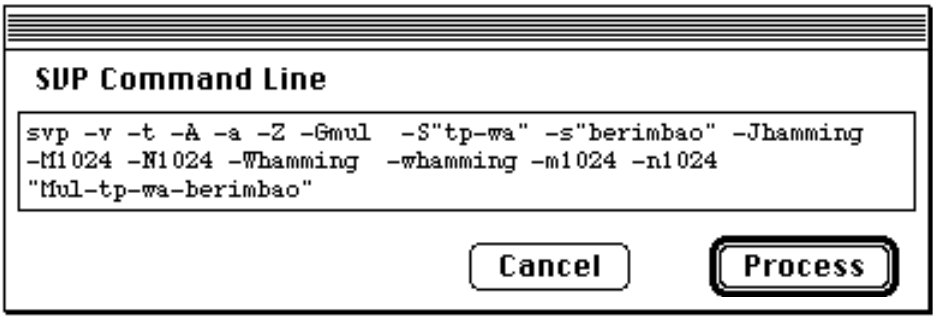
These two examples complement the preceding ones. Using identical cross synthesis parameters, the trumpet will be crossed with a small sequence played on the berimbao. The resulting sound contains a counterpoint between the timbral fluctuations and rhythmic action contained in the two sounds.

The command line to filter the berimbao by the trumpet with wah-wah mute is as follows:



Example 3 . 2 index 10

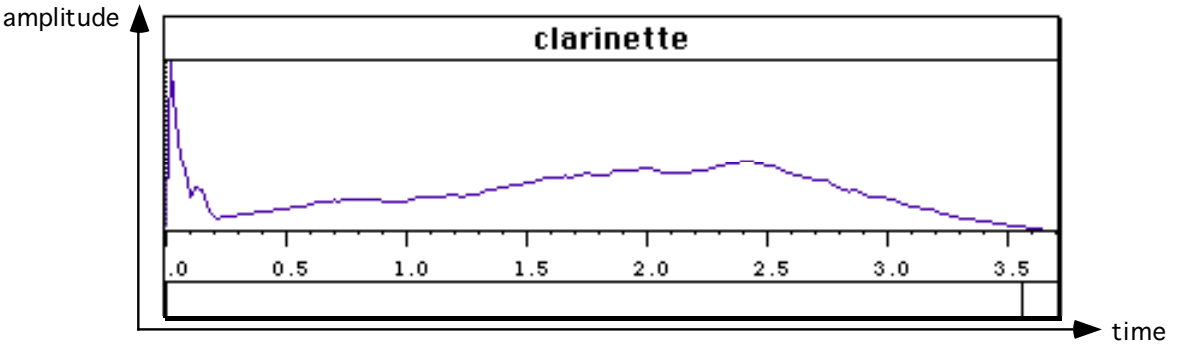
This is the command line to filter the trumpet with wah-wah mute by the berimbao:



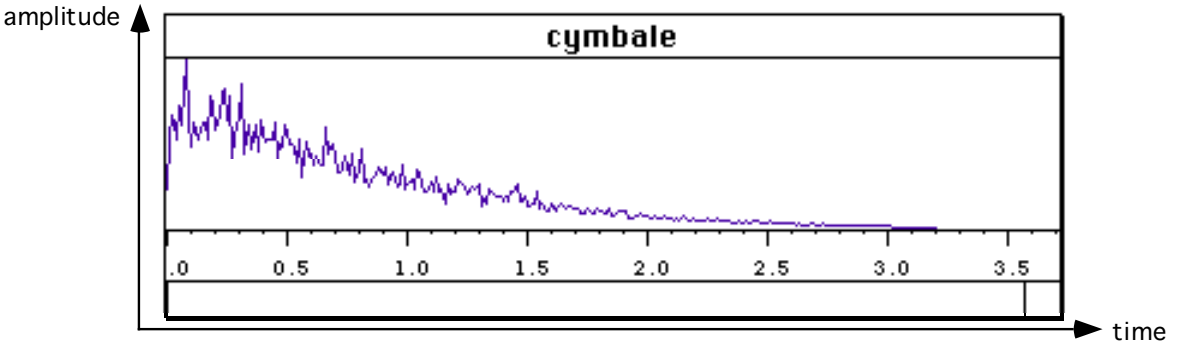
At this point we will begin presenting sound examples constructed from several processing operations in order to go deeper into the workings of AudioSculpt.

Example 4 . 1 index 11, 12 and 13

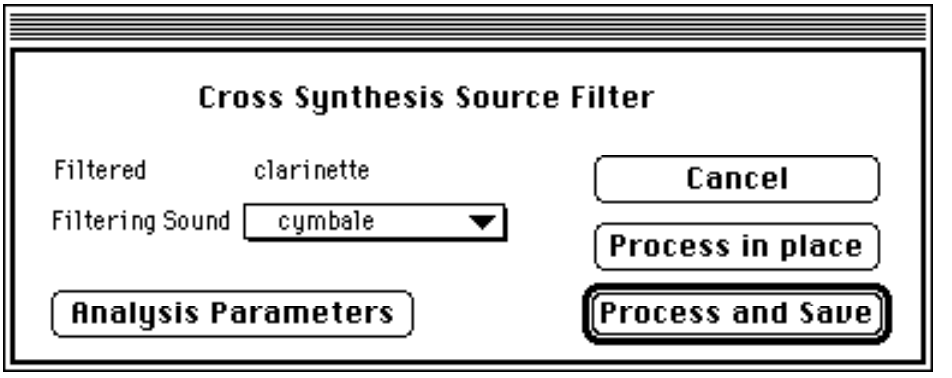
We will begin with a source-filter cross-synthesis between a clarinet with a sforzato attack followed by subito piano, crescendo and diminuendo:



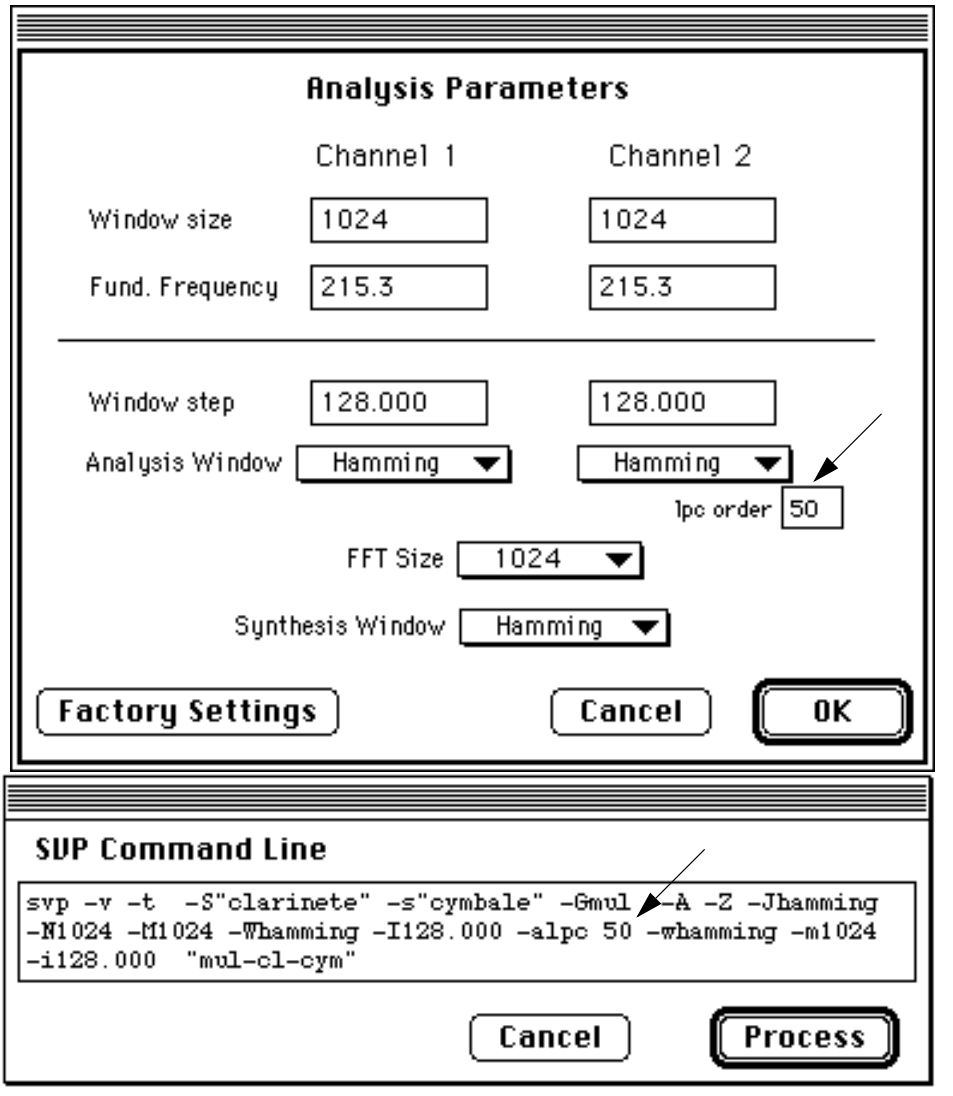
and a cymbal with a strong attack followed by a diminuendo:



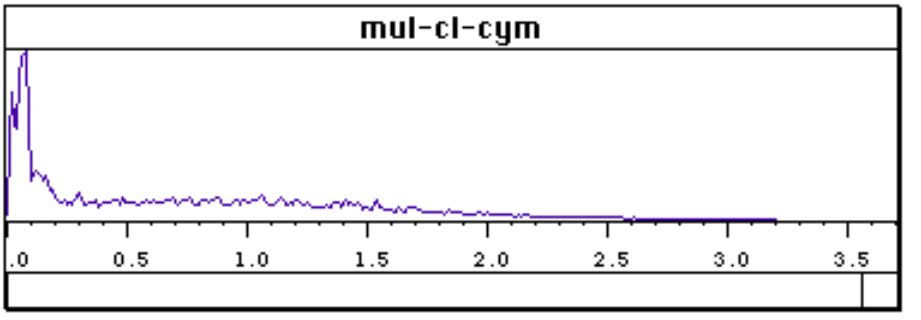
The cymbal will be used to filter the clarinet:



Since the cymbal's sound contains many partials, we will increase the definition of the spectral envelope of the analysis by setting the order (i.e. the number of poles) of the linear prediction algorithm (LPC) to 50:



The resulting sound has the following time/amplitude envelope:



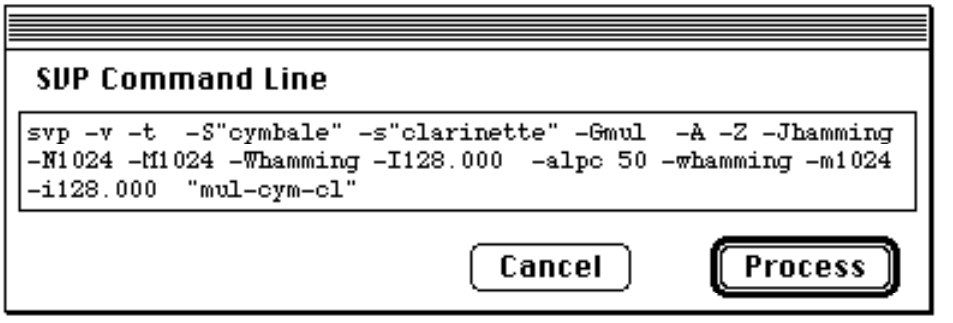
Remember that the filtering sound allows only those frequencies to pass which are contained in its spectrum. The source sound passes through this filter.

Note that the two source sounds have amplitude envelopes which are rather different, and that the envelope of the crossed sound itself has a profile quite unlike those of the sound sources. Its envelope more closely resembles that of a plucked string, and in it one can distinguish four zones as marked in the figure.

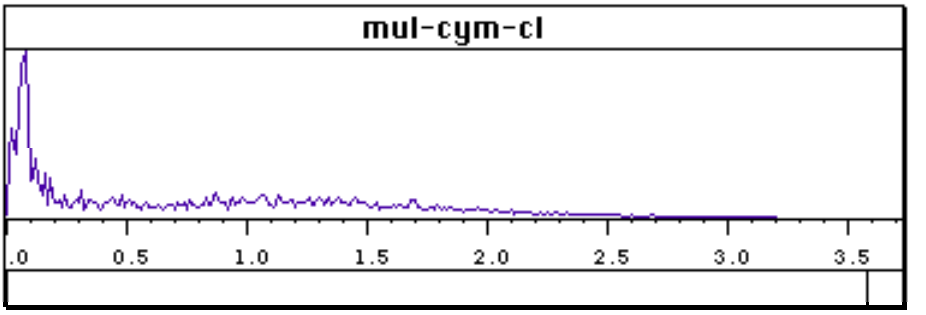
- The amplitude of the attack is very strong because it is an important part of each of the source sounds, containing many frequencies common to both.
- Just after the attack, the envelope falls sharply because, even though the "cymbal filter" remains open with strong amplitude levels, the clarinet partials have rather weak amplitudes.
- The amplitude of the crossed sound then remains stable, because, even though the clarinet's amplitude increases, the "cymbal filter" begins to close.
- After 3.2 seconds, the clarinet is filtered by an empty signal. The spectra of the clarinet are multiplied by empty spectral envelopes, since there is no sound in the cymbal file after this time. The resulting amplitude is therefore nil.

Example 4 . 2 index 14

This is the inverse of the preceding example. Here, the cymbal is filtered by the clarinet; all of the cross-synthesis parameter values remain the same.



The resulting amplitude envelope resembles that of the preceding example, but the predominant timbre is that of the cymbal:

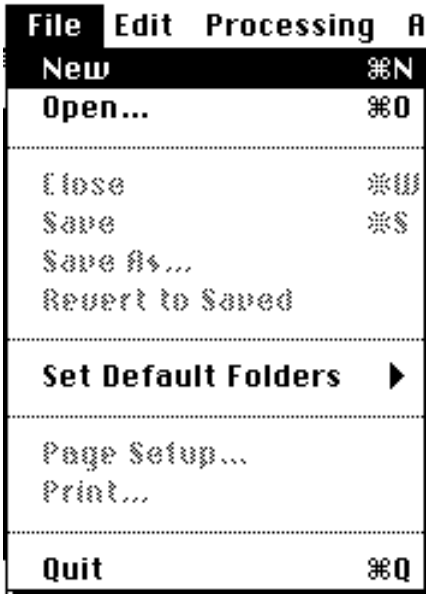


Many successive treatments may be applied to a sound in order to arrive at a particular type of sound desired. The user is encouraged to try different types of treatments (time-stretching, transposition, etc…) using either fixed or dynamic parameter values. Dynamic parameter values are those which change over time, and can be supplied to AudioSculpt in the form of a text file.

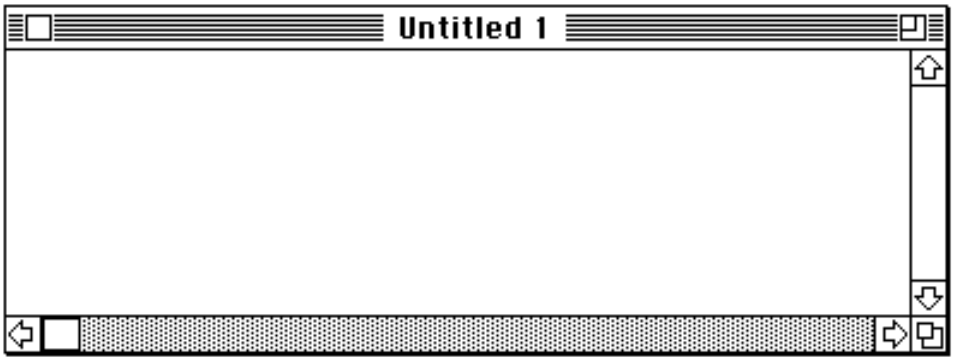
The following section gives an example of applying a dynamic time-stretch to the sound we have just created.

Creating a Text File for Dynamic Sound Processing

By selecting **New** from the **File** menu (or by typing command-N), you can open an empty text-editing window to create a dynamic parameter file:



The commands results in a new window:

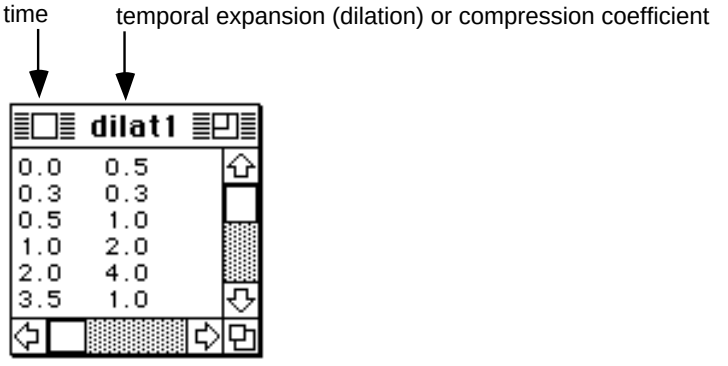


In the figure below, we have created a simple file which contains points in time with corresponding coefficients for temporal compression/expansion.

Caution: If the coefficient values are less than 1 there will be a temporal compression, if they are greater than 1 there will be a temporal expansion or dilation (time-stretching); a value of 1 will not change the duration of the original sound.

Between any two points in time there will be a linear interpolation between the coefficients. In other words, each coefficient will change continuously in value until it reaches the next coefficient value indicated.

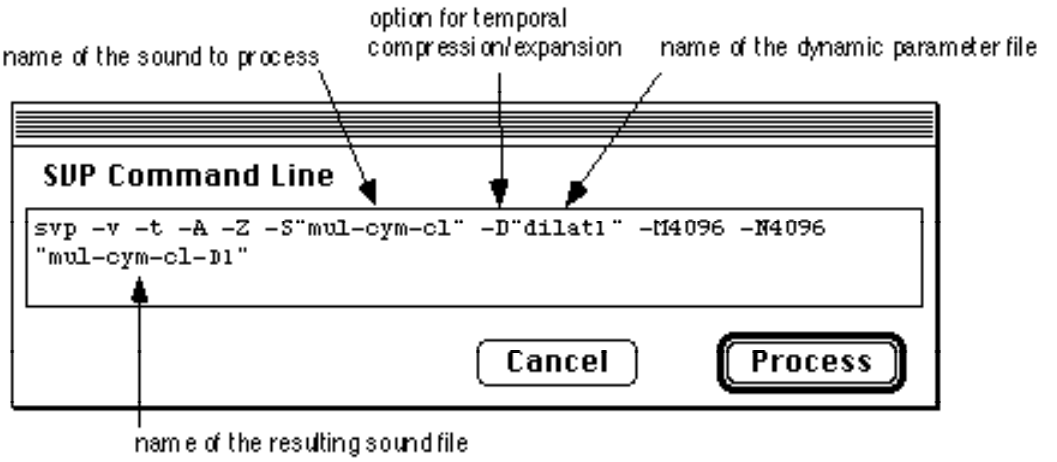
When you save the file, you will be prompted to name it and should be sure to remember the folder where you have saved it. In this example the parameter file has been named **dilat1**.



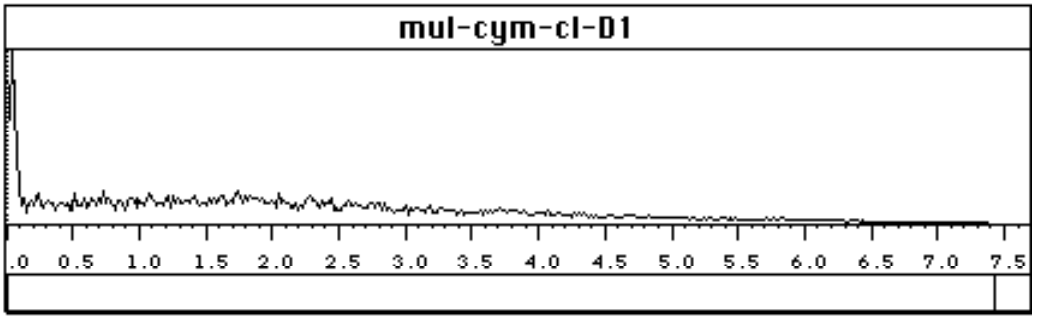
The resulting sound will begin with a time-compression which becomes stronger until 0.3 seconds, at which point the compression slowly becomes weaker until 0.5 seconds where the compression coefficient is equal to 1.0 (no compression or expansion). Between 0.5 and 1.0 seconds, the coefficient becomes greater, thereby time-stretching the sound, and continuing until 2.0 seconds, before returning to 1.0 over the remaining part of the sound. All in all, the sound will be time-compressed in a variable manner from 0.0 to 0.5 seconds and time-stretched from 0.5 to 3.5 seconds.

Example 4 . 3 index 15

In order to use a text file to control dynamically the temporal expansion and compression of a sound, begin the processing by selecting **Time Scale Modification** from the **Processing** menu with the window of the dynamic parameter file active, or, if using a command line, indicate the name of the file just after the option -D.

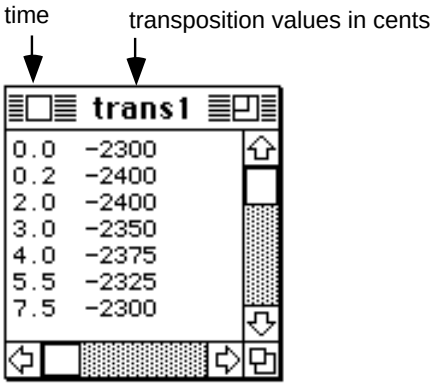


The time scale modification gives the file a total duration of about 7.5 seconds. The attack stays very short, but the resonance was stretched by more than twice in an irregular fashion.



Example 4 .4 index 16

The following example further modifies the previously created sound by applying a transposition to it. In order to make a dynamic transposition, you should use a text file such as:

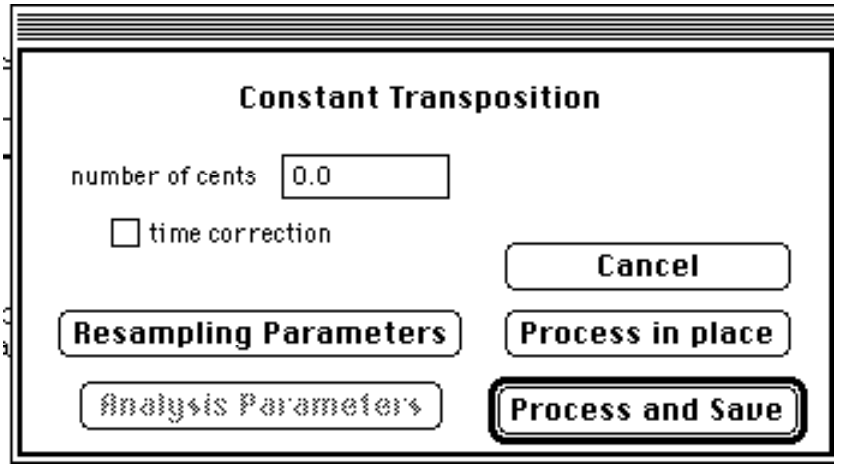


Transposition values are expressed in cents, for example:

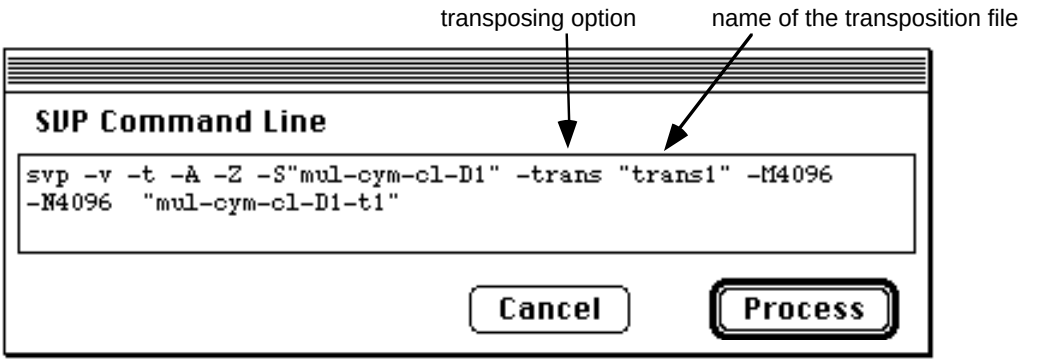
- 100 cents = one half step higher
- -200 cents = a whole step lower (or two half steps lower)
- 0 cents = no transposition

The maximum transposition allowed is two octaves in either direction. (+2400 or -2400 cents).

Begin the processing by selecting **Transposition** from the **Processing** menu, with the dynamic transposition file in the active window…

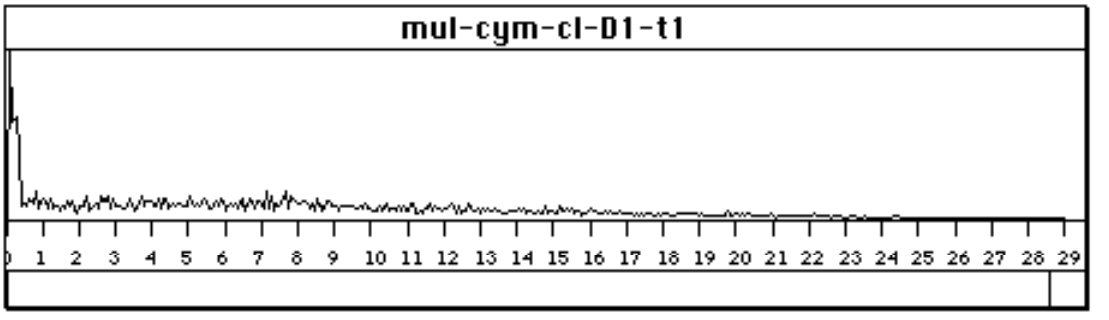


or, when using a command line, indicate the name of the transposition file after the option -trans:



Transposition without time correction will modify the duration of the sound; if you would like to preserve the sound's original duration, click on the **time correction** checkbox in the **Constant Transposition** dialog box (or add the option -D in the command line). If you wish to make a constant transposition, type the transposition interval in cents directly into the space provided in the dialog box.

In our example, the transposition has given the sound a deep drone with little fluctuations in pitch — it is evident that the sound has evolved into something far from the cymbal and clarinet of its origins.



4 Generalized cross-synthesis

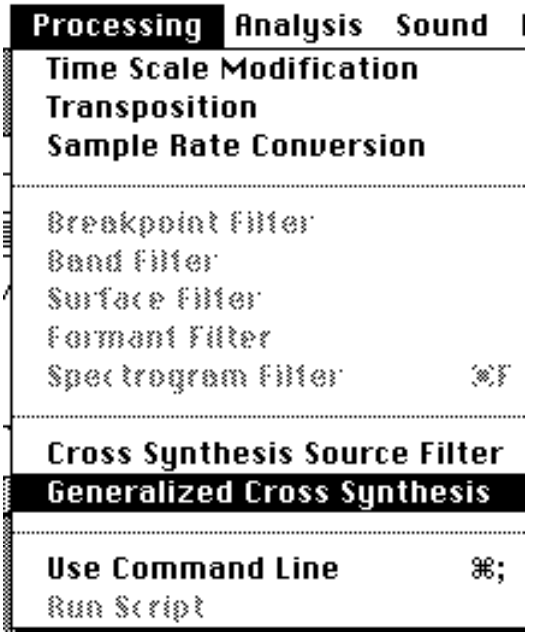
This type of cross-synthesis allows the separate combination of the amplitude and instantaneous frequency spectra of two input sounds. For the amplitude spectra of the primary and secondary sounds, A1 and A2, and their instantaneous phase spectra, $f1$ and $f2$,(related to instantaneous frequency) on a given window, the spectrum of the output sound for that window is defined as:

$$A = X\ A1 + x\ A2 + q\ A1\ A2$$

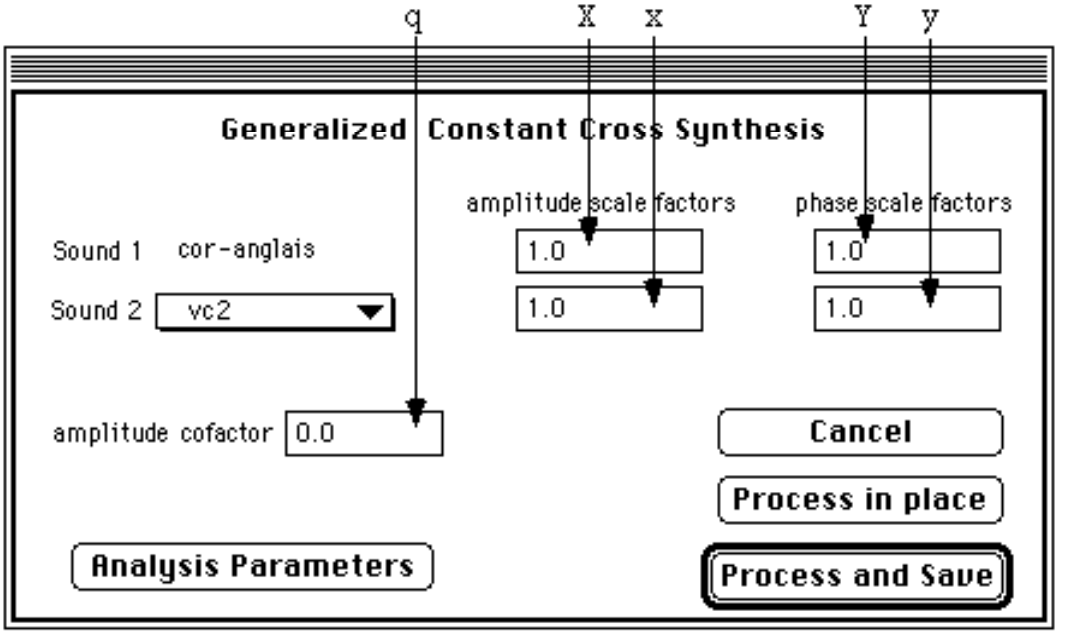
$$f = Y\ f1 + y\ f2$$

The Amplitude Scale Factors correspond to the coefficients **X** and **x** of SVP, the Phase Scale Factors to coefficients **Y** and **y** and the Amplitude Cofactor to coefficient **q**.

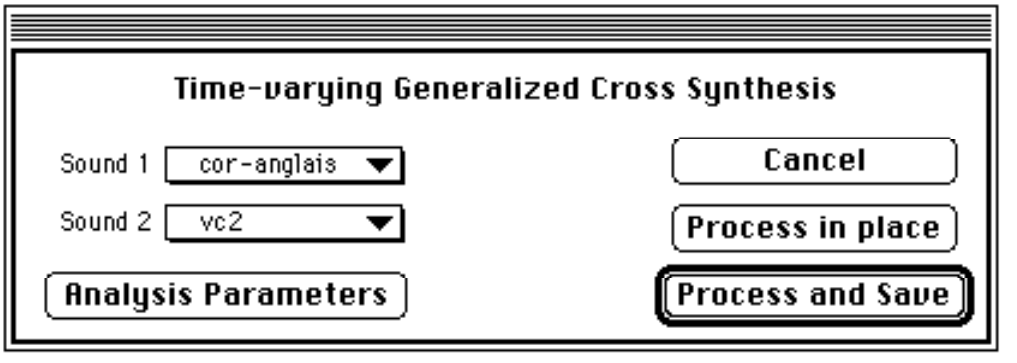
To begin processing, open the files that you would like to cross in AudioSculpt using the command by selecting **Open** from the **File** menu (or command-O). As you now know, you have the option to use either fixed or dynamic (time-varying) parameter values. To use fixed parameter values just choose the type of cross synthesis desired from the **Processing** menu:



In the dialog box which opens, select the sound files and adjust the cross-synthesis parameter values. Then, click on **Analysis Parameters** to enter the analysis parameter values, as in the case of the source-filter cross-synthesis.



Using dynamic parameters with a cross-synthesis is much like using them with time-stretching or transposition. Begin by creating a text file with the crossing parameters. Next, make the window containing the text the active window and select the type of cross-synthesis from the **Processing** menu. The following dialog box will open:

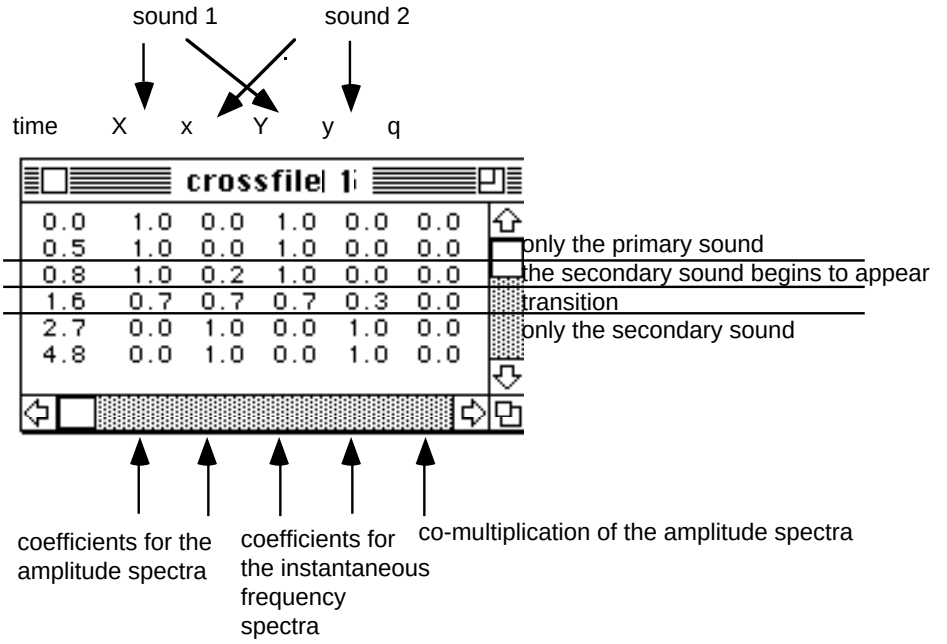


Select the sound files you wish to use, click on **Analysis Parameters** to modify the analysis parameter values, as before, then begin the processing by clicking on **Process and Save**.

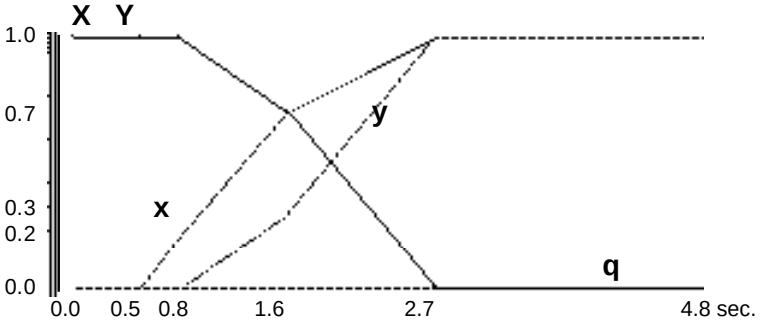
Example 5 index 17, 18 and 19

This is an example of spectral interpolation from one sound to another, a typical application of the generalized cross-synthesis mode. To begin with we will choose two sounds that resemble each other — an English horn and a cello — and proceed by constructing a transition from one to the other using a dynamic parameter file. The text file used (**crossfile-1**) indicates the temporal changes of the crossing parameters **X**, **x**, **Y**, **y**, and **q**. Do not forget that this file must be in the currently active window when the **Generalized Cross Synthesis** command is selected, otherwise AudioSculpt will assume the use of fixed parameter values.

Here is the parameter file used for the crossing:



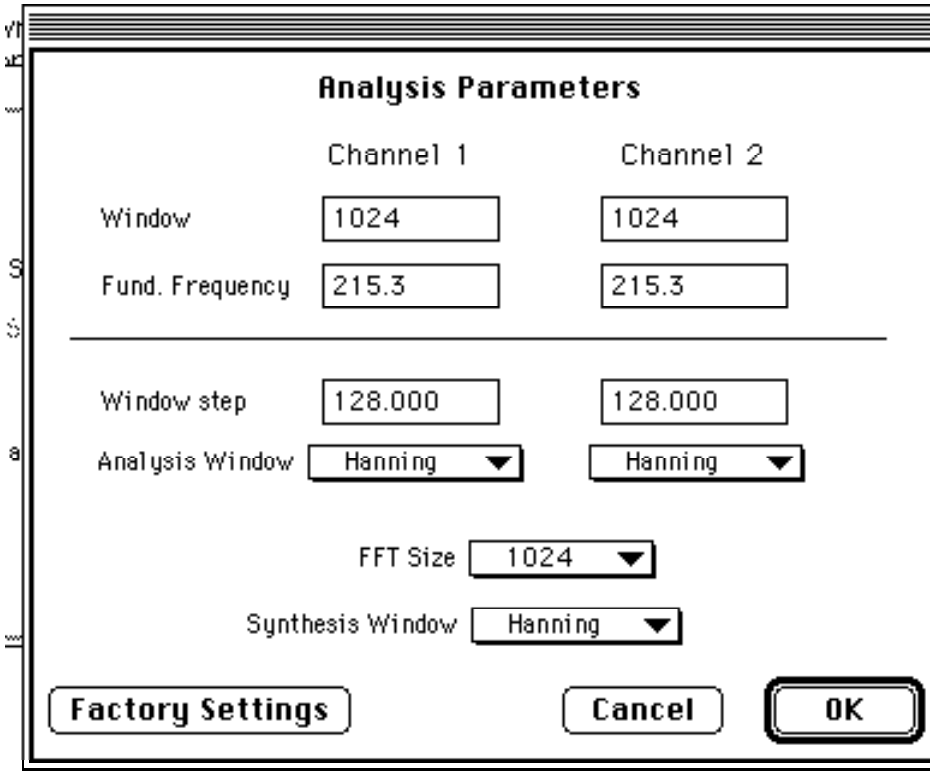
The following is a graphic representation of the parameter file **crossfile-1**:



(The X and Y curves are superimposed)

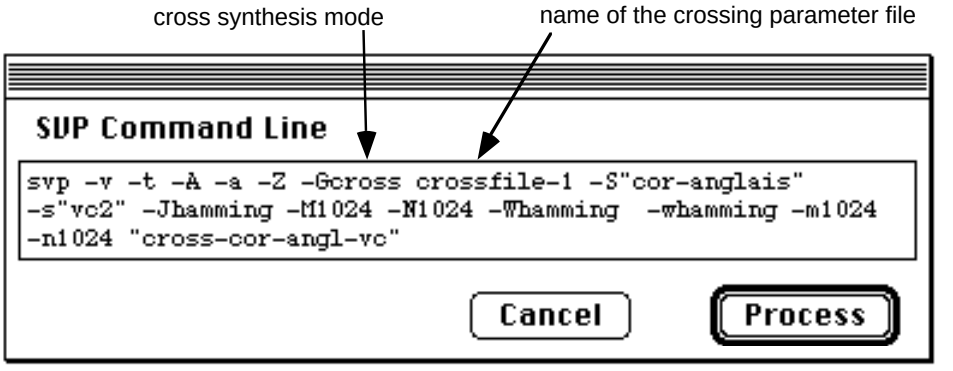
The English horn is the only sound at the beginning of the output file since both the **X** and **Y** values are equal to 1, and the **x** and **y** values (for the cello) are set to 0. These values are retained until 0.5 seconds, so during the first half second only the English horn will be heard. The transition to the cello sound begins after this point, when, from 0.5 to 0.8 seconds, the amplitude of the secondary sound begins to increase[1]. By 1.6 seconds the transition is already evident, and, from 2.7 seconds until the end (at 4.8 seconds) only the secondary (cello) sound is heard.

We will use the following analysis parameters:



This is the equivalent command line:

1. One should always remember to initiate such a cross at a very subtle rate, since our hearing perception is very sensitive to changes made to a stable signal.



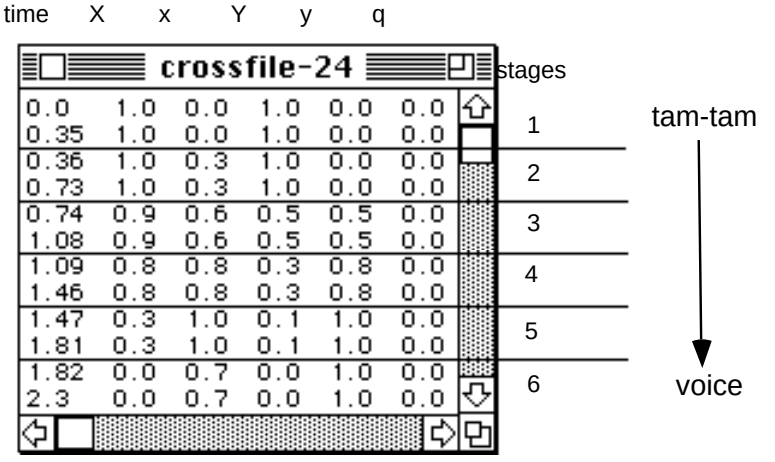
This timbral interpolation was greatly facilitated because the both English horn and the cello have a large quantity of similar timbral characteristics and a similar acoustic ambiance at many pitch levels. As we can see, the choice of sounds to interpolate is a very important decision in this kind of work. When two sounds are similar, an interpolation between them may be made solely by the interpolation of their timbres.

By using a parameter file for cross-synthesis you can obtain a very smooth and homogenous interpolation; any potential imperfections in this gradual timbral transformation are masked by the other parameters of the sound. However, in a cross between two sounds which have different acoustic qualities, each change in the sound, however slight, will bring these differences immediately to our attention.

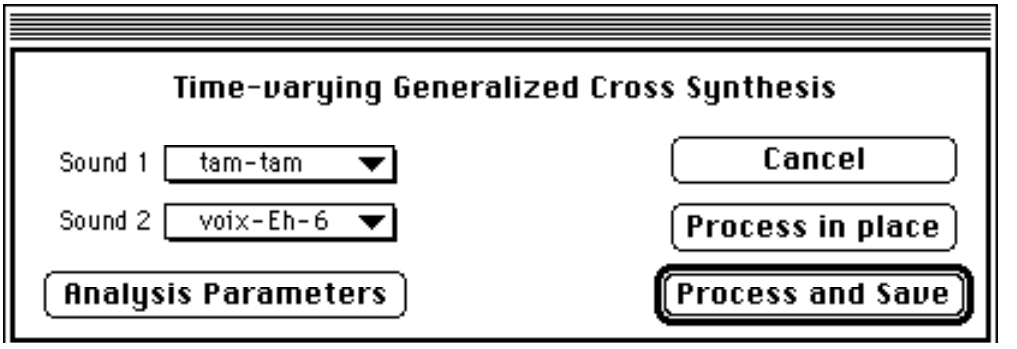
Example 5.1 index 20, 21 and 22

The following example offers different types of interpolation between two very different sounds. Here, the sound of a tam-tam is crossed with a succession of short vocal attacks.

The interpolation will take place in “stages”. To this end, we will use a dynamic crossing file which will change the parameter values just before each vocal attack, and remain stable until the next attack.

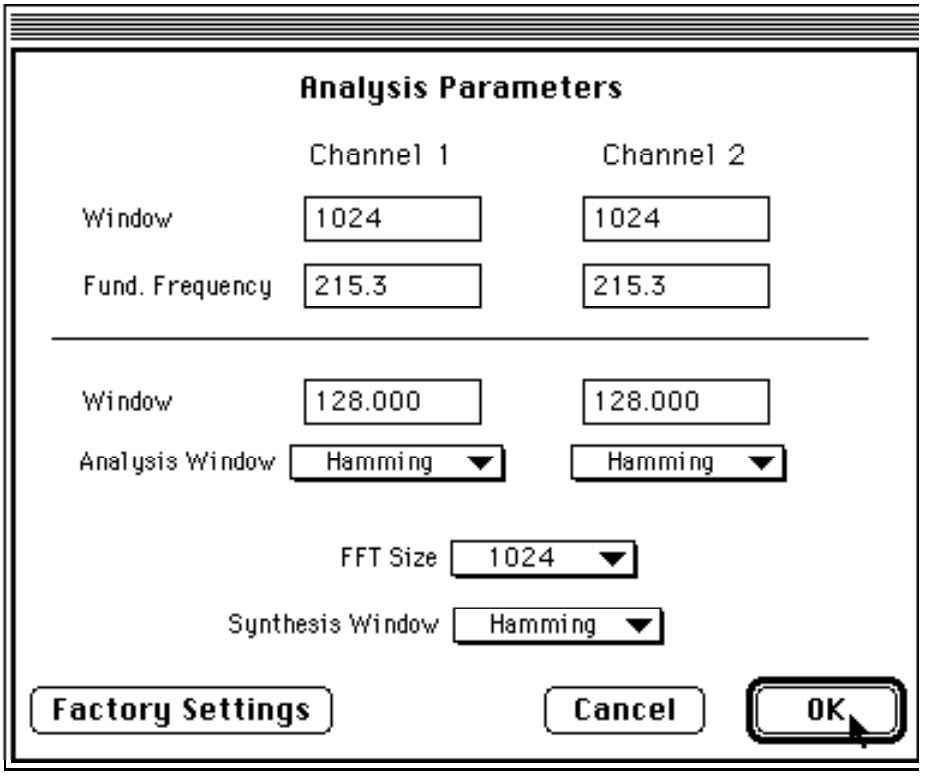


With the window of the crossing file active, select **Generalized Cross Synthesis** from the **Processing** menu, to open the dialog box where you can select the files to be crossed.



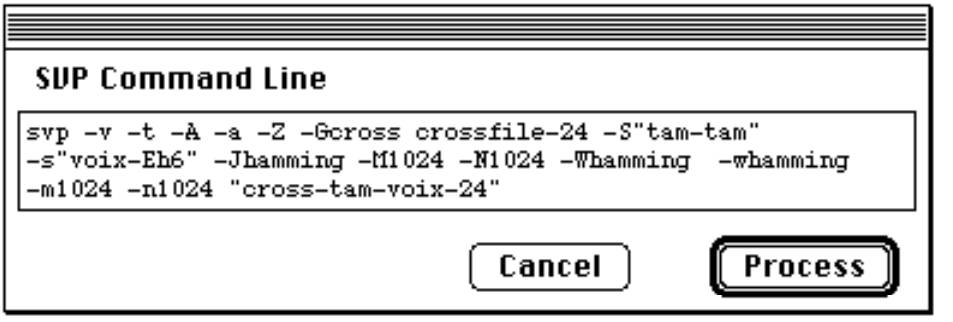
Click on **Analysis Parameters** to open the dialog box to adjust the analysis parameters.

We will use the following analysis parameters:

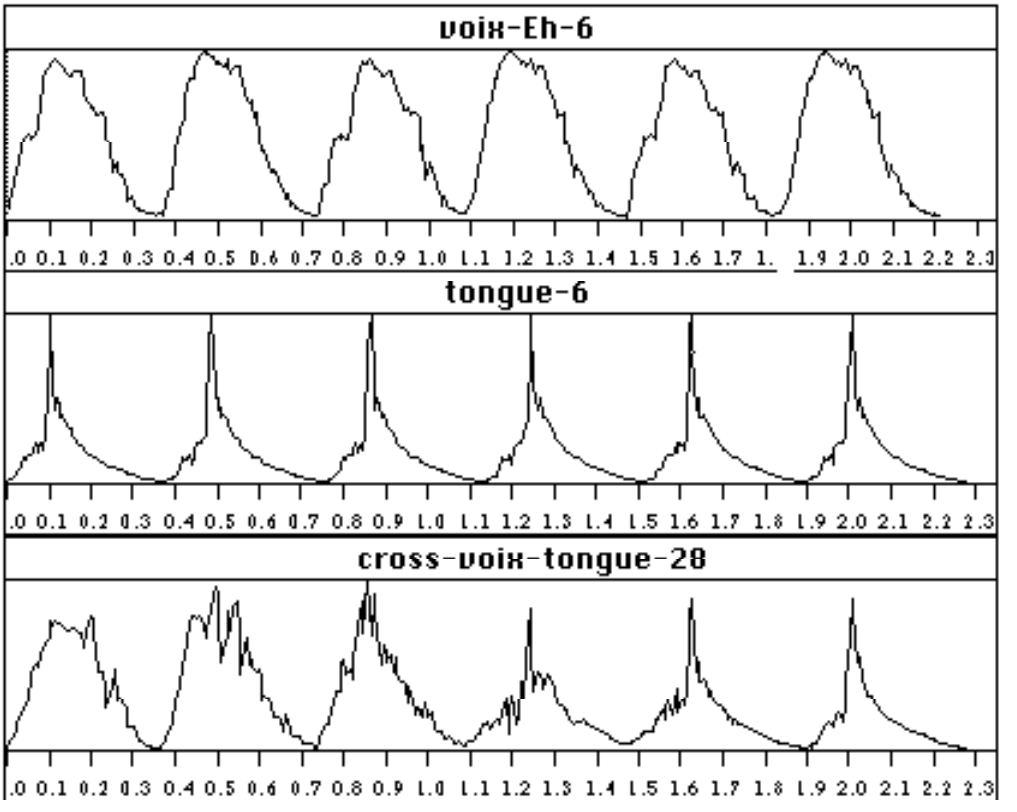


Click on **OK**, then **Process and Save** to begin the cross-synthesis.

The following equivalent command line may also be used:



Here is the amplitude/time envelope representation of the two source sounds followed by their resulting cross-synthesis:



The interpolation has been made discreetly at the attack of each vocalization.

We will now create a series of examples using the same two sound files with the same analysis parameters as above. Our variations will be made only to the dynamic parameter file controlling the cross-synthesis. In this manner you can begin to better understand the many diverse possibilities inherent in this type of cross synthesis.

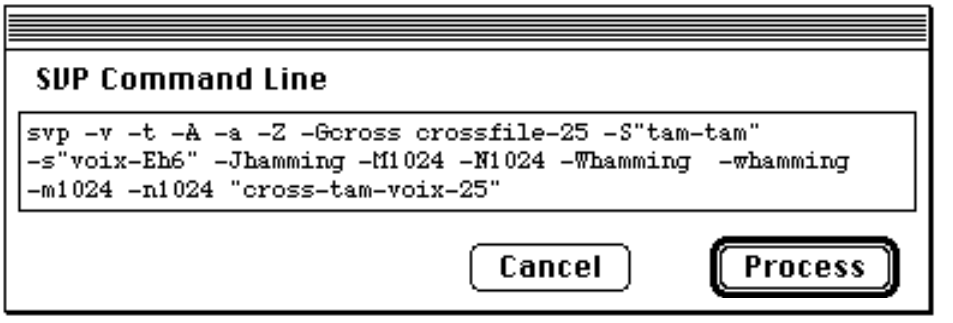
Example 5.2 index 23

In this file the “stages” progress by “jumps”.

crossfile-25

time	X	x	Y	y	q
0.0	0.0	1.0	1.0	0.0	0.0
0.35	0.0	1.0	1.0	0.0	0.0
0.36	1.0	0.3	1.0	0.3	0.0
0.73	1.0	0.3	1.0	0.3	0.0
0.74	0.9	0.6	0.5	0.5	0.0
1.08	0.9	0.6	0.5	0.5	0.0
1.09	1.0	0.8	0.3	0.8	0.0
1.46	1.0	0.8	0.3	0.8	0.0
1.47	0.3	1.0	0.5	1.0	0.0
1.81	0.3	1.0	0.5	1.0	0.0
1.82	0.0	0.7	0.0	1.0	0.0
2.3	0.0	0.7	0.0	1.0	0.0

In order to process this cross-synthesis by using a command line, it will suffice to change only the name of the crossing file and the name of the output file.

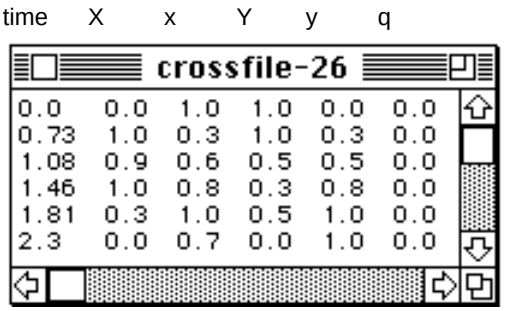


The resulting sound:

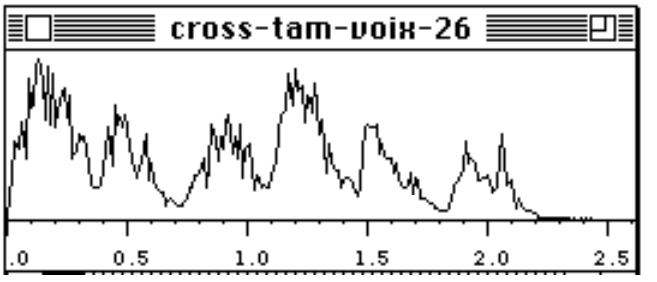
The “jumps” give a different color to each vocalization.

Example 5.3 index 24

Here there are no more “stages”;

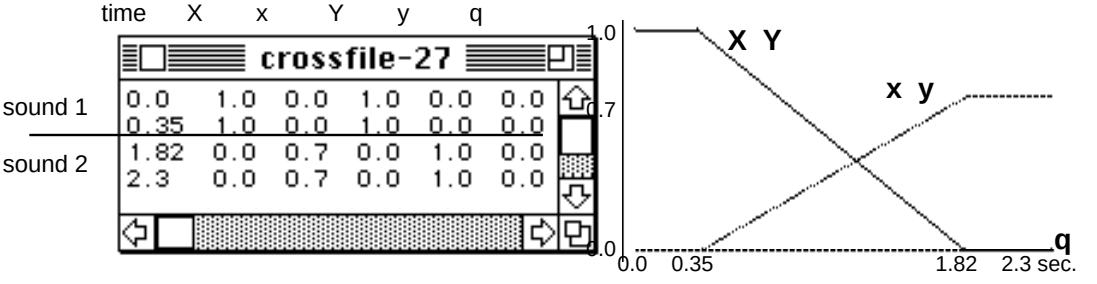


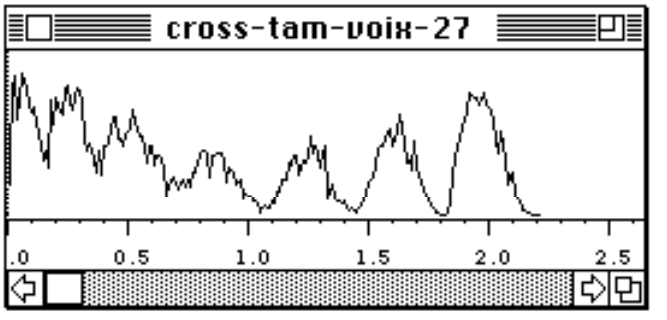
The changes are continuous, but made in a zigzag fashion.



Example 5.4 index 25

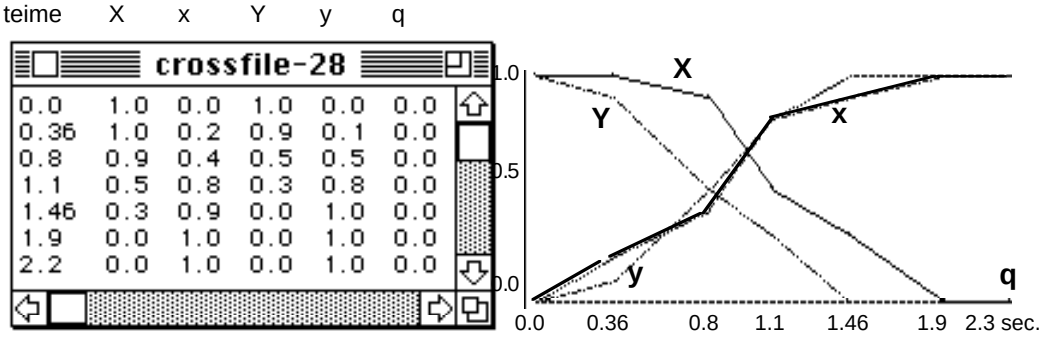
The primary sound file is heard alone at the beginning (between 0 and 0.35 seconds), and the secondary sound file is heard alone at the end (from 1.82 to 2.3 seconds). The transition takes place between 0.35 and 1.82 seconds.



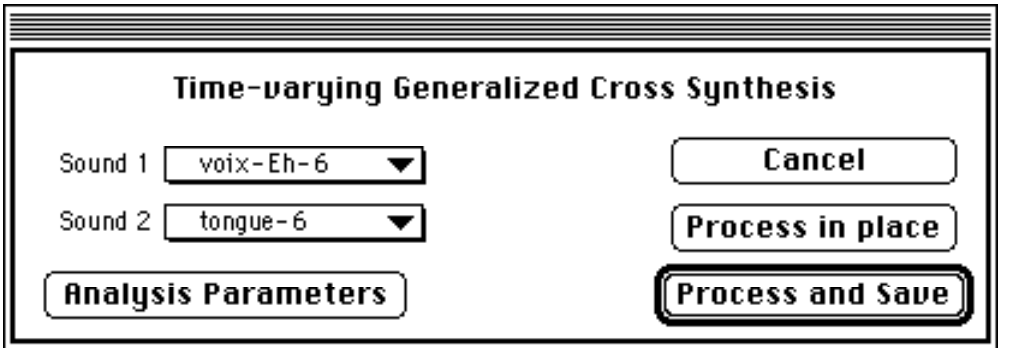


Example 5.5 index 26 and 27

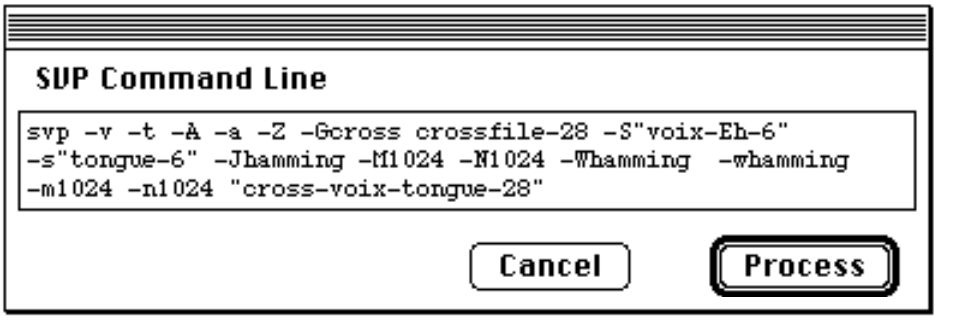
In developing the same idea, the voice will now be crossed with “tongue-rams” on the flute; the analysis parameters are the same as before. Below are both the dynamic crossing file and its graphic representation.



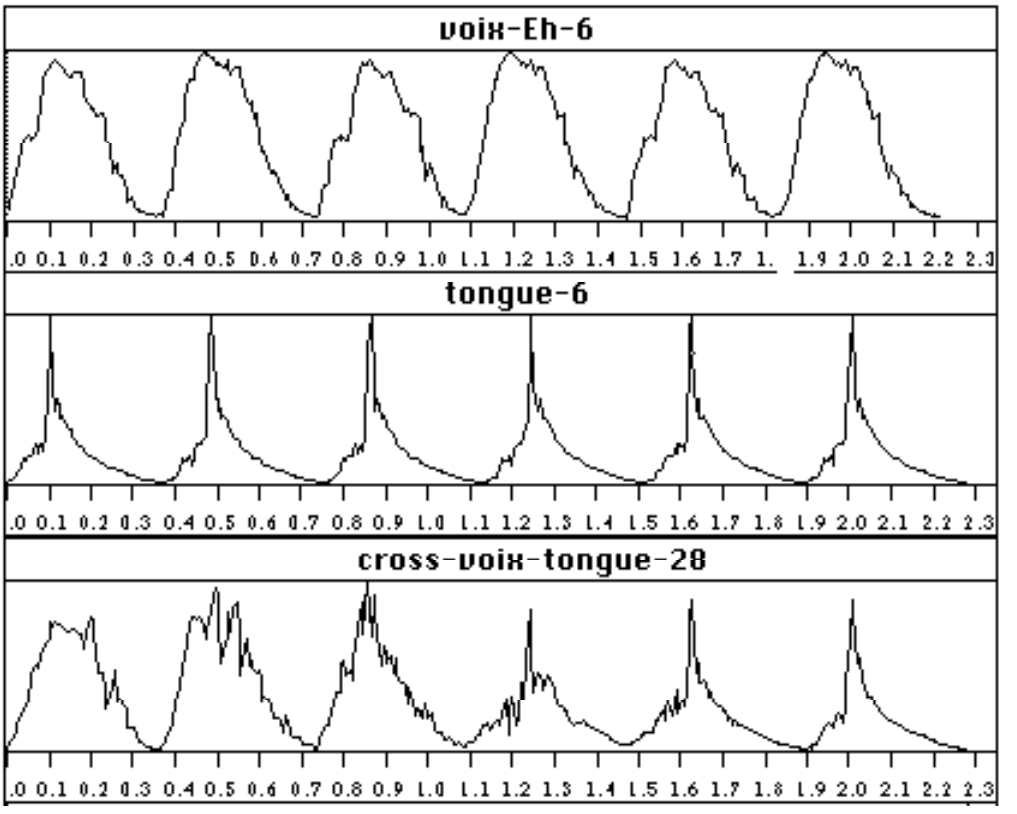
As we can see, the crossing between the two sound files will not be made in a linear fashion, in order to make a more convincing interpolation.



Here is the command line equivalent:



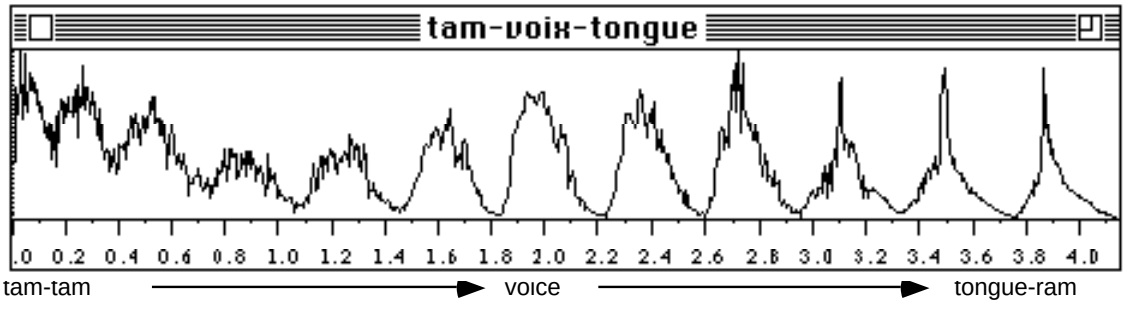
The file **cross-voix-tongue-28** is the resulting interpolation between the vocal attacks and the “tongue-rams” on the flute.



Example 5.6 index 28

Here is an example of cross-synthesis in sequence.

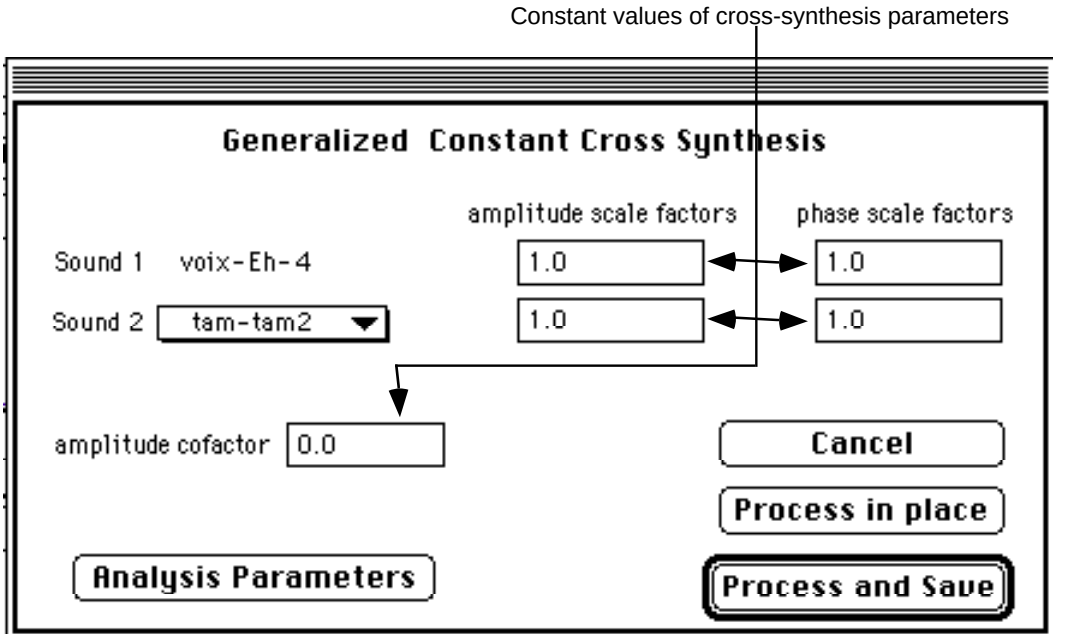
By using a digital sound editing program (such as Sound Designer by Digidesign, for instance), you can paste example **5.4** after example **5.5** to obtain an interpolation in two steps.



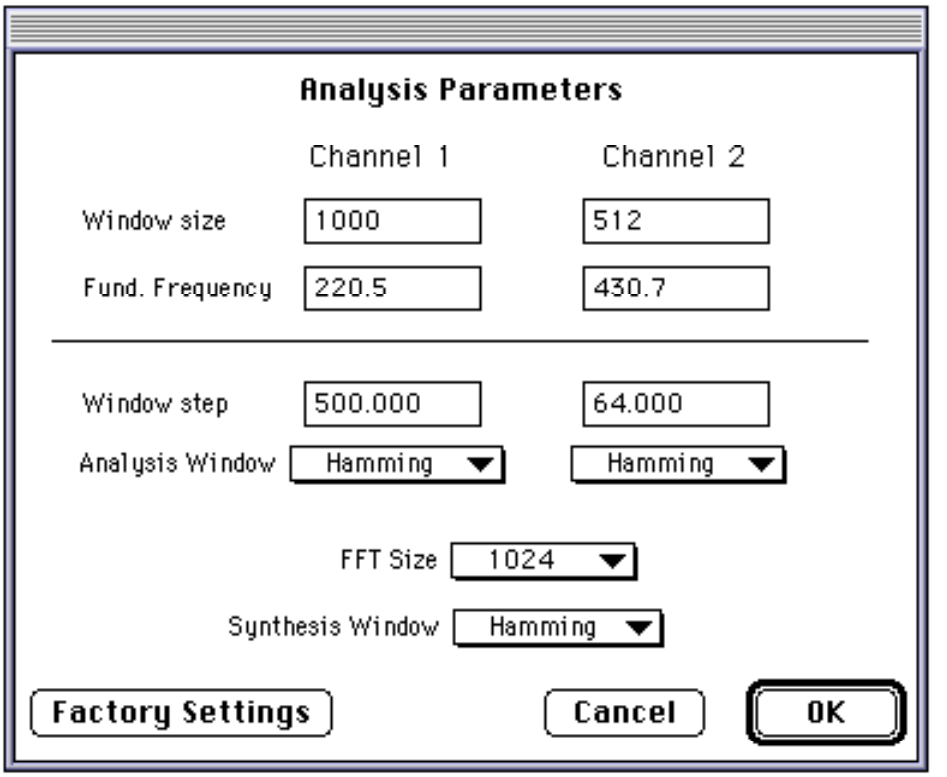
One can further develop this idea by editing a larger series of cross-syntheses together. A return to the point of departure may even be made by performing a cross-synthesis from the last to the first sound in the sequence.

Example 5.7 index 29, 30 and 31

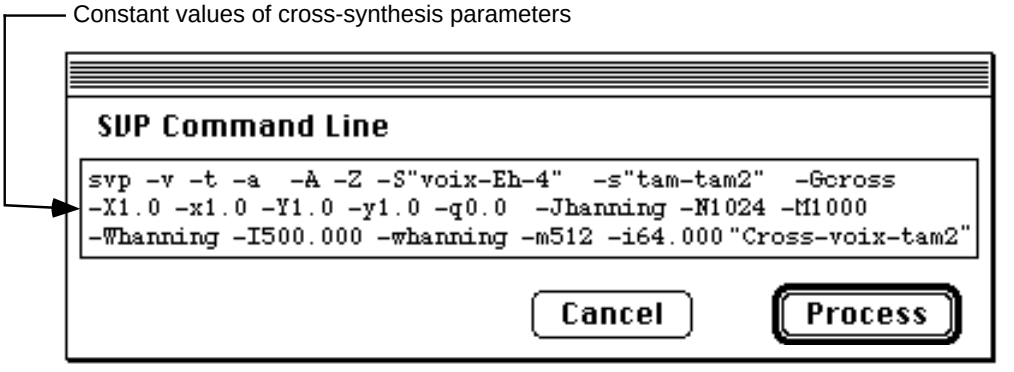
Our last example in this series will not use a parameter file, since our crossing parameters will remain constant. Our source sound files will be almost the same: the voice will contain only four attacks, and the tam-tam's resonance will be longer. Select **Generalized Cross Synthesis** from the **Processing** menu to open the dialog box to select the sound files and enter the parameter values for the cross-synthesis.



We will use the following analysis parameters:



The equivalent command line is as follows:



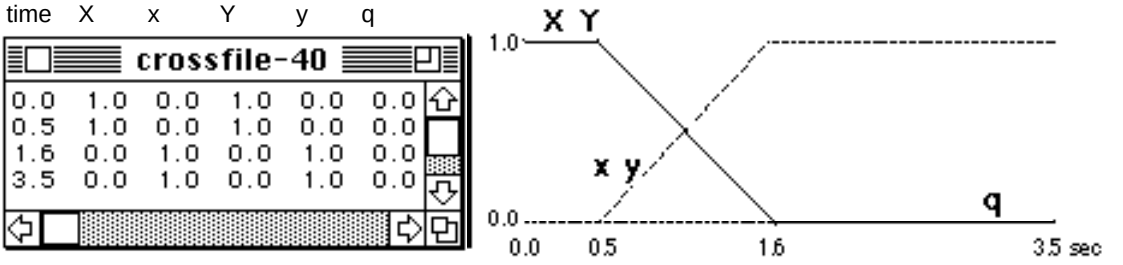
The values entered in the **Analysis Parameters** boxes will create a distortion in the time scale of the secondary sound **tam-tam2**, due to the use of different hop sizes for the two sounds' analysis windows. The resulting sound file will have a duration of 50 seconds, in spite of the fact that the two source sounds **tam-tam2** and **voix-Eh-4** have durations of 1.474 and 6.402 seconds, respectively. If the two hop sizes are set to equal values, the duration of the resulting sound file will be equal to that of the larger input file (**tam-tam2**), whose duration is 6.402 seconds. Although AudioSculpt will synthesize sound in the output sound file as long as it finds a frequency spectrum in one of the input sounds, it will, nonetheless, continue the cross-synthesis using silence if it does not.

In this example the two hop sizes are respectively equal to 500 and 64. As such, the spectra of the sound **voix-Eh-4** are spaced much farther apart than those of **tam-tam2**. The ratio between the two rates is therefore 500/64, or 7.81. In performing the cross-synthesis, AudioSculpt will calculate the sound **Cross-voix-tam2** using the temporal scale of the primary sound file **voix-Eh-4**, which will thereby distort the time scale of the file **tam-tam2** by a factor of 7.81. Because **tam-tam2** contains more spectra than does **voix-Eh-4**, the cross will continue until the end of the file **tam-tam2**. The resulting duration can be calculated by multiplying the size of the sound file **tam-tam2** (6.402 seconds) by this dilation factor, which gives us a total of 50 seconds.

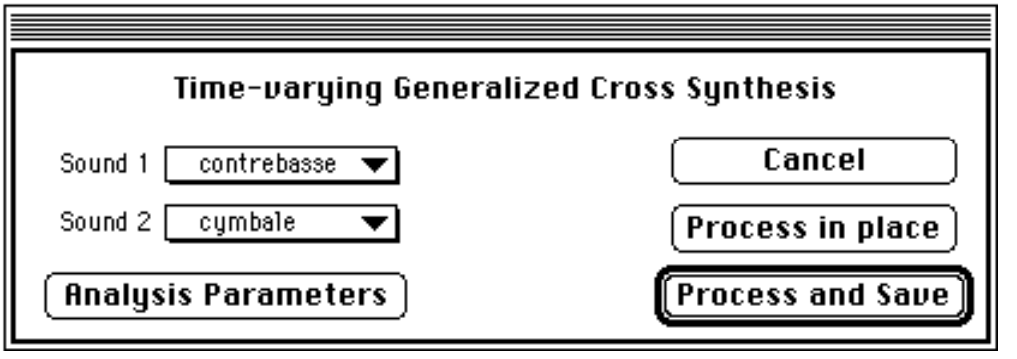
The next two examples show how large modifications can be made to a resulting cross-synthesis by slightly varying the parameters used for the cross. A very low bass clarinet sound will be crossed with a cymbal. The analysis parameters for both examples **7.1** and **7.2** are identical, but different dynamic crossing files will be used.

Example 6.1 index 32, 33 and 34

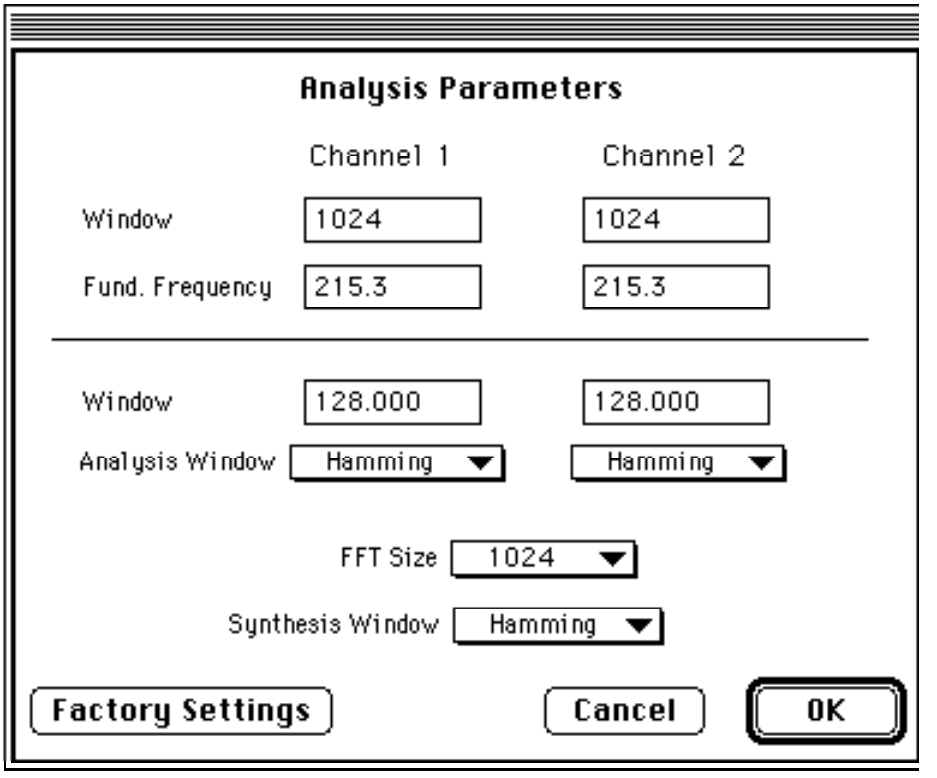
The following parameter file will be used to control the cross:



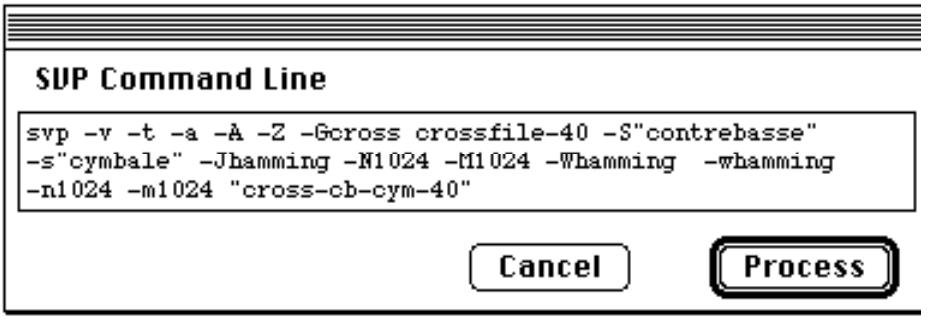
This file controls a very simple cross. The output file begins with the primary sound alone, followed by a rapid transition to the secondary sound beginning at 0.5 seconds, and, from 1.6 seconds, the secondary sound is heard alone. With the crossing file's window active, select **Generalized Cross Synthesis** from the **Processing** menu and select the sound files in the dialog box.



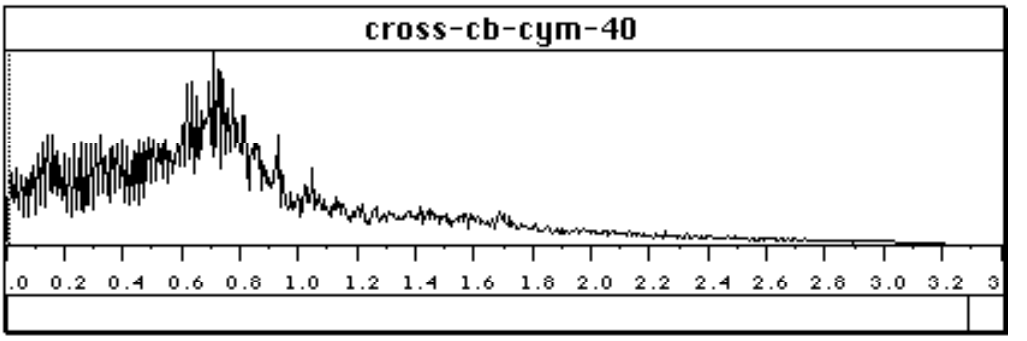
Here are the analysis parameters we will use…



…and the equivalent command line for the whole cross-synthesis:

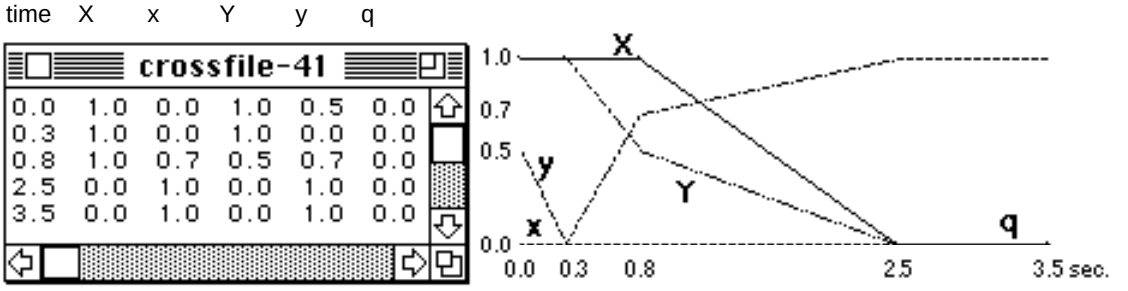


The resulting sound has the following amplitude envelope:

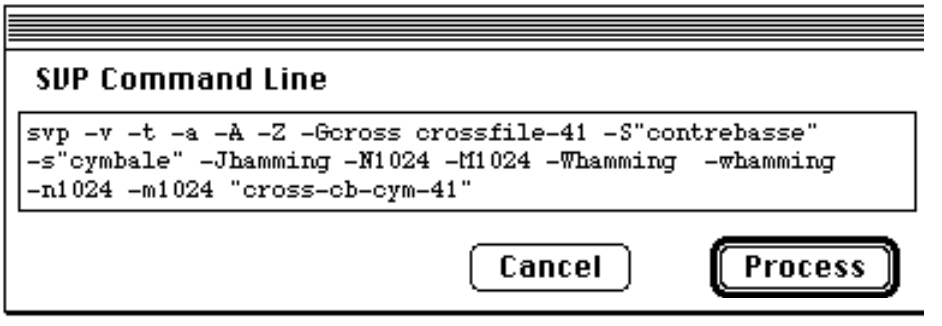


Example 6.2 index 35

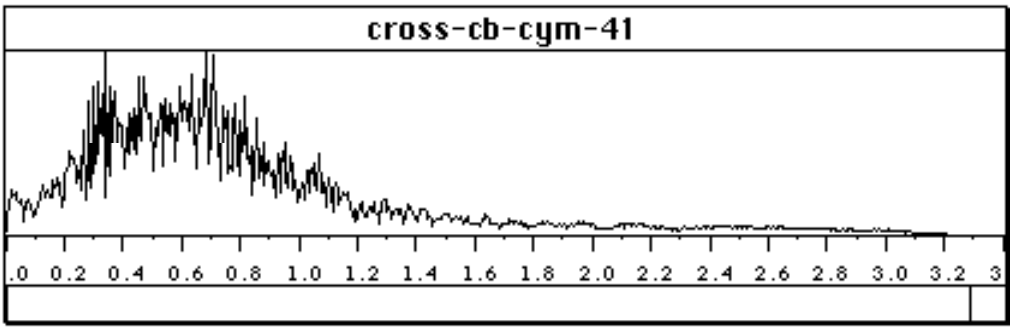
The modifications made to the dynamic crossing file are few, but they nonetheless produce noticeable differences. Here is the modified crossing file and its graphic representation:



The command line equivalent is as follows:



This is the amplitude/time envelope of the resulting sound:



The resulting sound is a hybrid which moves and gets transformed over time. In particular, you can follow the glissando produced by the parameter **y** from the beginning of the sound to 0.8 seconds. Around 2.5 seconds the resonance of the cymbal becomes more apparent.

Example 7 index 36, 37 and 38

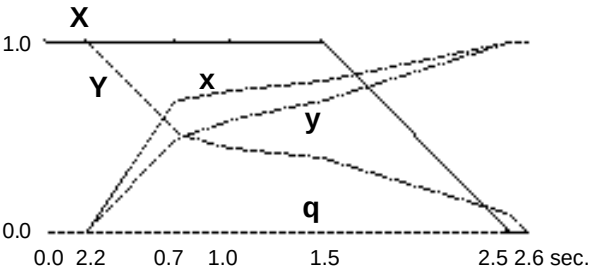
Here, a short double-attack on the tabla is crossed with a tam-tam with long resonance. As the attack is generally the most characteristic part of an instrumental sound, its modification will render the sound difficult to recognize. During the attack, a sound is unstable since the spectrum is made up of transients which stabilize only after the attack. This example modifies the transients in the tam-tam's attack. This type of cross may be affected on any combination of instruments and can be applied to the idea of micro orchestration of complex sounds.

The parameter file used for the cross is carefully calculated to modify the resonance of the tam-tam.

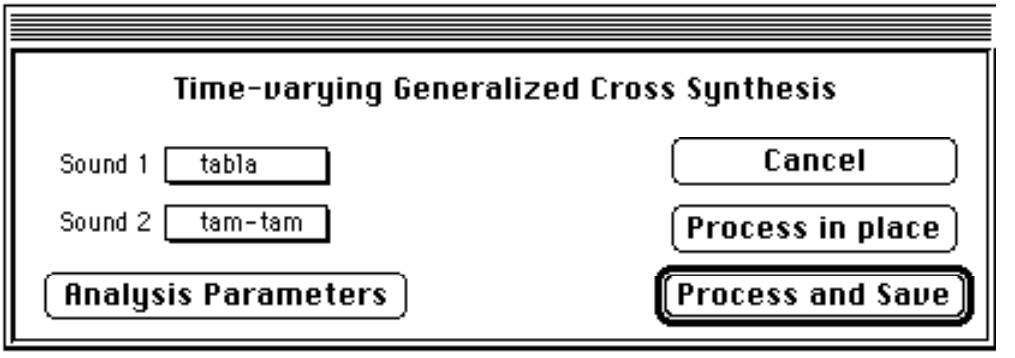
time X x Y y q

crossfile-74

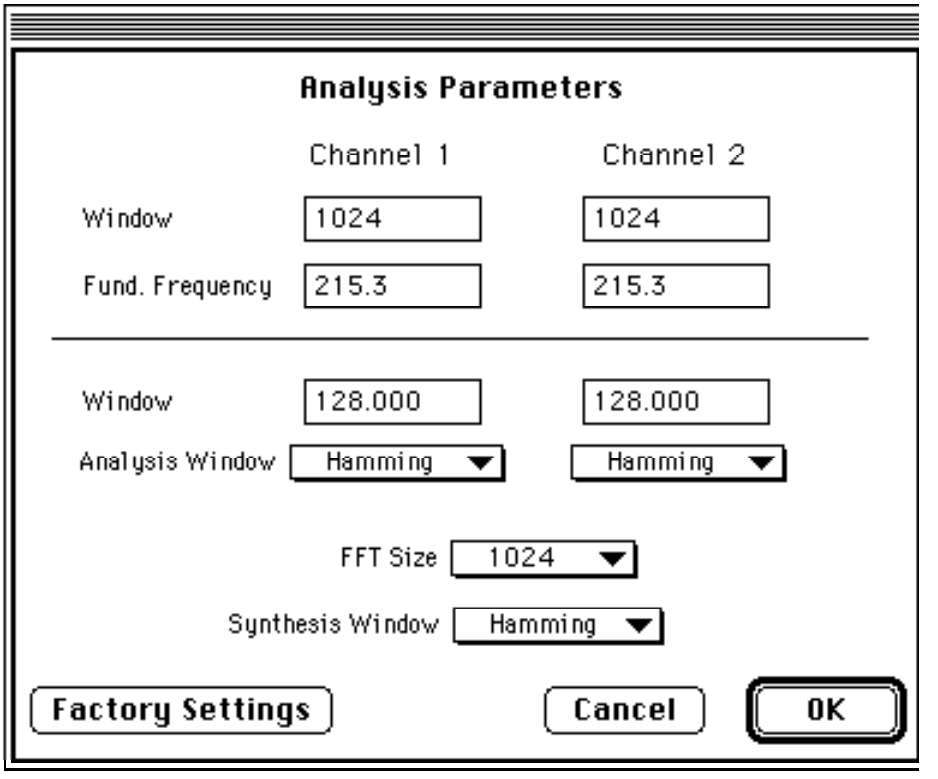
0.0	1.0	0.0	1.0	0.0	0.0
0.22	1.0	0.0	1.0	0.0	0.0
0.7	1.0	0.7	0.52	0.5	0.0
1.0	1.0	0.75	0.45	0.6	0.0
1.5	1.0	0.8	0.4	0.7	0.0
2.5	0.0	1.0	0.1	1.0	0.0
2.6	0.0	1.0	0.0	1.0	0.0



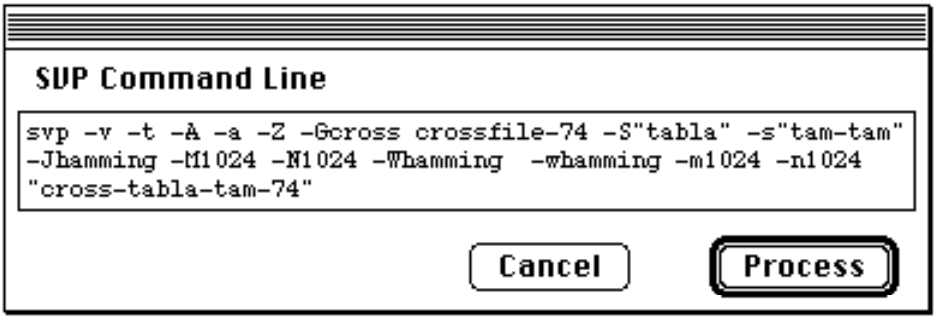
As before, two sound files are selected,



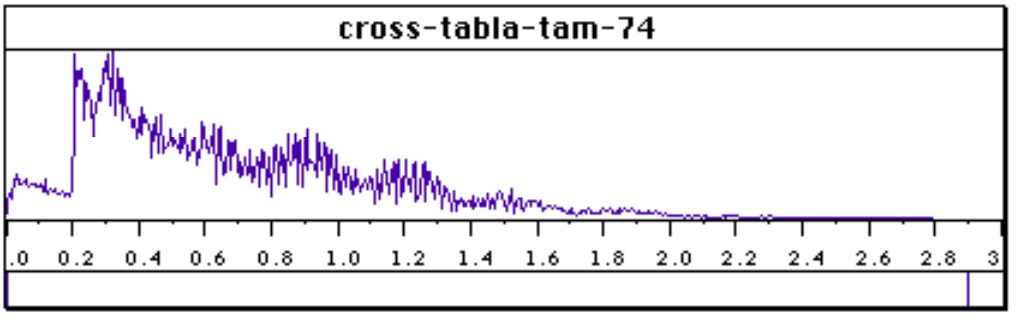
and the following analysis parameters are used:



The equivalent command line:



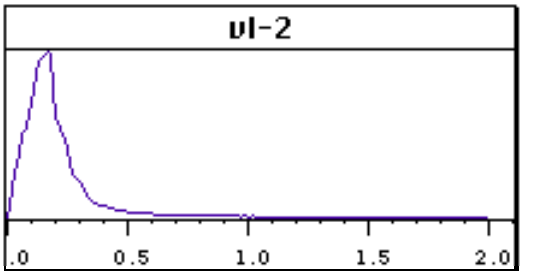
and the resulting sound:

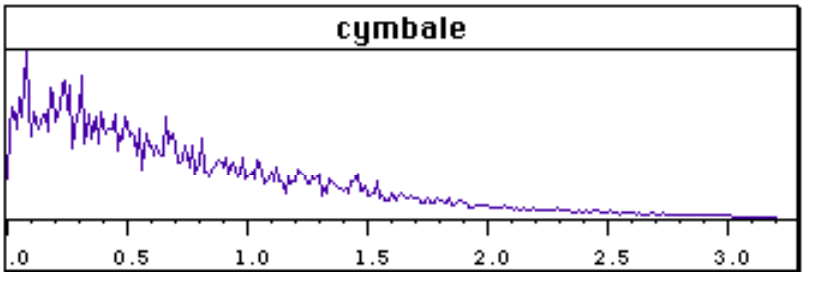


The following example describes one possible method for a better realization of a cross-synthesis.

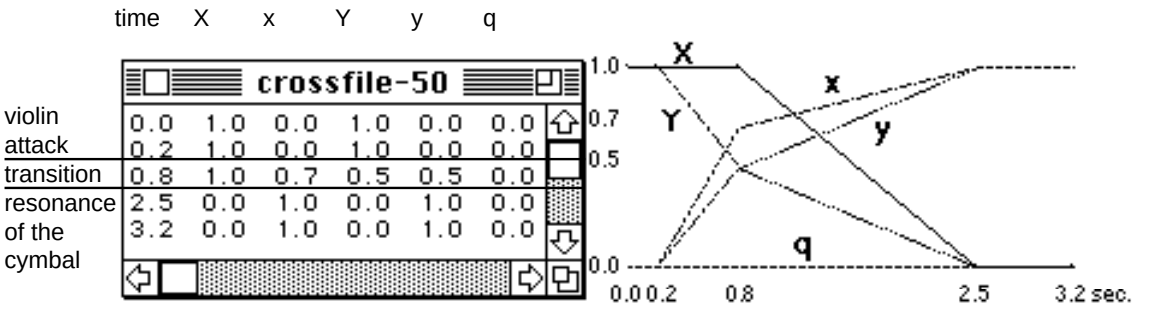
Example 8.1 index 39, 40 and 41

A violin attack is crossed with a cymbal.

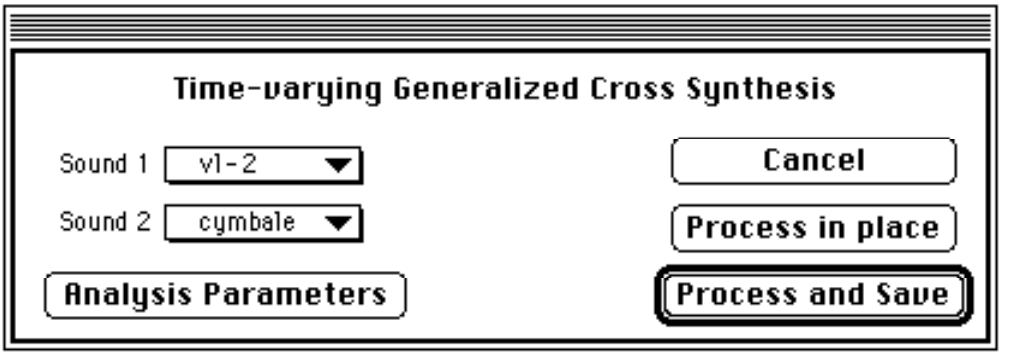




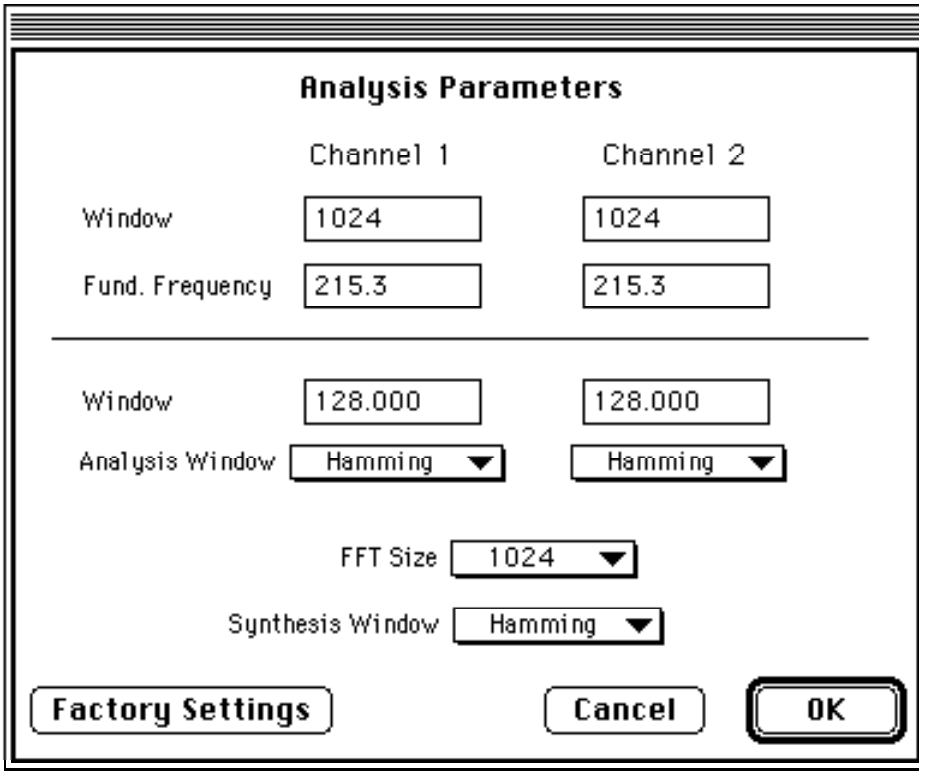
We will first try a cross using the following parameter file:



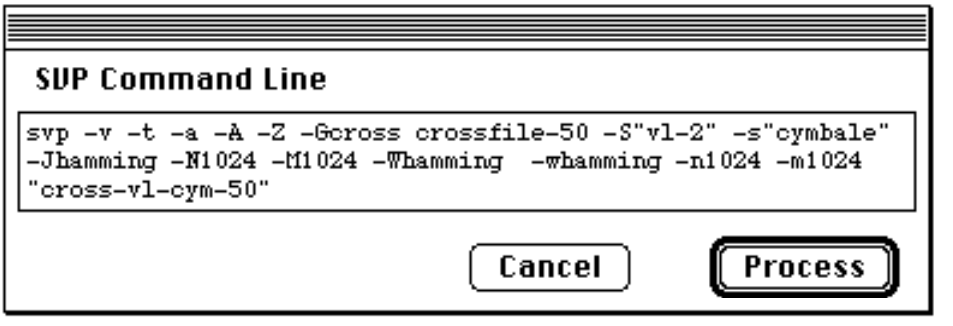
The following files will be selected in the same manner as before…



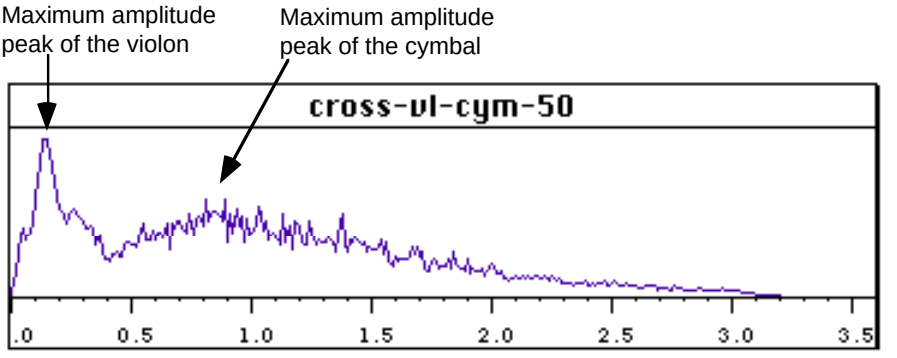
… using the following analysis parameters:



The equivalent command line:

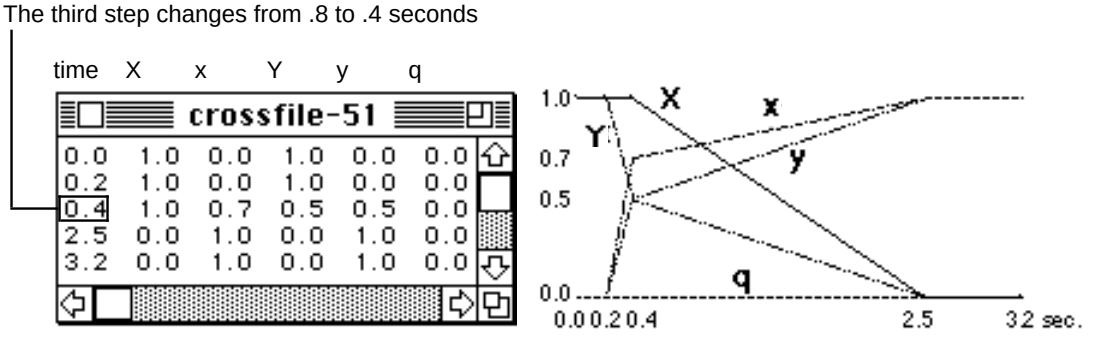


and the resulting sound:

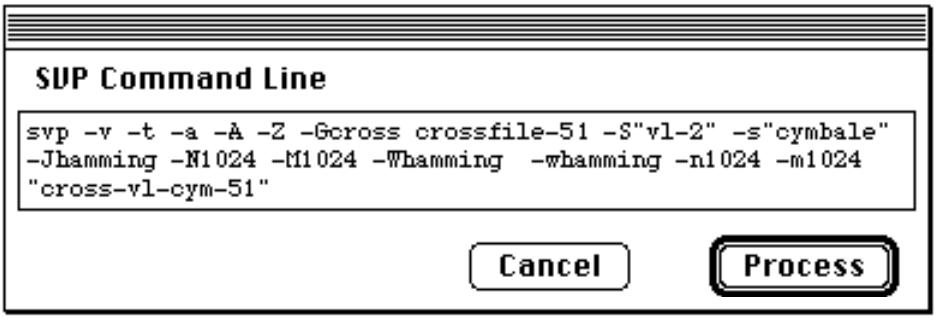


Example 8.2 index 42

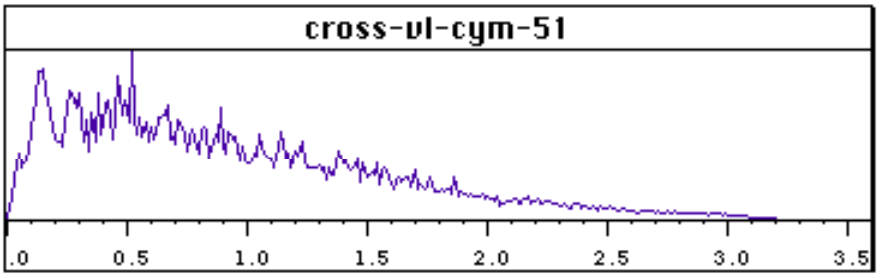
The maximum amplitudes of the two instruments are too distant from each other in the resultant sound, provoking a discontinuity (see above similar figure in example 9.1). In order to better realize this cross, it is necessary to bring these peaks closer together by modifying the point in time where one finds the cymbal's peak in the envelope of the resultant sound.



The following command line equivalent may also be used:



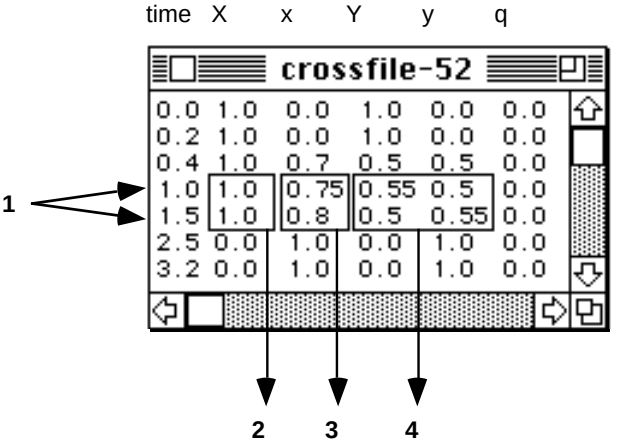
The resulting sound:



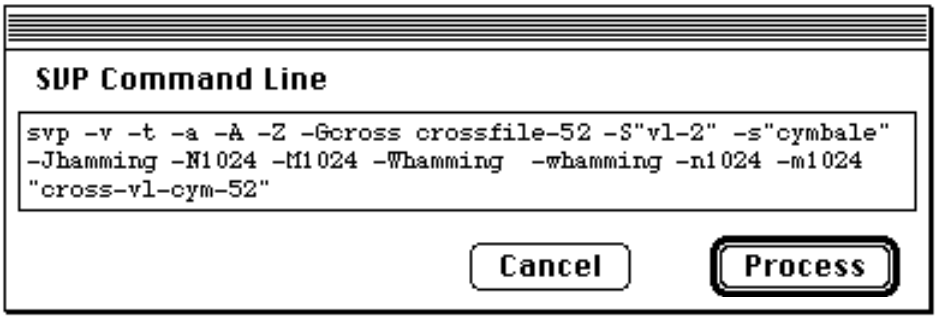
The continuity between the two sounds is evident in the resulting sound both upon listening to it and in its graphic representation (in the form of an amplitude envelope) as well.

Example 8.3 index 43

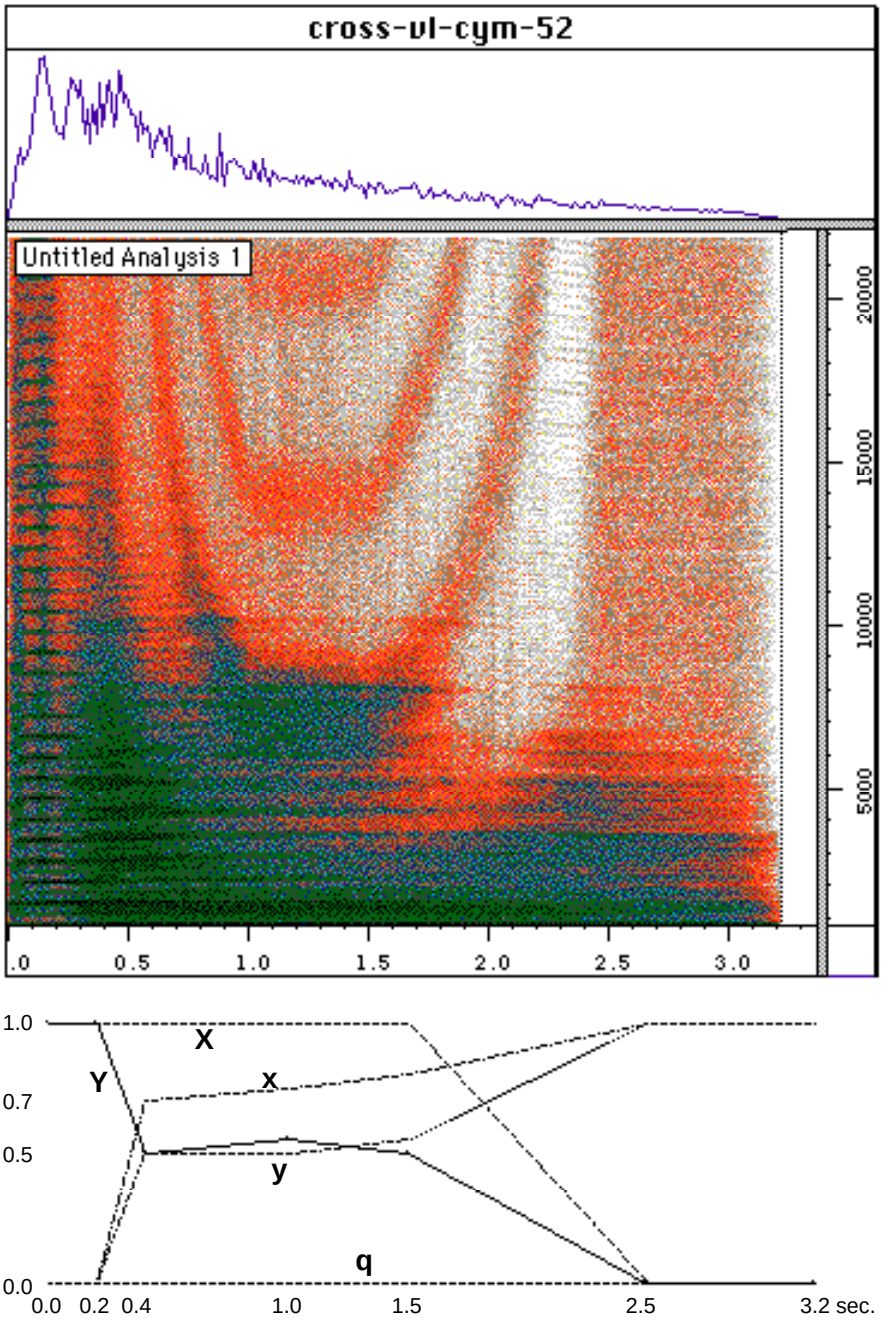
Here, two lines have been added to the crossing file (1) to further improve the continuity between the two sounds. The violin is held at maximum until 1.5 seconds (2) while the amplitude of the cymbal continues to increase (3). To make the cymbal's resonance less evident,envelope a slight oscillation is added to the parameters which control the sounds' phases (4).



The command line equivalent:



Here is the resulting sound represented by both an amplitude/time envelope and a sonogram, with a graphic representation of the crossing parameter file **crossfile-52**, below.



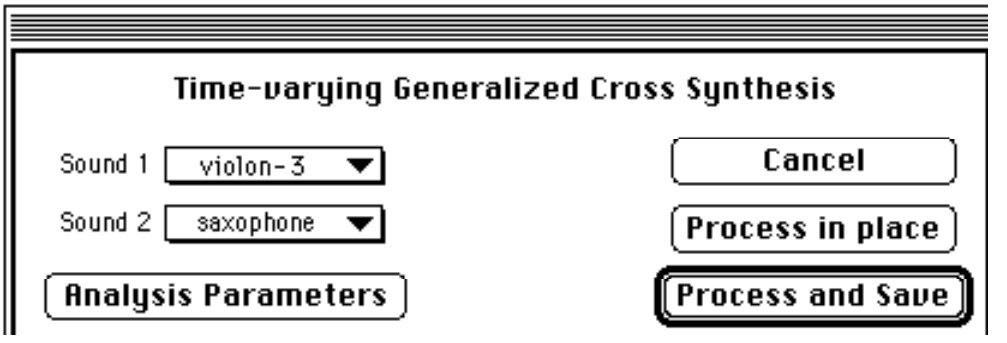
Example 9.1 index 44, 45 and 46

In this example a violin is crossed with a saxophone. The crossing parameter file used produces a rather strange result; to better understand the reason behind this, one may compare this with examples 9.2, 9.3 and 9.4. The following dynamic parameter file is used:

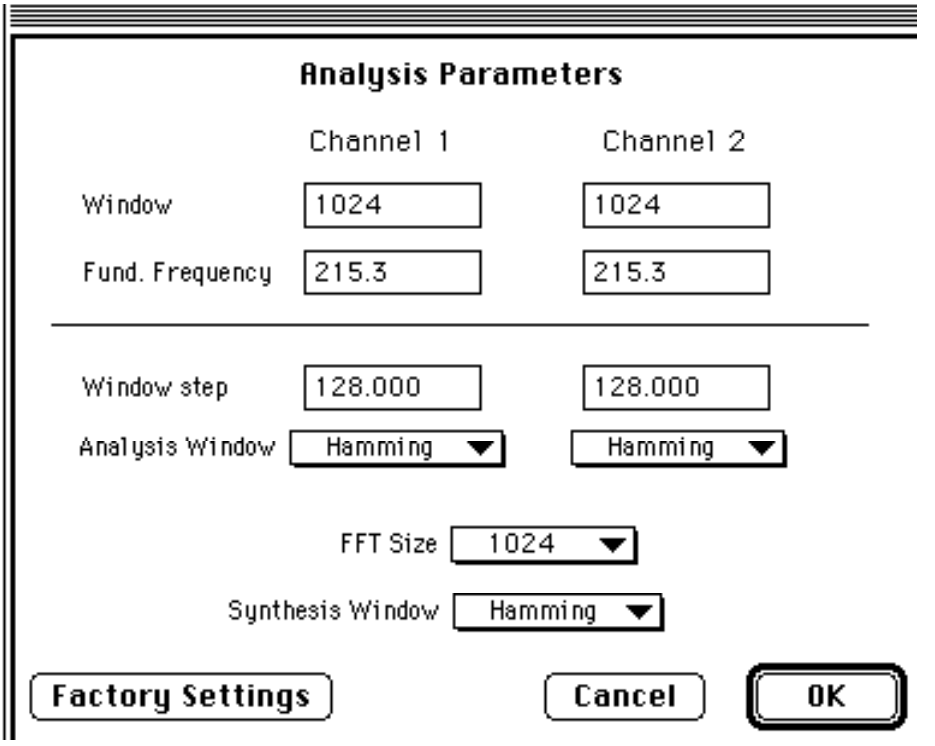
crossfile-60

time	X	x	Y	y	q
0.0	1.0	0.0	1.0	0.0	0.0
0.88	1.0	0.31	1.0	0.07	0.0
1.44	0.76	0.51	1.0	0.12	0.0
1.84	0.89	0.66	1.0	0.15	0.0
2.24	0.76	0.79	1.0	0.18	0.0
2.32	0.73	0.76	1.0	0.19	0.0
2.64	0.85	0.66	1.0	0.22	0.0
2.72	0.88	0.64	0.98	0.22	0.0
2.88	0.94	0.58	0.94	0.24	0.16
2.96	0.97	0.6	0.92	0.24	0.25
3.28	0.74	0.69	0.84	0.27	0.59
3.68	0.46	0.79	0.74	0.35	0.8
4.0	0.4	0.8	0.75	0.4	0.75
4.1	0.3	0.9	0.7	0.5	0.7

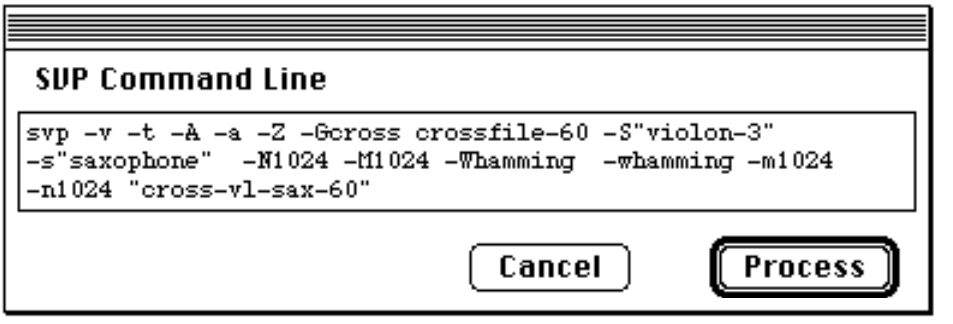
The files are selected in the dialog box, as shown,



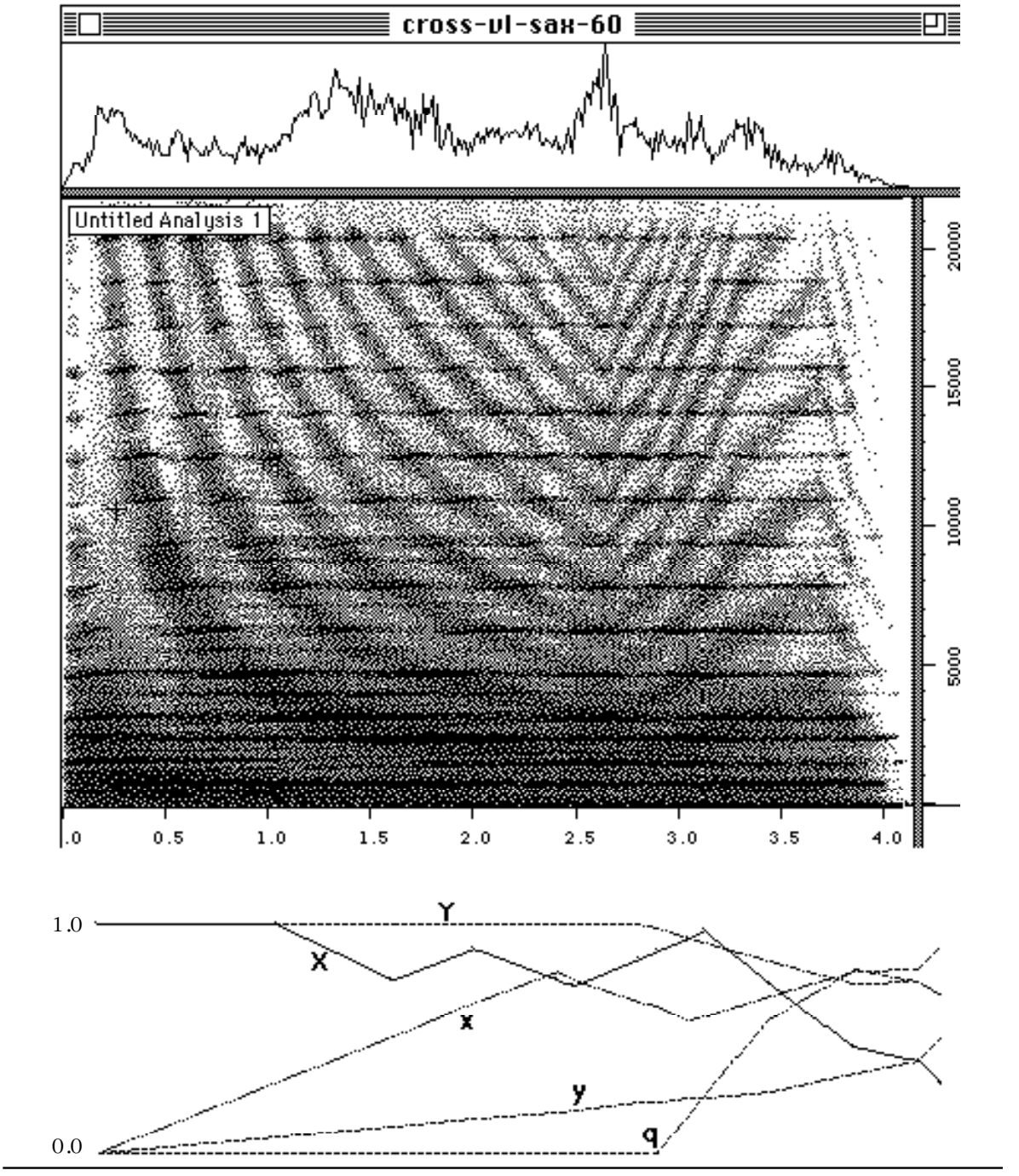
using the following analysis parameters:



The command line equivalent:



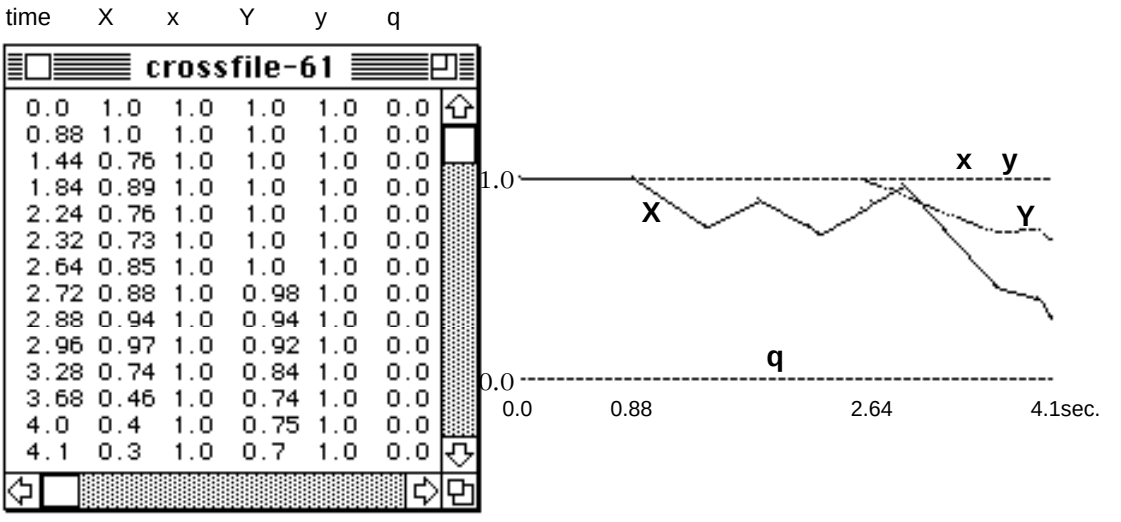
The resulting sound:



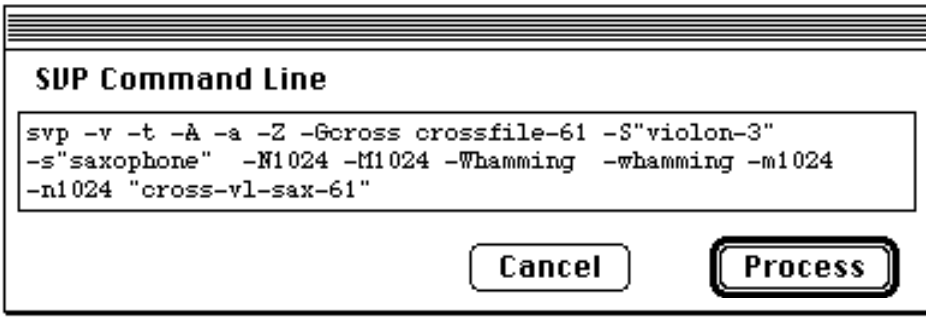
By carefully observing the sonogram, one notices that the glissando heard in the sound does not correspond to its spectral representation — in fact, it seems to move in contrary motion to it. This is a typical example of a paradoxical sound.

Example 9.2 index 47

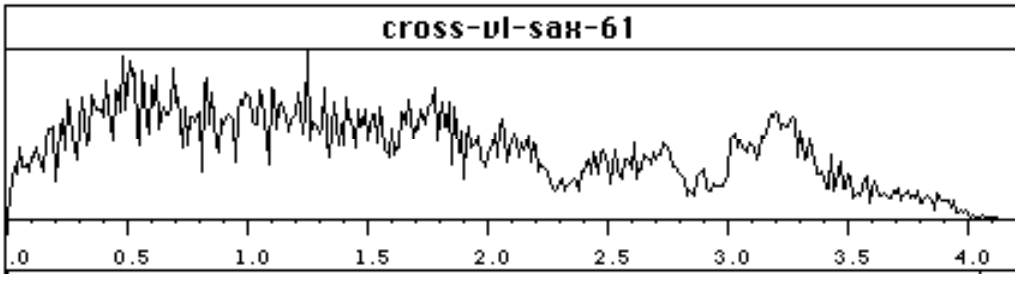
Here the coefficients relating to the primary sound have been kept, but those relating to the secondary sound are constant and equal to 1 for **x** and **y** and 0 for **q**.



This is the equivalent command line…

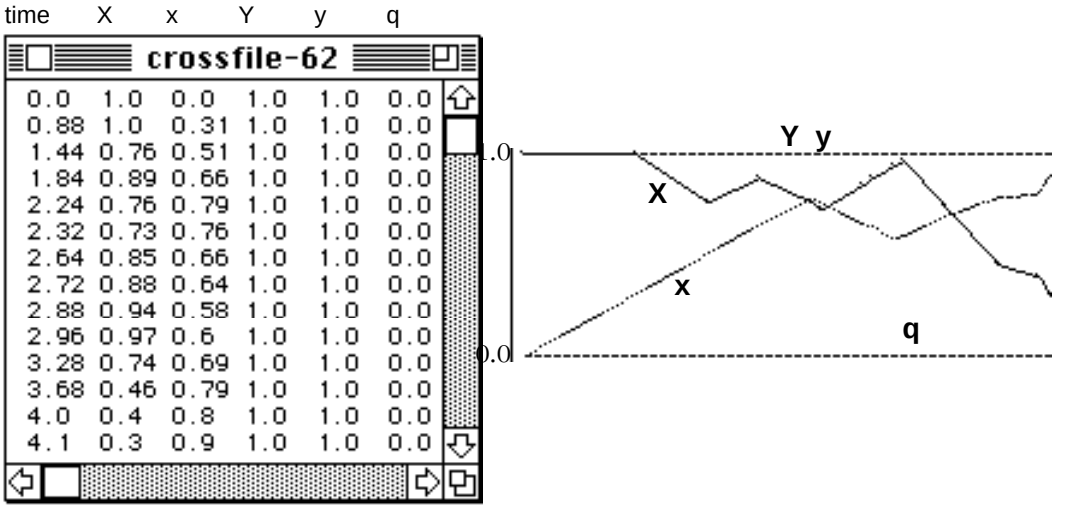


… and the resulting sound:

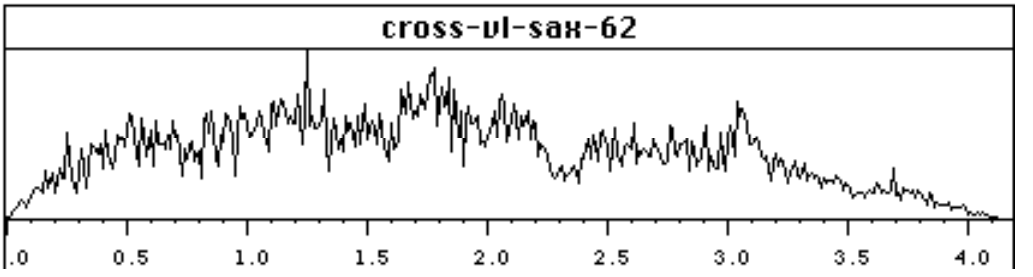


Example 9.3 index 48

Here, the amplitude coefficients (**X** and **x**) for the two sounds have been preserved (with respect to Example 10.1), but coefficients **Y** and **y** have a constant value of 1.

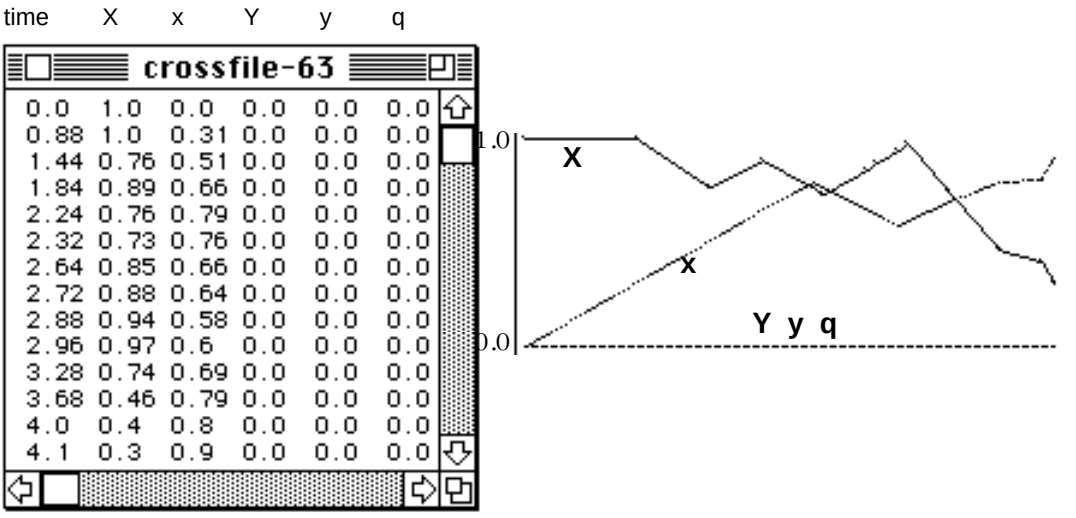


The resulting sound:

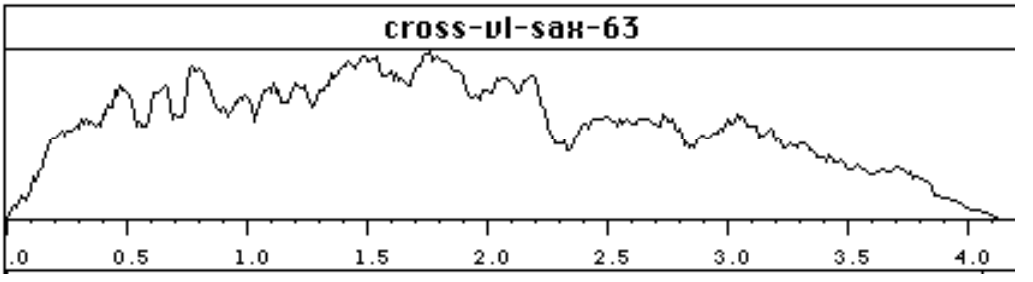


Example 9.4 index 49

For the last example in this series, the amplitude coefficients (**X** and **x**) are conserved as before, but the three remaining coefficients **Y**, **y** and **q** are constant and equal to 0.



This combination creates a very metallic sounding output.



Example 10 index 50, 51 and 52

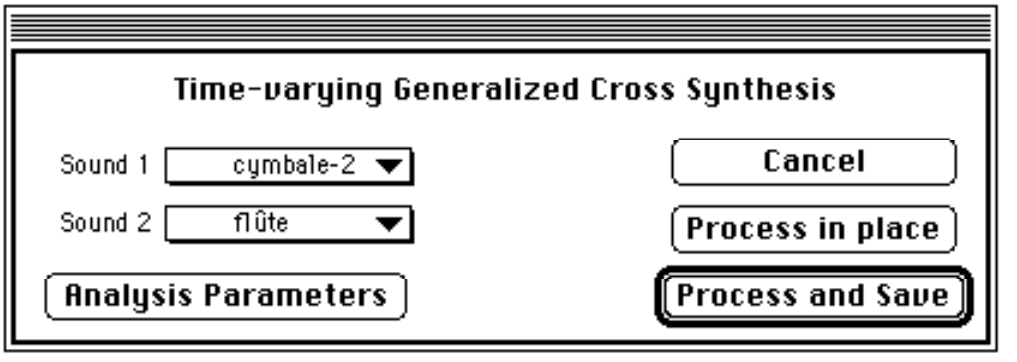
A cymbal is crossed with a flute using a rather complex parameter file:

time X Y x y q

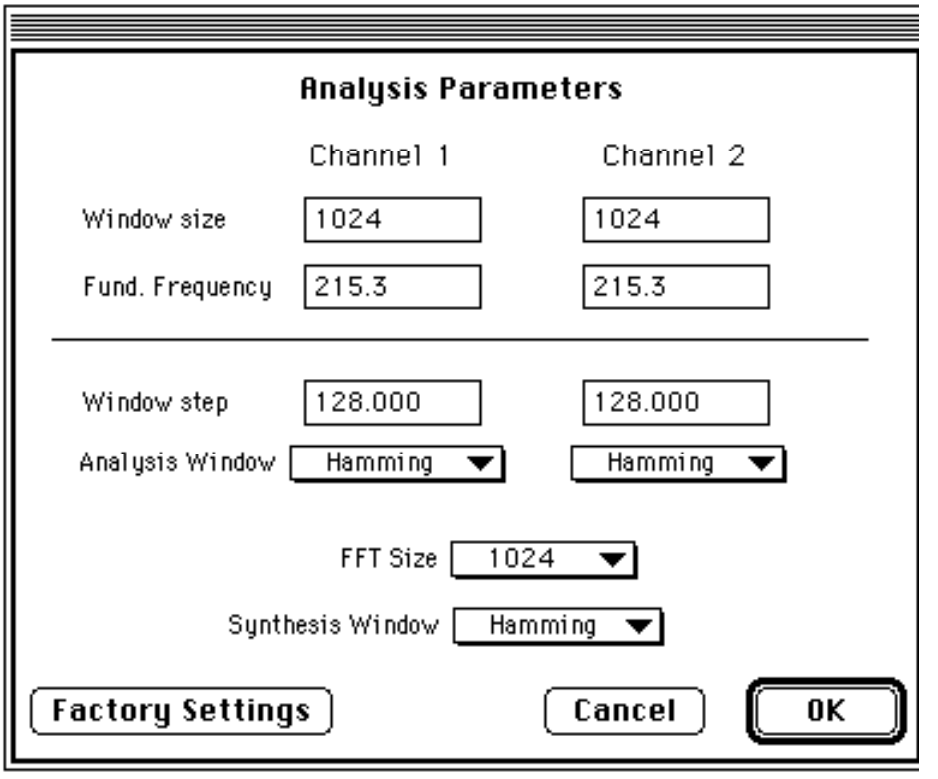
crossfile-32

0.0	1.0	0.0	1.0	0.0	0.0
0.09	1.0	0.0	0.92	0.15	0.83
0.19	1.0	0.0	0.85	0.29	0.76
0.38	1.0	0.3	0.72	0.59	0.61
0.59	0.78	0.64	0.56	0.92	0.44
0.66	0.71	0.76	0.51	0.77	0.39
0.75	0.61	0.91	0.62	0.56	0.31
0.92	0.44	0.96	0.8	0.2	0.18
0.94	0.42	0.96	0.75	0.15	0.17
1.08	0.28	1.0	0.48	0.35	0.06
1.15	0.2	1.0	0.34	0.46	0.0
1.18	0.18	1.0	0.3	0.49	0.07
1.25	0.11	1.0	0.16	0.52	0.27
1.32	0.03	1.0	0.15	0.55	0.48
1.34	0.0	1.0	0.14	0.56	0.44
1.53	0.0	1.0	0.11	0.64	0.16
1.74	0.0	1.0	0.07	0.74	0.94
1.88	0.0	1.0	0.04	0.8	0.46
1.93	0.0	1.0	0.04	0.8	0.3
1.97	0.0	0.89	0.05	0.79	0.14
2.23	0.0	0.28	0.08	0.77	0.32
2.35	0.0	0.0	0.09	0.76	0.33

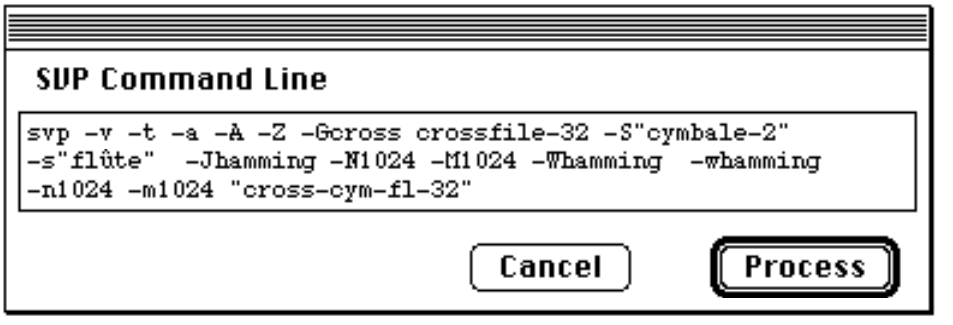
Here is the selection of files in the generalized cross-synthesis dialog box:



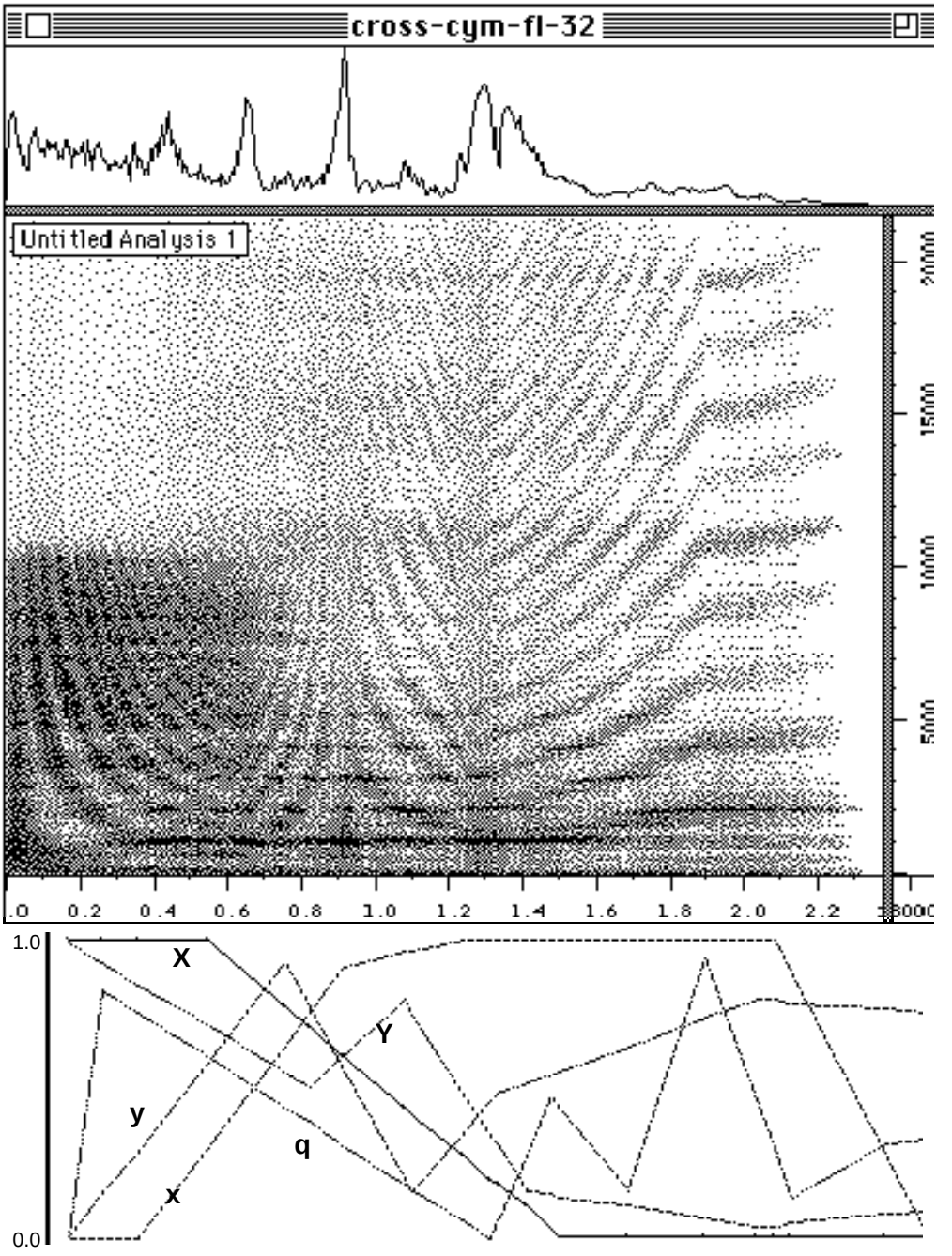
The analysis parameters:



The command line equivalent:



The resulting sound:53



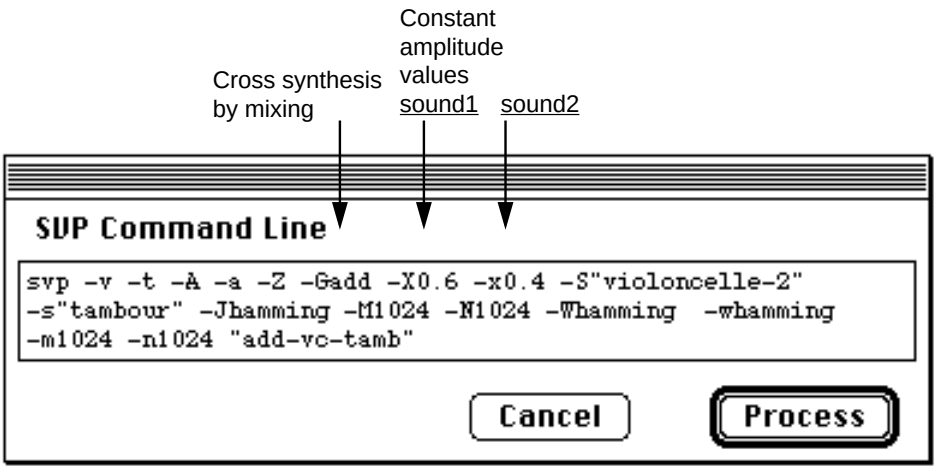
The parameters **Y** and **y**, as they are defined in the file, impose rather large variations in pitch. The most evident to the ear is probably the glissando produced by parameter **Y** at the beginning of the sound.

5 Cross-Synthesis by Mixing

This type of cross-synthesis is analogous to the mixing of two sounds. The coefficients **X** and **x** play the role of faders and allow control of the weight of each sound in the mix. This type of cross-synthesis is currently only available by using a command line, using the option -Gadd.

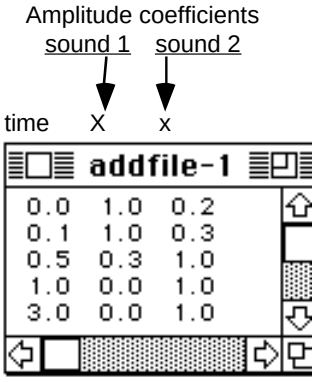
To begin processing, open the command line dialog box by selecting **Use Command Line** from the **Processing** menu.

If you are using constant parameter values, you must type these values just after the options -X and -x (respectively, the two options are relative to the primary and secondary input sounds).

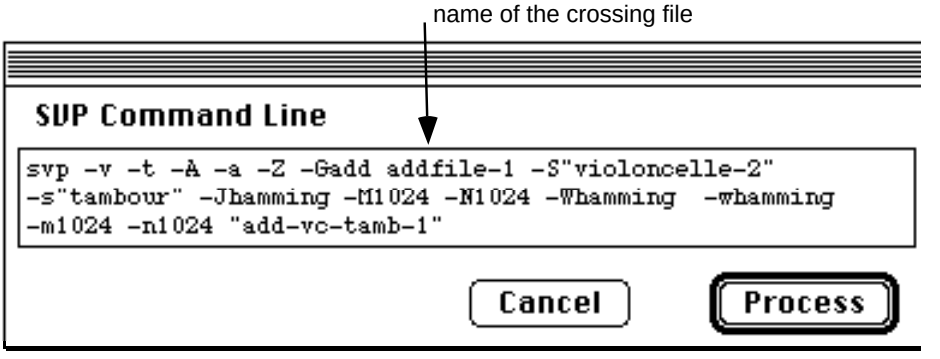


Example 11 index 53, 54 and 55

If you use dynamic crossing parameters (those which vary over time), you should create and save a text file containing these parameter values.



The name of this file must be given just after the option **-Gadd** in the command line.



Do not forget to correctly specify the folder where AudioSculpt will look for both the parameter files and sound files you will be using (by selecting **Set Default Folders** from the **File** menu) before initiating the processing.

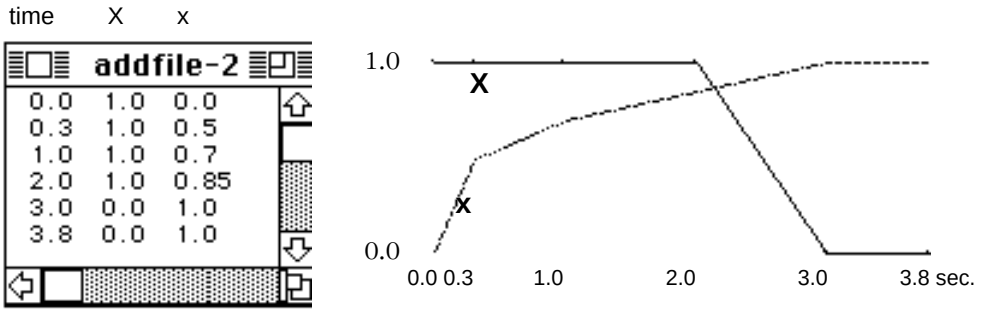
Here, our resulting sound has the attack of a cello with the resonance of a drum. The crossing file creates a very rapid transition.

Graphical representation of file `addfile-1`

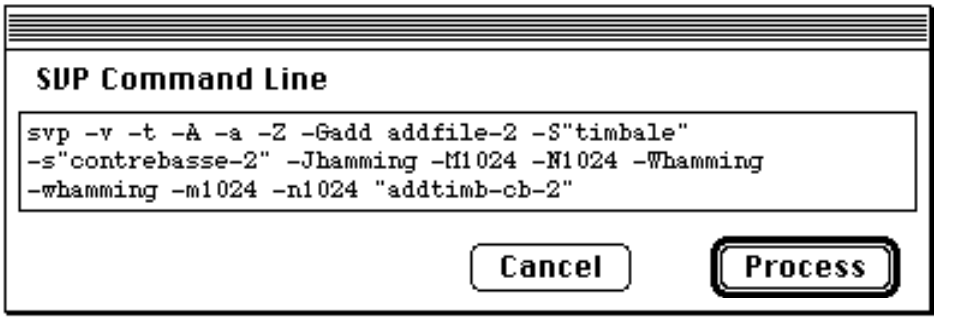
Example 12 index 56, 57 and 58

In the last example of this series, a timpano is crossed with a contrabass. The parameter file used for the cross creates a transition which is much slower than in the previous example.

The following is the dynamic parameter file used for the cross:



This is the command line which will be used:



Cross-synthesis by mixing (add mode) is a useful tool not only for the mixing of two sounds, but, more importantly, for the creation of micro-mixes. The two preceding examples illustrate the precise control of amplitude one has over the mixing of two sounds when using this type of cross-synthesis.

6 Appendix

This appendix includes those examples which “cross the boundaries” of the diverse types of cross-syntheses so far explained in this tutorial. These examples are presented to provide a springboard for ideas for those who are searching for new fields of audio processing in which to explore.

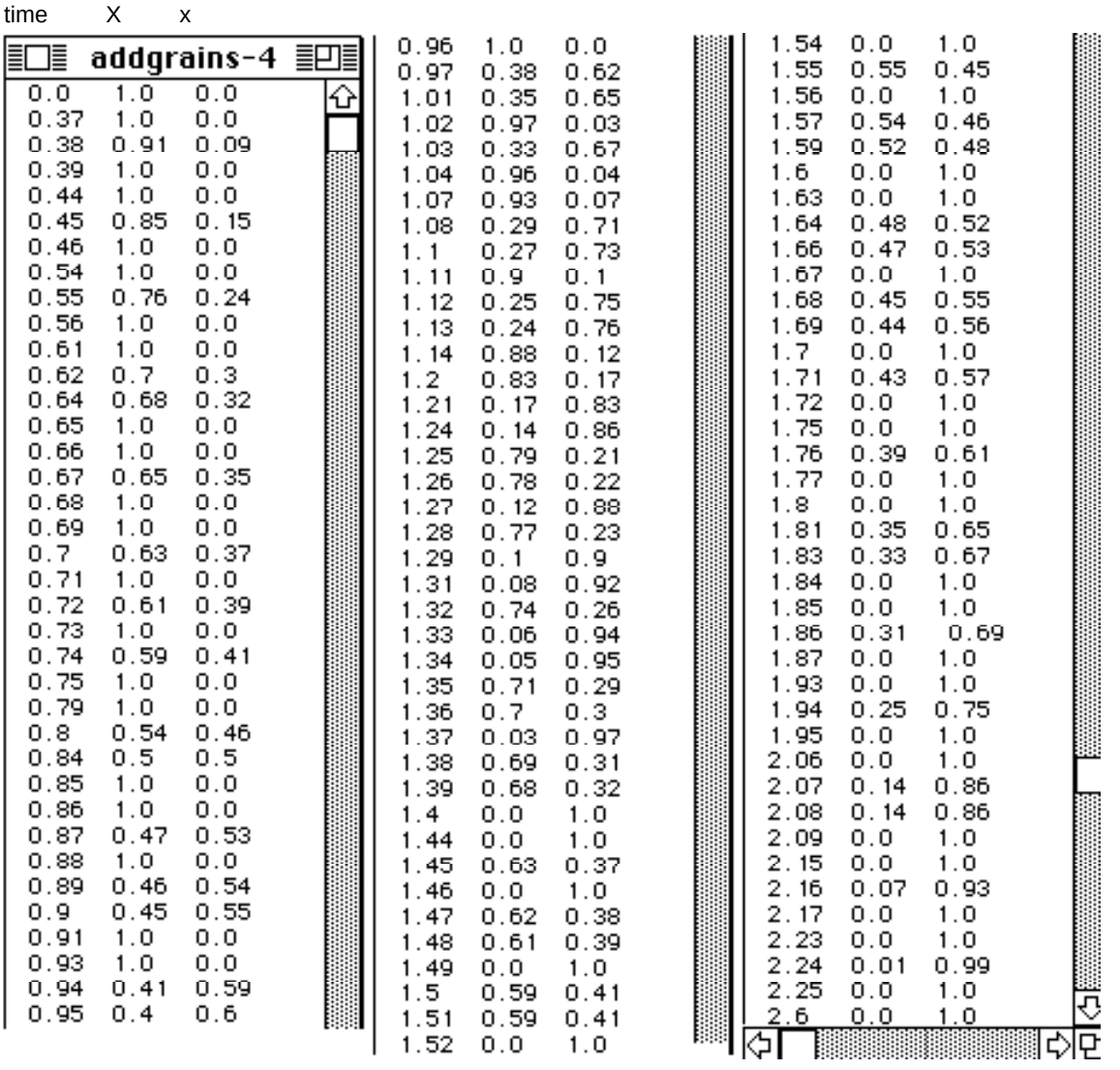
The examples were realized with the help of the program PatchWork[1] (also developed at Ircam), which served to create complex dynamic parameter files for unconventional control of cross-synthesis, transposition, and time expansion/compression.

1. In PatchWork, the selected time resolution (shown in the time column of the parameter file) is not necessarily in accordance with the step size used in AudioSculpt. When it is smaller, AudioSculpt reproduces only the variations which match the step size. the finer ones are skipped.

Example 13. 1 index 59, 60 and 61

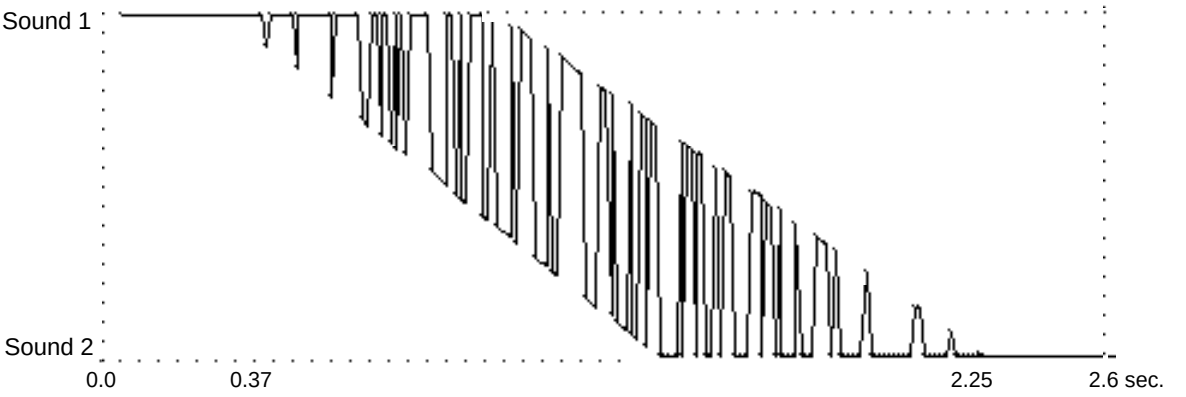
This example uses cross-synthesis by mixing (add mode) in which the parameter file contains variables for **X** and **x** whose sum is always equal to 1. This is equivalent to using two faders to control the output level of the two sound files. The amplitude values in the parameter file oscillate very rapidly in an irregular fashion, which gives the global impression of a progressive transition between a trumpet and a flute accompanied by a rapid granulation effect.

Here is the crossing parameter file:

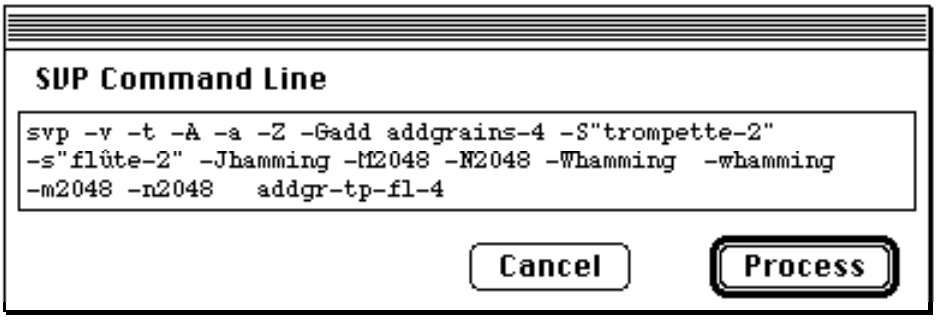


To create this long file, a patch was constructed in PatchWork, using a stochastic algorithm to generate the oscillations while controlling the percentage of predominance of one sound or the other, as well as the interval of oscillations.

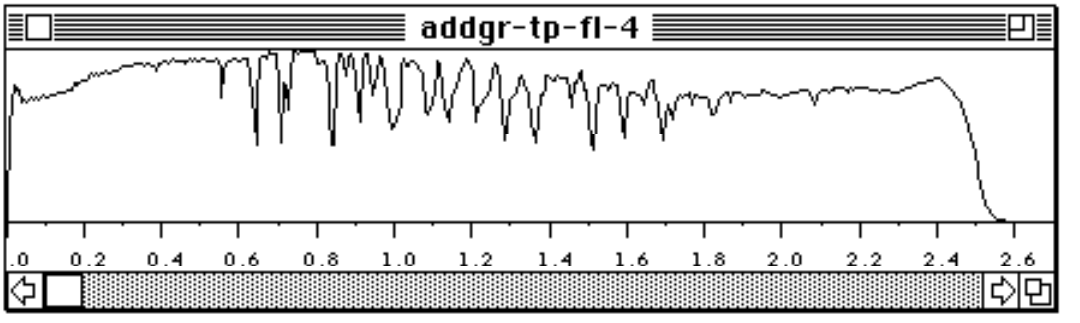
Here is the graphic representation of the file **addgrains-4**:



The command line used:



The amplitude/time envelope of the resulting sound:

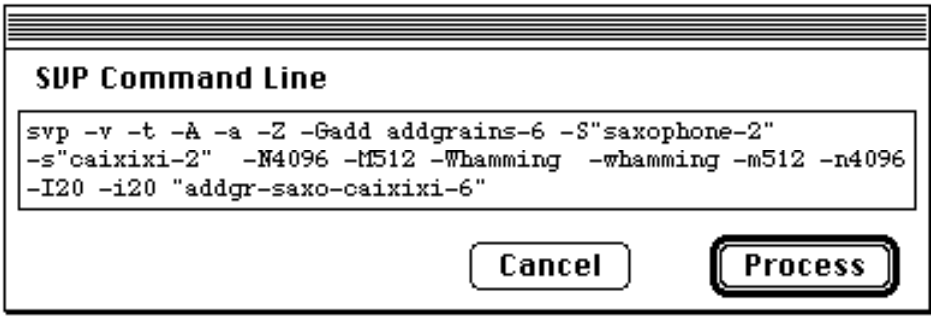


The oscillation between the two sounds produces different types of “grains” during the transition. This type of granulation has a masking effect during the transition to the flute sound. The idea to introduce elements which capture the auditory attention just at the moment of transition is just one of the many possible means leading to inexhaustible creative possibilities.

Example 13.2 index 62, 63 and 64

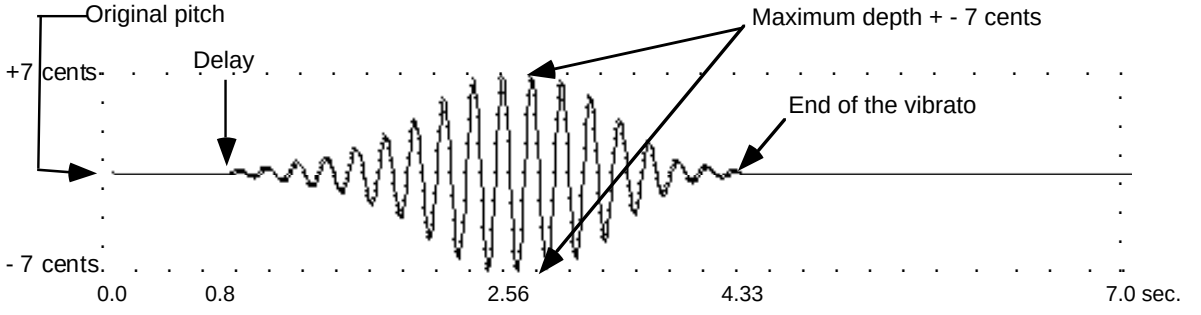
With the same type of cross synthesis, and using a crossing parameter file similar to that in the previous example, a saxophone melody is crossed with the sound of a caixixi. The granulation produced by this parameter file softens the rugged character of the caixixi.

The parameter file **addgrains-6** resembles that of the preceding example, and will not be displayed here due to its length. Here is the command line:



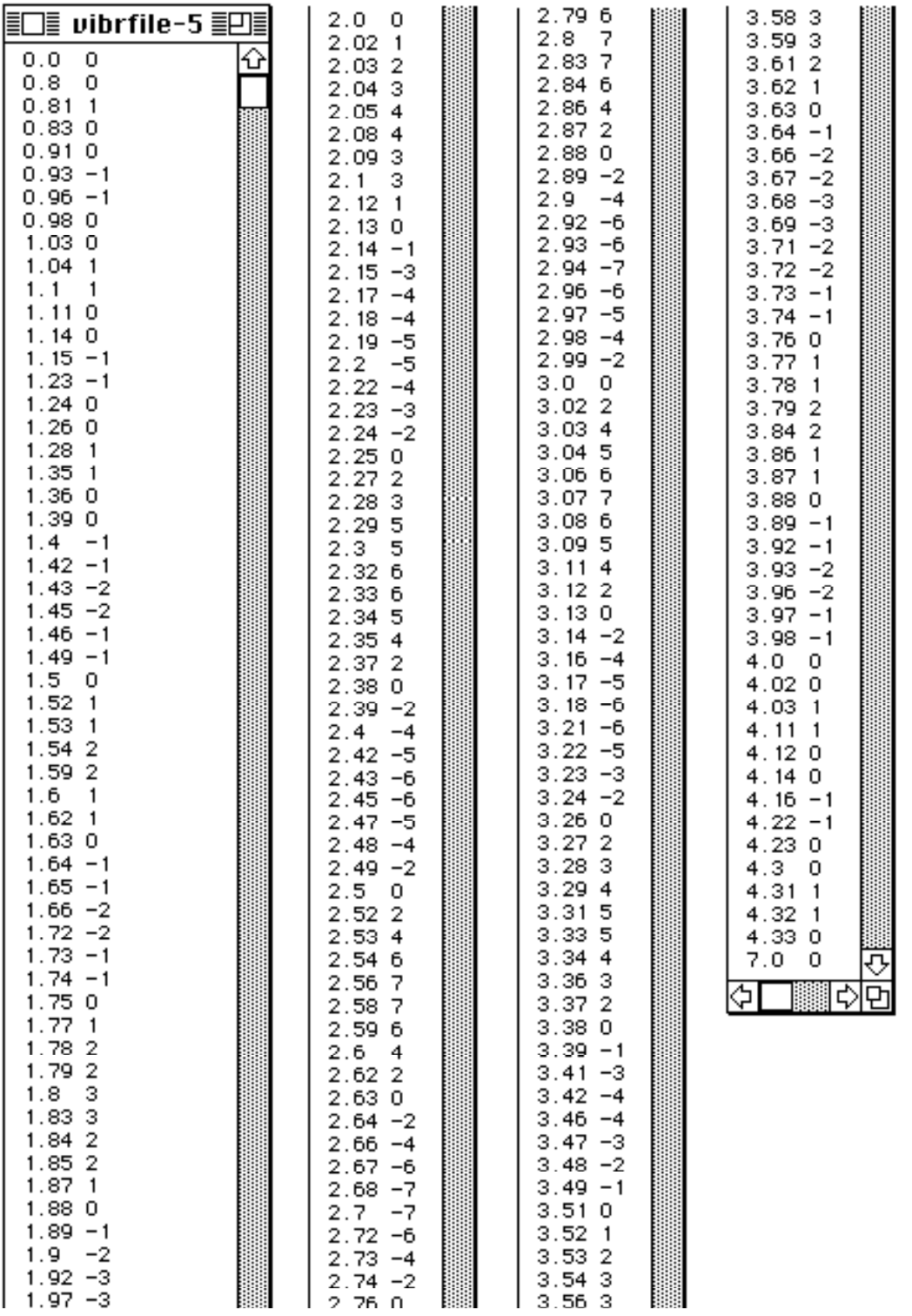
Example 14.1 index 65 and 66

This example uses the option -trans to generate a vibrato for the sound of the cello. Again, the dynamic parameter file which controls the transposition was created within PatchWork by using a patch to determine the frequency, amplitude envelope, waveform and delay leading to any kind of vibrato.

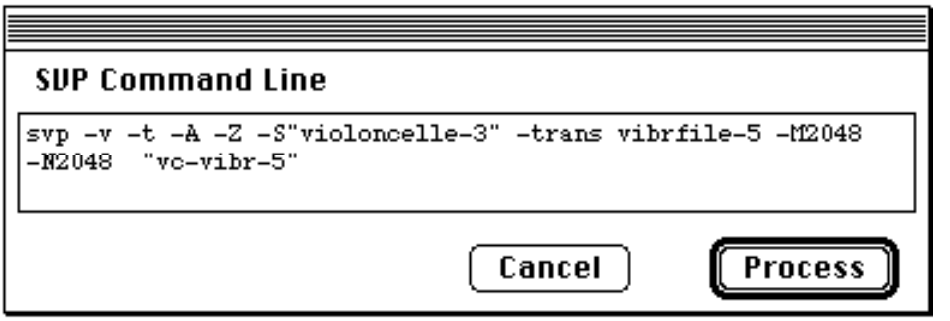


The dynamic transposition file used to produce the vibrato:

time cents

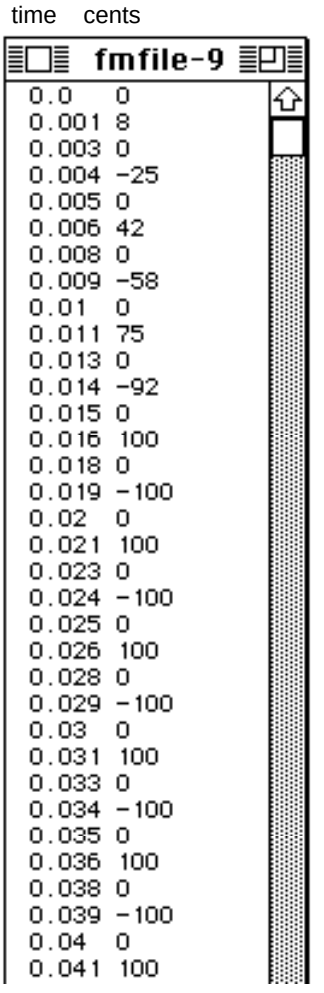

vibrfile-5
0.0 0
0.8 0
0.81 1
0.83 0
0.91 0
0.93 -1
0.96 -1
0.98 0
1.03 0
1.04 1
1.1 1
1.11 0
1.14 0
1.15 -1
1.23 -1
1.24 0
1.26 0
1.28 1
1.35 1
1.36 0
1.39 0
1.4 -1
1.42 -1
1.43 -2
1.45 -2
1.46 -1
1.49 -1
1.5 0
1.52 1
1.53 1
1.54 2
1.59 2
1.6 1
1.62 1
1.63 0
1.64 -1
1.65 -1
1.66 -2
1.72 -2
1.73 -1
1.74 -1
1.75 0
1.77 1
1.78 2
1.79 2
1.8 3
1.83 3
1.84 2
1.85 2
1.87 1
1.88 0
1.89 -1
1.9 -2
1.92 -3
1.97 -3
2.0 0
2.02 1
2.03 2
2.04 3
2.05 4
2.08 4
2.09 3
2.1 3
2.12 1
2.13 0
2.14 -1
2.15 -3
2.17 -4
2.18 -4
2.19 -5
2.2 -5
2.22 -4
2.23 -3
2.24 -2
2.25 0
2.27 2
2.28 3
2.29 5
2.3 5
2.32 6
2.33 6
2.34 5
2.35 4
2.37 2
2.38 0
2.39 -2
2.4 -4
2.42 -5
2.43 -6
2.45 -6
2.47 -5
2.48 -4
2.49 -2
2.5 0
2.52 2
2.53 4
2.54 6
2.56 7
2.58 7
2.59 6
2.6 4
2.62 2
2.63 0
2.64 -2
2.66 -4
2.67 -6
2.68 -7
2.7 -7
2.72 -6
2.73 -4
2.74 -2
2.76 0
2.79 6
2.8 7
2.83 7
2.84 6
2.86 4
2.87 2
2.88 0
2.89 -2
2.9 -4
2.92 -6
2.93 -6
2.94 -7
2.96 -6
2.97 -5
2.98 -4
2.99 -2
3.0 0
3.02 2
3.03 4
3.04 5
3.06 6
3.07 7
3.08 6
3.09 5
3.11 4
3.12 2
3.13 0
3.14 -2
3.16 -4
3.17 -5
3.18 -6
3.21 -6
3.22 -5
3.23 -3
3.24 -2
3.26 0
3.27 2
3.28 3
3.29 4
3.31 5
3.33 5
3.34 4
3.36 3
3.37 2
3.38 0
3.39 -1
3.41 -3
3.42 -4
3.46 -4
3.47 -3
3.48 -2
3.49 -1
3.51 0
3.52 1
3.53 2
3.54 3
3.56 3
3.58 3
3.59 3
3.61 2
3.62 1
3.63 0
3.64 -1
3.66 -2
3.67 -2
3.68 -3
3.69 -3
3.71 -2
3.72 -2
3.73 -1
3.74 -1
3.76 0
3.77 1
3.78 1
3.79 2
3.84 2
3.86 1
3.87 1
3.88 0
3.89 -1
3.92 -1
3.93 -2
3.96 -2
3.97 -1
3.98 -1
4.0 0
4.02 0
4.03 1
4.11 1
4.12 0
4.14 0
4.16 -1
4.22 -1
4.23 0
4.3 0
4.31 1
4.32 1
4.33 0
7.0 0

The command line used:



Example 14.2 index 67 and 68

One can further extend the idea of using the option -trans to impose a vibrato upon a sound by generating a frequency modulation to be imposed upon a sound file. It is sufficient to create a vibrato faster than 20 Hz for the first lateral frequency bands to appear.

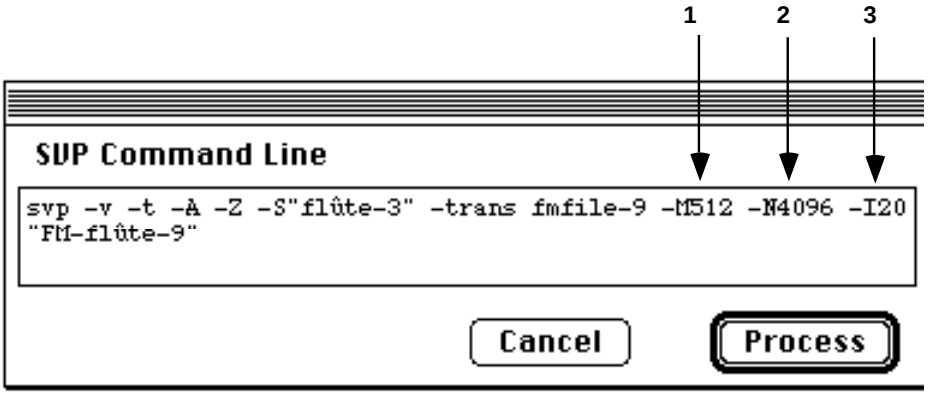


In this example, the sound of a flute is colored with frequency modulation induced from a dynamic transposition file.

The beginning of the file **fmfile-9** is shown in the figure. This file defines a vibrato whose frequency is 200 Hz, and whose amplitude varies between, +100 and -100 Hz.

To achieve good spectral and temporal definition, the analysis is made with a large FFT (flag -N4096) and a very small hop size (flag -I20).

The following command line is used:

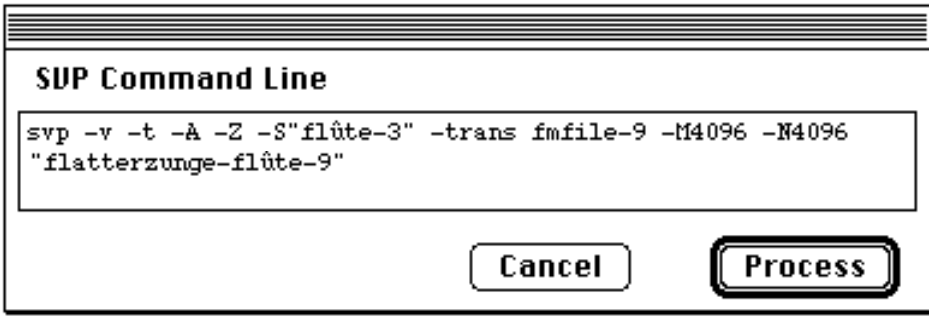


This example of frequency modulation imposed on a sound file is given, once again, to provide a point of departure for the advanced AudioSculpt user. Once this principle is understood, it can be applied to other types of modulation, such as amplitude modulation and wave shaping.

Example 14.3 index 69

The same transposition can be used to simulate a flutter-tongue by using a large window size and the default hop size value.

The following command line is used:



The transformation from flutter-tongue to ordinario is obtained by using a version of the above transposition file whose vibrato begins at the maximum amplitude and gradually decreases to nothing.

7 Conclusion

The use of text files containing parameter values allowing the realization of sound processing which varies over time opens inexhaustible possibilities when one takes into consideration that these files may be created in a program other than AudioSculpt itself (for example, PatchWork).

At the beginning of this work, I constructed different systems to create text files based on parameter envelopes, as well as for the graphic representation of existing files. I named these systems “editors”, and, with their help, I extended the possibilities of cross-synthesis within AudioSculpt. Some of these experiments, described in the appendix of the AudioSculpt manual show that, with a little imagination, one can discover new creative applications of the program.

In the appendix to this tutorial, the reader will also find several examples of unconventional sound processing which, I hope, can be musically very powerful.

8 Annex: List of options

Here is a summary of available SVP options:

-v	display errors in the console window.
-t	display the analysis window's average position in the console window.
-A	analysis of the primary input sound.
-a	analysis of the secondary input sound.
-a	analysis by linear prediction (LPC). Default order = 30 poles.
-Z	perform a re-synthesis.
-G	cross synthesis between two files followed by one of the three modes: add, mul or cross.
-S	name of the primary sound file.
-s	name of the secondary sound file.
-M	size of the analysis window (in samples) for the primary sound file.
-m	size of the analysis window (in samples) for the secondary sound file.
-N	size of the FFT (in samples) for the primary sound file (the option -n for the secondary sound file is not used since the FFT size must be the same for both sound files.)
-I	hop size (in samples) for the primary sound file.
-i	hop size (in samples) for the secondary sound file.
-J	type of synthesis window.
-W	type of analysis window for the primary sound file.
-w	type of analysis window for the secondary sound file.

For a detailed explanation of SVP options, refer to your AudioSculpt or your SVP manual.

Index

T

U

W