Diphone Studio Applications

Additive Analysis and Synthesis
Creating Sequences
Analysis of Resonance Models and Chant Synthesis

by Hans Tutschku

Enter

ircam Centre Pompidou

PDF Adobe

ForumNet

Mac

QuickTime

Forum Ircam

Diphone Studio

Diphone

Design : Xavier Rodet
Macintosh Programming : Adrien Lefèvre
Unix Programming : Thierry Galas, Philippe Depalle
Research assistants : Guillaume Bouriez, Marteen de Boer, Xavier Hosxe, Gilbert Nouno

SuperVP

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

Pm

Design : Xavier Rodet et Guillermo Garcia
Program : Guillermo Garcia, Diemo Schwarz

Chant

Design : Xavier Rodet, Yves Potard
Program : Yves Potard, Gerhart Eckel, Francisco Iovino, Dominique Virolle

Sdif

Design : Xavier Rodet
Program : Dominique Virolle, Diemo Schwarz

ModRes

Design: Xavier Rodet
Program : Pierre-Francois Baisnee, Francesc Marti

Table of Contents

Diphone Studio. Applications

Additive Analysis

Chant

Resonance Models

By Hans Tutschku

Presentation of this book

This Acrobat version of this book comes with a set of sound examples coded in MP3. In order to play them, QuickTime version 4 must be installed on your machine. This electronic book has been realized by Marc Battier.

These examples will be available at the DEGEM Web site Deutsche Gesellschaft Für Elektroakustiche Musik):

`http://www.degem.de`

They can also be played from the author’s Web site:

`http://www.multimania.com/hanstutschku/diphone/diphone.htm,`

This book was first published in German as three articles in *Mitteilungen*, journal of the Deutsche Gesellschaft Für Elektroakustiche Music, issues 35-36-37, PFAU Neue Musik, Saarbrücken, 2000.

The original articles have been translated into English by Diemo Schwarz.

This document was translated from LATEX by HEVEA into Adobe FrameMaker and converted to Acrobat PDF files using MicroType SP TimeSavers FrameMaker Edition utilities.

Introduction

The name Diphone Studio stands for a bundle of three related programs, developed in recent years at IRCAM for the Apple Macintosh. The two analysis programs AddAn and ResAn read sound files and reduce their data to representations. These can then be used to compose new musical units using the interpolator of Diphone.

Diphone is the user interface to interpolate sound segments that allows to create new musical sequences. The graphical interface contains several editors and is the framework for various synthesis plugins, such as additive synthesis and Chant synthesis.

The dictionary editor allows to select segments from existing dictionaries libraries of segments) and to create and store new ones.

The sequence editor allows the composition of new sequences out of existing segments. The parameters of the segments time and function of interpolation of a segment to the following one) can be freely manipulated.

The BPF editor allows to manipulate break point functions. All values of frequencies, amplitudes, phases, transpositions, etc. are stored as BPFs.

The segments that will be used to compose new sequences are assembled into dictionaries by the analysis stage. You can work straight away with the dictionaries provided with Diphone Studio, and compose new sounds with existing segments, or you can use the analysis programs to use your own sounds as starting point for new dictionaries.

AddAn is a program for additive analysis, ResAn is a program for the analysis of resonance models.

Since the program bundle is very comprehensive, containing various synthesis models and their respective analyses, the article is split into several parts.

This **first part** describes the possibilities for additive analysis with the program AddAn, the creation of user dictionaries, and the manipulation of segments in the additive signal model in Diphone.

The **second part** discusses the creation of segments.

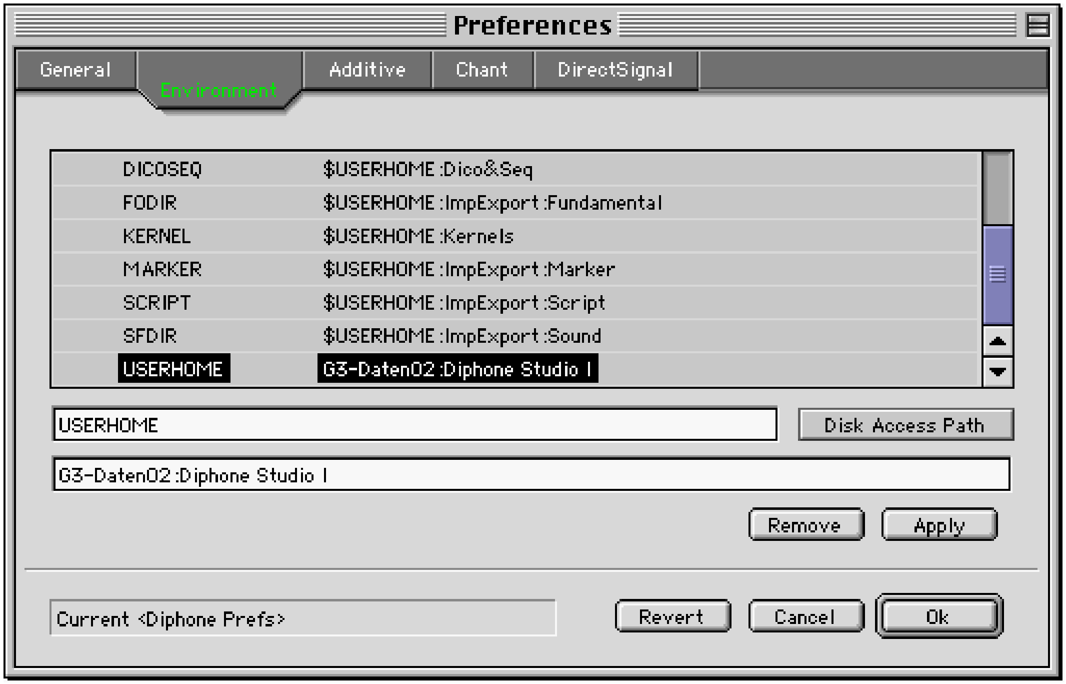
Finally, Chant synthesis, resonant filters, and the analysis of resonance models are presented in the **third part**.

At the end of this book, you will find a **list of Sound examples** and a general **index**.

1 Additive Analysis and Synthesis

1.1. System Settings

Because the file system structure of Diphone is based on Unix, it is important to set, in the Preferences of the program, where on your disk sounds, analysis files, and other data can be found. The folder of the program contains three important sub-folders: Container, Dico&Seq, and ImpExport. The folder Container stores the dictionaries Dico&Seq contains important files for Chant and newly created sequences. ImpExport contains more sub-folders with the original sounds Sound), or folders to store analysis files Fundamental and Partial).



DICOSEQ	$USERHOME:Dico&Seq
FODIR	$USERHOME:ImpExport:Fundamental
KERNEL	$USERHOME:Kernels
MARKER	$USERHOME:ImpExport:Marker
SCRIPT	$USERHOME:ImpExport:Script
SFDIR	$USERHOME:ImpExport:Sound
USERHOME	G3-Daten02:Diphone Studio I

USERHOME

G3-Daten02:Diphone Studio I

Current <Diphone Prefs>

If you want to use a different place on your hard disk than the program folder for these files, you have to set the variable USERHOME in Preferences -> Environment to that place.

1.2. AddAn --- Additive Analysis

Diphone does not treat the sounds itself, but the parameters representing them, like in many other analysis--synthesis methods.

In additive analysis--resynthesis, sounds are represented as a sum of sinusoidal oscillations. They are characterised by the parameters frequency, amplitude, and phase of these partials. Additive analysis estimates these parameters, along with the fundamental frequency f0). For harmonic sounds, the fundamental frequency corresponds to the first harmonic and is variable over the duration of the sound. This type of synthesis works best for harmonic sounds with a small amount of noise. Later on, we will nevertheless learn about ways to analyse less harmonic sounds.

To synthesise a sound in Diphone, a dictionary of segments is used, that contain the data for frequencies, amplitudes, and phases. To create such a dictionary, one analyses a sound with AddAn, and saves the resulting data in the dictionary. The first step is the calculation of the fundamental frequency, then the frequencies, amplitudes, and phases of the partials are estimated. Finally, a validating synthesis of these data is possible to check the quality of the analysis.

These three steps result in various data files stored on disk: Starting from a sound file example.aiff, a data file containing the fundamental frequency example.F0 is created in the folder :ImpExport:Fundamental, along with a file containing the data of the partials example.ADD in :ImpExport:Partial, and the resynthesised sound example.synth.AIFF in the folder :ImpExport:Sound.

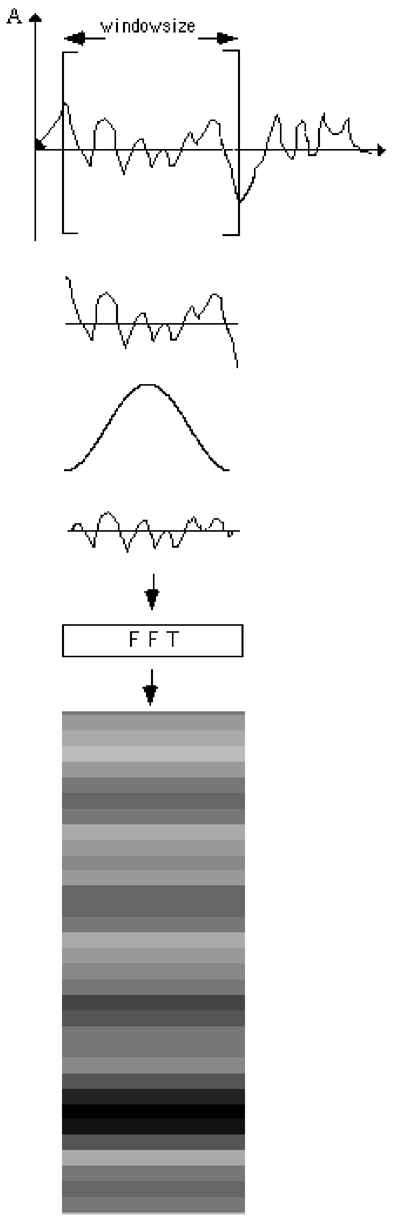
1.3. Fourier Analysis of the Spectrum (FFT)

During the search for the fundamental frequency, as well as while calculating the existing partials, the sound is analysed with a Fourier transform FFT). While some of the readers will already know the following explanation of the FFT, I will nevertheless go into a certain detail, because its understanding is crucial for a good analysis result, and many recurring questions are asked about it. It might prove helpful to some readers for the choice of parameters in other programs, such as AudioSculpt or SoundHack).

However, I will describe the FFT and their parameters WindowSize, FFT-Size, and WindowStep not with mathematical-scientific rigor, but rather musician-friendly and schematically.

Because a single FFT-analysis yields a static spectrum, without temporal evolution, the analysis of a whole sound with one single FFT would not yield a representation of the sound itself, but a kind of ``global average'' of the energies contained.

To obtain the variations of the spectrum over time, the sound is split into small segments that are analysed independently. These yield one FFT-spectrum short-time-spectrum each. The entire analysis of the sound is then a sequence of these short-time-spectra. The length of such a segment is determined by the parameter WindowSize, expressed either in seconds or in number of samples. As said before, because the calculation of the FFT of a segment is a static description of the energies in the sound segment, a melody contained in the segment would not appear as a sequence of pitches, but as a chord.



Therefore, good values for the window size depend on the sound to be analysed: If it contains fast variations, the window size should be small e.g. 500--1000 samples = 11.3-- 22.6 ms). This assures that multiple acoustic events that form the sound do not fall into the same window. If the sound is evolving slowly, larger values e.g. 2000--4000 samples) can be chosen. Because the window size is selected before analysis starts, and is constant for all the following short-time-spectra, the start and end points of the windows will not coincide with the zero crossings of the sound. This means that the chosen segment abruptly starts and ends, which introduces clicks that disturb analysis. See the second row of the figure.) Therefore, it is necessary to fade the sound segment in and out. This is done by multiplication with a so-called window function Hamming, Hanning, Blackman, Kaiser, etc.).

The sound segment faded in and out is then analysed by an FFT. The FFT works on a sample buffer the size of which FFT-size) must be at least as large as the window size, and always a power of 2 for instance window size = 1000, FFT size = 1024 or 2048 or 4096, etc.). The buffer of the FFT can not be smaller than the window size, since then the sound segment faded in and out would be cut once more and new clicks introduced. The FFT size can, however, be larger than the window size: In this case, the remaining samples are filled with zeros zero padding).

The FFT size is measured in number of samples, although it is not related to the temporal resolution of the analysis a frequent point of misunderstandings). The FFT size determines the frequency resolution of the analysis, and thus the quality of the image of the frequencies. One can think of it like this: The entire frequency spectrum from 0 Hz up to half the sampling frequency i.e. 22050 Hz with a sampling frequency of 44100 Hz) is split into frequency bands of equal size the FFT bins). For each of these bands, the energy contained in it is calculated. To obtain a high quality of analysis, as many and as small frequency bands as possible are chosen, so that the energies of different partials fall into different bands and can thus be distinguished.

The number of these bands and thus their bandwidth) depends on the FFT size. With an FFT size of 1024, 512 bands between 0 Hz and 22050 Hz are calculated. The bandwidth is thus 22050 / 512 = 43.07 Hz. With an FFT size of 4096, 2048 bands with 10.77 Hz bandwidth are calculated. A larger FFT size yields a larger number of smaller bands and thus a better resolution of the analysed frequencies. If several partials fall into the same band, they are averaged with such drawbacks as beating or cancellation.

A "short time FFT analysis" comprises thus a fixed grid of frequencies, for which the current amplitudes and phases are found.

To illustrate resynthesis, one can imagine that each of the frequency bands are reproduced by a sine generator, the frequency of which is tuned to the central frequency of the band, and fine-tuned by the phase. The amplitude of the generator follows the analysed amplitude of the band. These data are interpolated from one short time analysis to the next, and thus yield a continuous signal again.

As a rule of thumb, we can say that the temporal resolution of the analysis is determined by the window size the faster the signal varies, the smaller the windows that should be used), and that the quality of the frequency resolution is determined by the FFT size.

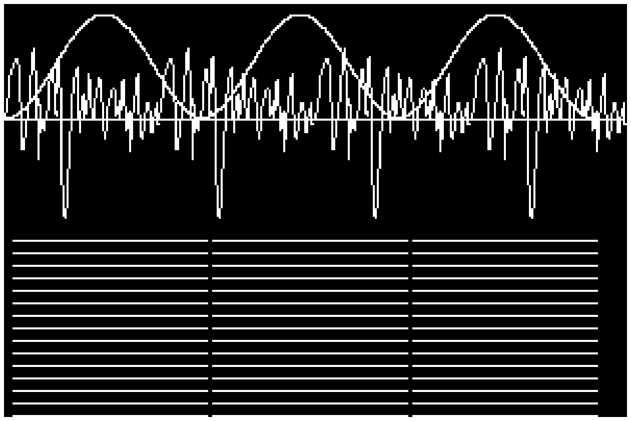
However, the window size determines one more property: the lowest analysable frequency. At least three periods of it must be within the window size for it to be analysed. This is not a strict limit. Four to five periods are even better to precisely determine a frequency. If the sound to be analysed varies between 200 and 500 Hz, the longest period is 1/200 Hz = 0.005 s. To get four periods into the analysis window, the window size should therefore be at least 0.005 * 4 = 0.02 s long.

Here, we hit a difficulty with rapidly varying sounds with low frequency components. To analyse the temporal variation of the sound we'd want to choose a small window size. To analyse the low frequencies, however, we'd need a larger window size. In cases like this, the only way to find a compromise is to try and experiment.

In programs where window size and FFT size can not be chosen independently e.g. SoundHack, SoundDesigner, ProTools Audiosuite), you always encounter such a compromise. Small values lead to a good temporal resolution, but also give large frequency bands of the FFT and thus a bad resynthesis of the frequencies. Large values yield a good frequency resolution, but decrease the temporal resolution many sound events fall into the same analysis window, so that their energies will be averaged).

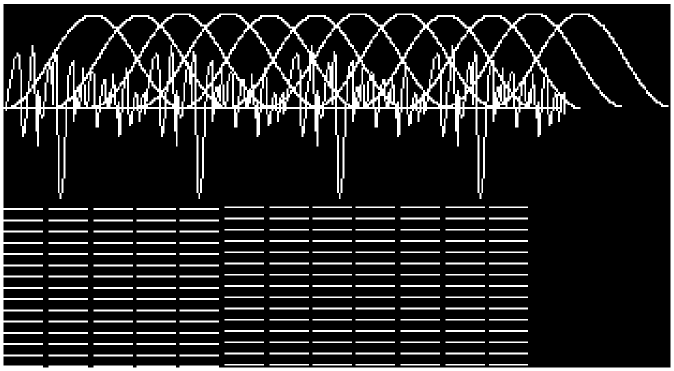
The next parameter is the window step, sometimes called *stepsize*.

If an analysis with length window size) would follow immediately the previous one, parts of the sound could not be analysed because of the fading in and out of the window.

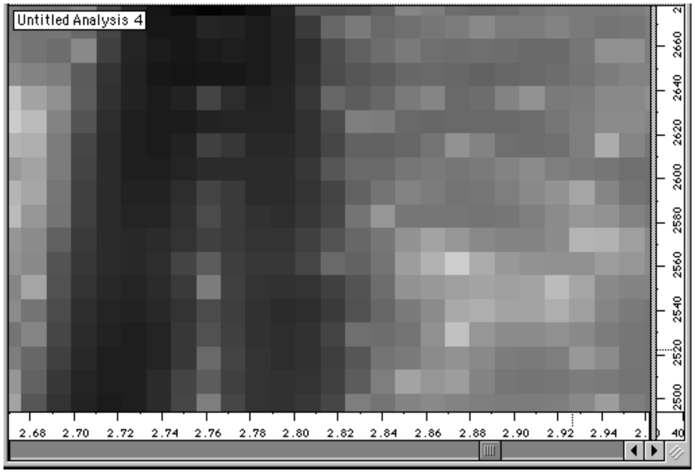


That's why the following analysis starts with a temporal shift smaller than the window size --- the analysis windows overlap. Typical values for the step size are 1/4 or 1/8 of the window size fig. 4).

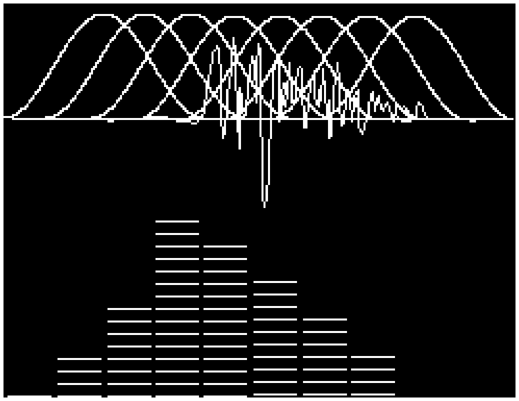
Each analysis window results in a short-time-spectrum that is used only for the duration of the step size, then the next analysis follows. The advantage is that every point in the sound appears several times more or less precisely in the middle part of the window function. Therefore, its energy is not affected so much by the fading in and out.



If we zoom into the sonagram in Audiosculpt, we see little boxes showing the short-time-spectra in their temporal order. Their length corresponds to the step size. On the frequency axis, the boxes correspond to the FFT bins here approximately 10 Hz large, since the FFT size was 4096)

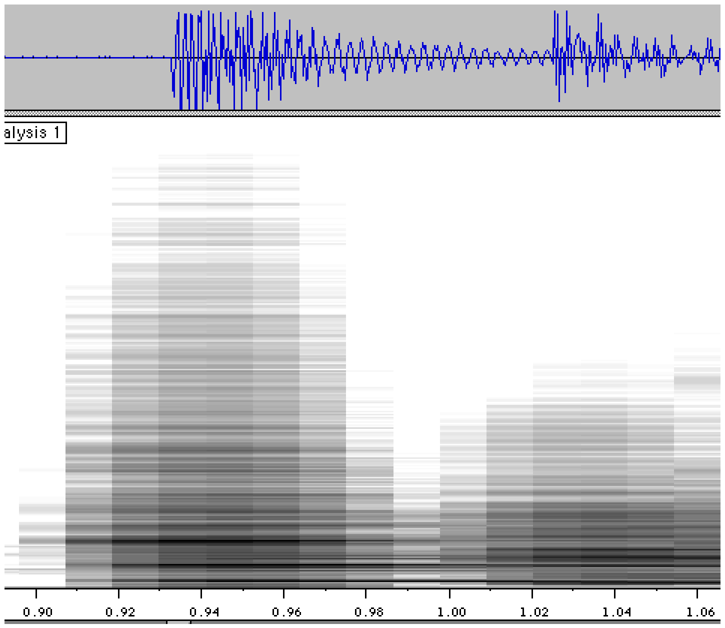


The disadvantage of overlapping analysis windows is that each point in the sound is analysed multiple times e.g. four times with a step size of 1/4 the window size). Let's examine this effect with the example of the analysis of an attack, a sound event with a lot of energy in the whole spectrum



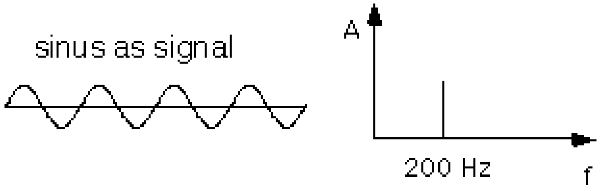
The first analysis window touches the attack only slightly, and because of the fading out there's little energy left: the resulting spectrum hardly has any energy. The following analysis windows will have more and more of the attack in their centre, and will therefore find more and more high frequencies. In the figure, the fourth window has the attack in its centre.) In the following windows, the energy of the high frequencies will decrease again.

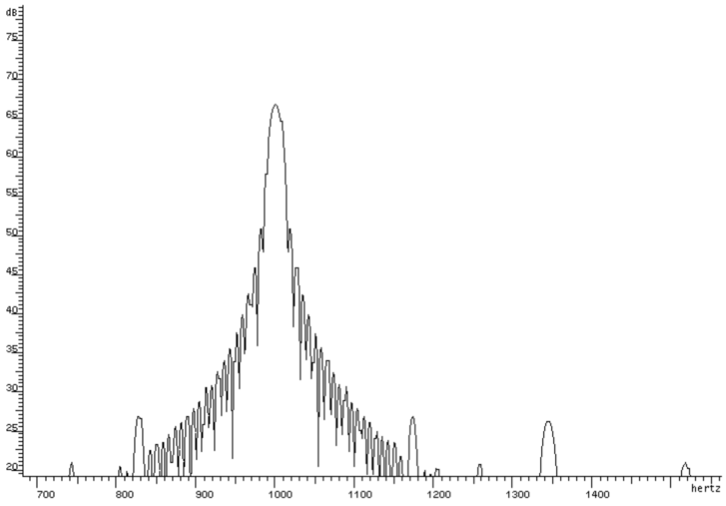
Because the short-time-spectrum corresponding to the current analysis window is positioned at the beginning of the window, some energy is found that actually appears later in the sound. This causes the effect of a sort of "spectral fade in" of the high frequencies:

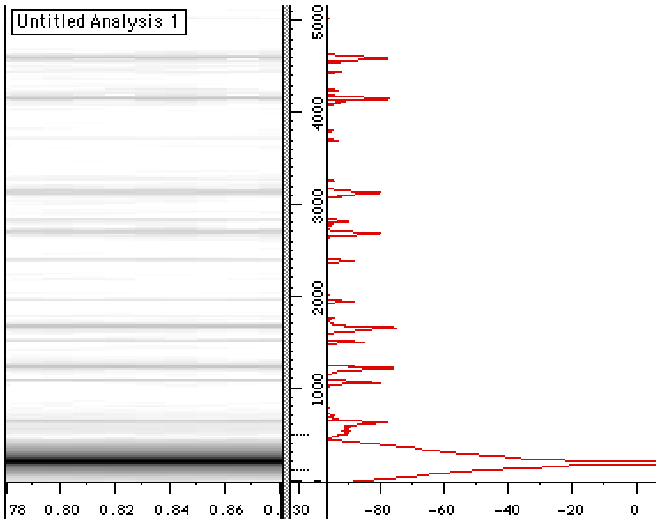


The fourth parameter for the analysis is the window function. Several programs offer a choice of window functions, among which Hamming, Hanning, Blackman, and Kaiser are often found. Their influence on the quality of the analysis is subtle, however not to be neglected.

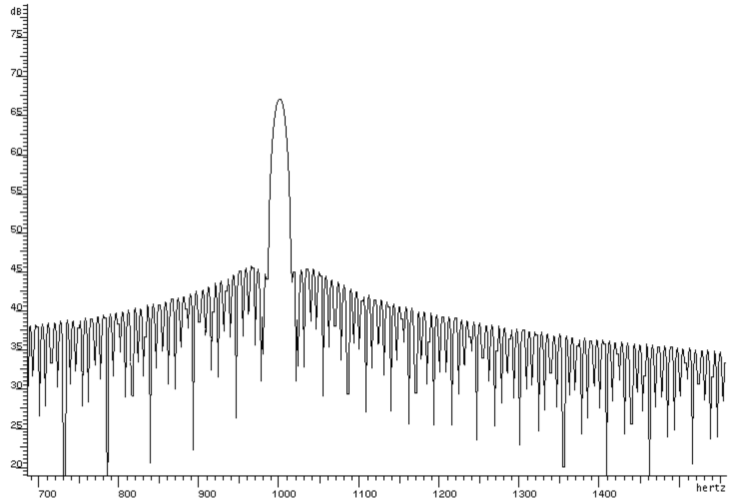
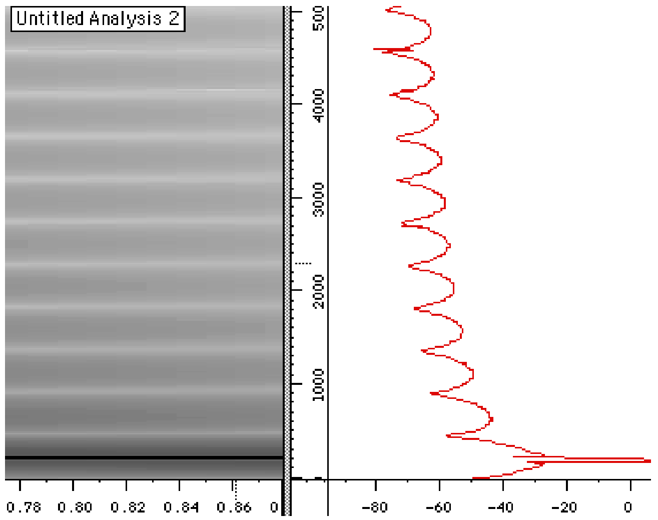
Theoretically, the spectral image of a sine oscillation is a line in the spectrum



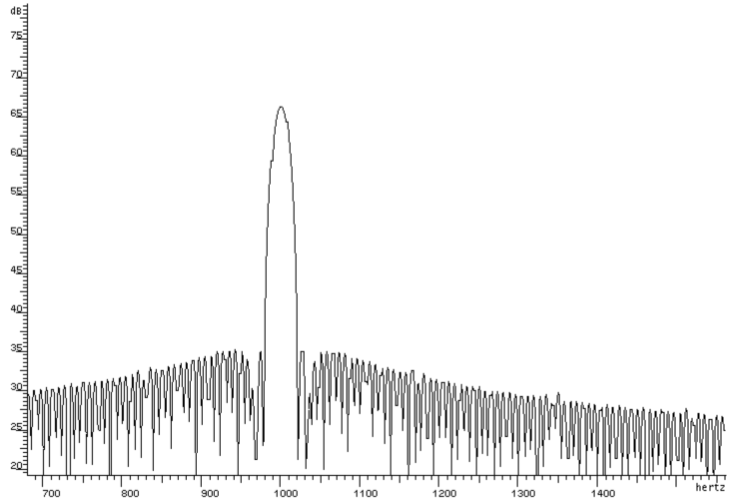
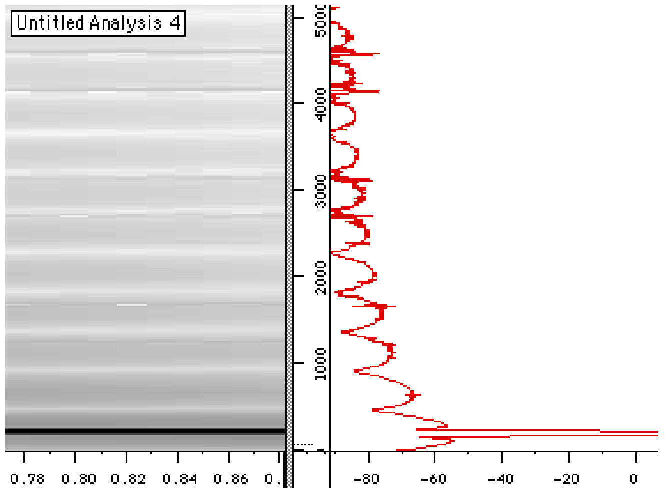




A Hanning window generates a rather large central peak and some side-bands.

The frequency definition of the central peak of the Hamming window is better. However, it generates substantially more spread-out side-bands.

The spectrum of the Blackman window has a good frequency definition, too. The side-bands have a lower amplitude and are not as widely spread as those of the Hamming window.

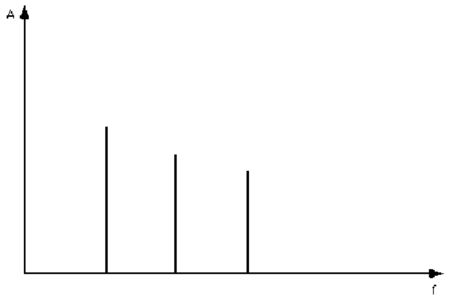
During analysis, this signal is now multiplied with a window function. What happens in fact is that two signals are multiplied similar to ring modulation). The resulting spectrum is obtained by a convolution of the spectrum of the signal and the spectrum of the window function.

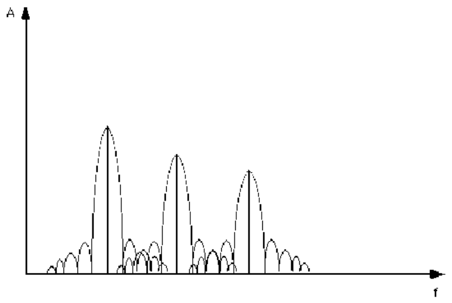
The qualitative differences of the various window functions stem from the structure of their respective spectra. To illustrate this, in the three spectrum figures, a sinus of 1000 Hz frequency is multiplied with the three most typical window functions. You can see that the energy of the sinus is not concentrated on a single line left column).

This can also be seen in the corresponding sonagrams here with a sinus of 200 Hz in the right column): While the Hanning window produces clearly visible "partials", the other two functions rather generate a louder Hamming) or softer Blackman) "noise floor". The signal-to-noise ratio between the central peak and the side-bands is large enough, however, for them to be not audible. Only with several analysis/ resynthesis steps in a row, the side effects of the window function can be perceived.

So far only the multiplication of one single sinus frequency with the three most common window functions was shown. During the analysis of a complex sound this energy redistribution indeed happens for every single partial.

If the complex sound contains three partials, its theoretical spectrum should be a sequence of three lines. Because these three frequencies are multiplied with the spectrum of the window function during analysis, the following energy distribution is obtained:



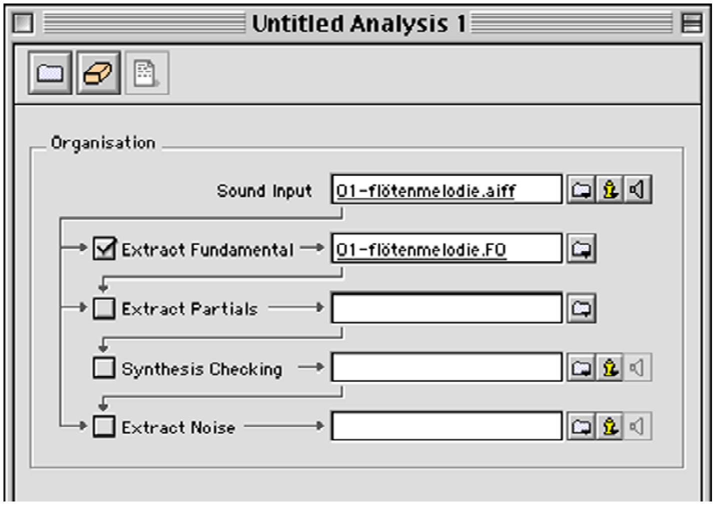


1.4. Analysis of the Fundamental Frequency

After the preceding summary about the effects of the four parameters, we go back to the analysis in Diphone. First, the fundamental frequency f0) is calculated.

Sound example 01-floetenmelodie

If the sound to be analysed resides in the folder :ImpExport:Sound, you can select it in AddAn by simply clicking on the folder symbol in the line Sound Input, and play it by clicking on the loudspeaker symbol.


Untitled Analysis 1

Organisation

Sound Input: 01-flötenmelodie.aiff

☑ Extract Fundamental → 01-flötenmelodie.F0

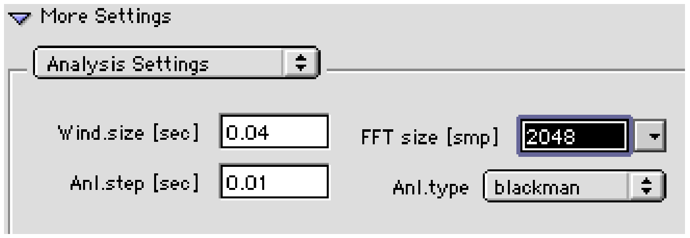
☐ Extract Partials →

☐ Synthesis Checking →

☐ Extract Noise →

First, we choose only Extract Fundamental.

In the lower part of the window More Settings) we set the parameters for the analysis. Window size and window step called here Analysis Step), are specified in Diphone in seconds:


More Settings

Analysis Settings

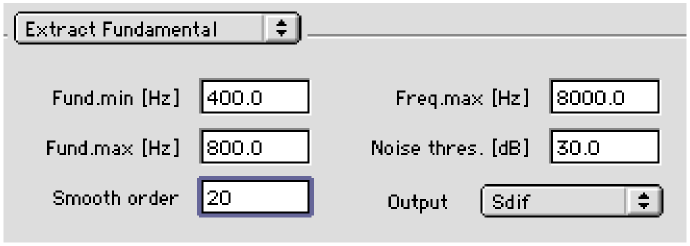
Wind.size [sec]: 0.04

FFT size [smp]: 2048

Anl.step [sec]: 0.01

Anl.type: blackman

In the following, we set the parameters for the analysis of the fundamental frequency:


Extract Fundamental

Fund.min [Hz]: 400.0

Freq.max [Hz]: 8000.0

Fund.max [Hz]: 800.0

Noise thres. [dB]: 30.0

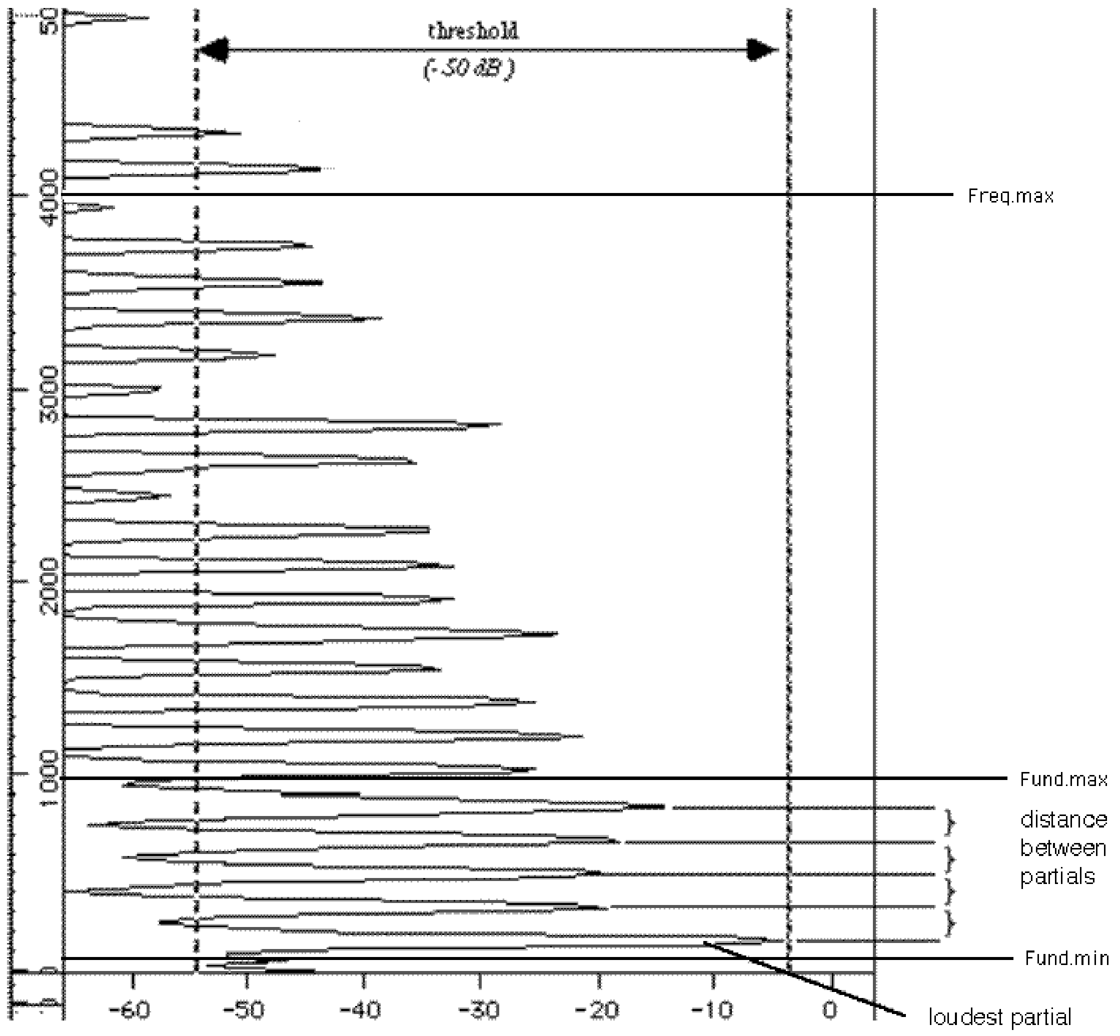
Smooth order: 20

Output: Sdif

During this analysis, the distances of the energy bands found are calculated, and examined if they refer to a lowest common multiple. To obtain good results, and also to reduce computation time, we limit the search range. If not, all FFT bins within one analysis would be compared with each other.)

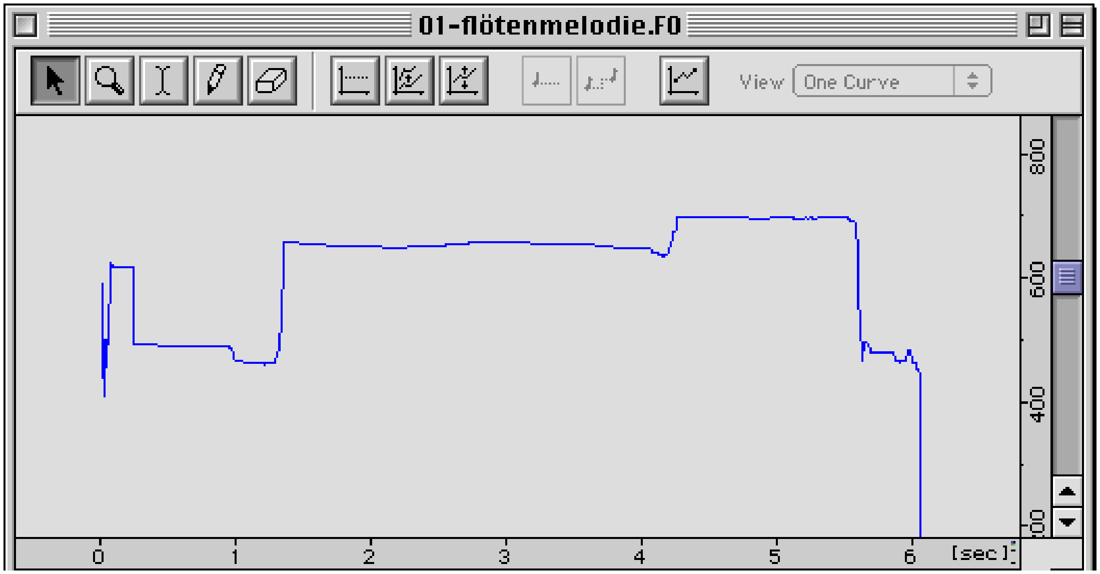
Fund.min and Fund.max specify limits between which the f0 is searched. Noise threshold is a relative amplitude distance from the loudest partial. All energies softer than this threshold are discarded during the search for harmonic multiples. For sounds with a large noise part one would choose a lower!) Noise threshold, to only include the loudest energies in analysis.

Freq.max determines the upper limit of the search space for harmonic multiples. If 4 to 5 multiples of a fundamental frequency have been found, this can be considered the correct f0. It is thus not necessary to search up to the end of the spectrum 22050 Hz). Smooth order values between 3 and 30) smoothes the curve of the fundamental frequency after analysis, to level small deviations.

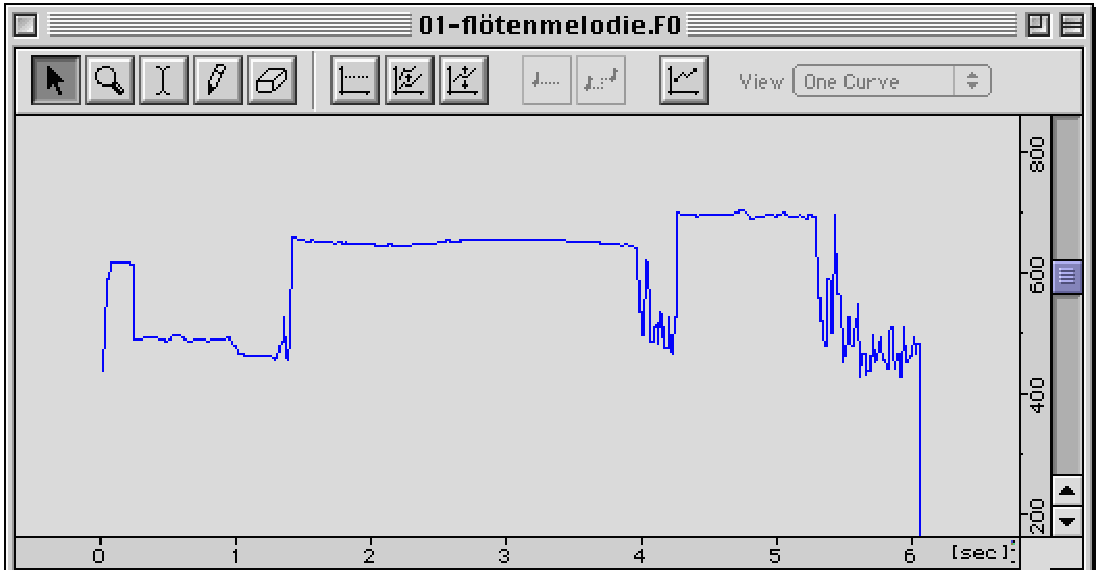


The output format is SDIF Sound Description Interchange Format), a binary file. To display the data, or to use them in older programs, it is still possible to output an ASCII file.

The output can then be loaded and viewed in Diphone. For our sound example, the settings above yield this result:

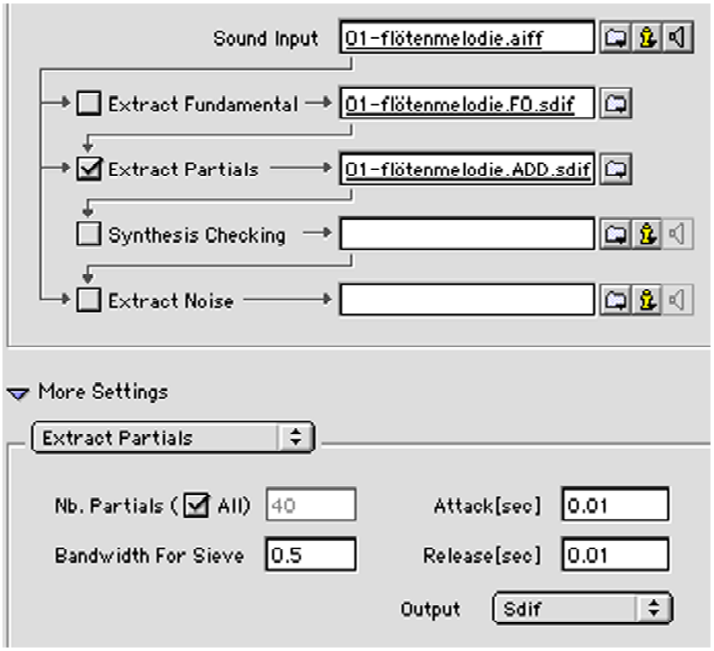


Some irregularities at the beginning and the end of the sound can be seen, however, these are irrelevant. Bad parameters for the analysis of the f0 result in erratic jumps in the frequency curve:



To use the f0 data in the next steps, the output-format must be set to SDIF. ASCII was only used for visual checking.)

1.5. Reduction to Harmonic Partials


Sound Input: 01-flötenmelodie.aiff

☐ Extract Fundamental → 01-flötenmelodie.F0.sdif

☑ Extract Partials → 01-flötenmelodie.ADD.sdif

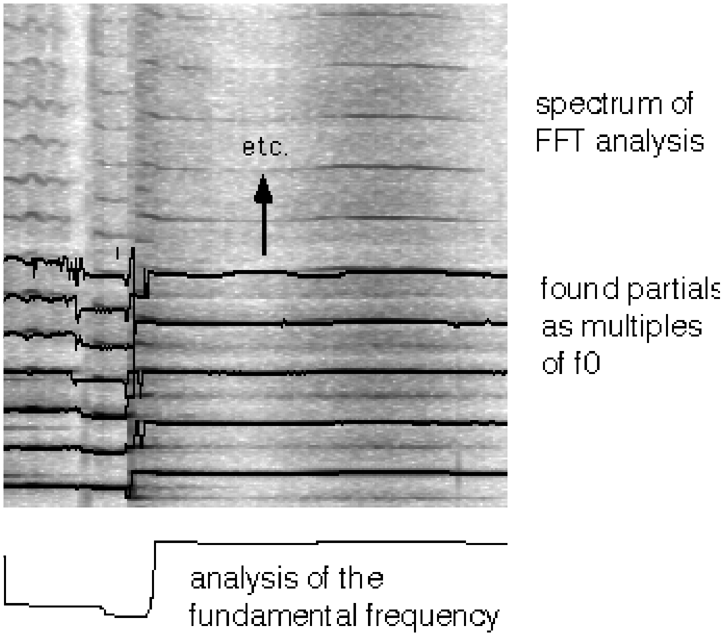
☐ Synthesis Checking →

☐ Extract Noise →

More Settings

Extract Partials

Nb. Partials (☑ All)	40	Attack[sec]	0.01
Bandwidth For Sieve	0.5	Release[sec]	0.01
		Output	Sdif

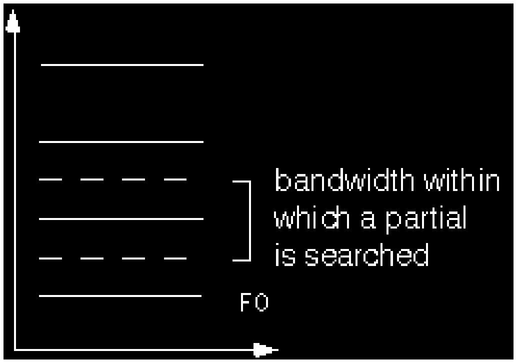


This analysis step takes the calculated fundamental frequency as a basis to extract energies from the short time FFT spectra that are multiples of the fundamental frequency, i.e. they correspond to the harmonic partials of the sound.

This point shows the importance of a good analysis of the fundamental frequency. If it would show unnatural jumps, also the frequencies of the partials would contain these errors.

Furthermore, the above analysis shows that the noise part of the sound, in our example caused by the air in the flute, can not be analysed. Later I will present an example how to represent the noise part in a better way.)

The parameters for Attack and Release allow the fading in and out of partials the moment they appear in the spectrum or cease to exist, respectively. This is to avoid clicks.



Bandwidth for Sieve determines a frequency range for each partial, situated around the exact multiple of the fundamental frequency. This permits deviations from the strict harmonic structure of a sound. A value of 0.5 ranges from the frequency half way between two partials up to the frequency half way between the following two partials.

1.6. Resynthesis and Extract Noise

Synthesis Checking activates the resynthesis of the freshly analysed partials. It offers to check the quality of the analysis, before you start segmenting the data and calculate dictionaries containers).

Sound example 02-floetenmelodie.synth

If Extract Noise is selected, a sound example.noise.AIFF is created, containing the subtraction of the resynthesized sound from the original sound. The result contains all the sound components that don't lie on the harmonic grid.

Sound example 03-floetenmelodie.noise

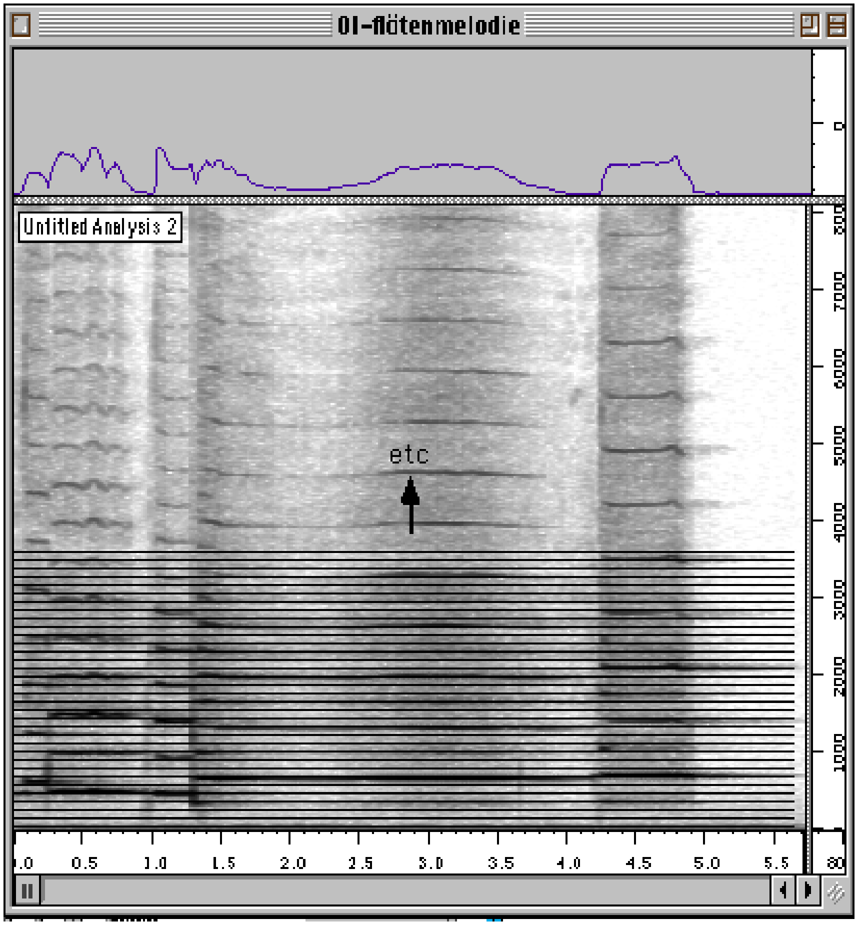
Of course, all the analysis steps can be calculated in one pass. To calculate only the fundamental frequency first and look at its form has the advantage, though, that one can recognise a possibly bad f0-analysis before calculating the next passes.

1.7. Improvement of the Analysis of the Noise Part

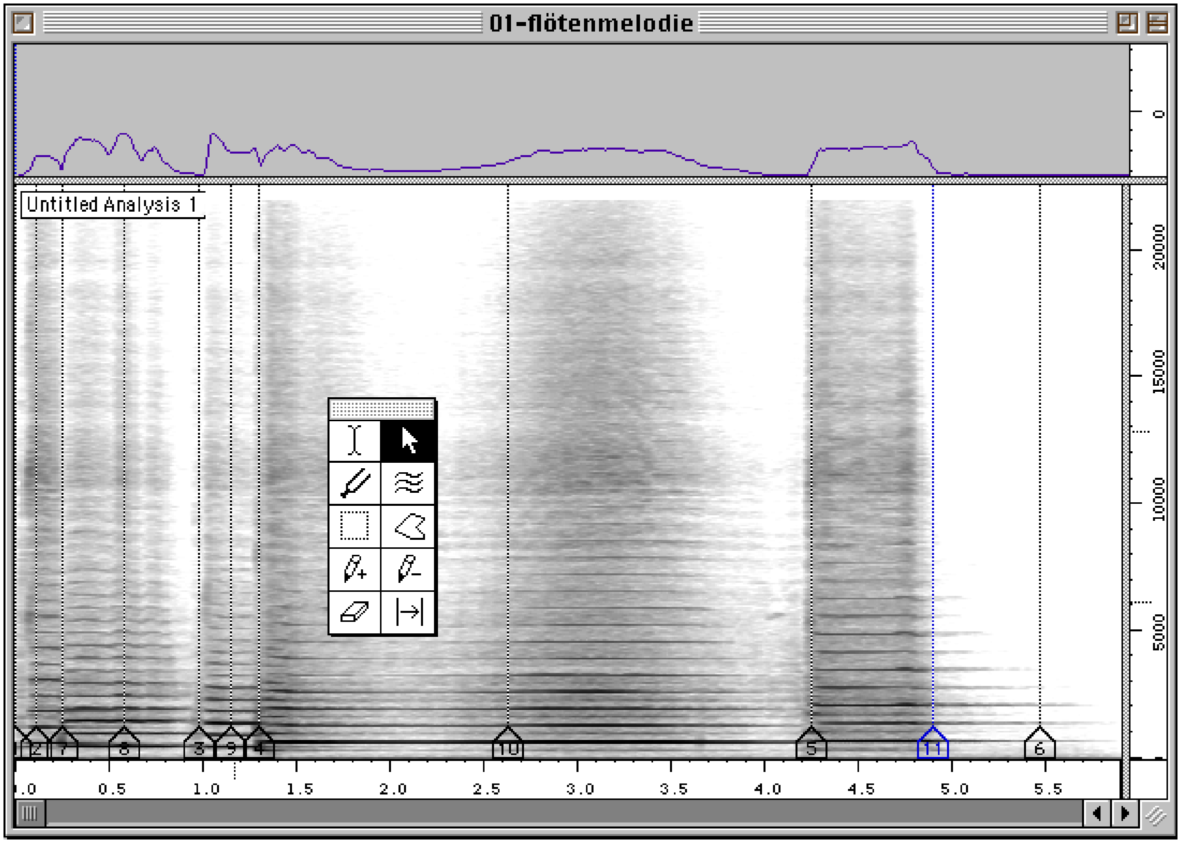
As mentioned above, the analysis of the noise part of the flute is impossible because of the reduction to the purely harmonic sound component. However, if we supply an ``artificial'' fundamental frequency of, say, 30 Hz, the partial analysis searches for all multiples of 30 Hz and yields a much greater number of sinusoidal partials to resynthesis.

Sound example 04-floetenmelodie.synth.30Hz

Because these are not only located on the multiples of the real fundamental frequency, but also between the ``real'' partials, the energies pertaining to the noise component of the sound are reproduced.

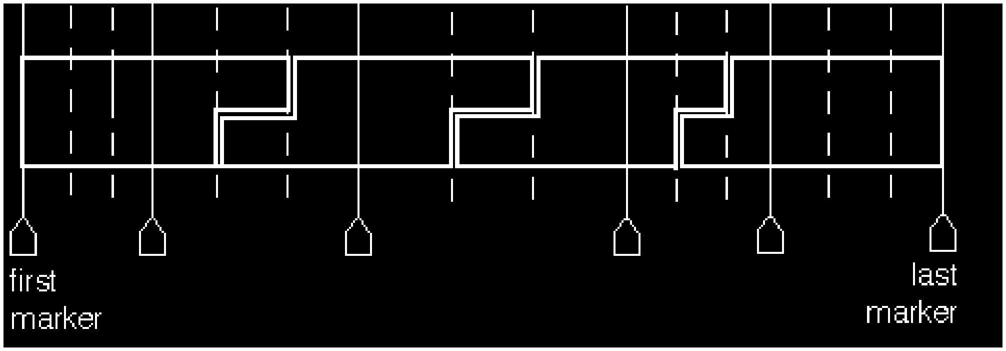

01-flötenmelodie
Untitled Analysis 2
etc
.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
80

30 Hz


01-flötenmelodie
Untitled Analysis 1
0
20000
15000
10000
5000
.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5

1.8. Segmentation of the Source Sound

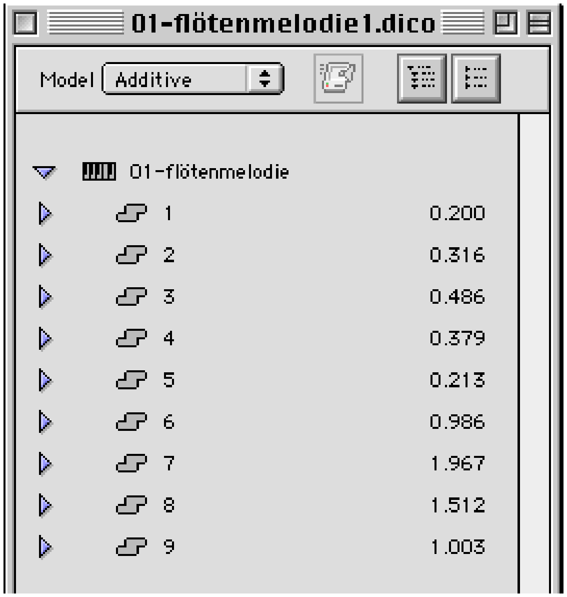
To perform segmentation, we can use the program AudioSculpt. By Ctrl-click into the sonagram, we can place markers the times of which can be written into a text file in Diphone's folder :ImpExport:Marker) by selecting the menu item File -> Export -> Markers. Each marker stands for the middle of a segment.



1.9. Calculation of the Dictionary containing the Segments

Starting from this marker file, we can create a script in Diphone menu PlugIns -> Create Script from Markers), and then calculate the container menu PlugIns -> Create Dictionary from Script).

Now the folder container contains a dictionary the segments of which contain the analysis data that can be used in new sequences. If we open this container in Diphone, we can drag and drop the segments into a new sequence and edit them there.



01-flötenmelodie

1	0.200
2	0.316
3	0.486
4	0.379
5	0.213
6	0.986
7	1.967
8	1.512
9	1.003

The possibilities of sound creation with these segments are described in the next part.

2 Creating Sequences

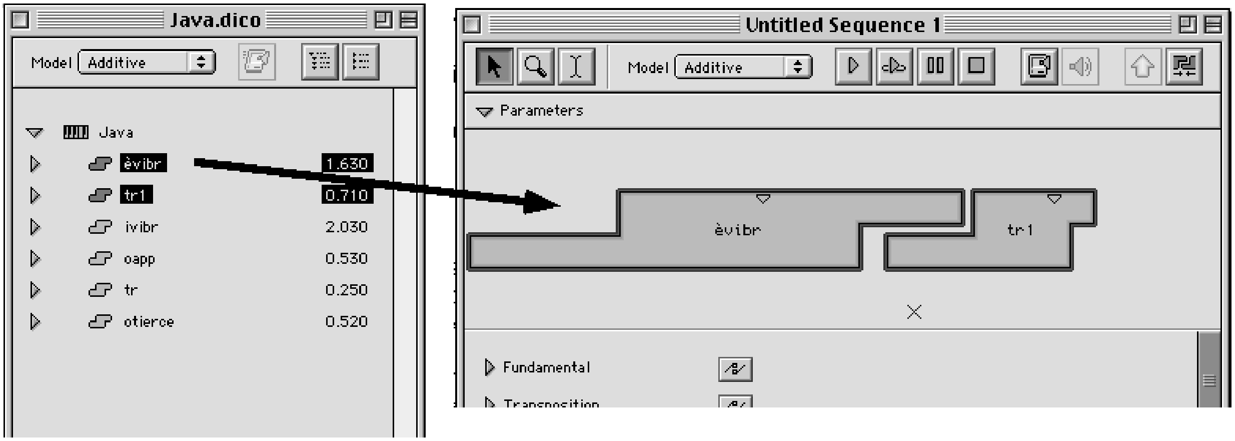
The second part describes the possibilities of sound manipulation within Diphone. This article is not meant to replace the good French/English manual, but rather to provide a quick introduction.

2.1. Some Definitions

A container is a document that resides in the folder container. It contains the data from additive analysis and the segmentation of the source sound. Depending on the analysis parameters and length of the source sounds, this file can have a size of several megabytes.

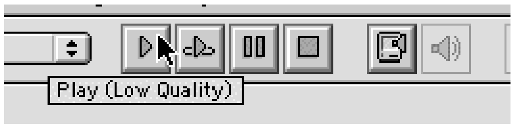
If we open a container in Diphone, it appears as dictionary, which is so to say an alias to the container. All modifications of the dictionary never change the data of the container, but are only stored in the dictionary. This assures that data of various sequences that use the same container can not be modified by changing a sequence. We can compare this to sound files and regions in Protools: if we change a region, the sound files on disk are left intact.

A dictionary can contain one or more instruments from which some or all segments can be dragged into a sequence:



Java	
èvibr	1.630
tr1	0.710
ivibr	2.030
oapp	0.530
tr	0.250
otierce	0.520

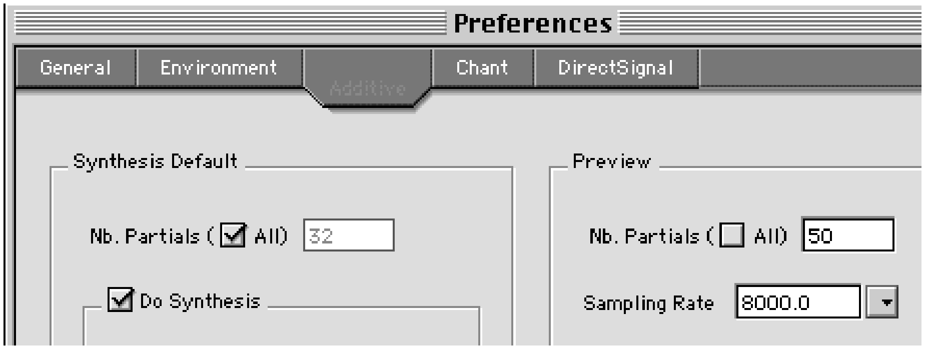
A BPF break point function) is the notion for the representation of data over time. A simple BPF is, for instance, the fundamental frequency. Multi-BPFs are frequencies, amplitudes and phases together.



If we play a sequence in Diphone, we get a preview.

Diphone can, depending on the processor speed of the computer, play a certain number of partials in real time. This number is set in the preferences.

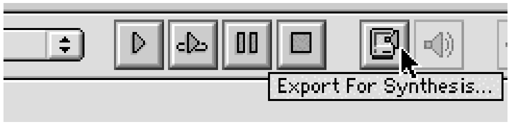
If Diphone can't play the desired number, clicks occur and we have to either select a lower sampling frequency or less partials in the preferences.


Preferences

Synthesis Default
Nb. Partials (☑ All) 32
☑ Do Synthesis

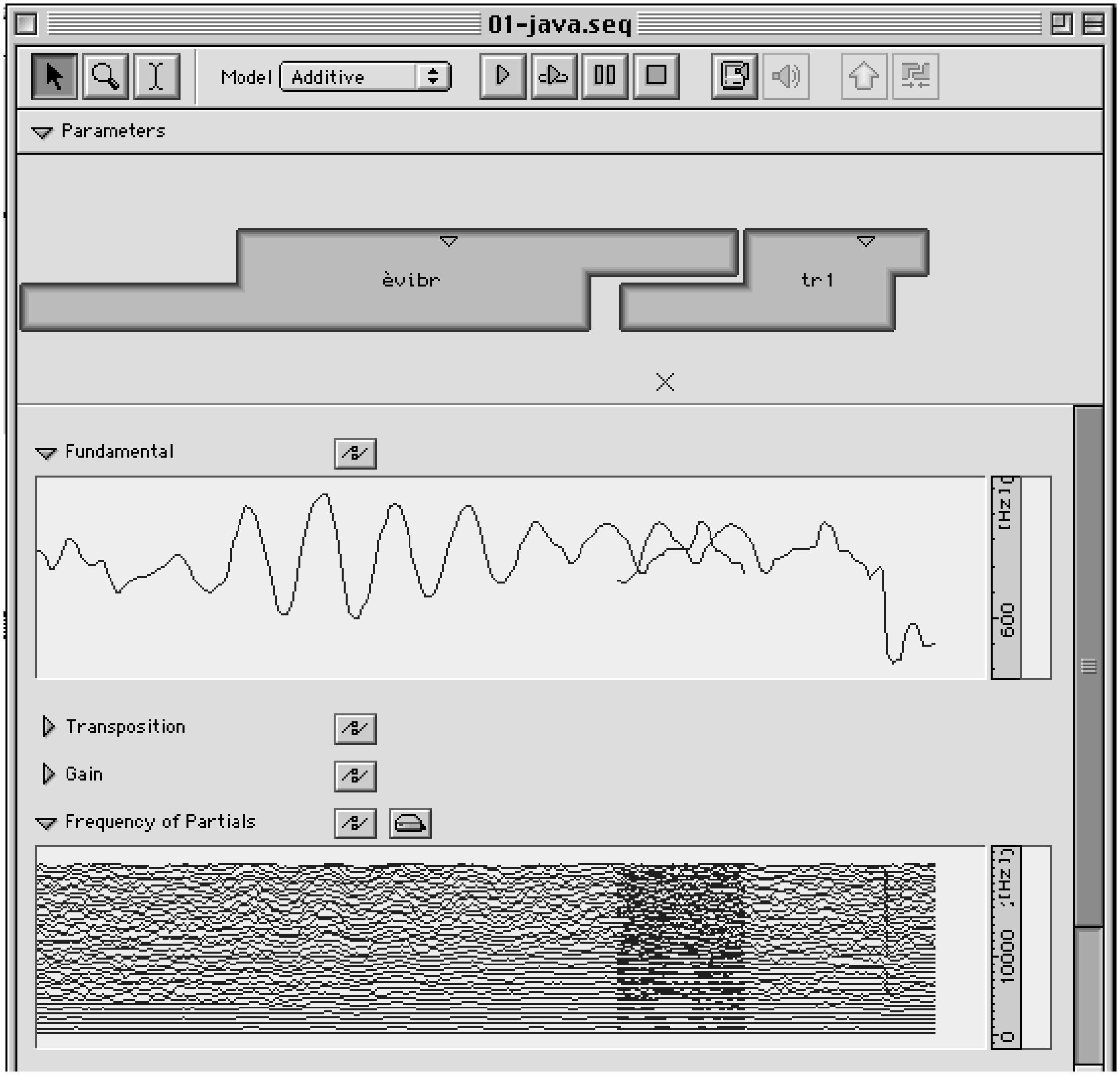
Preview
Nb. Partials (☐ All) 50
Sampling Rate 8000.0

If we want to obtain the result in best quality, we create a new sound file with Export for Synthesis.

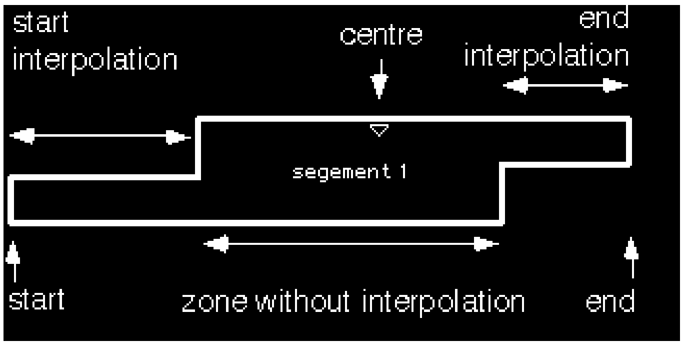


2.2. Displaying and Editing of Parameters

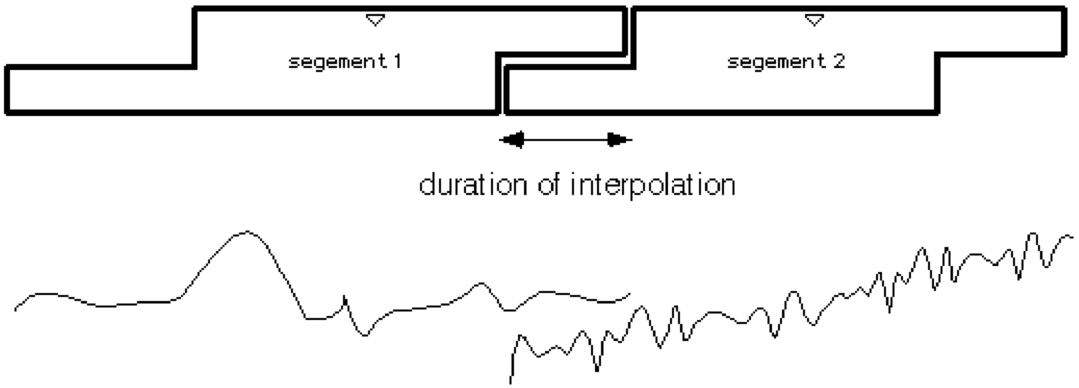
Below the sequence containing the segments, all parameters pertaining to the segment are listed. We find here the values resulting from analysis of the fundamental frequency, the frequencies, amplitudes, and phases of the partials and some additional parameters described further on. Clicking on the triangle next to the name of the parameter opens a graphical display of it.



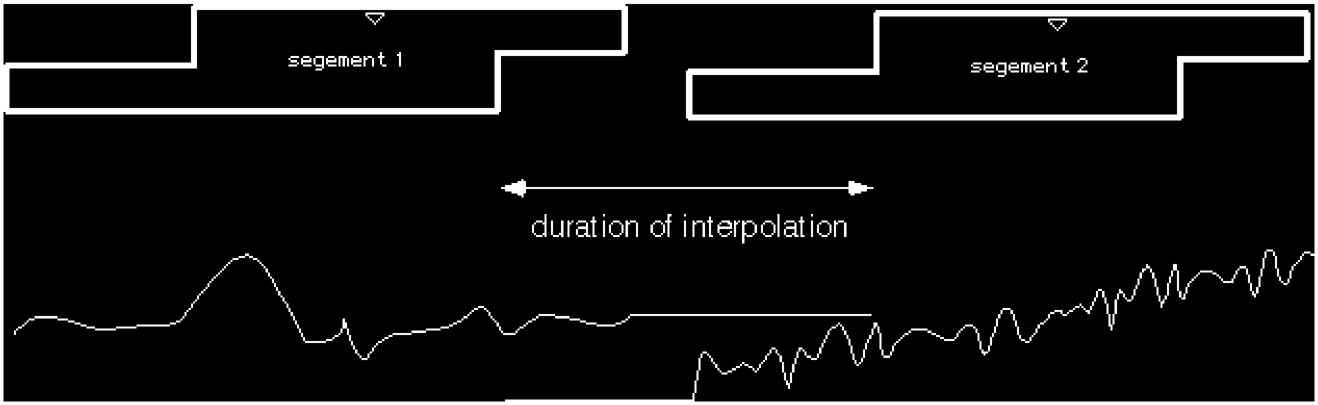
A segment is made up from three zones: start interpolation, central zone without interpolation, and end interpolation:



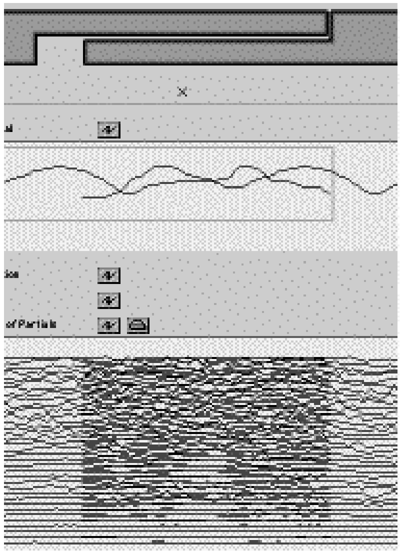
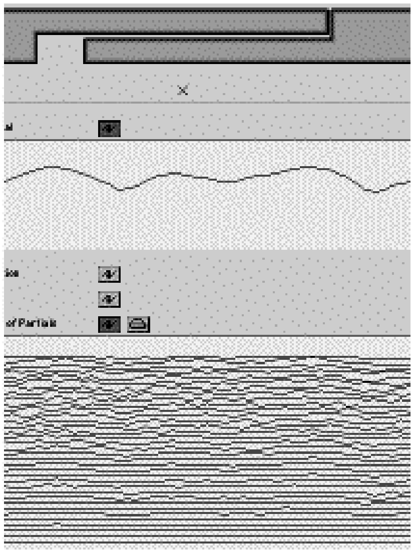
The data of two consecutive segments are interpolated from the end of the central zone of the first segment up to the start of the central zone of the following segment:



If two segments are not directly touching each other, there is no gap in the sound. Rather, the duration of interpolation is prolonged, and the data are interpreted like this: The last value of the first segments and the first value of the following segment are held for the necessary duration of the interpolation.

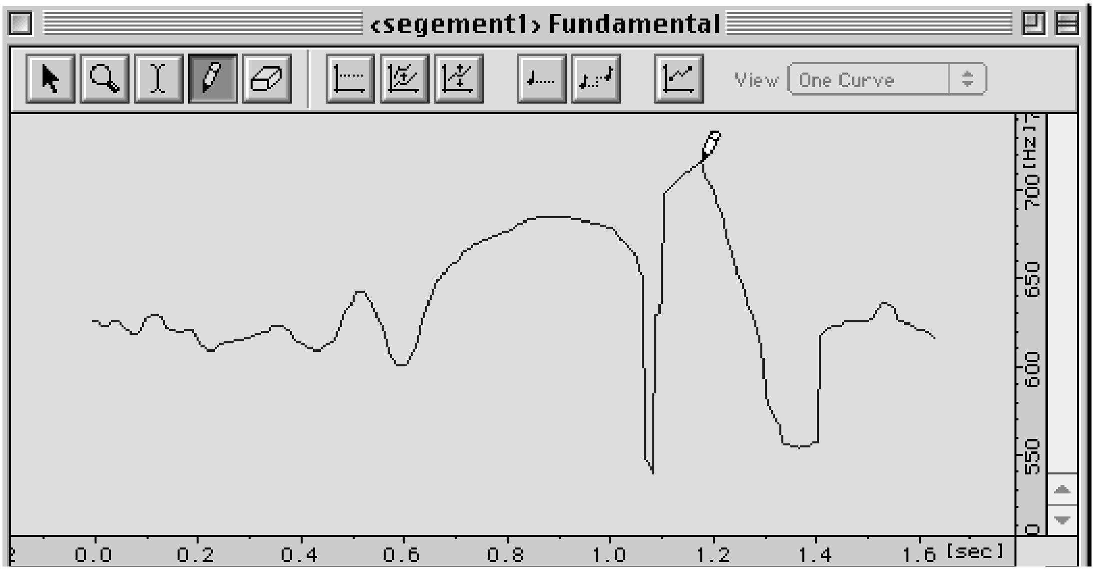


We can choose between two display modes: Either we see the segment data without interpolation, or we click on the interpolation icon and visualise thus the result of interpolation.

If we are in the non-interpolating display mode and move the mouse over a parameter curve, the latter is framed with a blue rectangle.

Double click then opens the editor, where we can modify the data either with the pencil or with arithmetic functions:

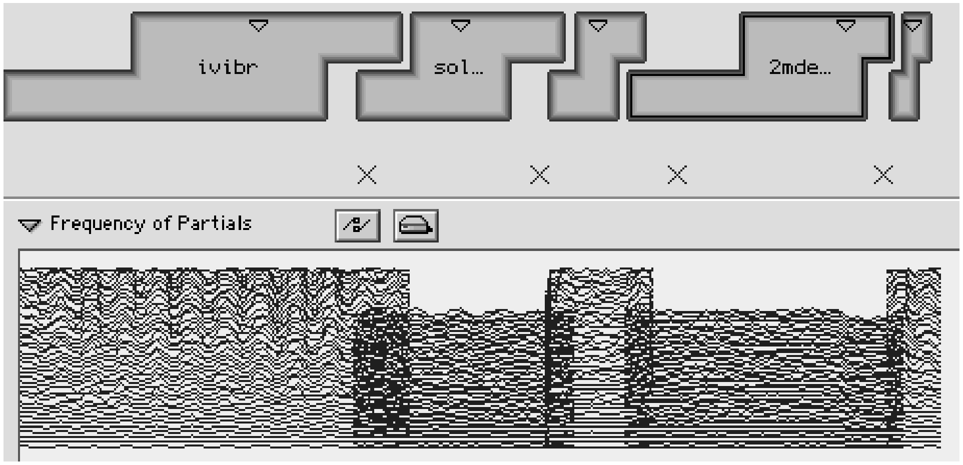


To zoom into the display, we can use the magnifying glass, or select a range on the time or value axis while holding shift). Double clicking on the axis puts us back to the original view.

All changes of the data have an immediate effect on the sound being played and of course also on the resynthesis.

2.3. Interpolation between Segments

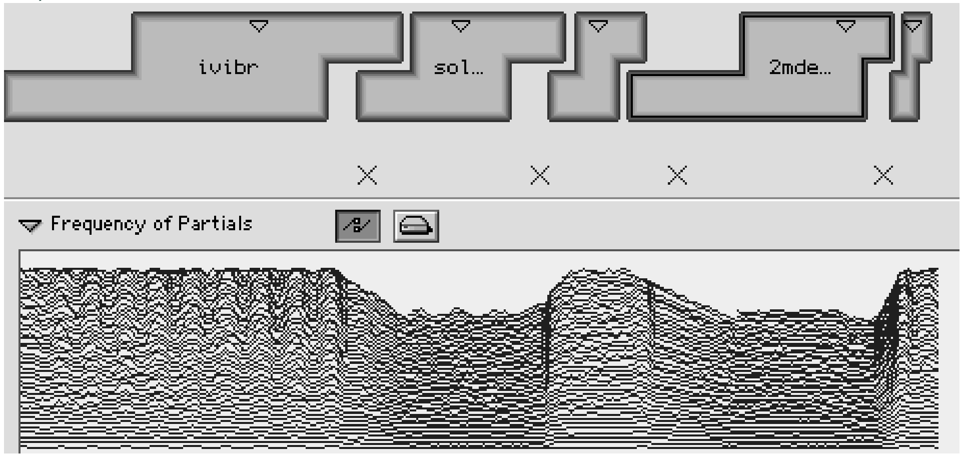
The following example combines segments from two different dictionaries. I use the dicos that are provided with the program, so that everyone can follow the steps taken. From the dicos Java and Shaku some segments were placed alternatingly into a new sequence. In the non-interpolating display mode we can clearly see the different spectra of both source sounds:



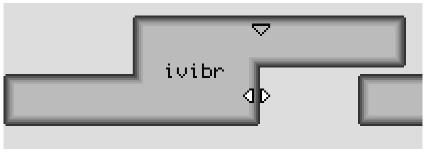
In the interpolating mode we see the glissandi arising from interpolation.

Sound example java-shaku01

)

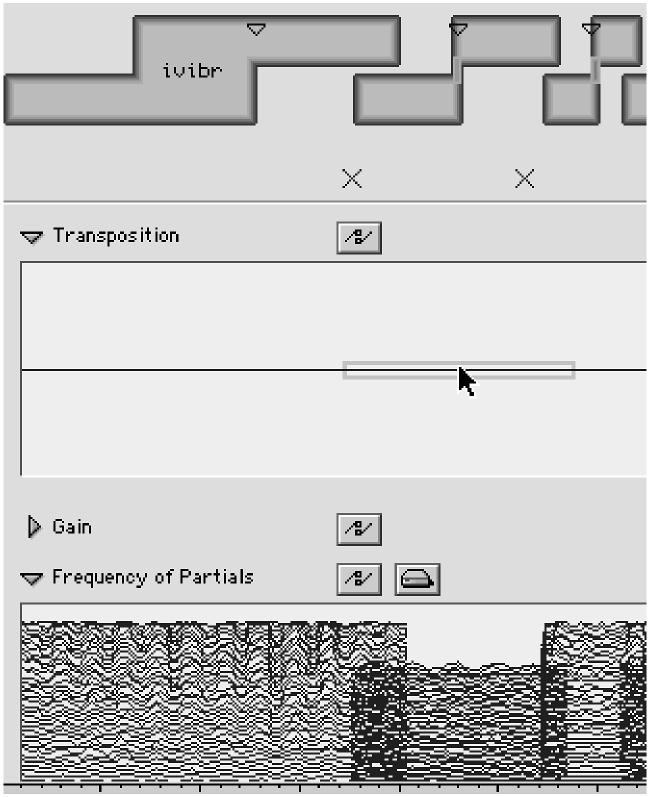


Moving the mouse over the start or end of the central zone, the cursor changes into a white double arrow with which we can change the length of the central zone.

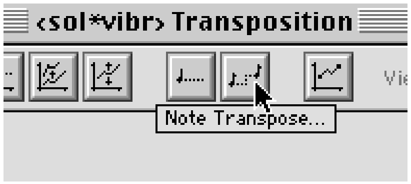


This changes the length of the interpolation between the segments. As we saw in the beginning, only the data of a segment that are within the central zone, are played in their original form. Reducing the length of the central zone, results in longer periods of interpolation, as in the following example

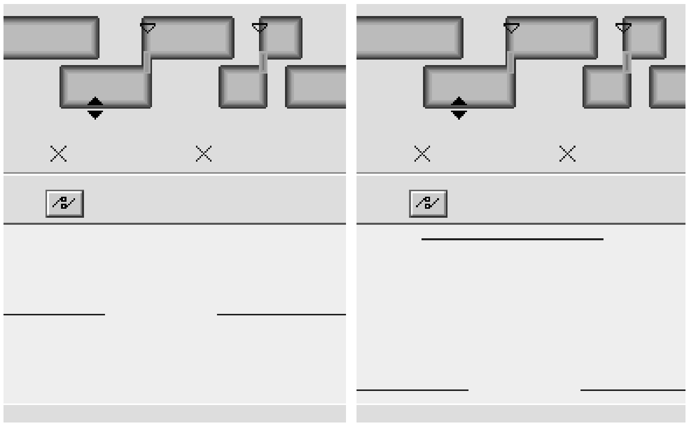
Sound example java-shaku02



With the function Note Transpose, we transpose the segment 5.5 half tones upwards, to approximately reach the same pitch as the preceding and the following segment.

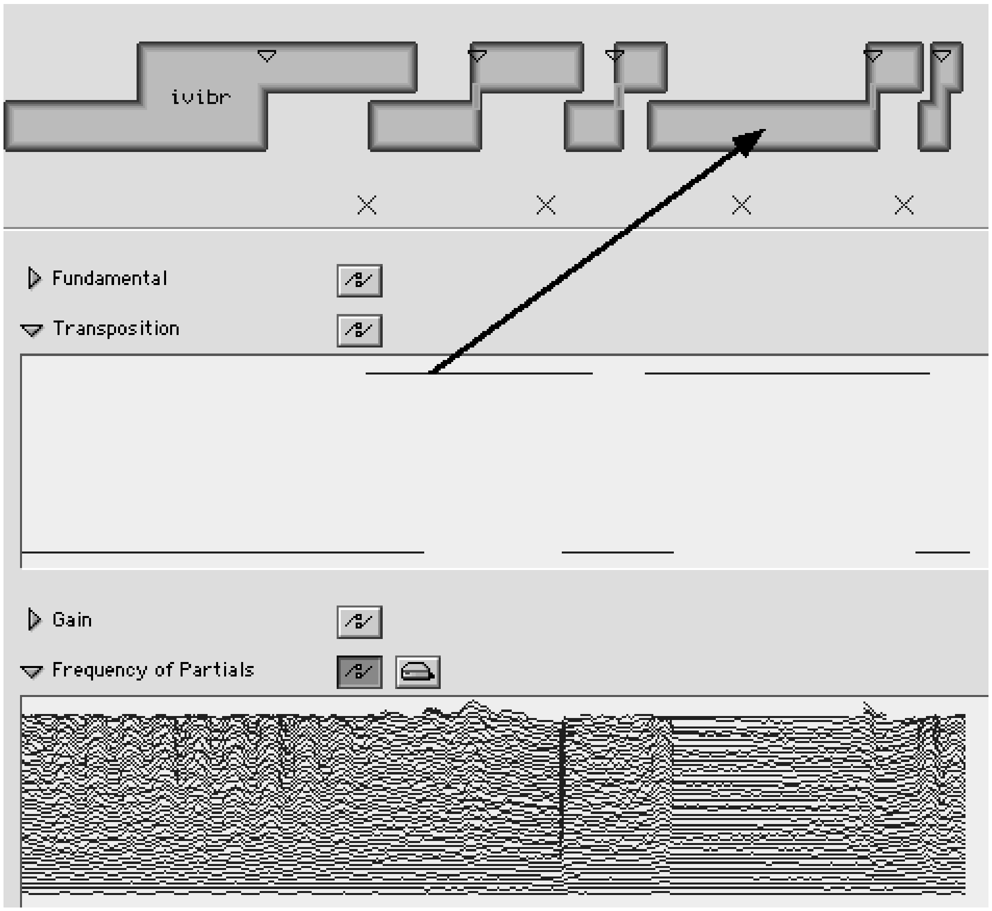


Afterwards, we have to move the mouse pointer to the lower border of a segment --- it changes into a black double arrow --- and click once to rescale the display of the edited parameter.

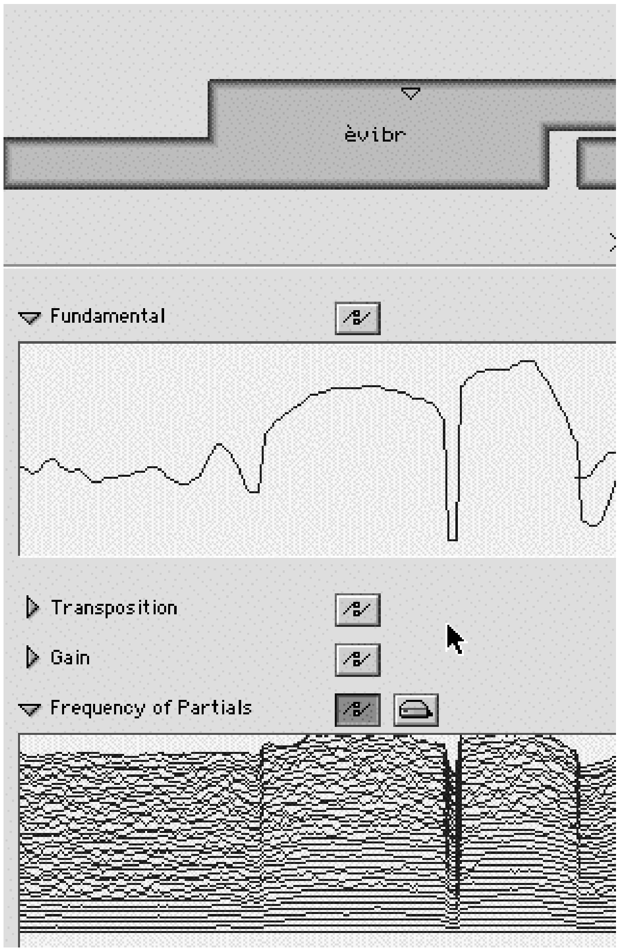


If we want to reuse the same transposition for the fourth segment, we can drag it into that segment directly.In the interpolating display mode of the frequencies we see that there are now no more glissandi between the segments.

Sound example java-shaku03



In the interpolating mode, the effects of all changes of transpositions and changes of the fundamental frequency on the partials are displayed immediately.

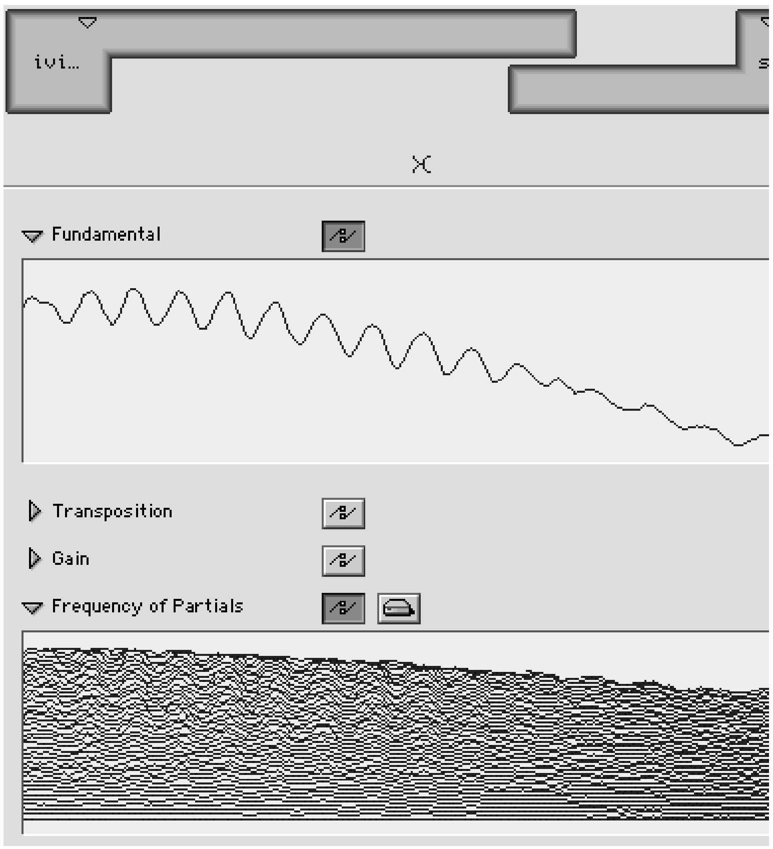

èvibr
Fundamental
Transposition
Gain
Frequency of Partials

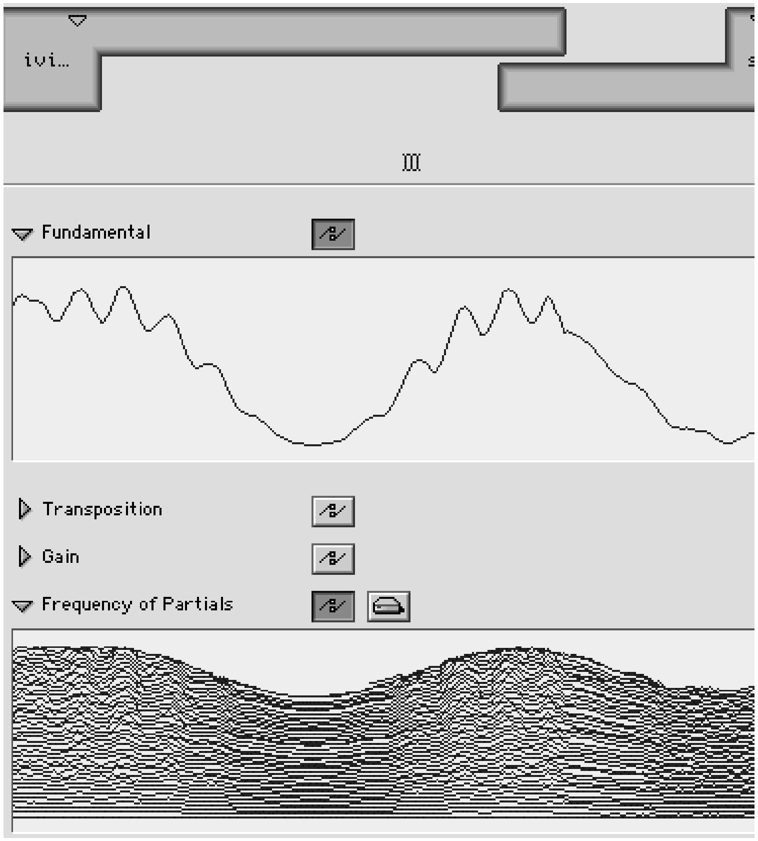
2.4. Articulations

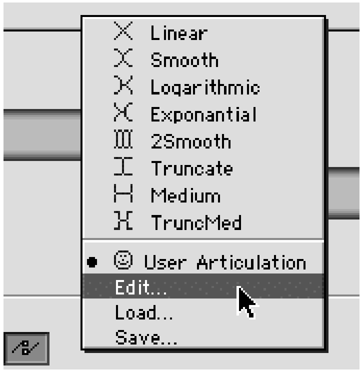
Between two segments there is a menu for the articulation of the interpolation. The following example shows the difference between an exponential and a 2 Smooth interpolation.

The latter is a sort of back-and-forth movement between the data of both segments.

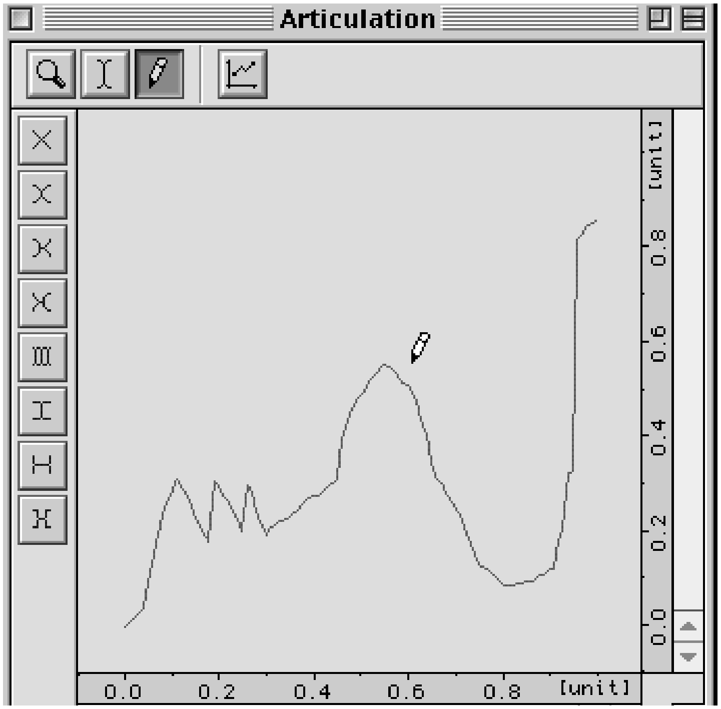
Sound example java-shaku04

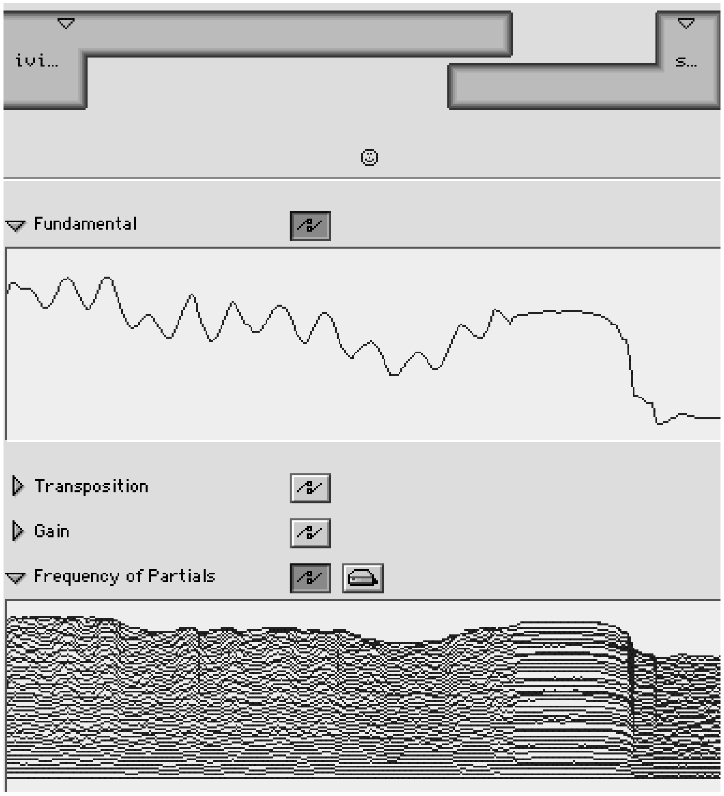






It is also possible to edit a user-defined interpolation function.





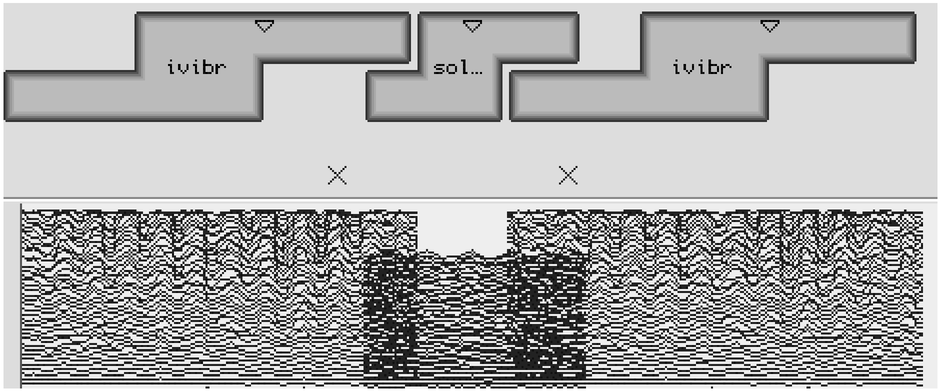
The result of this interpolation curve influences, like in the preceding examples, the interpolation of all parameters between two segments Sound example java-shaku05).

Sound example java-shaku05

2.5. Exchanging Frequencies between Segments

This example is a sequence of the original segments from the two dicos.

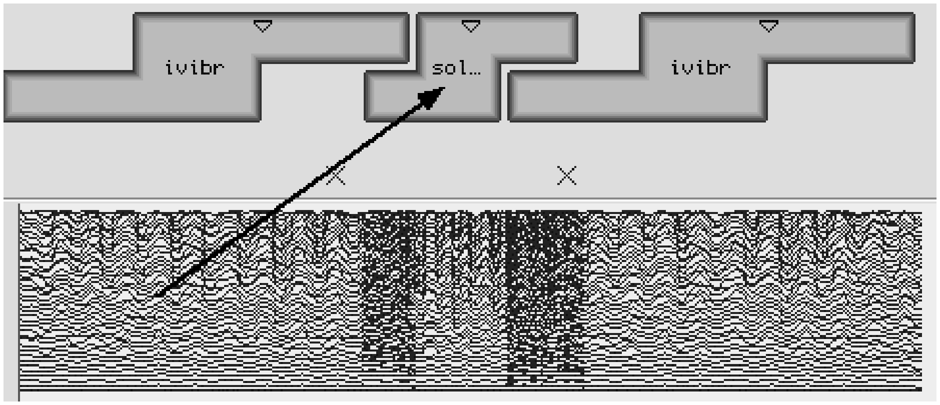
Sound example java-shaku06



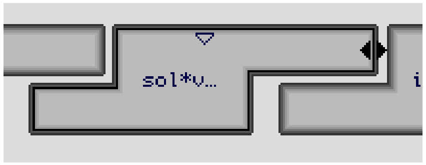
I we now drag the frequencies of the first Java segment into the following Shaku segment, the frequencies of the voice segment are played with the amplitudes of the Shaku segment.

Sound example java-shaku07

Moreover, the time axis is compressed, because the Shaku segment is shorter and the length of the data is adapted to the length of the segment.



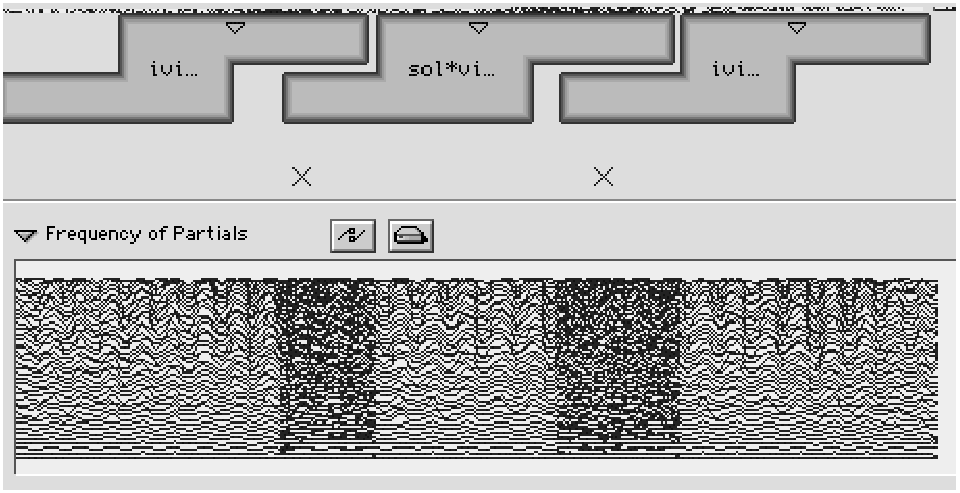
Moving the mouse pointer over the start or the end of a segment changes it into a black double arrow with which we can change the length of the segment.



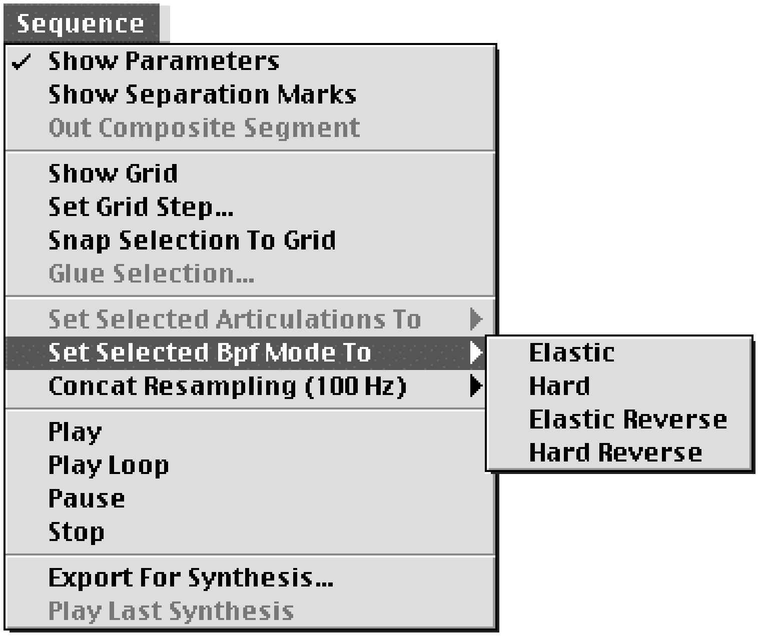
After stretching the middle segment, the frequency data of the voice are played back approximately in their original speed Sound example java-shaku08)

Sound example java-shaku08

Following this schema, all sorts of sound combinations can be generated. Amplitudes and frequencies, as well as fundamental frequencies of different segments can be combined.

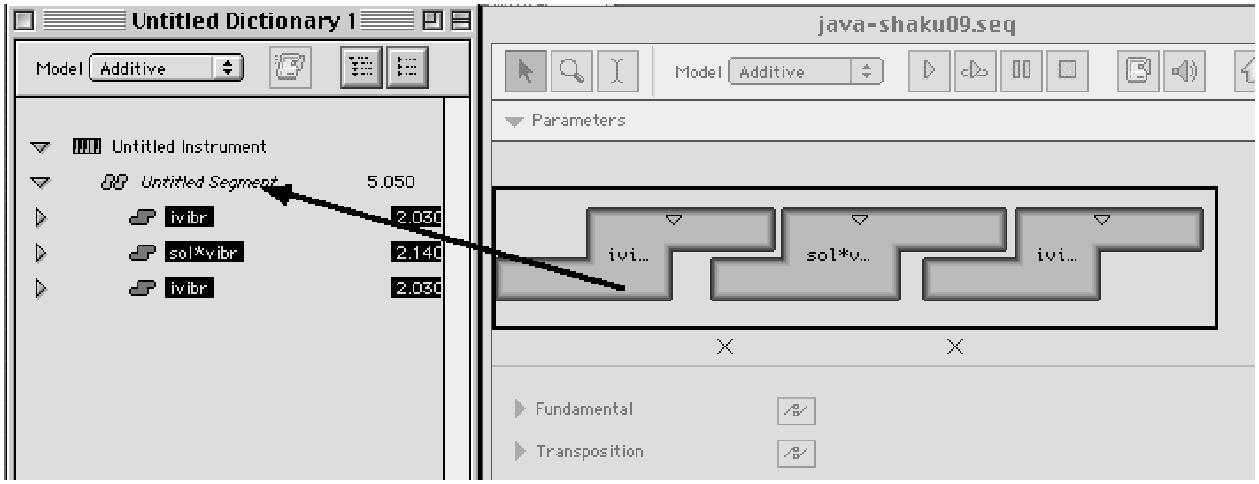


If we choose the BPF mode Hard from the menu Sequence, the data do not adapt to the length of the segment any more.

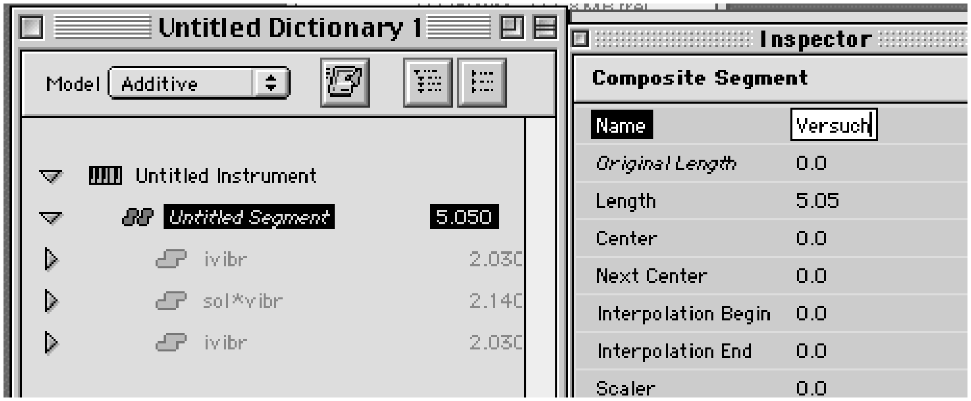


2.6. Composite Segments

From the last sequence, we will now create a so-called composite segment. We create a new dico menu File -> New Dictionary and within the dico a new instrument menu Dictionary -> Create Instrument, and then a new composite segment menu Dictionary -> Create Composite Segment. Then we select all segments of the sequence and drag them into the composite segment:



To change the name, we use the inspector:

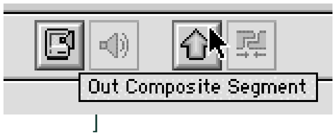


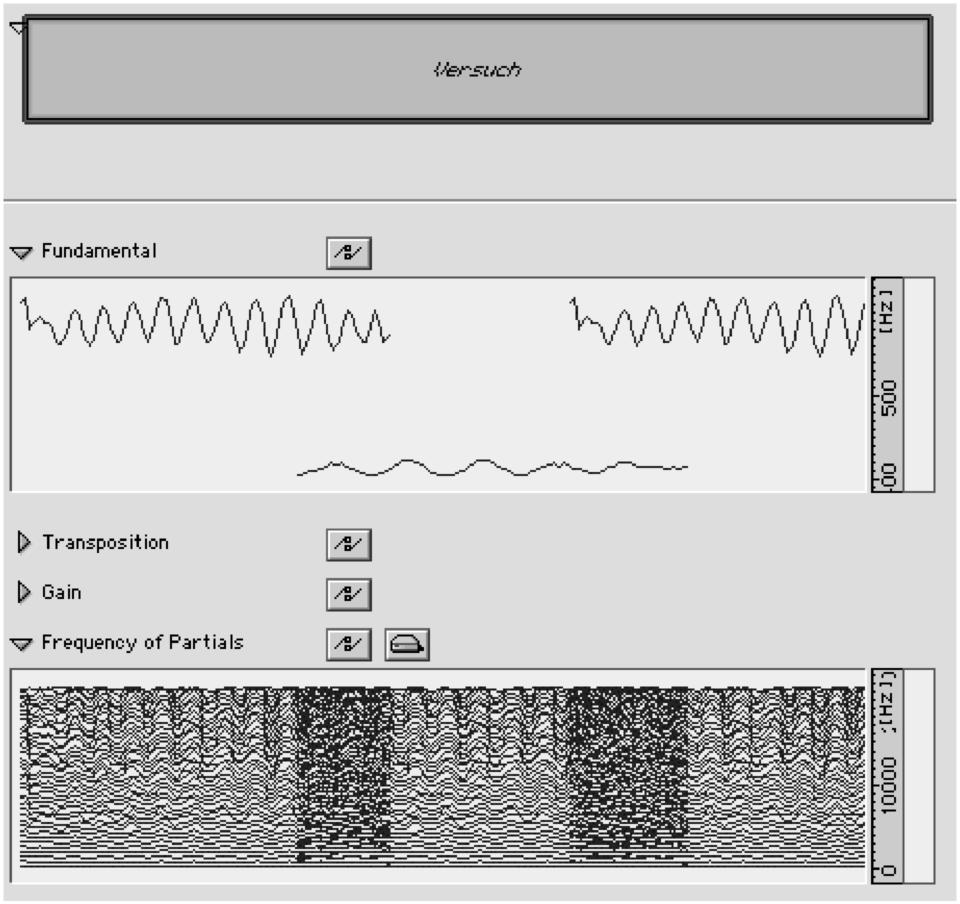
Composite Segment

Name	Versuch
Original Length	0.0
Length	5.05
Center	0.0
Next Center	0.0
Interpolation Begin	0.0
Interpolation End	0.0
Scaler	0.0

We drag this composite segment into a new sequence. It contains the data of the three single segments.

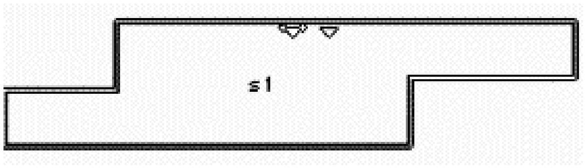
If we want to change the data within a composite segment, we can open it by double click and edit it just like a normal sequence. Clicking the upward arrow we leave the composite segment again.

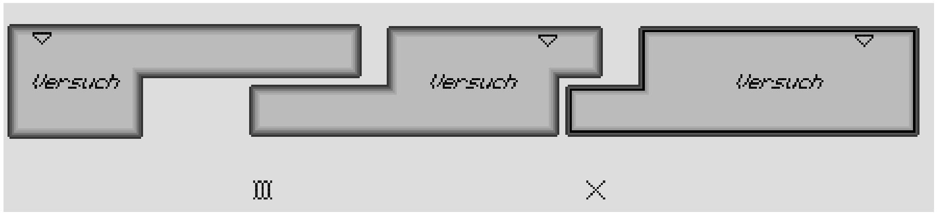




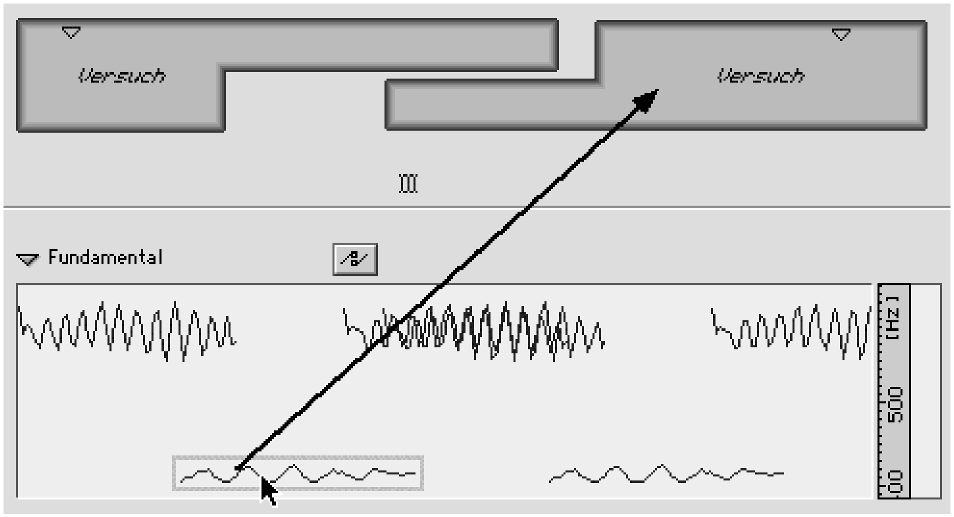
An advantage of composite segments is that they can again be used to build sequences. Here, the data within the composite segment are seen as a unit and the interpolation to the following segment starts at the end of the central zone of the composite segment.

To change the size of the central zone, we have to first move the centre while holding the Alt key.

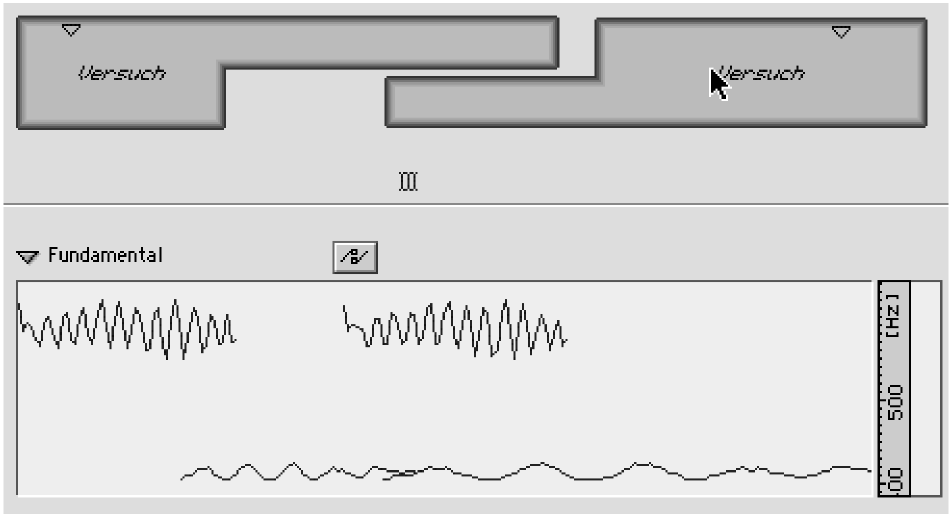




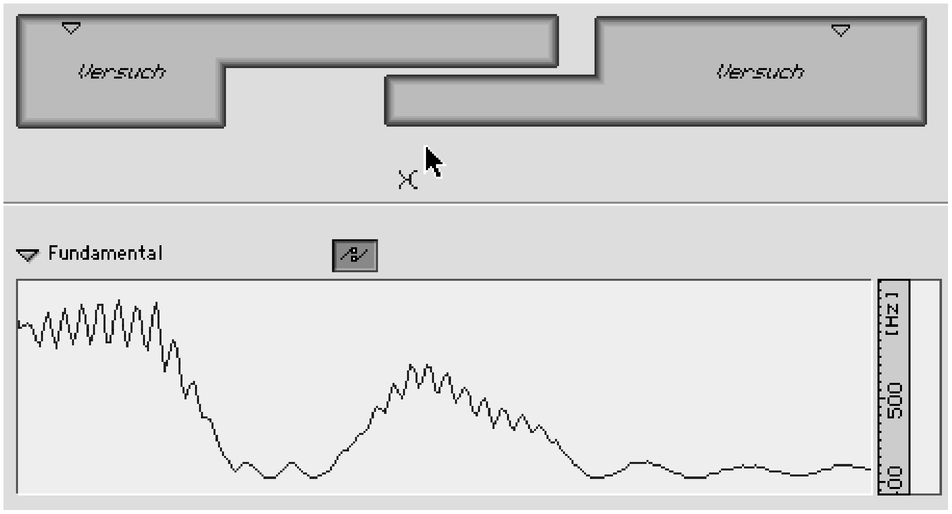
Another possibility is to change the data, which changes all data of the respective type fundamental frequency, partial frequencies, etc.) within the composite segment.



For instance, if we drag the fundamental frequency of a segment into a composite segment, it replaces the individual fundamental frequencies with the BPF.

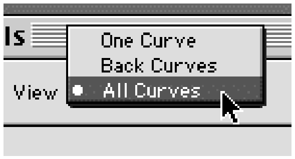


Here, the display of the data is in interpolating mode, so that the evolution of the exponential interpolation between the two composite segments is clearly visible.

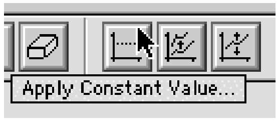


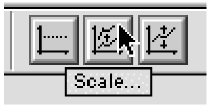
2.7. Editing Multi-BPFs

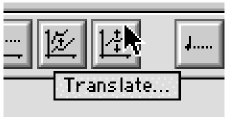
Double clicking on a parameter like frequencies or amplitudes brings up the same editor as for simple BPFs. For multi-BPFs we simply have to select which curves are to be affected by the transformations.

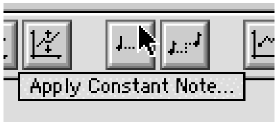


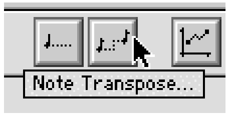
The following transformations are available:











Sets one or all curves to a fixed value.

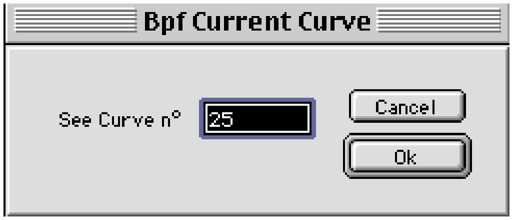
Scales the data with a factor in relation to a value. This value can be 0.0 or relative: the minimum, the average, or the maximal value of the current curve.

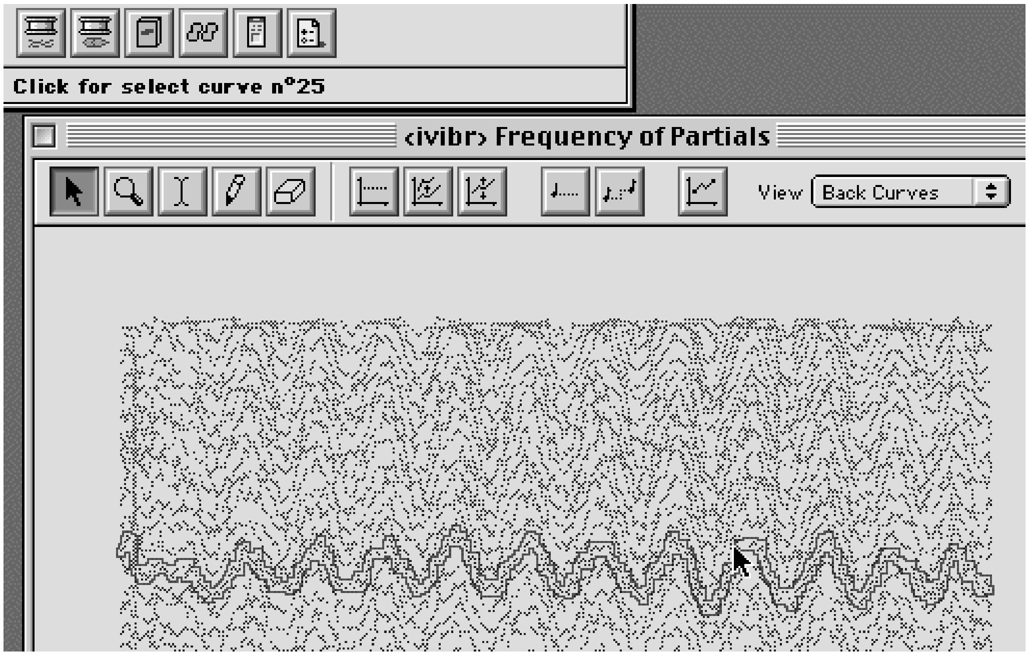
Adds a fixed value.

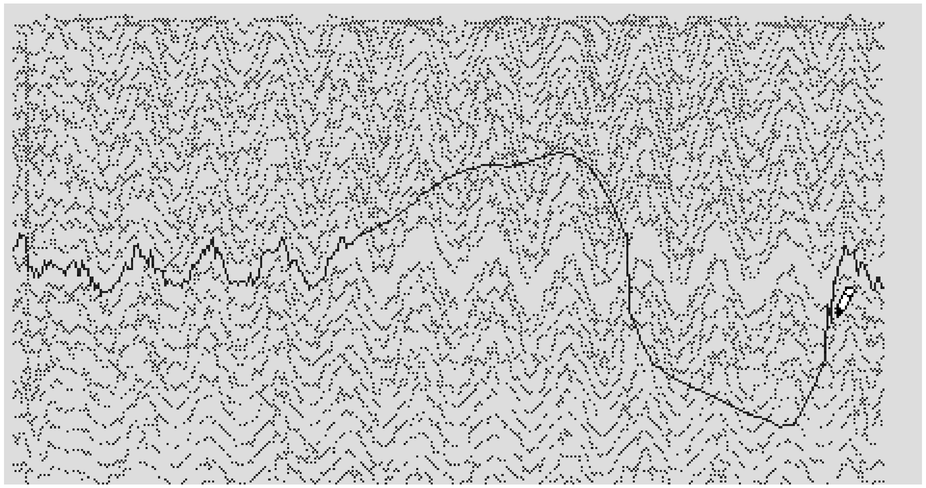
Allows the conversion of note values to frequencies and their application to the curves' values.

Transposition by intervals.

To edit single curves of a multi-BPF, we choose Back Canvas and can then either click on the desired curve, or enter an index number in the menu Bpf.

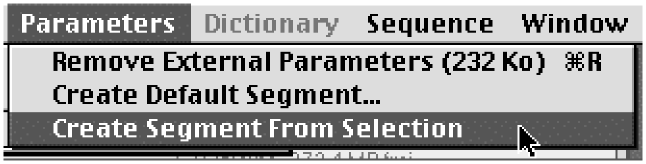


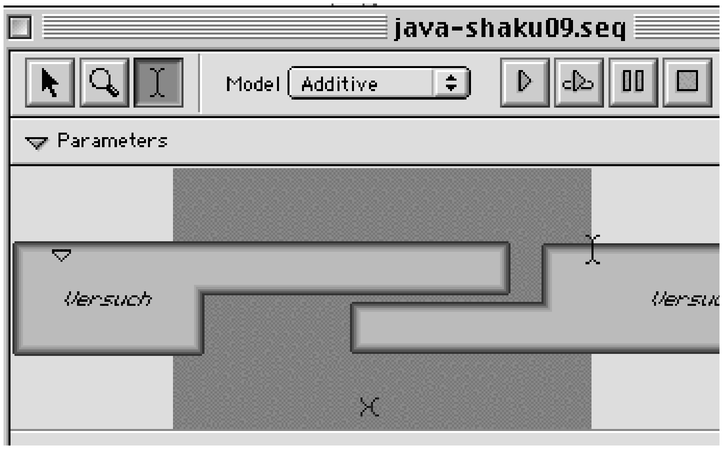

Click for select curve n°25



2.8. Creating a Segment from a Sequence or from a Part of a Sequence

If we edited a sequence, we can define it, or a part of it, as a new segment. This creates a new dico that contains only this one segment. This method can be fruitfully used when we want to edit single interpolation curves between far away segments.


Parameters
Dictionary
Sequence
Window
Remove External Parameters (232 Ko) ⌘R
Create Default Segment...
Create Segment From Selection



2.9. Known Bugs

To convert SDIF to ASCII files, we need, for the moment, the small external application SDIF/ASCII Converter 1.0b1. It can be downloaded from http://www.ircam.fr/ForumNet, or I can send it by e-mail 66KB).

Don't install the program to deeply in the directory tree, i.e. in not too many sub-folders. The paths for the sound files would get too long, and Diphone doesn't work correctly. If your structure is like this:

```
:Macintosh:Project:Music:Programs:Sound:Ircam
     :SoundDesign:Diphone
```

problems are likely to occur. In this case, try to move the Diphone folder to the top level: Macintosh:Diphone.

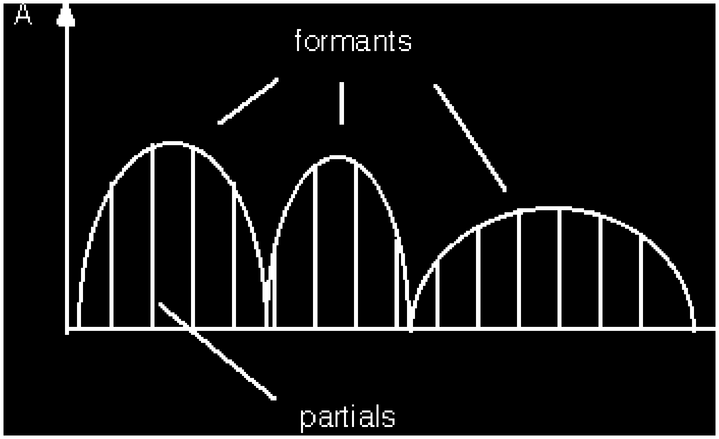
3 Analysis of Resonance Models and Chant Synthesis

The third and last part will treat in concise form the very large area of resonance models and Chant synthesis. The analysis of resonance models is realised with the program ResAn. Even without knowledge of the program Diphone, very interesting sounds can be generated with resonance models.)

Chant synthesis can be split into two functional areas, where each one can be compared with additive or subtractive synthesis, respectively. In the first case, spectra are generated by adding formants. In the second case, the formant definitions are used to filter existing sounds. The formant parameters can, in both cases, be either determined arbitrarily, or by analysing existing sounds.

3.1. Formants

The usual definition of a formant describes an area in the spectrum that has more energy than its neighbouring frequency ranges. It is thus a group of partials that ``stand out'' in the spectrum:

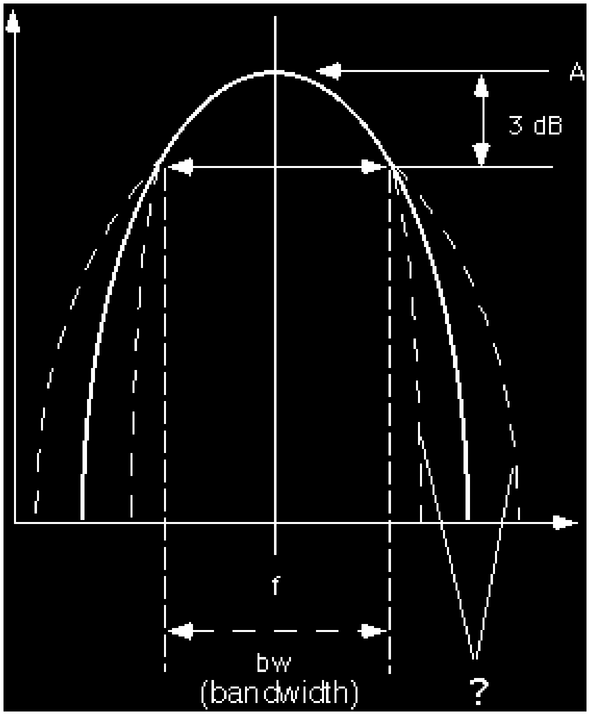


A formant is described by three parameters:

central frequency f)

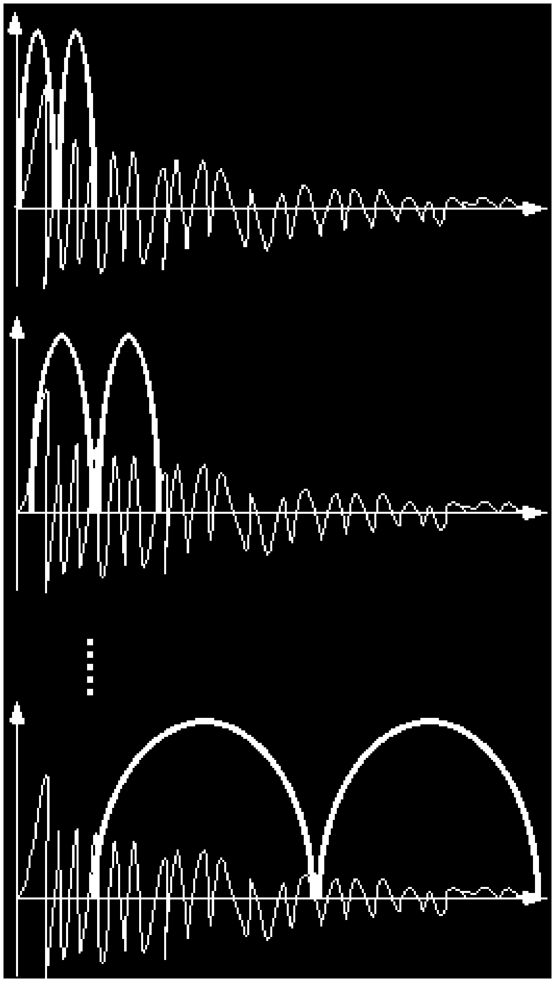
amplitude at the central frequency A)

bandwidth bw), the opening of the formant at the point where the amplitude is 3 dB lower than the maximum amplitude --- the bandwidth is thus expressed in Hz:



We see here, that the definition of a formant is rather inaccurate, because these three parameters don't define how the spectral opening behaves in the softer parts.

3.2. Analysis of Resonance Models



What we call resonance models are models of vibration, that are excited by a short burst of energy and are then let decay. The frequencies stay constant during the decay. For instance, struck or plucked sound bodies belong to these models.

Starting from a sound recording, the analysis of these models tries to estimate the resonance qualities, and to determine a series of formants with their frequencies, amplitudes, and bandwidths. This series is fixed and does not change over time.

The analysis is calculated in multiple passes. It has to describe the moment of attack, as well as the resonance. That's why the analysis uses two neighbouring windows, where the left one analyses directly the attack, and the right one the following resonance.

Both sound segments windows) are analysed by an FFT the principle of this analysis has been described in the first part of the article), and the determined frequencies spectral peaks) of both windows are compared.

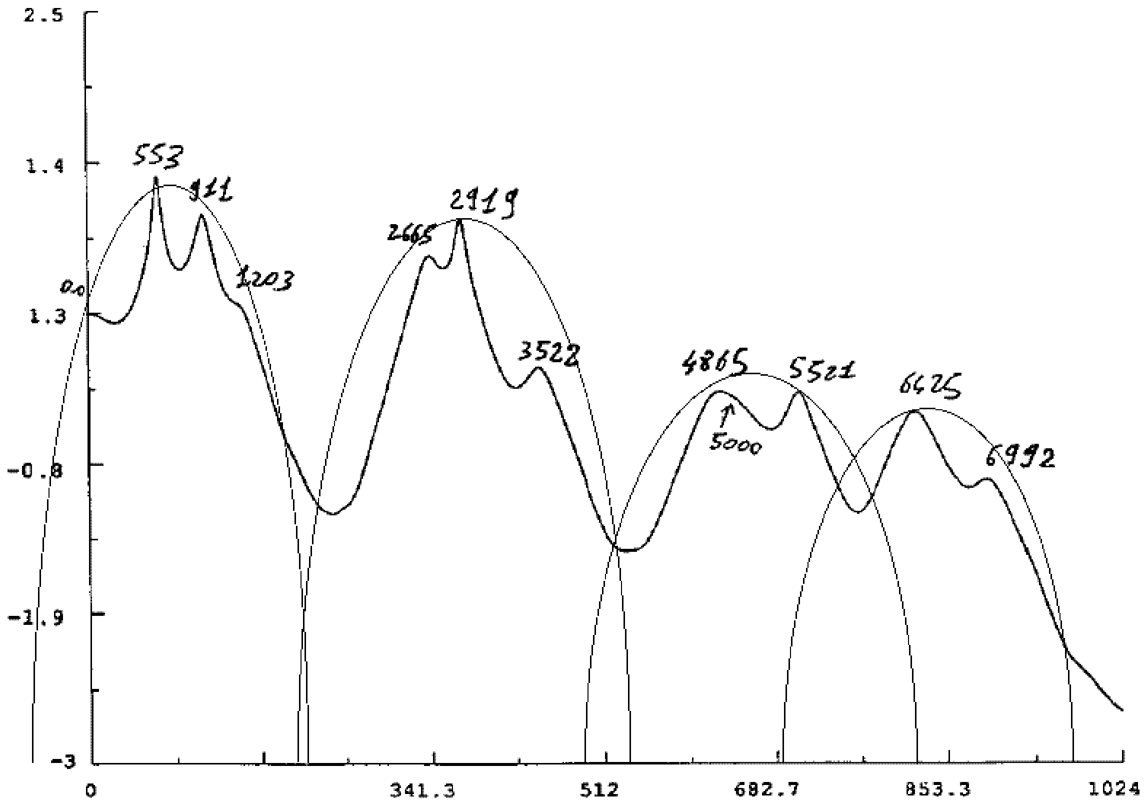
If a resonance frequency occurs in both analysis windows, we assume that it is a spectral part of the analysed model. By comparing the amplitudes, the decay time of the respective frequency can be found.

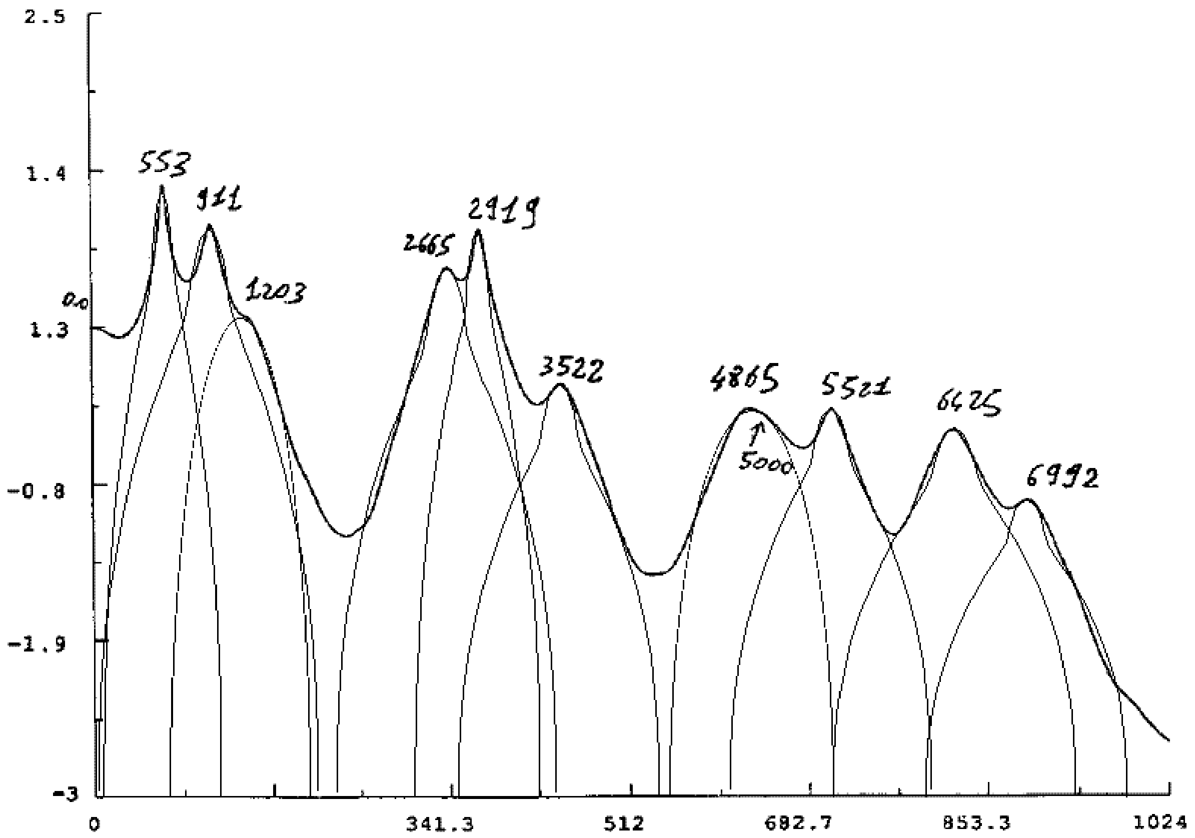
This first pass generates a so-called elementary model. In the following passes, the windows are enlarged and shifted slightly from the attack towards the resonance.

At the end of each pass, the model of the current pass is compared with the preceding model. If matching frequencies are found or frequencies very close to each other), they are included in the resulting enhanced model. Their parameters for amplitude and bandwidth are adapted after comparing the old and new values. If certain resonances do no longer occur, or new ones appear, they stay in the model, or are added, respectively.

Because the window size and the according FFT) of the first analysis passes is rather small, a quite course frequency resolution is obtained for the attack moment. This results in very broad formants, that let a large amount of energy pass. This corresponds to the natural behaviour of the models, that exhibit, at the moment of attack, a very rich spectrum, and vibrate with all ``parts'' of the vibrating body. The energy of the attack fades away quickly, so that later only the resonances of the models are left.

By enlarging the windows in every analysis pass, more and more precise frequencies can be distinguished and finer grades of formants can be reached.

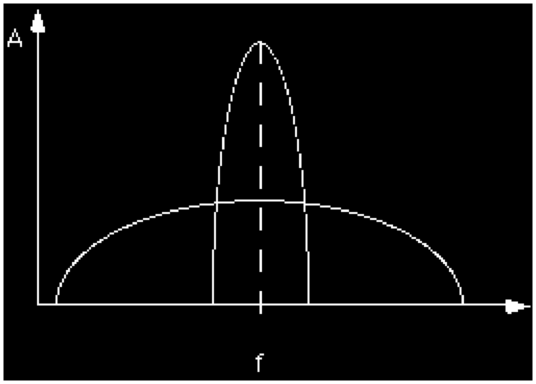

2.5
1.4
1.3
-0.8
-1.9
-3
0
341.3
512
682.7
853.3
1024
553
911
1203
0.0
2665
2919
3522
4865
5000
5521
6425
6992



The analysis of a given sound with small windows yields few, broad formants. The analysis with larger windows allows the calculation of more precise formant data. Broad formants let energy pass and resonate little, narrow formants ``take more time'' to let a given amount of energy pass and resonate thus longer. This is a simplified explanation of the relation between bandwidth and resonance time: The narrower the formant, the longer it sounds.

With enhanced models, it can happen that one frequency is represented with several formants, that have different amplitudes and bandwidths.

Then, the broader formant models the moment of attack, the narrower formant the resonance that follows:



All parameters for the comparison of elementary and enhanced models can be controlled individually approximately 25 parameters per analysis pass). A relatively simple first try is possible by using the presets. I prefer preset 6 with 6 passes, that yields very good results for a large range of input sounds:

After the 6 analysis passes, we find in the folder ImpExport:Chant the 6 SDIF files that correspond to the enhanced models of the single passes. On the dialog page Synthesis in ResAn the single passes can be resynthesised.

We can hear in this example that each analysis pass enriches the formant spectrum.

Sound example: Original: crotale

Sound example: Resynthesis: crotale.m1

Sound example: Resynthesis: crotale.m2

Sound example: Resynthesis: crotale.m6

3.3. Creation of ''out of the ordinary'' Resonance Models

The analysis method described above can not only be applied to real resonance sounds, but also longer sound sequences can be analysed that don't correspond to this model. This results in very surprising resonance sounds, that correspond to the sum of the most prominent formants of the analysed sound.

Some of my favourite sound examples for this application are, as usual, on the web site:

Original	*Resynthesis*
huhn	huhn.m1, huhn.m2, huhn.m5
bulg-2	bulg-2.m5
bulg-frau	bulg-frau.m5
sundanesie-tembang	sundanesie-tembang.m5

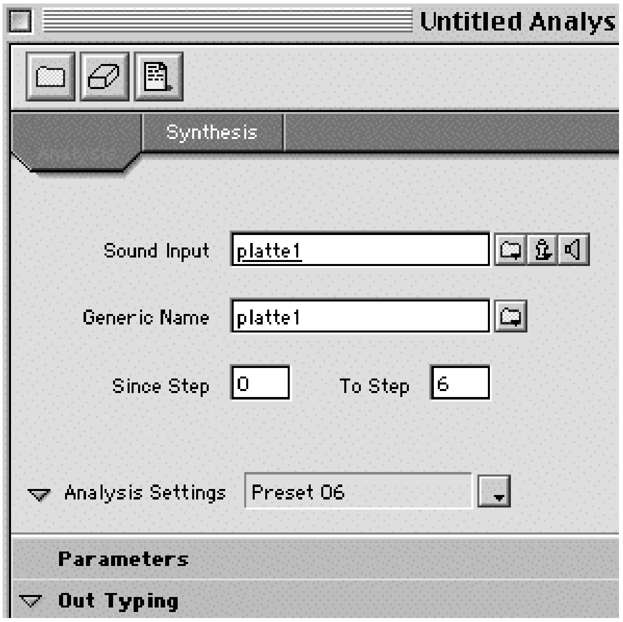
Sound example: huhn

Sound example: huhn.m1)

Sound example: huhn.m2

Sound example: huhn.m5

Sound example: bulg-2


Untitled Analys

Synthesis

Sound Input: platte1

Generic Name: platte1

Since Step: 0 To Step: 6

Analysis Settings: Preset 06

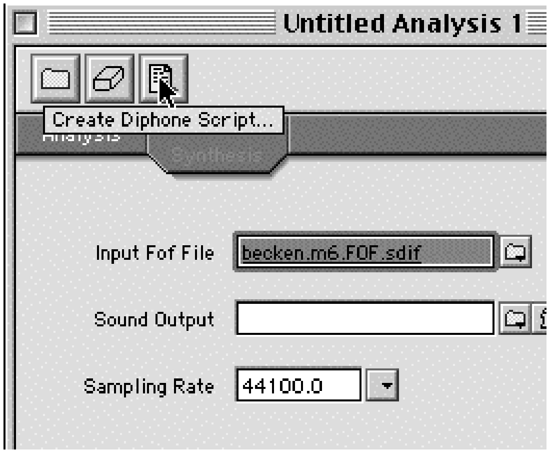
Parameters

Out Typing

3.4. Dictionary

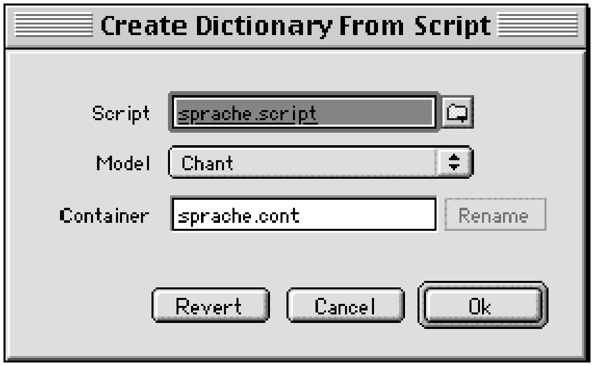
As described in the part about additive synthesis, the analysis data have to be stored in so-called dictionaries. These contain instruments and segments that can be used to create new sequences.

If we are satisfied with the resynthesis in ResAn, we can generate a script needed for the creation of the dictionary. The steps are the same as for AddAn.)



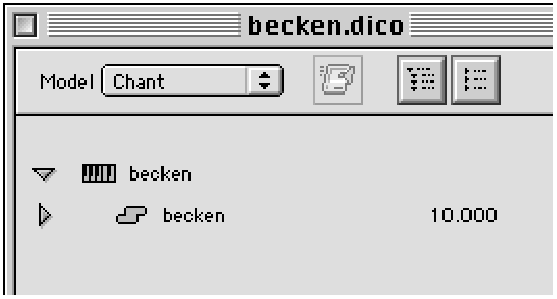
Input Fof File: becken.m6.FOF.sdif
Sound Output:
Sampling Rate: 44100.0

We generate the dictionary in Diphone that uses, in this case, the Chant model:



Script: sprache.script
Model: Chant
Container: sprache.cont

Since formant definitions don't change over the duration of the segments, no segmentation of the source sound is necessary. One single segment per sound is created, containing all formant data:



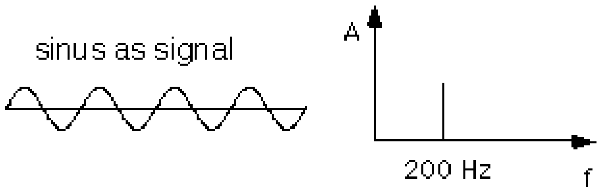
Model: Chant

becken
becken 10.000

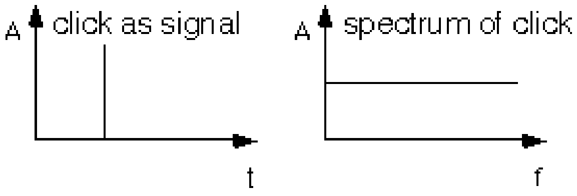
3.5. FOF-Synthesis

In FOF-synthesis, analysed formant data are used to generate purely synthetic formants. A FOF fonction d'ondes formantiques) is a sinus oscillation with an amplitude envelope. Here again we will see that there is a direct relationship between the temporal duration of a sound event and its spectral definition frequency). This is described in more detail in the first part of this article about Diphone. I will nevertheless repeat the basic principles:

Theoretically, a sinus oscillation has the spectral definition of a single frequency



The shortest possible sound, a click, has a completely flat spectrum. A frequency definition is impossible, since the sound has no period and consists of a single sample:



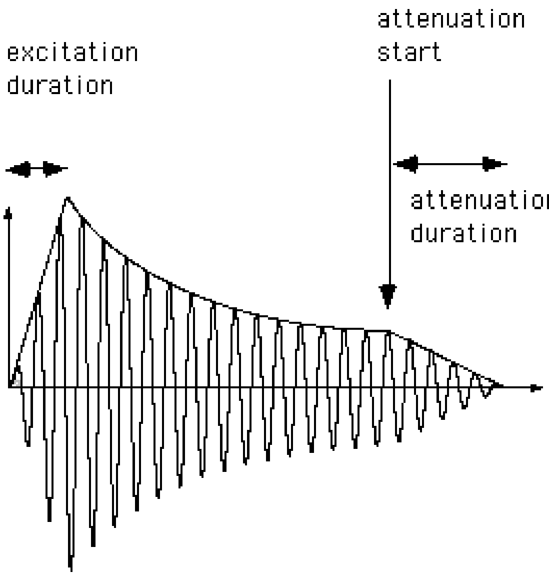
However, the spectral definition for a sinus from above is only valid if it always existed and would continue forever. The moment we start or end a sinus, we multiply the signal with an amplitude envelope. This envelope can itself be seen as a signal with its own spectrum. The multiplication of both signals sinus * envelope) results in a convolution of both spectra and the simple definition of a single frequency of a sinus is no longer valid.

We exploit this fact to use the duration of the sinus to control its spectral definition. The sinus oscillation of a FOF is subjected to the following envelope:

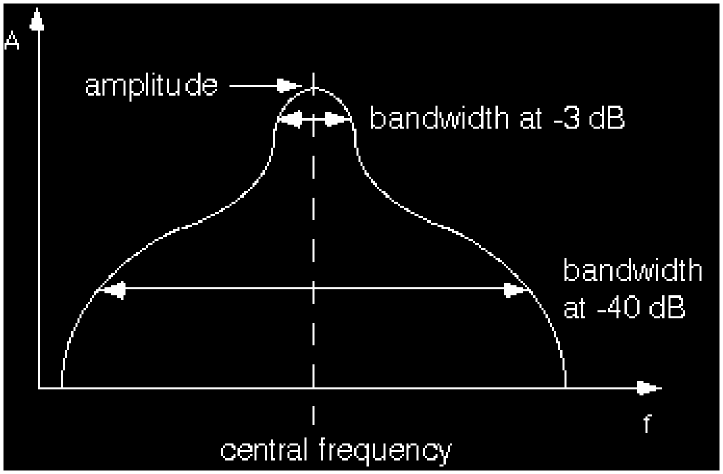
linear attack with controllable time excitation duration

immediate exponential decay

starting at attenuation start, linear decay in time attenuation duration

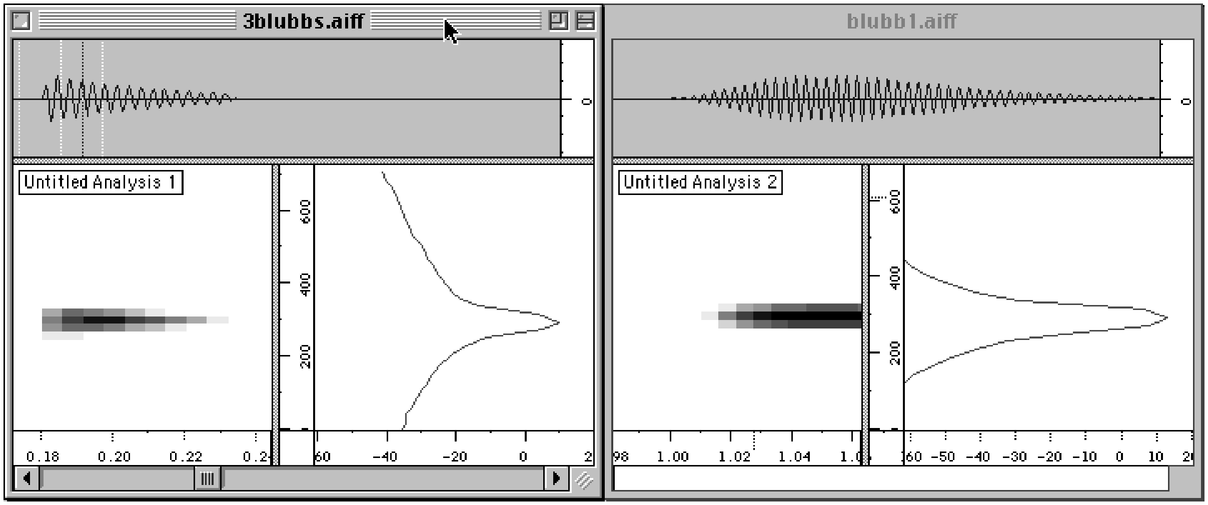


This signal results, in spectral representation, in a formant. Its spectral definition bandwidth) depends on the duration of the signal. Additionally, there is one more factor which is the bandwidth at -40 dB relative to the maximum amplitude of the formant) that depends reciprocally on the attack time: The longer the excitation duration, the narrower the bandwidth at -40 dB:



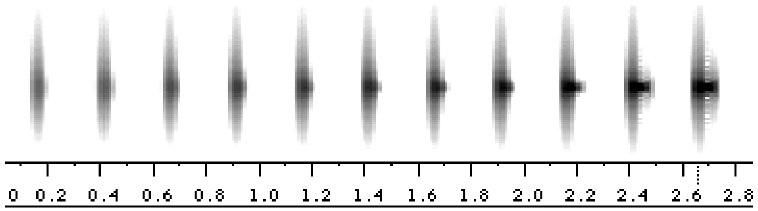
If we play such a FOF, we get a short ``bleep'', with the formant spectrum described.

In the following figure, we see the signal with its sonagram representation and the resulting formants. The central frequency of the formants corresponds here to the frequency of the sinus played. For the right image, the excitation duration has been increased and we can clearly see that the bandwidth at -40 dB is substantially narrower.

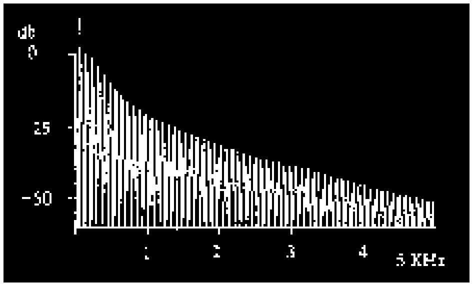


If we lower the bandwidth of the formant, its resonance time becomes longer.

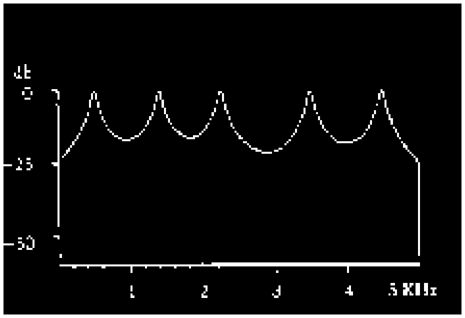
Sound example: bandwidth



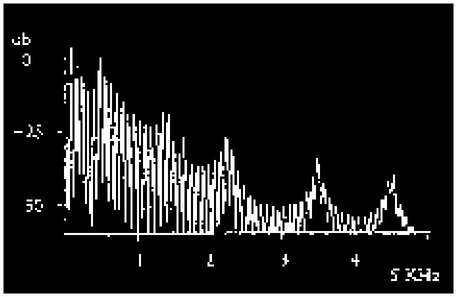
Chant has been developed at the end of the seventies at Ircam. It was primarily targeted for the synthesis of the singing voice. While singing, the vocal folds generate a rich harmonic spectrum, that is then filtered by the vocal tract.



Rich harmonic spectrum

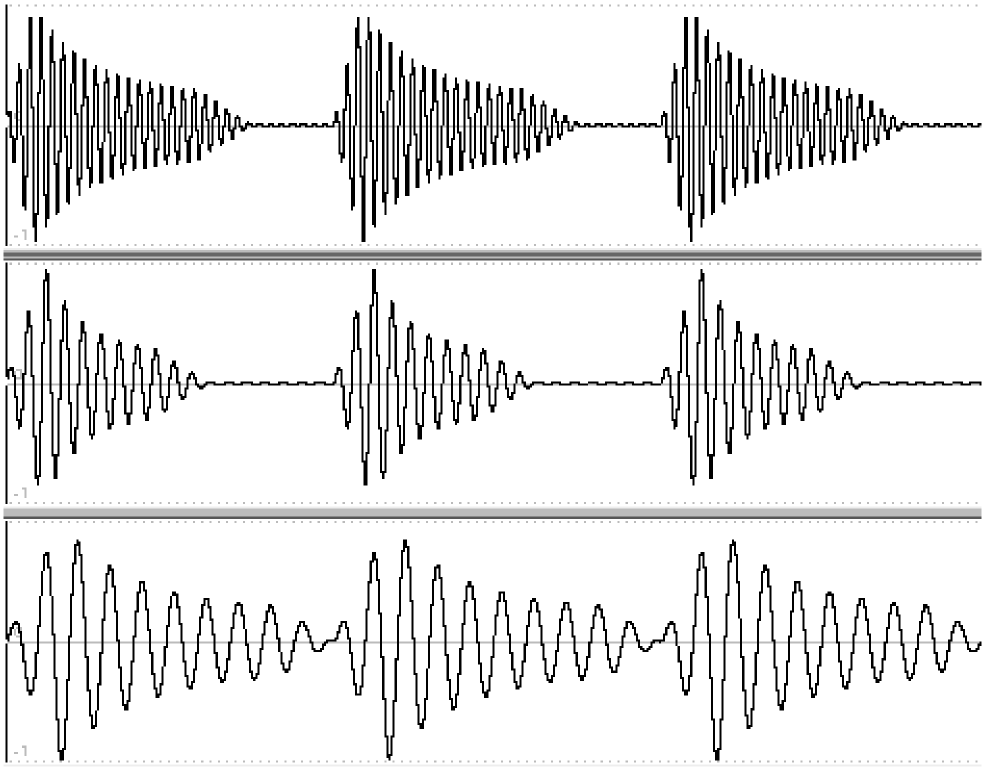


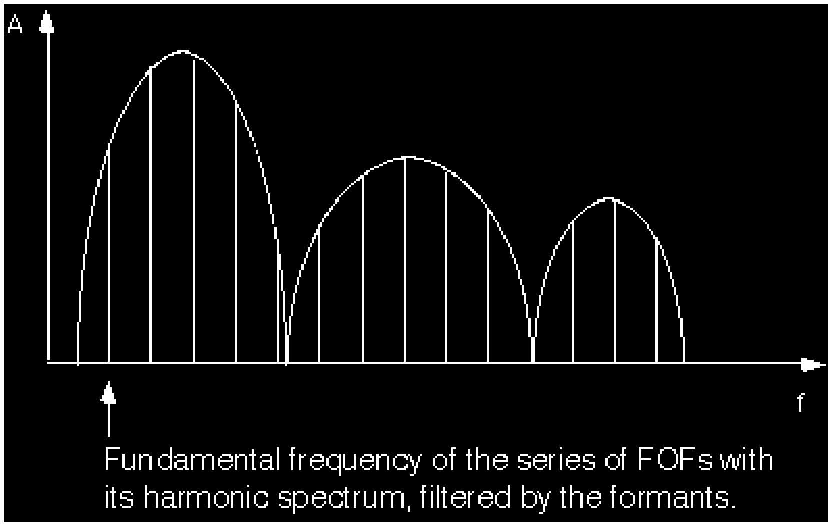
Filtering of the vocal tract



Resulting spectrum

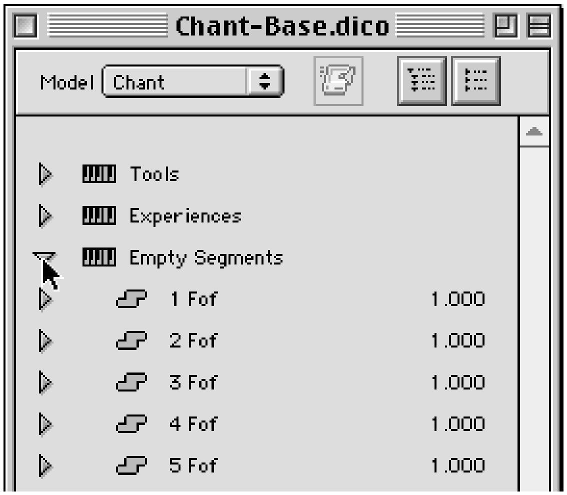
If we want to generate multiple formants with FOFs, we need multiple sinus signals in parallel that are started synchronously, i.e. have the same fundamental frequency. Each stream of sinuses generates a formant:


-1
-1
-1


A
f
Fundamental frequency of the series of FOFs with its harmonic spectrum, filtered by the formants.

The fundamental frequency the same for all the FOFs) is the basis of a harmonic spectrum, filtered by the formants. Each stream of FOFs can have its own envelope, resulting in different bandwidths of the formants. The amplitudes of the formants result from the amplitudes of the sinus signals.

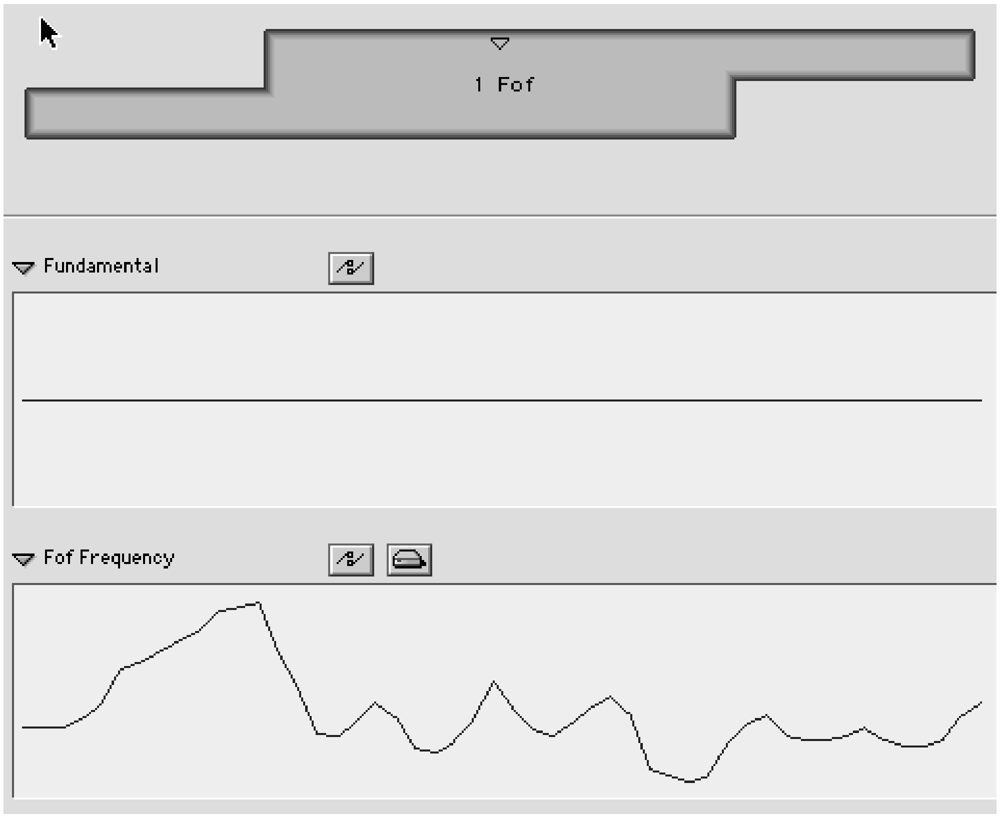
The Diphone folder dico&seq contains the Chant dictionary Chant-Base.dico that contains an instrument with 16 ``empty'' segments. From these default segments can we construct sequences of 1--16 formants from scratch:



1 Fof	1.000
2 Fof	1.000
3 Fof	1.000
4 Fof	1.000
5 Fof	1.000

As for additive synthesis, all parameters can be reached as unfoldable editors under the sequence.

Also, all manipulations of the segments within a sequence like articulation, zone of interpolation, composite segments, apply to Chant synthesis also:



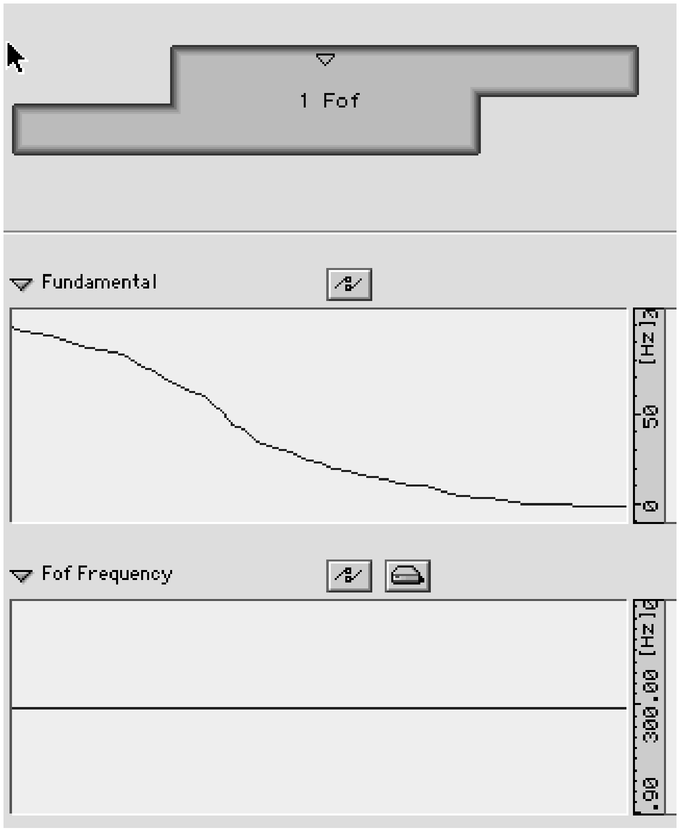
In the following, I will give a series of sound examples that illustrate the most important parameters.

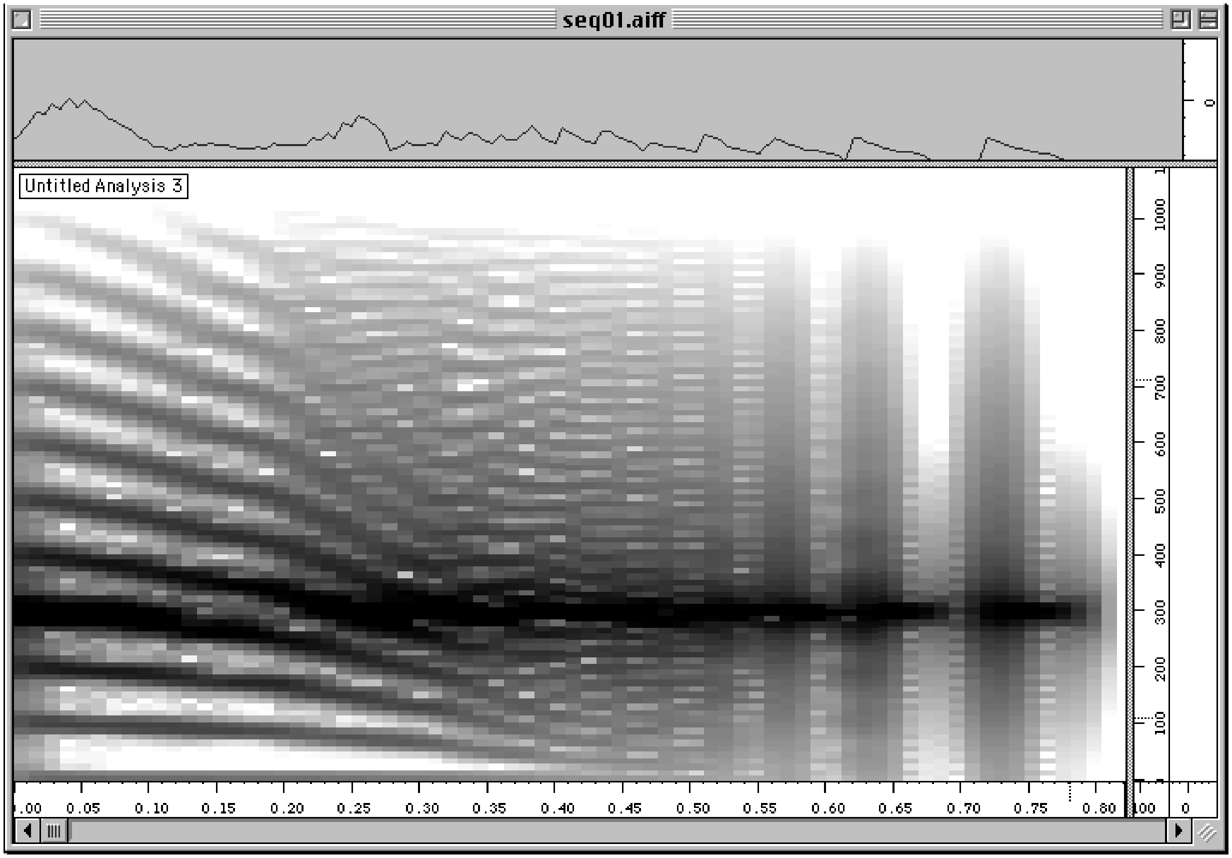
The sound example chant-seq01 is a segment with a central frequency of 300 Hz. Only the fundamental frequency, i.e. the frequency with which the individual FOFs are started, descends from 100 Hz to 0 Hz.

Below 16 Hz we hear the rhythm of the individual FOF starts.

Sound example: chant-seq01

The spectral representation clearly shows the harmonic spectrum based on 100 Hz, performing a downward glissando. We can also see the central frequency of the formant around 300 Hz:

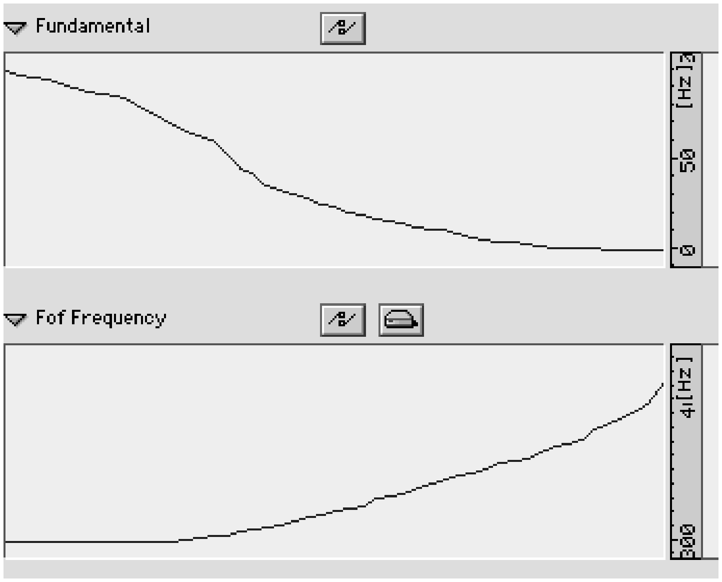


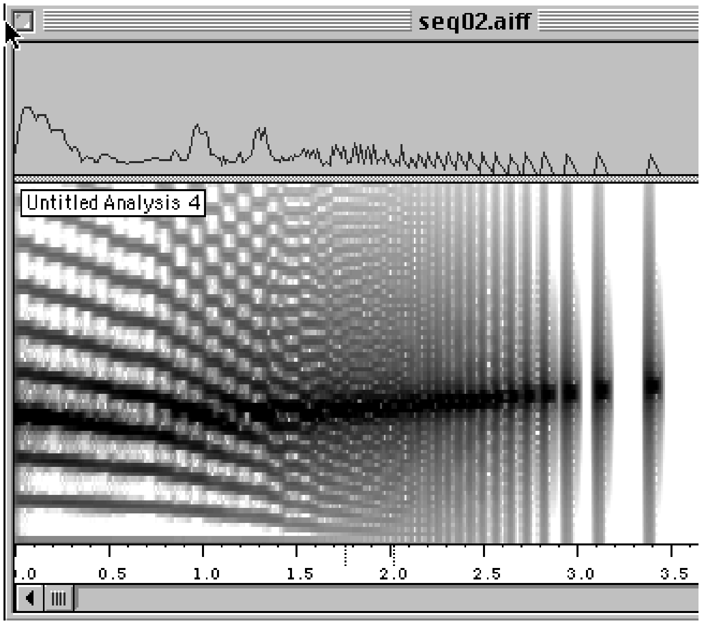


In the example chant-seq02, the fundamental frequency descends like in the previous example.

Sound example chant-seq02

However, the frequency of the formant rises from 300 to 400 Hz:





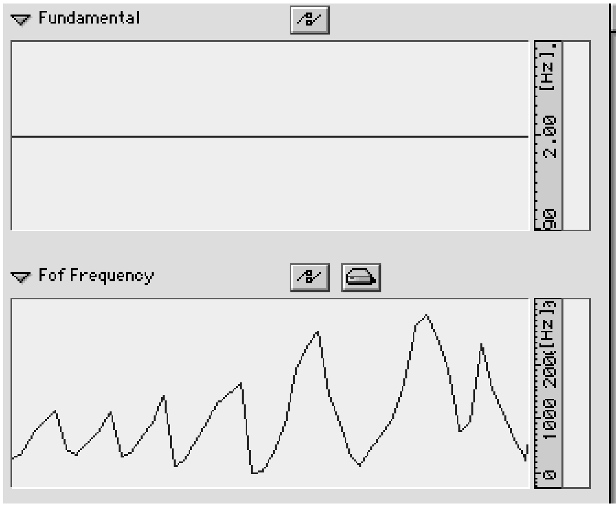
Parameter changes for FOFs are not continuous. That means that, when a FOF is started, the parameters current at that point in time are applied. During the lifetime of a FOF, the values stay constant.

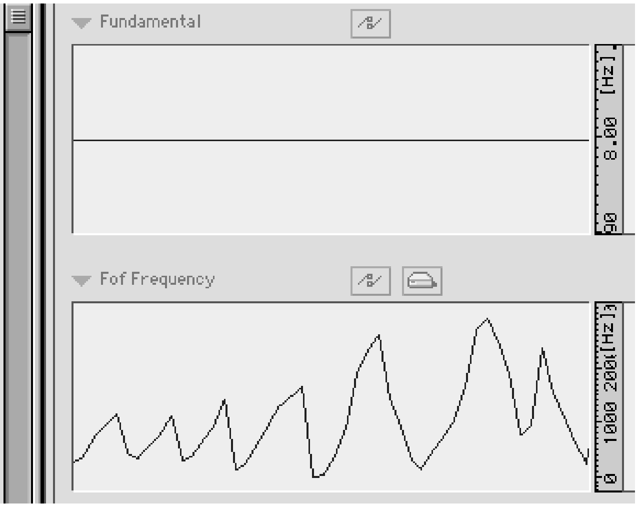
The example chant-seq03 has the same frequency evolution as chant-seq04.

Sound example: chant-seq03

Sound example: chant-seq04

But because the fundamental frequency in chant-seq03 is only at 2 Hz, the current values are read from the frequency curve only every 500 ms. The example chant-seq04 has a fundamental frequency of 8 Hz.

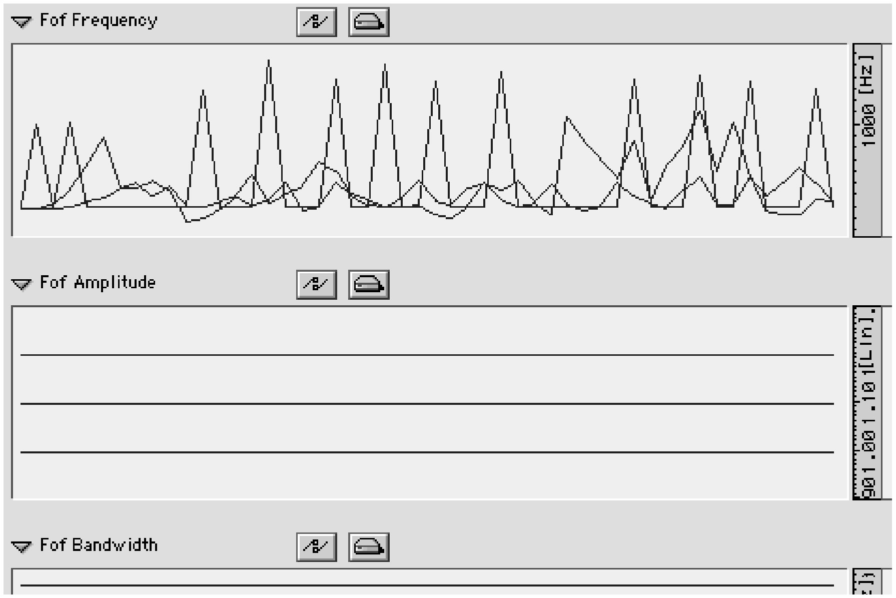




The example chant-seq05 is a sequence of three formants, the frequencies of which change continuously.

The three amplitudes and bandwidths stay constant:

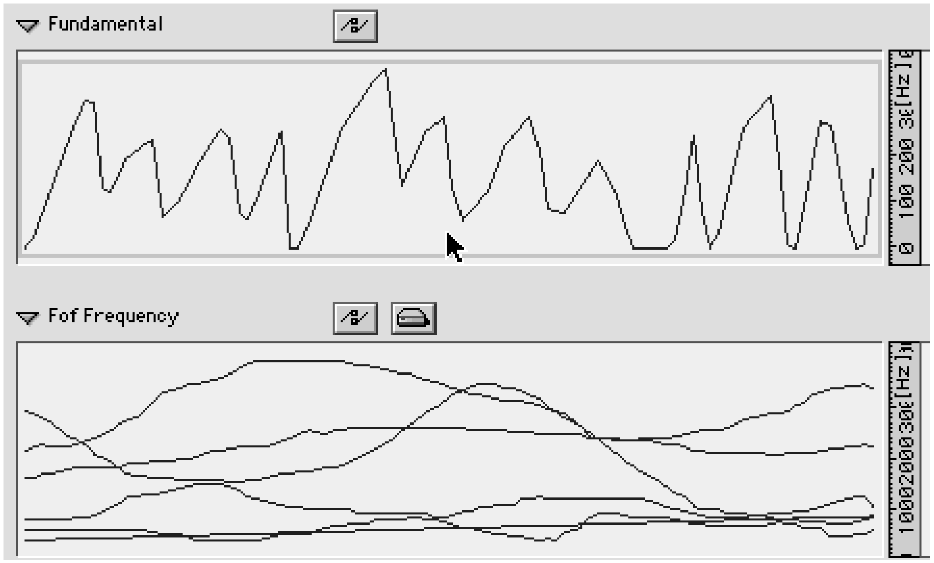
Sound example: chant-seq05



In chant-seq06, the fundamental frequency varies between 0 and 500 Hz.

Sound example: chant-seq06

Since it is a segment with 6 FOFs, at a fundamental frequency of 500 Hz 6000 FOFs are started. This results in very rich sounds, however, the computation time increases accordingly.



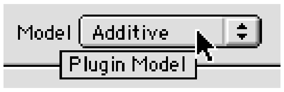
3.6. Using Additive Analyses for Chant Resynthesis

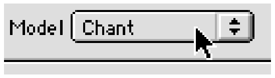
Infinitely reducing the bandwidth of a formant 0.001 Hz) practically yields a single partial. The following series of sound examples compares additive synthesis and Chant FOF synthesis.

Sound example java.additive

Sound example java.additive is the additive resynthesis of a sequence. The same analysis data can also be used as formant definitions. The frequency data of the partials then become the central frequencies of the formants, and the amplitude data of the partials the amplitudes of the formants. Only the bandwidths of the formants have to be ``invented''.

When we work with a sequence containing additive segments, we see, in the upper area of the window, the model Additive:





If we switch to Chant, the frequency and amplitude parameters are automatically used for the formants. java-as-chantfof is the Chant FOF synthesis of the same sequence with a very small bandwidth of 1 Hz. For the sound java-as-chantfof-bw1000, the bandwidth has been increased to 1000 Hz.

Sound example: java-as-chantfof

Sound example: java-as-chantfof-bw1000

If we no longer let the fundamental frequency follow the original frequency, we get only the changes in the formants, leading to very interesting sounds. java-as-f0-500-bw10 is the same sequence with a bandwidth of 10 Hz, only the fundamental frequency has been set to 500 Hz and stays constant.

Sound example: java-as-f0-500-bw10

3.7. Filter

The formant definitions can not only control the FOFs, but can also be used as filters. We can either filter white noise from a built-in noise generator) or an external sound file.

The example java-as-filter-noise is the result of filtering white noise with the formant data of the preceding sequence.

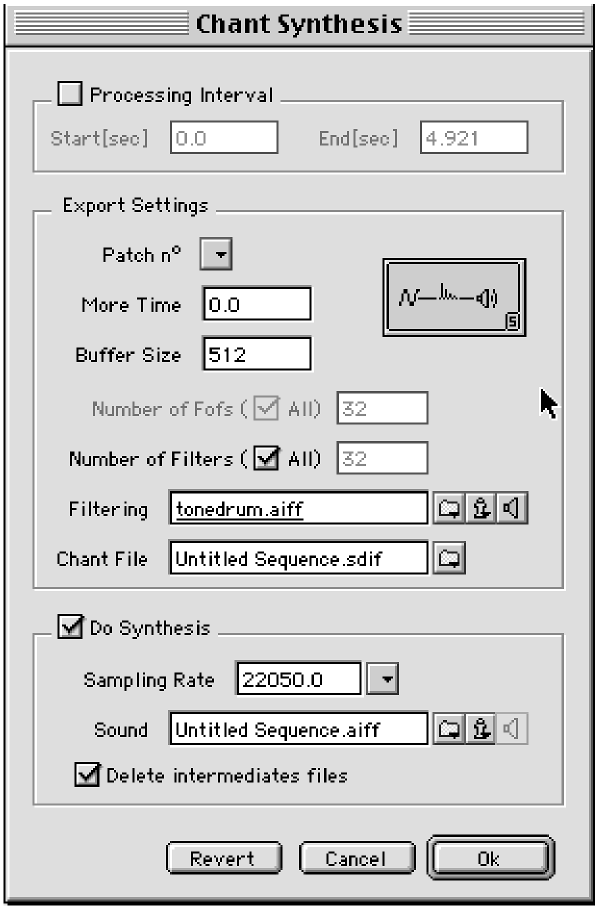
The sound file tonedrum has been filtered with the same formants as the example chant-seq06. Contrary to FOFs, the parameter changes of the filters are continuous, that means we get glissandi of the formants. The result is tonedrum.filtered.

Sound example: java-as-filter-noise

Sound example: tonedrum

Sound example: tonedrum.filtered

During resynthesis we have to specify which patch Chant should use. Up to now we have only used the first patch, that connects the FOF synthesis directly to the ``loudspeaker''. All sorts of combinations of FOFs, filtering noise, and filtering sound files can be chosen:



Chant Synthesis

☐ Processing Interval

Start[sec] 0.0 End[sec] 4.921

Export Settings

Patch n°

More Time 0.0

Buffer Size 512

Number of Fofs (✓ All) 32

Number of Filters (✓ All) 32

Filtering tonedrum.aiff

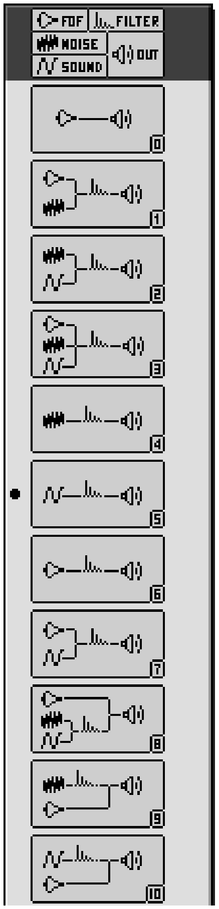
Chant File Untitled Sequence.sdif

✓ Do Synthesis

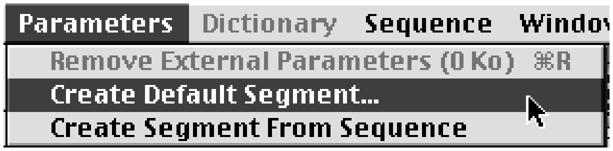
Sampling Rate 22050.0

Sound Untitled Sequence.aiff

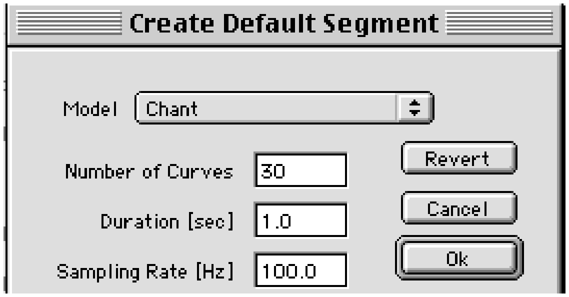
✓ Delete intermediates files



To use a sequence as a formant filter, the segments have to contain the parameters for the filters. If this is not the case, we can generate a so-called default segment, that contains all Chant parameters.

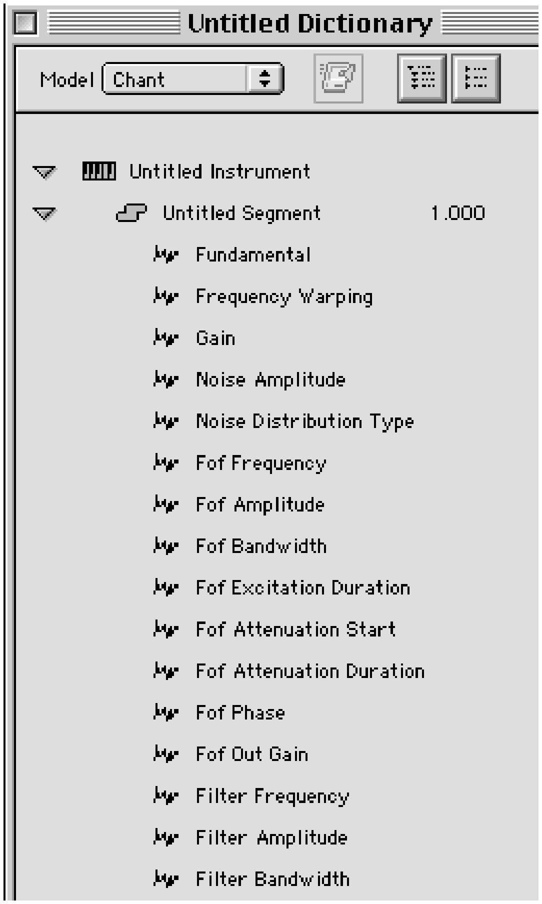

Parameters
Dictionary
Sequence
Windo
Remove External Parameters (0 Ko) ⌘R
Create Default Segment...
Create Segment From Sequence

What is important here is to select the Chant model for the default segment to be created.


Create Default Segment

Model: Chant
Number of Curves: 30
Duration [sec]: 1.0
Sampling Rate [Hz]: 100.0

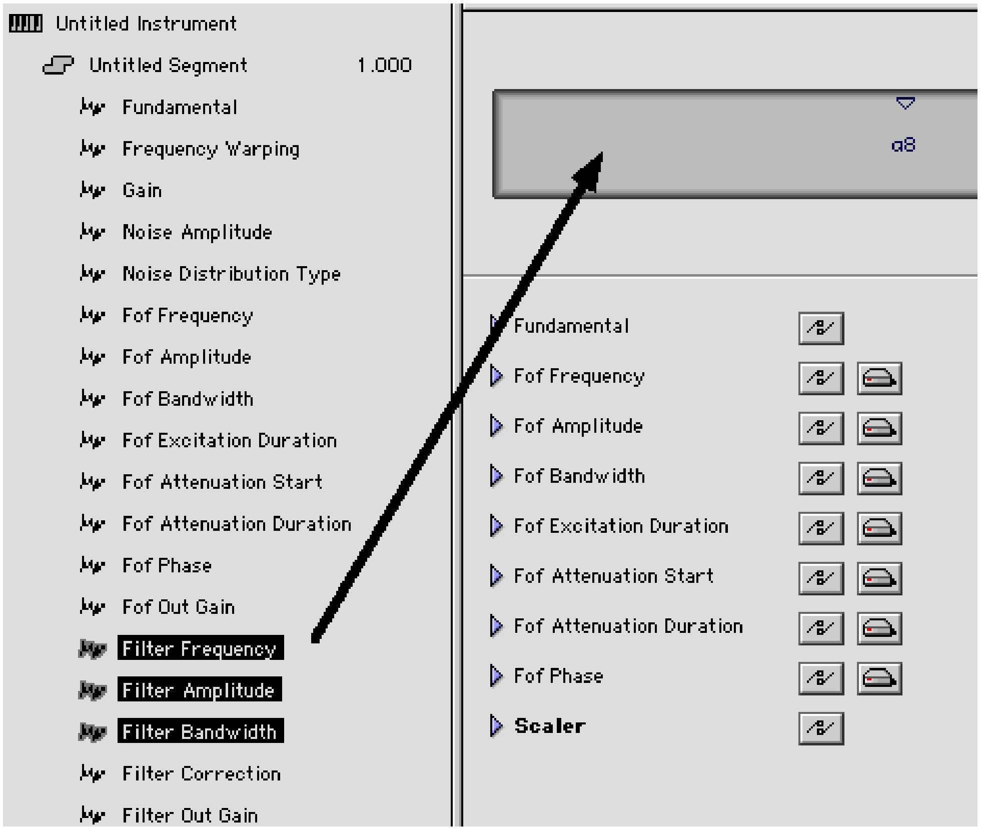
In the resulting new dictionary we find a segment with all Chant parameters:


Untitled Dictionary

Model: Chant

Untitled Instrument
Untitled Segment 1.000
Fundamental
Frequency Warping
Gain
Noise Amplitude
Noise Distribution Type
Fof Frequency
Fof Amplitude
Fof Bandwidth
Fof Excitation Duration
Fof Attenuation Start
Fof Attenuation Duration
Fof Phase
Fof Out Gain
Filter Frequency
Filter Amplitude
Filter Bandwidth

The necessary ones can be selected and dragged into the desired segment:



This way, we can create segments with all possible combinations of parameters.

Which ones are necessary depends of course on the selected Chant patch.

4 Conclusion

I hope that with this three part article I could give an insight into working with Diphone Studio --- a program that I consider very flexible and that allows some very unique possibilities of sound transformation. It is still in development and the internal) versions sometimes change within weeks. However, I think that with the possibilities of resonance and additive analysis now available, and with the editing capabilities in Diphone, a broad range of extraordinary facilities for sound transformation can be accessed.

The current development concentrates on the integration of new synthesis algorithms and better editing capabilities for multi-BPFs.

LIST OF SOUND EXAMPLES

INDEX