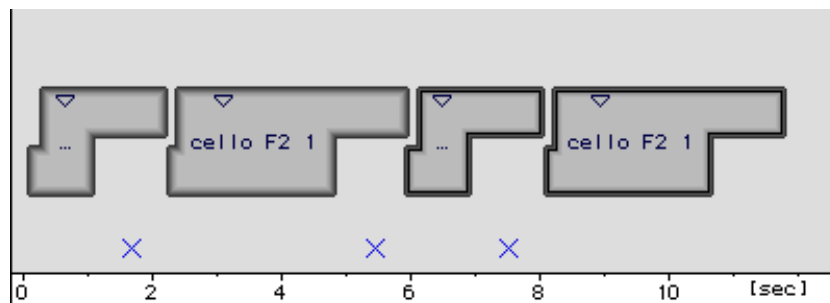


## Diphone Studio - Applications

### Synthèse additive Chant Modèles de résonance

Par Hans Tutschku



version originale : 2000  
version française : mars 2002

# 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

Apple Macintosh is a trademark of Apple Computer, Inc.

Diphone, SuperVP, ModRes, Sdif et Pm are trademarks of Ircam

© 2002, Ircam - Centre Pompidou



# Contenu

Diphone Studio - Applications	
Synthèse additive	
Chant	
Modèles de résonance .....	1
<b>Introduction .....</b>	<b>2</b>
<b>Analyse et synthèse additive .....</b>	<b>3</b>
Réglages systèmes .....	3
AddAn – Analyse additive.....	4
Analyse de Fourier (FFT) et spectre .....	5
Analyse de la fréquence fondamentale.....	12
Réduction en partiels harmoniques .....	16
Resynthèse et extraction de la fraction bruitée .....	17
Amélioration de la détection de la fraction bruitée.....	18
Segmentation du son original .....	20
Construction d’une bibliothèque contenant les segments.....	21
<b>La création de séquences .....</b>	<b>22</b>
Quelques définitions .....	22
Visualisation et édition des paramètres.....	24
Interpolation entre les segments .....	28
Articulations .....	33
Échanges des fréquences entre segments .....	35
Les segments composites .....	37
Édition de multi-BPFs .....	41
Créer un segment à partir d’une séquence ou d’une partie de séquence.....	43
Problèmes connus .....	43
<b>Analyse par Modèles de Résonance et synthèse avec Chant.....</b>	<b>44</b>
Les formants .....	44
Analyse par Modèles de Résonance .....	46
Création de modèles de résonance « hors du commun » .....	50
Dictionnaires .....	51
Synthèse par FOF .....	52
Utilisation de l’analyse additive pour des resynthèse avec Chant.....	64
Filtres .....	65
Conclusion .....	68
Liste des exemples sonores.....	69



# Diphone Studio - Applications

## Synthèse additive

### Chant

## Modèles de résonance

Par Hans Tutschku

## Présentation de l'ouvrage

Ce manuel a été initialement publié en allemand sous la forme de trois articles parus dans *Mitteilungen*, journal of the Deutsche Gesellschaft Für Elektroakustische Music, issues 35-36-37, PFAU Neue Musik, Saarbrücken, 2000

Traduction anglaise par Diemo Schwarz.

Traduction française par Laurent Pottier.



# Introduction

Le terme Diphone Studio regroupe un ensemble de trois programmes développés depuis quelques années à l'Ircam pour ordinateurs Macintosh et dont les fonctions sont interdépendantes. Les deux programmes d'analyse AddAn et ResAn permettent d'obtenir des représentations réduites de fichiers-sons, représentations qui peuvent ensuite être utilisées avec l'interpolateur de Diphone pour la production de nouveaux matériaux sonores musicaux.

Diphone est l'interface utilisateur pour interpoler des segments sonores afin de créer de nouvelles séquences musicales. L'interface graphique contient de nombreux éditeurs et constitue un noyau sur lequel peuvent être connectés divers plug-ins de synthèse, comme par exemple ceux pour la synthèse additive ou pour la synthèse avec Chant.

L'éditeur de dictionnaires permet de choisir des segments dans des bibliothèques de dictionnaires de segments, de construire de nouvelles bibliothèques et de les stocker en mémoire.

L'éditeur de séquences permet de construire de nouvelles séquences à partir de divers fragments dont les paramètres sont modifiables à volonté (le temps et les fonctions d'interpolation d'un segment au suivant etc.).

L'éditeur BPF (BPF = Break Point Function, fonction linéaire par segment) est un éditeur de courbes. Toutes les valeurs de fréquences, amplitudes, phases, transpositions, etc. peuvent être affichées dans ces BPF.

Les segments à partir desquels les nouvelles séquences seront fabriquées sont regroupés pendant l'analyse dans des dictionnaires. Il est possible de travailler simplement avec les dictionnaires fournis comme références avec le programme Diphone Studio et de créer de nouveaux sons avec les segments qu'ils contiennent, ou bien on peut utiliser les programmes d'analyse pour créer de nouvelles bibliothèques à partir de ses propres sons.

AddAn est un programme pour l'analyse additive et ResAn un programme pour l'analyse par Modèles de Résonance.

Puisque l'ensemble des programmes est relativement volumineux et contient plusieurs modèles de synthèse et leurs analyses associées, ce texte est divisé en plusieurs parties.

La première partie décrit les possibilités offertes par l'analyse additive en utilisant le programme AddAn, comment créer ses propres dictionnaires, ainsi que les manipulations de segments additifs dans Diphone.

La deuxième partie décrit la fabrication des segments.

Enfin, la synthèse avec Chant, avec des filtres résonants ainsi que l'analyse par Modèles de Résonances sont décrites dans la troisième partie.

Une liste d'exemples sonores et un index général sont donnés à la fin du texte.

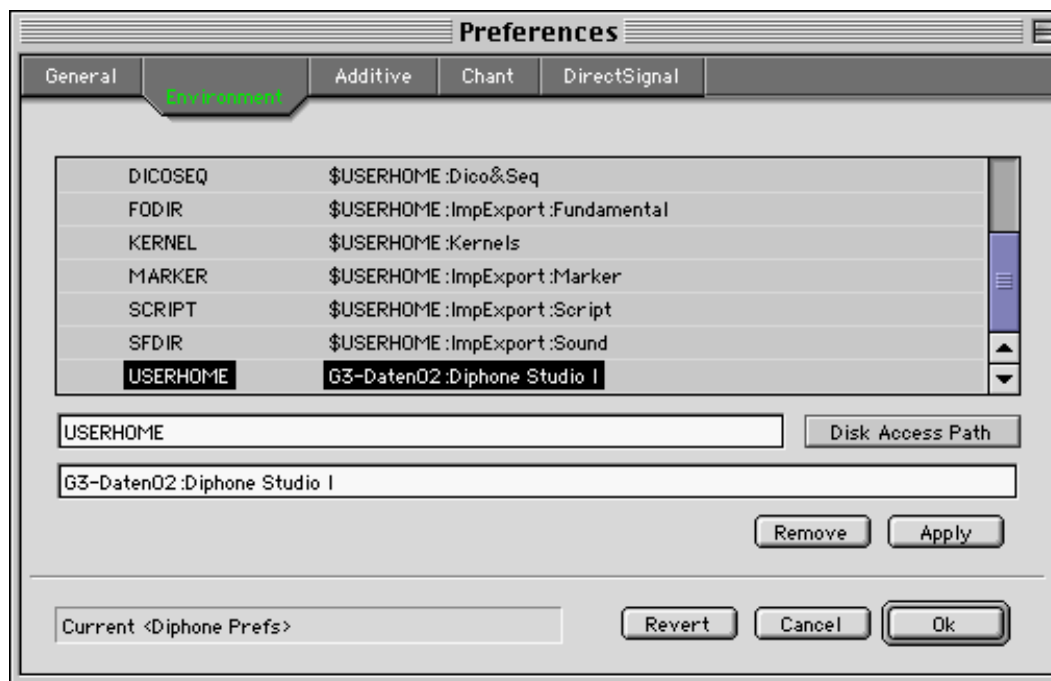
# I Analyse et synthèse additive

## 1.1. Réglages systèmes

Diphone a été conçu pour fonctionner à la façon d'un programme Unix, en particulier pour tout ce qui concerne la gestion des fichiers. Il est donc important d'indiquer dans les *préférences* du programme l'endroit sur le disque où sont situés les sons, les fichiers d'analyse et d'autres données. Le dossier de l'application Diphone contient trois sous-dossiers importants : *Container*, *Dico&Seq* et *ImpExport*.

Le dossier *Container* contient les dictionnaires, le dossier *Dico&Seq* contient des fichiers importants pour Chant et c'est aussi là que l'on place les séquences qui viennent d'être créées. Le dossier *ImpExport* contient de nombreux sous-dossiers dans lesquels se trouvent les sons à analyser (*Sound*) et les données d'analyse (*Fundamental* et *Partials*).

Si l'utilisateur décide de placer ces différents fichiers à un autre endroit de son espace disque (en dehors du dossier *Diphone*), il doit indiquer cette nouvelle position à l'aide de la variable *USERHOME* (dans *Preferences/Environment*).



## 1.2. AddAn – Analyse additive

Diphone ne traite pas le son directement, mais traite des paramètres le représentant comme le font d'autres techniques d'analyse/synthèse. Avec l'analyse/synthèse additive, les sons sont représentés par des sommes de signaux sinusoïdaux, ou partiels, dont les paramètres sont la fréquence, l'amplitude et la phase.

L'analyse additive consiste à calculer les valeurs de ces paramètres. L'analyse débute par une recherche de la fondamentale (*fundamental* ou  $f_0$ ) qui correspond, pour des sons harmoniques, au premier harmonique et qui peut varier dans le son en fonction du temps. Ce mode de synthèse fonctionne mieux avec des sons harmoniques qui ne contiennent qu'une faible quantité de bruit. Plus loin, nous indiquerons tout de même comment il est possible d'analyser des sons moins harmoniques.

Pour pouvoir synthétiser un son avec Diphone, on a besoin d'un dictionnaire de segments contenant chacun des listes de fréquences, d'amplitudes et de phases. Pour créer un dictionnaire, il faut analyser un son avec le programme AddAn et sauvegarder les données résultantes dans un dictionnaire.

La première étape est la détection de la fréquence fondamentale à partir de laquelle seront ensuite calculées les fréquences, les amplitudes et les phases des partiels. Alors, il est possible d'effectuer une resynthèse d'un son à partir de ces données. Celle-ci permet en particulier de tester la qualité de l'analyse effectuée.

Ces trois étapes produisent différents fichiers sur le disque : par exemple, si on part d'un son « exemple.aiff », on va obtenir un fichier décrivant la fondamentale intitulé « exemple.f0 », situé dans le dossier *ImpExport:fundamental*, un fichier contenant les paramètres des partiels : « exemple.ADD », situé dans le dossier *ImpExport:Partials* et enfin le son resynthétisé : « exemple.synth.aiff » qui sera situé dans le dossier *ImpExport:Sound*.

## 1.3. Analyse de Fourier (FFT) et spectre

Les opérations de recherche de la fréquence fondamentale ainsi que le calcul des partiels sont effectués à partir d'une analyse du son par transformée de Fourier (FFT). Même si, pour une partie des lecteurs, les explications qui suivent sur la FFT sont déjà familières, il me paraît nécessaire de donner certains détails dont la compréhension est indispensable pour une bonne réussite de l'analyse et sur lesquels certaines questions sont souvent soulevées (ces informations peuvent également être utiles pour l'utilisation d'autres programmes comme Audiosculpt ou Soundhack).

Je vais décrire la FFT et ses paramètres comme *WindowSize*, *FFTSize* et *WindowStep* non pas du point de vue strictement mathématique mais en insistant plutôt sur leur intérêt musical.

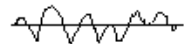
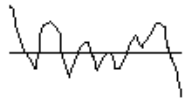
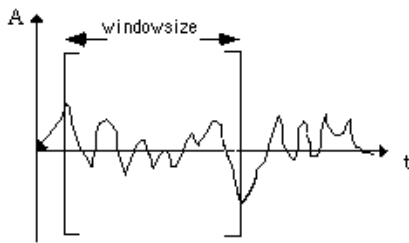
Du fait qu'une analyse FFT ne délivre qu'un spectre statique, sans information sur l'évolution temporelle du son, l'analyse FFT d'un son entier ne donnera pas une représentation du son lui-même mais plutôt une sorte de répartition moyenne globale des énergies qu'il contient. Pour connaître les variations spectrales du son en fonction du temps, il faut le diviser en petits fragments qui sont analysés les uns après les autres et délivrent chacun un spectre FFT à court terme. En réunissant ces analyses individuelles, on peut connaître l'évolution spectrale du son.

La longueur de chaque segment de son analysé est déterminée par un paramètre intitulé *WindowSize* (taille de la fenêtre) exprimé en secondes ou en nombre d'échantillons. Comme nous l'avons déjà signalé, du fait que l'analyse d'un segment est une analyse statique de l'énergie qu'il contient, si ce fragment contient une mélodie, celle-ci n'apparaîtra pas comme une succession de notes mais comme un accord.

En fait, la bonne valeur à attribuer à la taille de la fenêtre dépend du son à analyser. Si le son contient des variations rapides, la taille de la fenêtre doit être faible (par exemple 50 à 1000 échantillons, soit 11,3 à 22,6 ms). Ainsi, les différents événements acoustiques dont le son est formé ne tombent pas dans la même fenêtre. Si le son évolue plus lentement, il vaut mieux utiliser des fenêtres plus larges (par exemple 2000 ou 4000 échantillons).

Comme la taille de la fenêtre est choisie avant de réaliser l'analyse et qu'elle est constante sur toutes les analyses à court terme qui se succèdent, les positions de début et de fin de chaque fenêtre prise dans le son ne correspondent pas forcément à des valeurs d'intensité nulles du son. Ainsi, il peut se produire des clics au début et à la fin du son qui vont être détectés par l'analyse (cf. image 1-2, 2<sup>e</sup> ligne). Il est donc nécessaire d'atténuer le son au début et à la fin de la fenêtre (*fade in* et *fade out*). Cela est réalisé en multipliant le segment par une courbe nommée « *window function* » (Hamming, Hanning, Blackman, Kaiser, etc.). Une fois le segment ainsi modifié, il est analysé par FFT. La FFT agit sur un échantillon tampon (*buffer*) dont la taille (*FFTsize*) doit être au moins égale à la taille de la fenêtre et doit toujours être égale (en nombre d'échantillons) à une puissance de 2 (par exemple, si *window size* = 1000 échantillons, *FFTsize* = 1024 ou 2048 ou 4096 ...). Le tampon pour la FFT ne peut pas être plus petit que la taille de la fenêtre sans quoi les atténuations au début et à la fin sont coupées brutalement ce qui introduit à nouveau des clics. La taille de la FFT peut par contre être supérieure à la taille de la fenêtre. Dans ce cas, les échantillons manquants sont remplacés par des zéros (*zero padding*).

La taille de la FFT n'est pas liée à la résolution temporelle de l'analyse (c'est là souvent un point d'incompréhension). La taille de la FFT détermine uniquement la résolution en fréquence de l'analyse et donc la qualité de la représentation de ces fréquences. Ceci peut être expliqué ainsi : le spectre total des fréquences, depuis 0 jusqu'à la moitié de la fréquence d'échantillonnage du son (par exemple 22050 Hz pour un taux d'échantillonnage de 44100 Hz) est divisé en bandes de fréquences de largeurs égales (*frequency bins*). La FFT calcule l'énergie contenue dans chacune de ces bandes. Pour une analyse de qualité, il faut le plus de bandes possible et les bandes les plus fines possibles, de façon à ce que l'énergie des partiels du son tombent dans des bandes différentes et que ceux-ci puissent être distingués.



Le nombre de bandes (et donc leur largeur) dépend de la taille de la FFT. Avec une FFT dont la taille est de 1024 échantillons, on obtient 512 bandes entre 0 et 22050 Hz. Chaque bande a donc une largeur de  $22050/512 = 43,07$  Hz. Avec une taille de FFT de 4096 échantillons, on obtient 2048 bandes de largeurs 10,77 Hz. Plus la taille de la FFT est grande, plus on obtient de bandes, plus elles sont fines et plus la résolution en fréquence de l'analyse est bonne. Si plusieurs partiels tombent dans la même bande, la FFT calcule leur moyenne, qui peut inclure des effets de superposition comme des battements ou des oppositions de phase.

Une analyse de Fourier à court terme contient donc un tableau fixe de fréquences pour lesquelles sont simplement calculées les amplitudes et les phases associées.

Pour introduire à la resynthèse des sons, il suffit d'imaginer que chaque bande de fréquence est synthétisée par un générateur sinusoïdal dont la fréquence est choisie égale au centre de la bande considérée et qui est accordée avec précision par la valeur de la phase. L'amplitude de l'oscillation est donnée par l'amplitude détectée par l'analyse. Les données sont ainsi interpolées d'une analyse FFT à la suivante pour produire un signal continu.

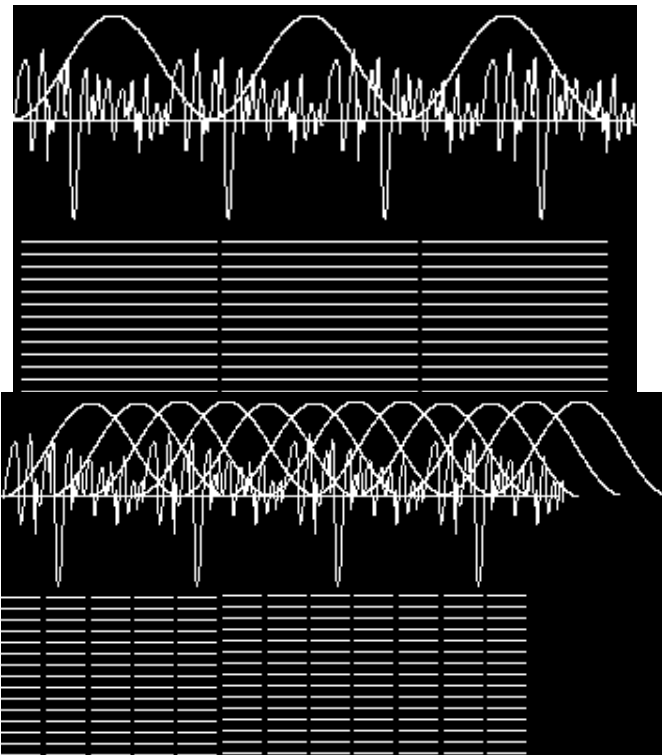
Pour éclaircir les idées, on peut dire que la résolution temporelle de l'analyse est déterminée par la taille de la fenêtre (plus le signal varie vite plus la fenêtre doit être petite) alors que la résolution de l'analyse en fréquence est déterminée par la taille de la FFT.

Par ailleurs, la taille de la fenêtre détermine une autre propriété : la fréquence la plus basse que l'on puisse détecter. Il faut au moins trois périodes de son dans la fenêtre d'analyse pour qu'une fréquence puisse être détectée. Ceci n'est pas une limite stricte. Quatre ou cinq périodes permettent une meilleure précision dans la détection de cette fréquence. Si le son à analyser contient des fréquences comprises entre 200 et 500 Hz, la période la plus longue est de  $1/200 = 0,005$  s. Pour obtenir quatre périodes dans la durée de la fenêtre d'analyse, il faut que cette fenêtre ait une taille d'au moins  $0,005 \times 4 = 0,02$  s.

On rencontre des difficultés lorsque les sons varient rapidement tout en comportant des composants à basses fréquences. Pour analyser les variations temporelles du son, on a besoin de choisir une faible taille de fenêtre. Pour analyser les basses fréquences, par contre, on a besoin d'une grande taille de fenêtre. Dans ce cas, la seule solution consiste à faire des essais pour trouver le meilleur compromis.

Avec des programmes pour lesquels les tailles de fenêtres et les tailles des FFT sont liées (comme avec SoundHack, Sound Designer, ProTools ou Audiosculpt), on doit toujours trouver un compromis. De faibles valeurs donnent une bonne résolution temporelle mais de larges bandes de fréquences avec la FFT et donc une mauvaise resynthèse des fréquences. Des tailles importantes donnent une bonne résolution en fréquence, mais diminuent la résolution temporelle puisque de nombreux événements sonores tombent dans la même fenêtre d'analyse et leurs énergies se compensent donc mutuellement.

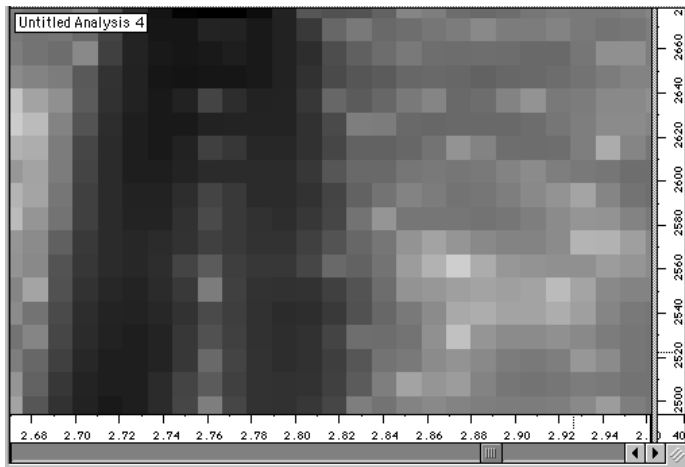
Le paramètre suivant de l'analyse est le pas d'analyse : *windowstep* ou *stepsize*.



Si une analyse (de taille *window size*) suit immédiatement l'analyse précédente, une partie du son n'est pas analysée, celle qui correspond aux atténuations situées au début et à la fin des fenêtres.

C'est pourquoi l'analyse démarre un peu avant la fin de l'analyse précédente. Le décalage entre une analyse et la suivante est inférieur à la taille de la fenêtre, de sorte que les fenêtres d'analyse se superposent. La distance entre le début d'une analyse et le début de l'analyse suivante est souvent de 1/4 ou 1/8 de la taille de la fenêtre.

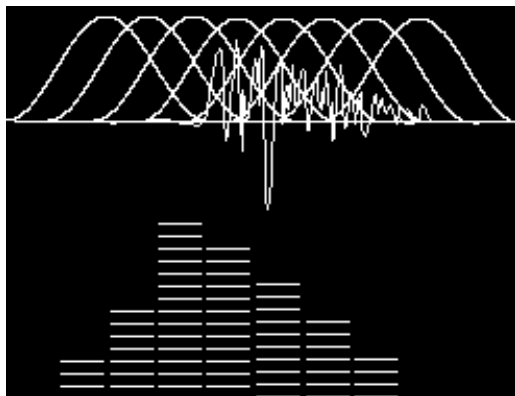
Chaque fenêtre d'analyse donne un spectre à court terme qui est utilisé seulement pour représenter la durée du pas d'avancement, avant que l'analyse suivante arrive. L'avantage est que chaque point du son apparaît au moins une fois dans une analyse de façon plus ou moins précise dans la partie centrale de la fenêtre. De cette façon, son énergie n'est pratiquement pas affectée par les atténuations de début et de fin.



Si on agrandit un sonogramme dans Audiosculpt, on voit de petites cases montrant les spectres à courts termes dans leur évolution temporelle. Leur longueur correspond au pas d'avancement. Sur l'axe des fréquences, ces cases correspondent aux largeurs des bandes de fréquences de la FFT (ici environ 10 Hz de large puisque la taille de la FFT est de 4096 échantillons).

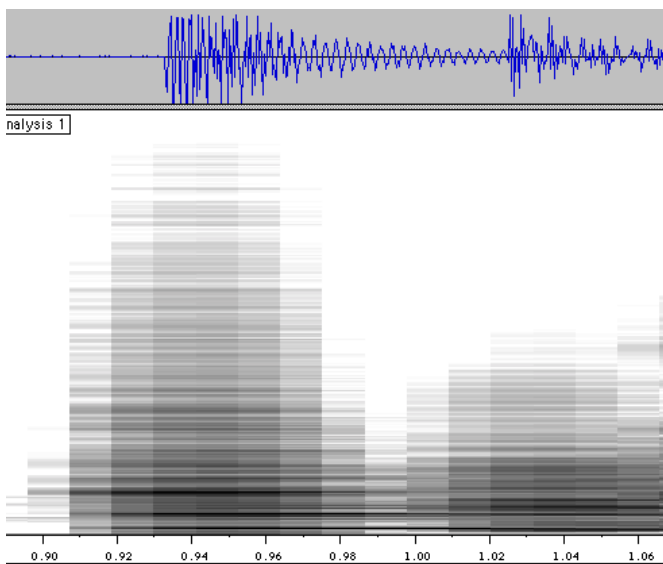
L'inconvénient des analyses qui se superposent est que chaque point est analysé plusieurs fois (par exemple 4 fois pour un pas d'analyse d'un quart de la taille de la fenêtre).

Étudions l'effet produit en examinant l'attaque d'un son comportant beaucoup d'énergie dans tout le spectre.



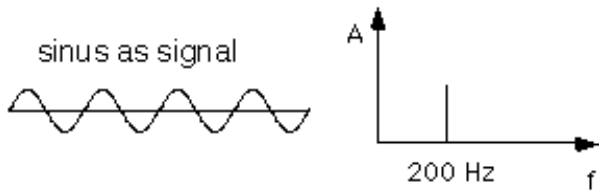
La première fenêtre d'analyse atteint à peine l'attaque du son et du fait de l'atténuation à laquelle ce son est soumis en fin de fenêtre, il ne reste que très peu d'énergie : le spectre résultant ne contient quasiment aucune énergie. Avec les fenêtres d'analyse suivantes, l'attaque va être de plus en plus présente dans la partie centrale de ces fenêtres ce qui va permettre de détecter de mieux en mieux les hautes fréquences (dans la figure, c'est la 4<sup>e</sup> fenêtre qui contient l'attaque en son centre). Avec les fenêtres qui suivent, l'énergie contenue dans les hautes fréquences va à nouveau diminuer.

Du fait que le spectre à court terme qui correspond à la fenêtre d'analyse courante est affiché dès le début de la fenêtre, on trouve à cet endroit de l'énergie qui n'apparaîtra que plus tard dans le son. Cela produit comme résultat une sorte d'attaque spectrale anticipée dans les hautes fréquences (*spectral fade-in*).



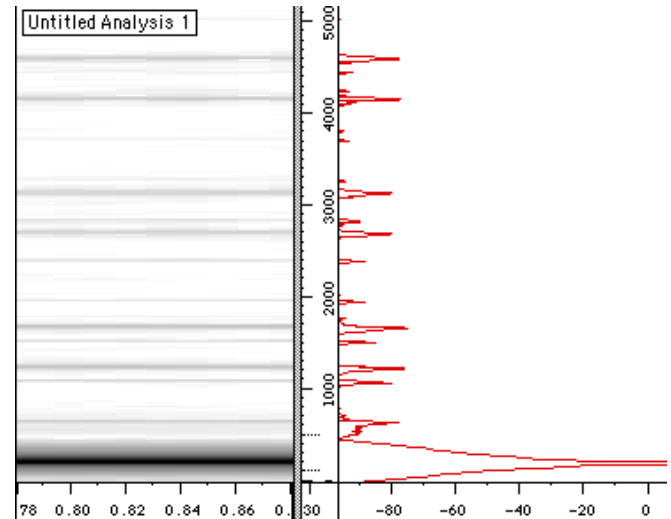
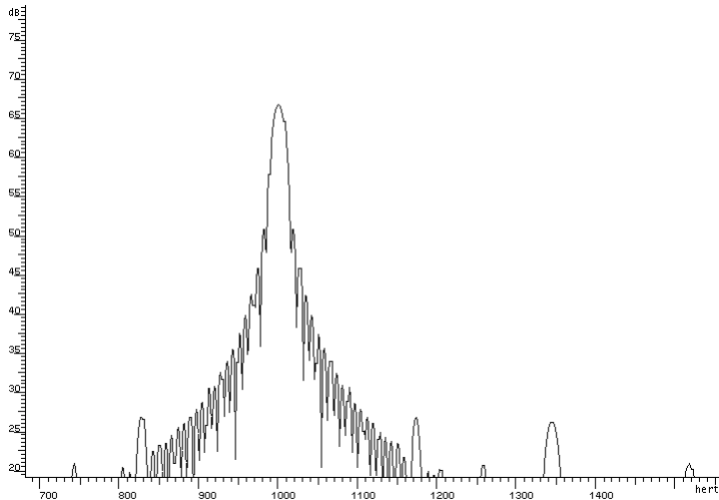
Le quatrième paramètre pour l'analyse est le choix de la forme de la fenêtre. De nombreux programmes proposent un choix de fonctions parmi lesquelles on trouve souvent les fenêtres de Hamming, Hanning, Blackman et Kaiser. Leur influence sur la qualité de l'analyse est assez faible mais pas assez pour être négligée.

En théorie, l'image spectrale d'une oscillation sinusoïdale devrait être un trait vertical dans le spectre.



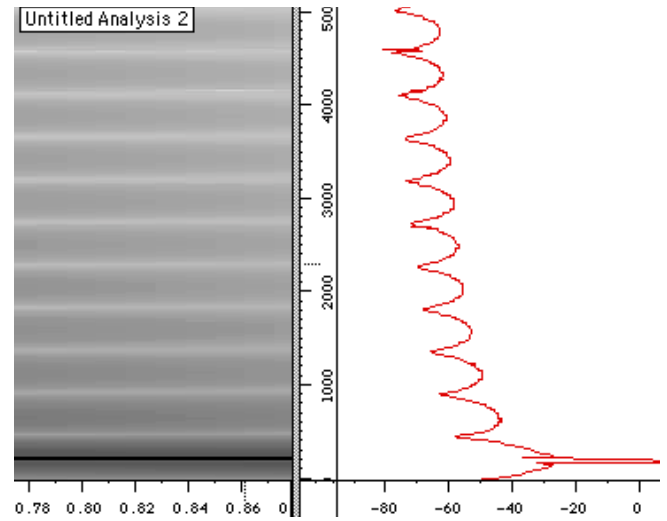
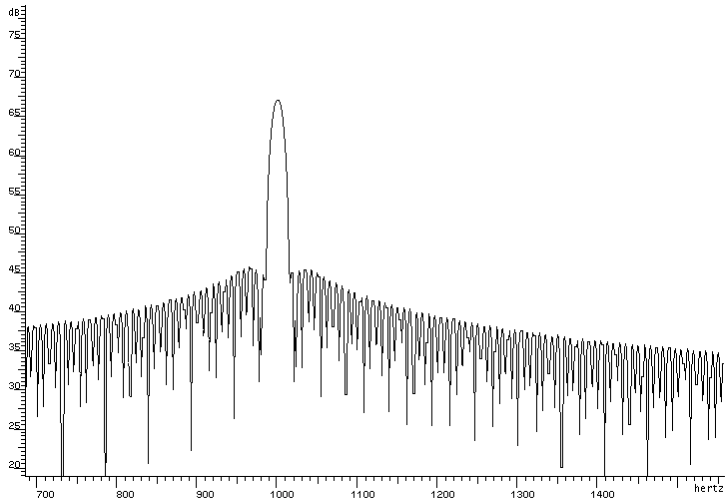
Si ce signal est multiplié pendant l'analyse par une courbe de fenêtrage, cela produit comme résultat deux signaux (un peu comme une modulation en anneau). Le spectre résultant correspond à la convolution du spectre du signal avec le spectre de la fonction de fenêtrage. Les différences de qualité entre les différents types de fenêtres dépendent de leur spectre. Pour illustrer cette question, nous allons multiplier une onde sinusoïdale de fréquence 1000 Hz par trois fenêtres types. L'énergie du sinus ne ressemblera alors pas à un trait unique dans le graphique.

Une **fenêtre de Hanning** génère un pic central assez large et quelques bandes latérales.

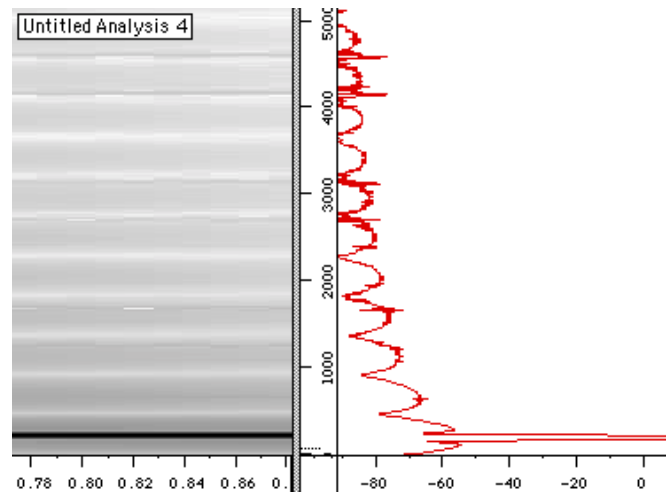
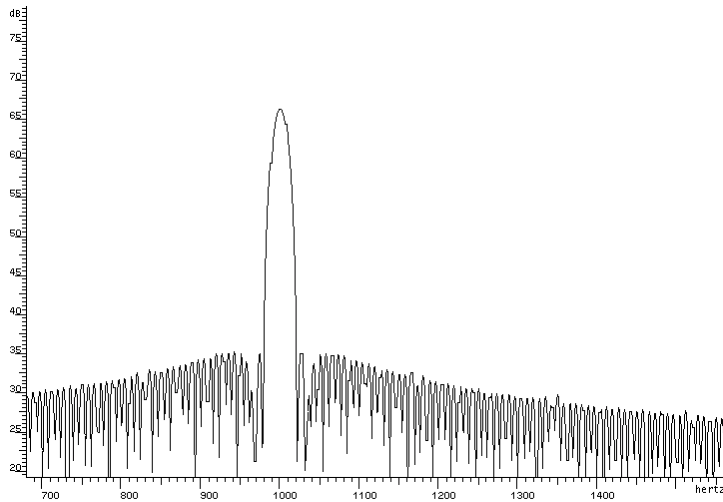


La définition en fréquence du pic central est meilleure avec une **fenêtre de Hamming**. Par contre, celle-ci produit des bandes latérales plus étendues.



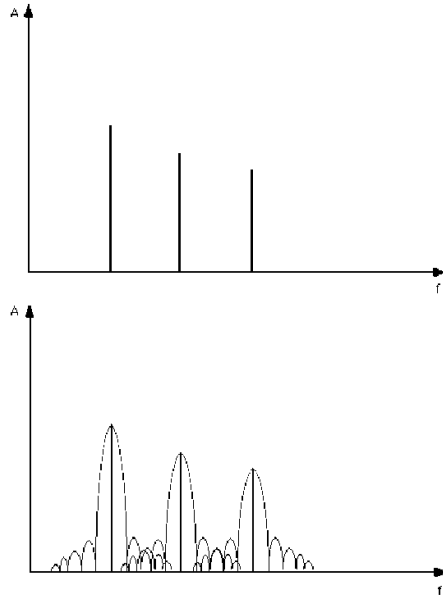


Le spectre obtenu avec la **fenêtre de Blackman** a également une bonne définition en fréquence. Les bandes latérales ont une amplitude plus faible et sont moins étendues qu'avec la fenêtre de Hamming.



Nous avons indiqué le résultat produit par l'action de trois types de fenêtrages sur une fréquence sinusoïdale unique. Lors de l'analyse de sons complexes, ce déploiement de l'énergie se produit alors pour chaque partiel. Si le son est formé de trois partiels, le spectre théorique devrait simplement comporter trois raies. Si l'on multiplie le son par une courbe de fenêtrage, on obtient la répartition d'énergie suivante :

On peut faire la même observation avec le sonagramme, en analysant une onde sinusoïdale et en choisissant des fenêtres de Hanning, Hamming et Blackman (respectivement de gauche à droite). Alors que la fenêtre de Hanning produit un partiel clairement identifiable, les deux autres fonctions engendrent un bruit de fond plus ou moins intense (Hamming : intense, Blackman, plus faible). Cela dit, le rapport signal/bruit entre la raie centrale et les bandes latérales est suffisamment important pour qu'on n'entende pas ces dernières. Ce n'est éventuellement qu'après plusieurs étapes d'analyse/resynthèse successives que les bandes latérales générées par les courbes de fenêtrage peuvent être perçues.

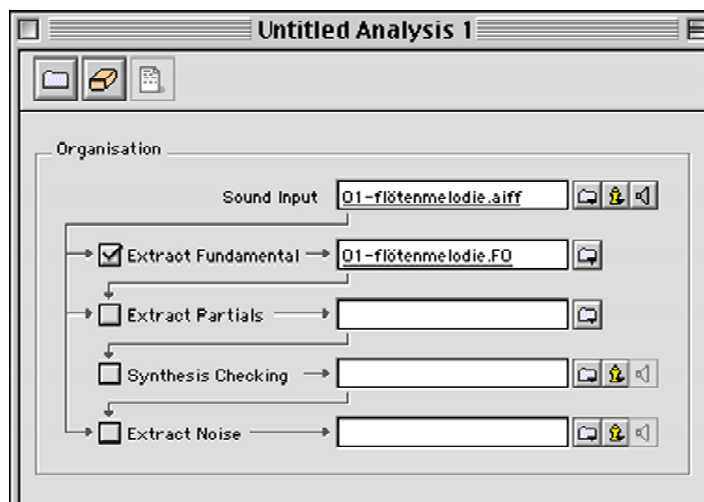


## 1.4. Analyse de la fréquence fondamentale

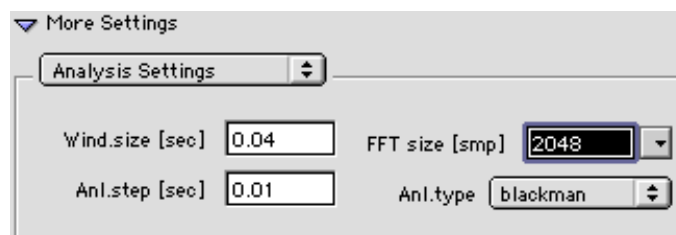
Après cet aperçu du rôle des quatre paramètres de la FFT, revenons à l'analyse dans Diphone. La première étape est l'estimation de la fondamentale (F0).

 Exemple sonore : 01-flötenmelodie

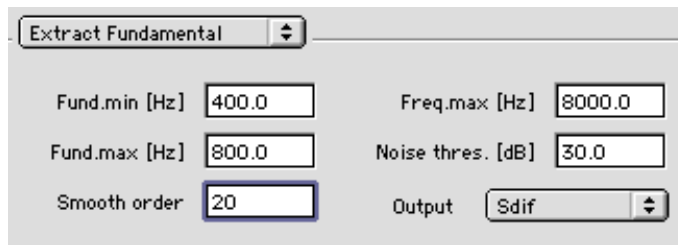
Si le son à analyser se trouve bien dans le dossier « :ImpExport:Sound », on peut alors le sélectionner dans AddAn en cliquant simplement dans la ligne « Sound Input » et le jouer en cliquant sur l'icône de haut-parleur.



On choisit d'abord uniquement l'option « *Extract Fundamental* ». Dans la partie inférieure de la fenêtre « *More Settings* », on peut régler les paramètres de l'analyse. La taille de la fenêtre et le pas d'avancement (ici *Analysis Step*) de l'analyse sont donnés dans Diphone en secondes.



Ensuite, il faut régler les paramètres pour l'estimation de la fréquence fondamentale.

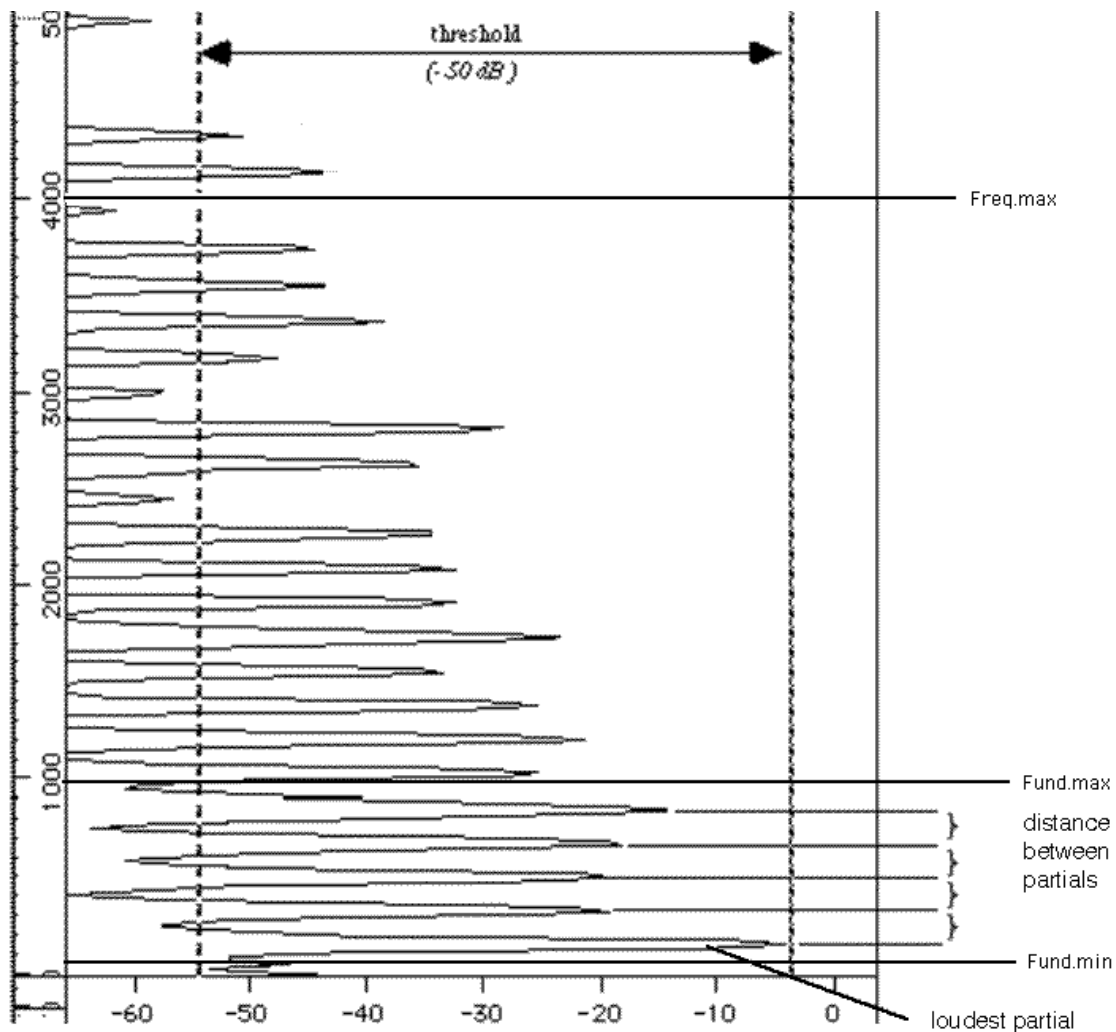


Pendant l'analyse, les distances entre les bandes de forte énergie contenues dans le son sont calculées et le programme recherche un éventuel plus petit commun multiple à ces fréquences. Pour de bons résultats et également pour diminuer les temps de calculs, on limite la plage de recherche (sinon toutes les raies fournies par l'analyse seraient comparées avec toutes les autres).

*Fund.min* et *Fund.max* fixent les limites entre lesquelles la fondamentale est recherchée. *Noise threshold* donne l'écart d'amplitude relatif maximum avec le partiel le plus fort. Toutes les énergies plus faibles que cette valeur sont éliminées pendant la recherche des divers harmoniques. Pour des sons contenant une fraction de bruit importante, on choisira un seuil de bruit (*Noise threshold*) moins élevé pour n'inclure que les énergies les plus importantes dans l'analyse.

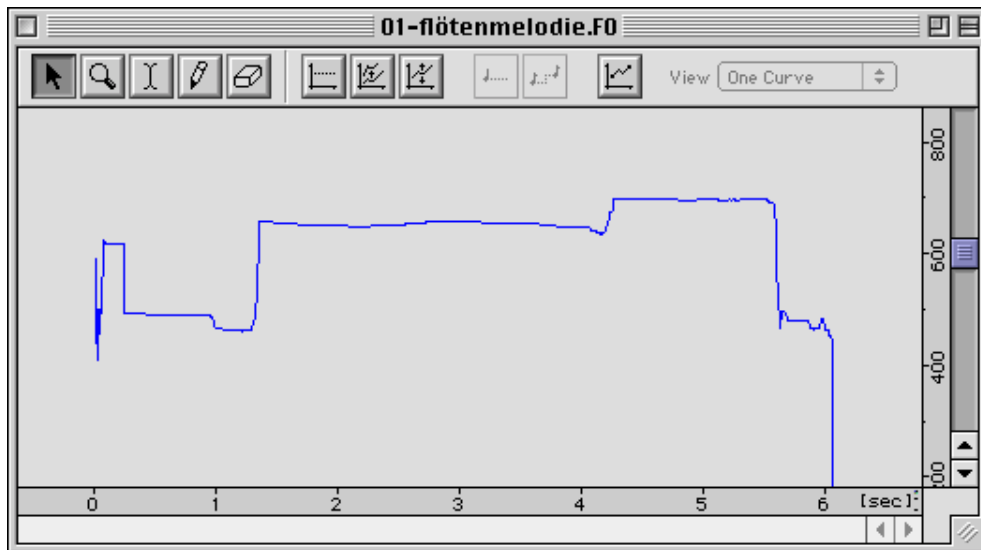
*Freq.max* indique la limite supérieure imposée pour la recherche des divers harmoniques. Si quatre ou cinq multiples de la fondamentale ont été trouvés, on peut penser qu'on a trouvé la bonne fondamentale. Il n'est pas nécessaire de continuer les recherches jusqu'à la fin du spectre (22050 Hz).

*Smooth order* (variant entre 3 et 30) effectue un lissage de la courbe de la fréquence fondamentale après l'analyse pour atténuer les petites déviations.

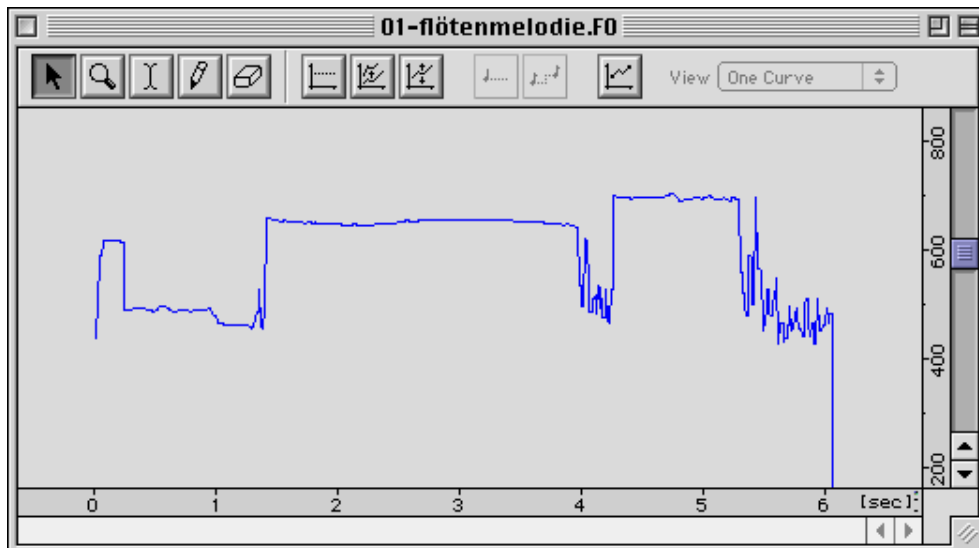


Le format de sortie est SDIF (Sound Description Interchange Format), un fichier binaire. Afin de visualiser les résultats, ou pour les utiliser dans d'autres programmes, il est également possible de sauver ces données en tant que fichier ASCII.

Les données peuvent être chargées et visualisées dans Diphone. Pour notre exemple sonore, les réglages précédents ont produit le résultat suivant :

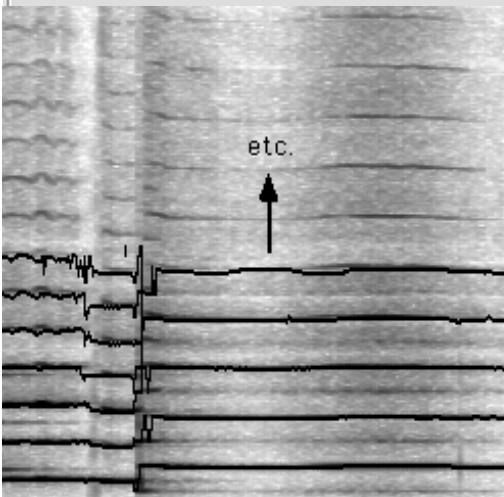
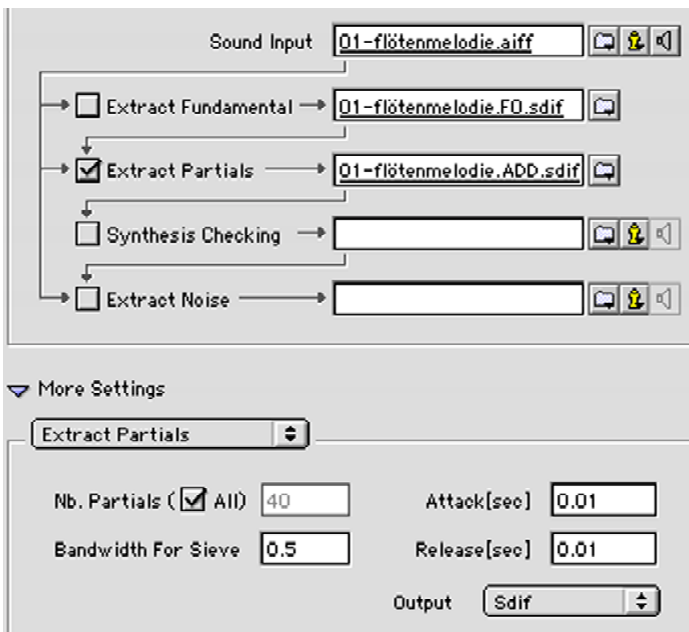


On peut déceler certaines irrégularités au début et à la fin du son qui n'ont toutefois que peu d'importance. Le choix de mauvais paramètres pour la recherche de la fondamentale peut entraîner des sauts irréguliers dans la courbe de fréquence.



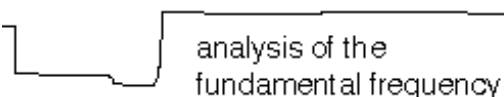
Pour utiliser les données de la F0 pour les étapes suivantes, le format de sortie doit être SDIF (ASCII n'est utilisé que pour permettre une vérification visuelle des données).

# 1.5. Réduction en partiels harmoniques



spectrum of  
FFT analysis

found partials  
as multiples  
of  $f_0$



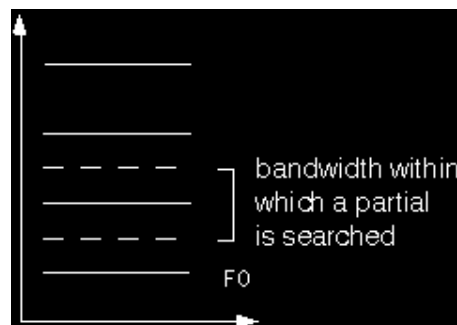
analysis of the  
fundamental frequency

Cette étape de l'analyse utilise les résultats donnés par le calcul de la fondamentale pour extraire l'énergie du spectre FFT à court terme dans des régions qui sont multiples de la fréquence fondamentale, c'est-à-dire qui correspondent aux partiels harmoniques du son.

Cela souligne l'importance d'effectuer une bonne estimation de la fondamentale. Si l'analyse génère des sauts indésirables de la fondamentale, les fréquences de tous les partiels en seront affectées.

Par ailleurs, l'analyse précédente montre que la partie bruitée du son, produite dans cet exemple par le souffle dans la flûte, ne peut être analysée (je présenterai plus loin un exemple indiquant comment on peut obtenir une meilleure analyse de la partie bruitée).

Les paramètres d'attaque (*Attak*) et de chute (*Release*) permettent une montée et une descente du volume des partiels au moment où ils apparaissent et où ils disparaissent respectivement dans le son. Cela permet d'éviter les clics.



La largeur de grille (*Bandwidth for Sieve*) détermine pour chaque partiel une bande de fréquence, centrée exactement autour d'un multiple de la fondamentale. Cela autorise des déviations de fréquence de ce partiel au-delà d'une structure strictement harmonique du son. Une valeur de 0,5 donne une étendue comprise entre le milieu de l'intervalle [harmonique théorique - harmonique précédent] et le milieu de l'intervalle [harmonique théorique - l'harmonique suivant].

## 1.6. Resynthèse et extraction de la fraction bruitée

En sélectionnant la synthèse (*Synthesis checking*), on active la resynthèse des partiels qui viennent d'être calculés. Cela permet de tester la qualité de l'analyse avant de la segmenter pour la répartir dans des dictionnaires (*container*).



Exemple sonore : 02-floetenmelodie.synth

Si Extract Noise est sélectionné, un fichier de son example.noise.AIFF est créé. Il contient la soustraction du son resynthétisé au son original, i.e. toutes les composantes qui ne sont pas sur la structure harmonique.



Exemple sonore : 03-floetenmelodie.noise

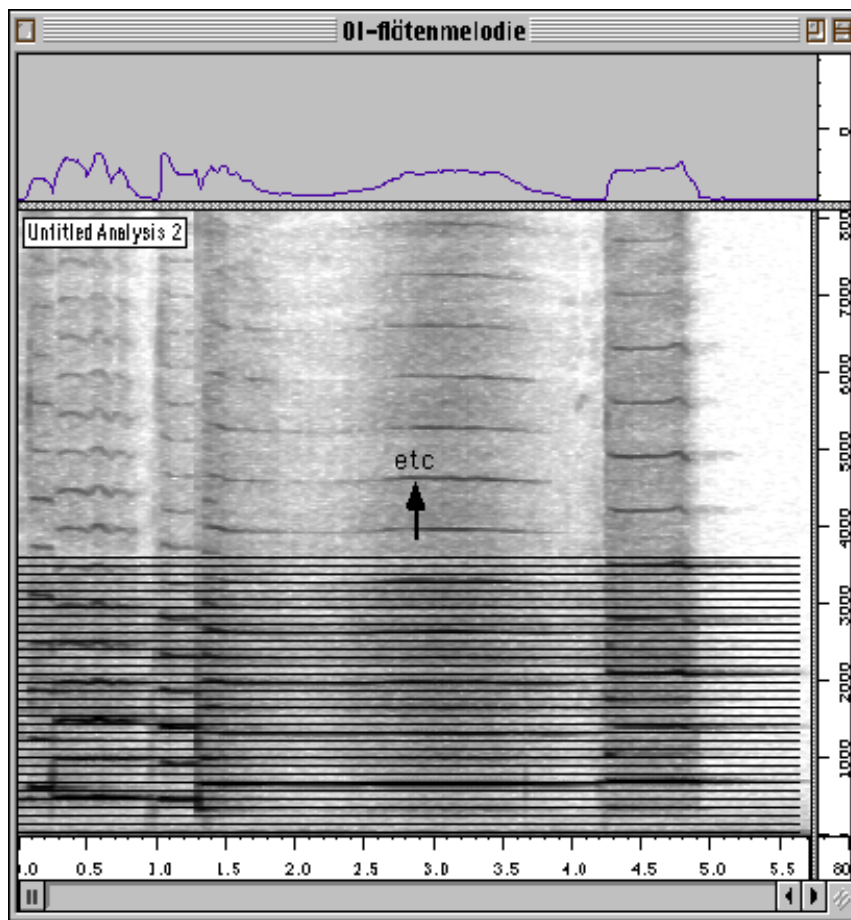
Bien sûr, toutes les étapes de l'analyse peuvent être réalisées en une seule fois. Le fait de calculer d'abord la fréquence fondamentale seule, et de regarder son allure, présente l'avantage d'éviter de lancer les autres étapes inutilement si l'estimation de la fondamentale n'a pas été réussie.



## 1.7. Amélioration de la détection de la fraction bruitée

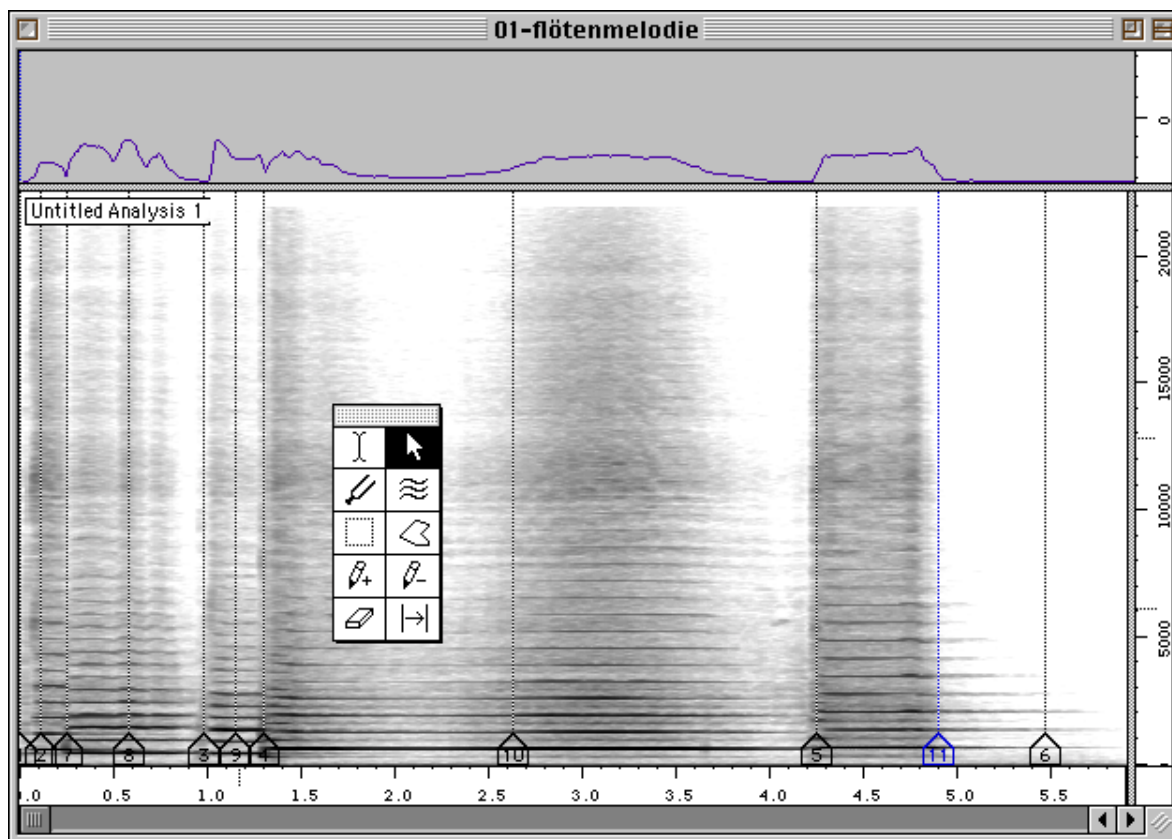
Comme nous l'avons indiqué, l'analyse de la fraction bruitée du son est impossible puisqu'on n'analyse que les composants du son purement harmoniques. Cela dit, si l'on peut fournir à l'analyse une fondamentale artificielle, de 30 Hz par exemple, l'analyse des partiels va rechercher tous les multiples de 30 Hz et ainsi utiliser un nombre de partiels beaucoup plus importants pour la resynthèse.

👂 Exemple sonore : 04-floetenmelodie.synth.30Hz



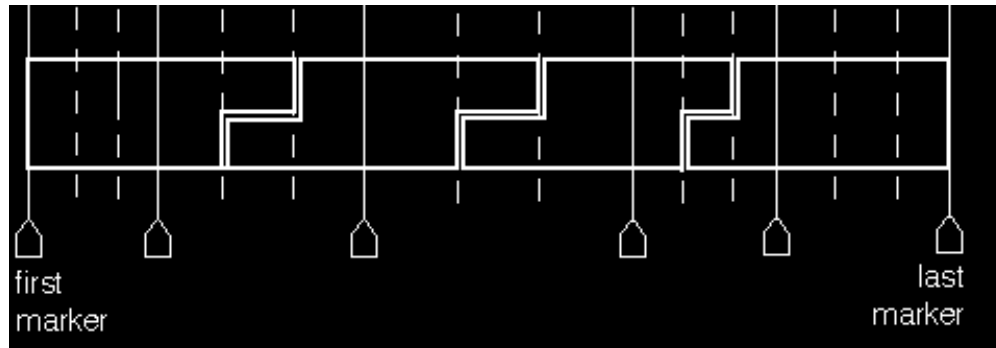
Du fait que ces partiels ne sont pas uniquement des multiples de la fondamentale réelle, mais sont placés entre les vrais partiels, l'énergie correspondant à la fraction bruitée est ainsi restituée.

30 Hz



## 1.8. Segmentation du son original

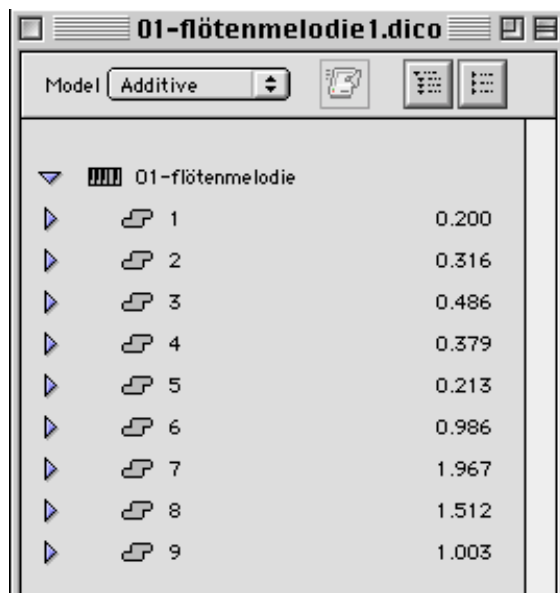
Pour réaliser la segmentation, on peut utiliser le programme Audiosculpt. En cliquant dans le sonogramme avec la touche *Ctrl* enfoncée, il est possible de placer des marqueurs sur l'axe du temps qui peuvent être sauvegardés (menu *File*→*Export*→*Markers*) dans des fichiers de type texte (dans le dossier *Diphone:ImpExport:Markers*). Chaque marqueur doit représenter le milieu d'un segment.



## 1.9. Construction d'une bibliothèque contenant les segments

À partir d'un fichier de marqueurs, on peut créer un script dans Diphone (menu *PlugIns*→*Create Script from Markers*) et ensuite construire une bibliothèque avec ce script (*PlugIns*→*Create Dictionary from Script*).

Le dossier *container* contient alors un dictionnaire dont les segments contiennent les données de l'analyse qui peuvent être utilisées pour créer de nouvelles séquences. Si ce container est ouvert dans Diphone, on peut alors glisser/déposer les segments dans une nouvelle séquence et là, les éditer.



Segment	Start Time
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

Les possibilités de création de sons avec ces segments sont décrites dans les parties suivantes.

## 2 La création de séquences

Cette deuxième partie décrit les différentes manipulations du son qui sont possibles dans Diphone.

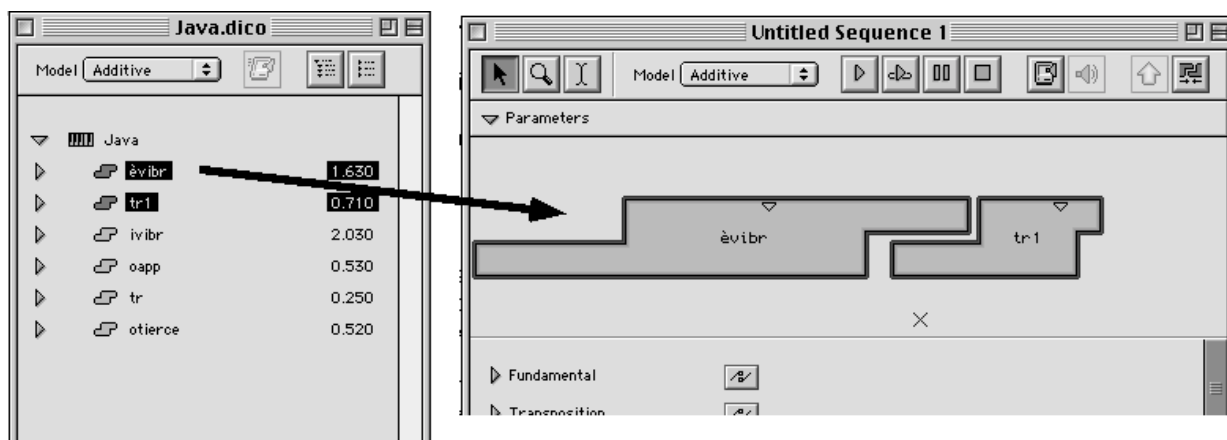
Ce texte ne remplace en aucun cas l'excellente documentation (française/anglaise). Il s'agit plutôt d'une rapide introduction.

### 2.1. Quelques définitions

Un *container* est un document situé dans le dossier *container*. Il contient des données provenant d'une analyse additive et la segmentation du son. Selon les paramètres de l'analyse réalisée et la taille du fichier-son source, ce fichier peut avoir une taille de plusieurs megabytes.

Si un container est ouvert dans Diphone, il apparaît en tant que dictionnaire, c'est-à-dire un alias vers le container. Les modifications de ce dictionnaire ne changent jamais les données du container, mais sont sauvegardées avec le dictionnaire. Ainsi, les données de diverses séquences qui utiliseraient le même container ne peuvent être modifiées en changeant une de ces séquences. On peut comparer ceci aux sons et aux régions dans Protocols : si on modifie une région, les fichiers-sons sur le disque restent intacts.

Un dictionnaire peut contenir un ou plusieurs instruments à partir desquels quelques segments, ou tous, peuvent être déplacés dans une séquence.

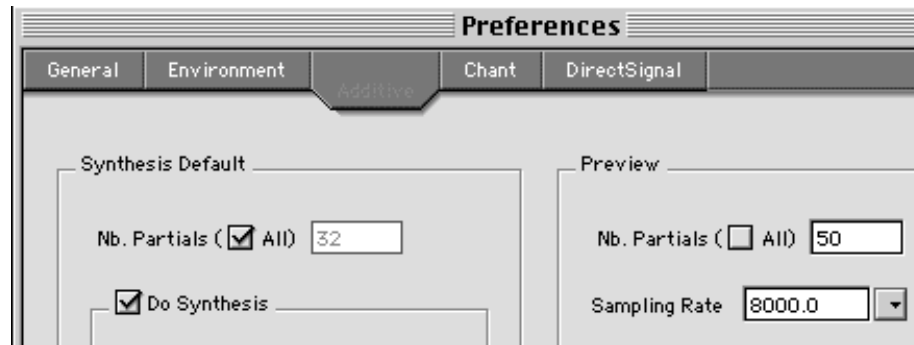


Une fonction par segments (BPF) correspond à une représentation graphique de certaines données en fonction du temps. Un exemple de BPF simple est, par exemple, la représentation de la fréquence fondamentale. Les fréquences, les amplitudes ou les phases sont par contre des BPFs multiples (multi-BPFs).



Si on joue une séquence dans Diphone, on entend seulement un aperçu du résultat (*preview*).

Selon la puissance du processeur de l'ordinateur, Diphone peut jouer jusqu'à un certain nombre de partiels en temps réel. Ce nombre est donné dans les préférences.



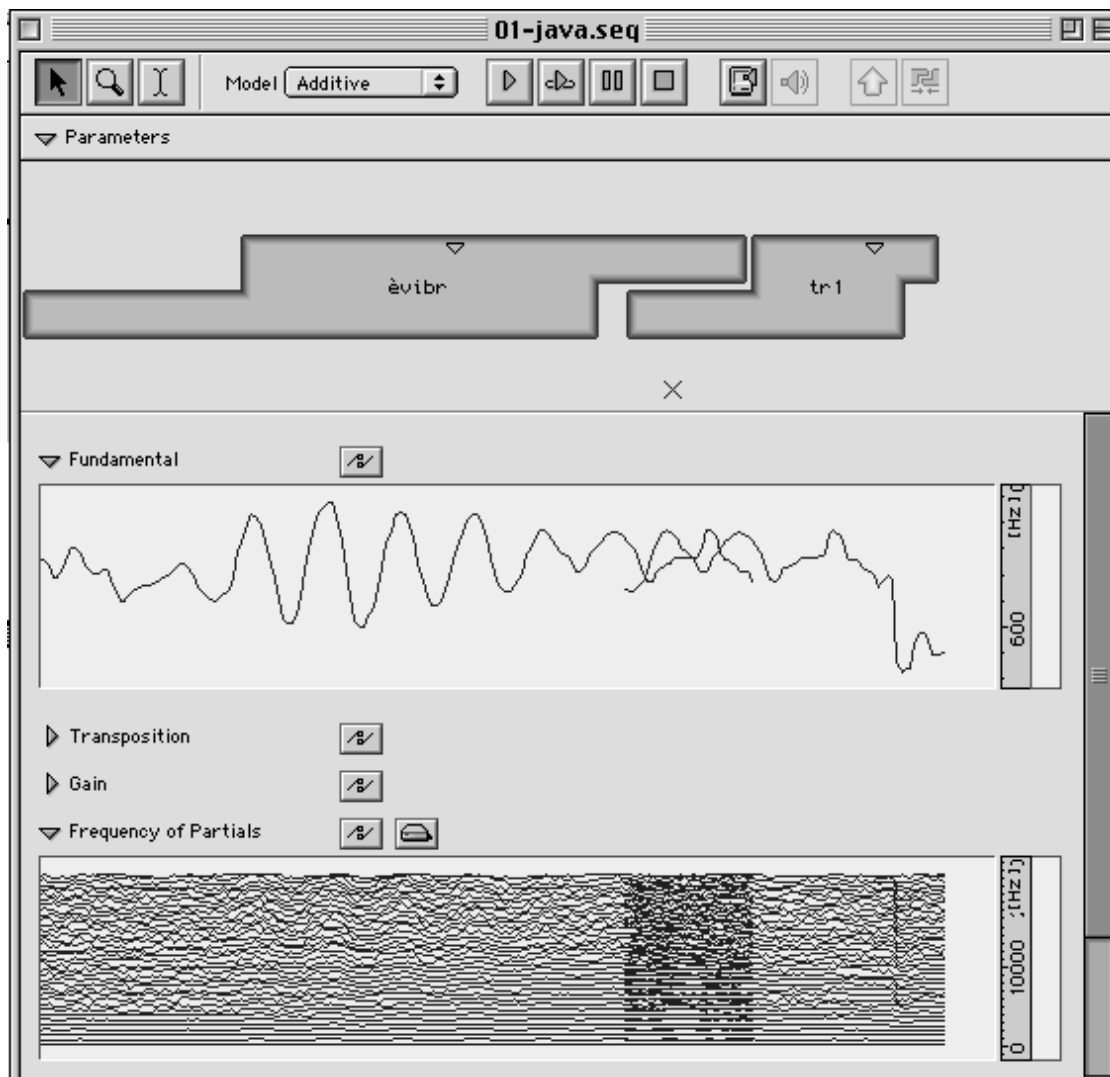
Si Diphone n'arrive pas à jouer le nombre de partiels choisi, cela peut produire des clics et il faut donc choisir un taux d'échantillonnage plus bas ou diminuer le nombre de partiels dans les préférences.

Si l'on veut obtenir un résultat de qualité optimale, il faut créer un nouveau fichier-son avec la commande *Export for Synthesis*.

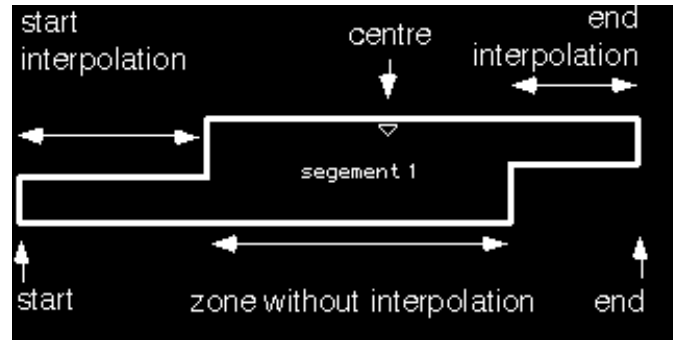


## 2.2. Visualisation et édition des paramètres

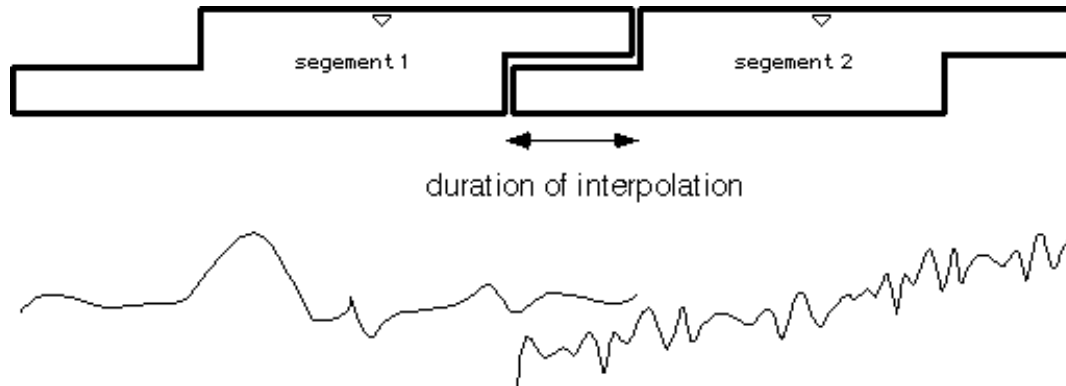
En dessous de la séquence contenant les segments, on peut observer tous les paramètres appartenant à ces segments. On trouve là les valeurs provenant de l'analyse de la fondamentale, les fréquences, amplitudes et phases des partiels et des paramètres supplémentaires, décrits ultérieurement. En cliquant sur le triangle à côté du nom du paramètre, on ouvre un affichage graphique de ces données.



Un segment est formé de trois zones : interpolation du début, zone centrale sans interpolation, interpolation de fin.

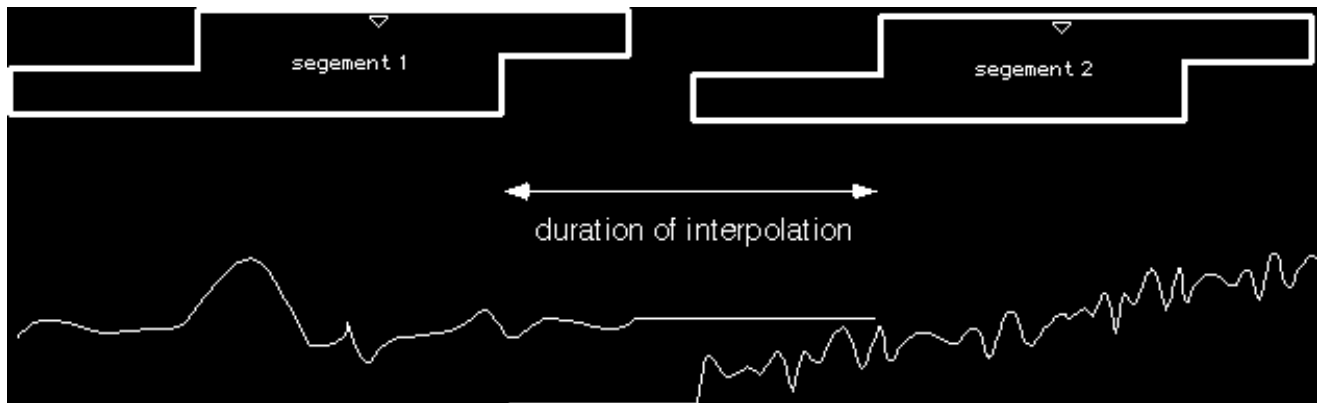


Les données de deux segments consécutifs sont interpolées depuis la fin de la zone centrale du premier segment jusqu'au début de la zone centrale du segment suivant.

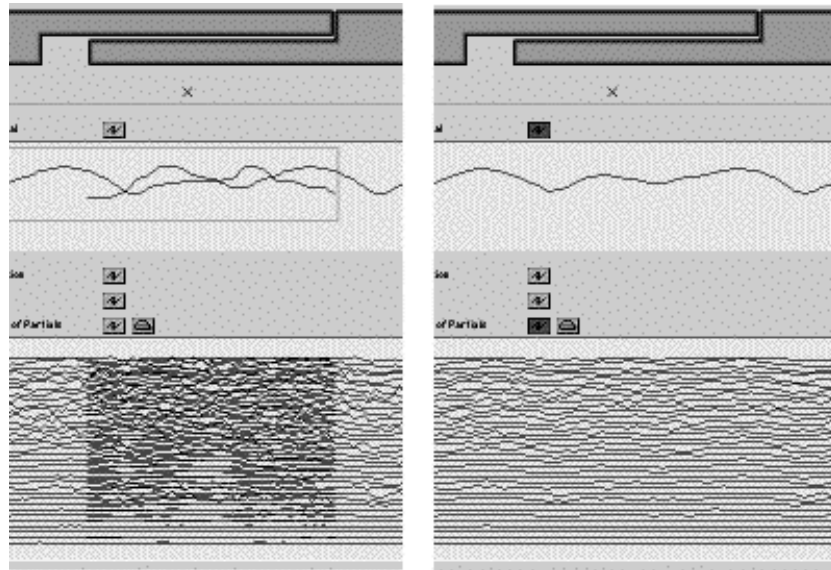


Si deux segments ne sont pas directement au contact l'un de l'autre, il n'y a pas de trou dans le son. En fait, la durée de l'interpolation est prolongée et les données sont interprétées de la façon suivante : la dernière valeur du segment précédent et les premières valeurs du segment suivant sont conservées pendant la durée d'interpolation nécessaire.



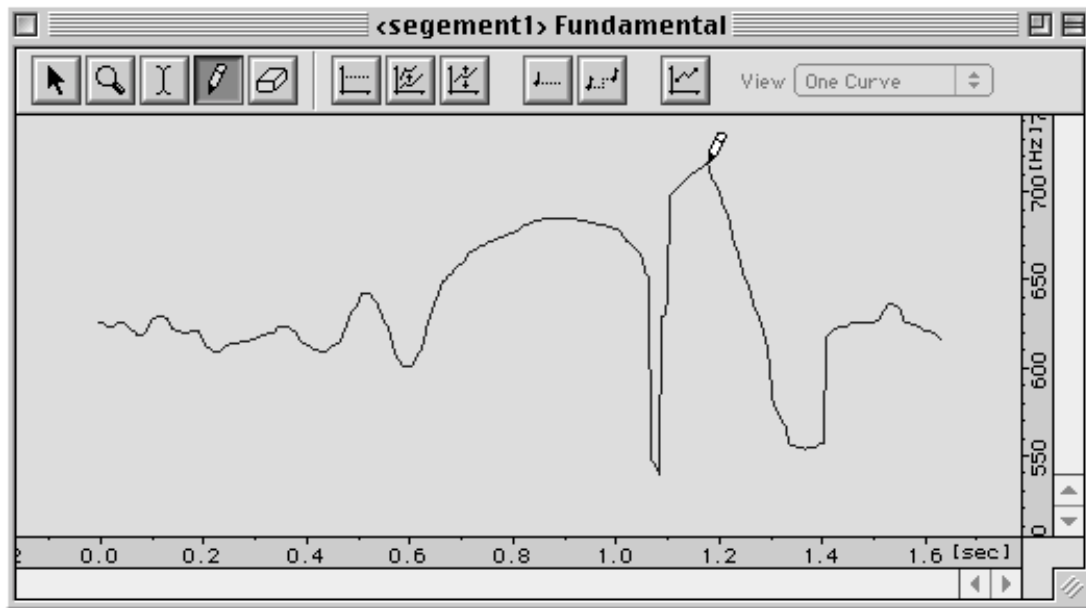


On peut choisir entre deux modes de visualisation : soit on voit les données des segments, sans interpolation, soit on clique sur l'icône d'interpolation et on voit alors le résultat de l'interpolation.



Si on est en mode de visualisation non-interpolé et qu'on déplace la souris au-dessus d'un paramètre de la courbe, celui-ci apparaît entouré par un rectangle bleu.

Un double-clic ouvre alors un éditeur où on peut modifier les données, soit avec le crayon, soit avec des fonctions arithmétiques.

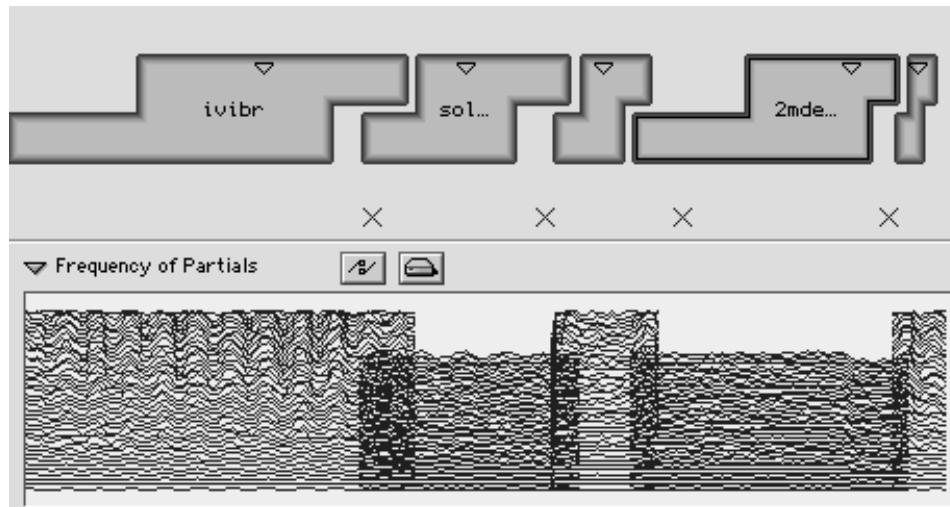


Pour agrandir ce qu'on voit, on peut utiliser la loupe ou sélectionner une plage sur l'axe du temps ou sur l'axe des données en tenant la touche *shift* enfoncée. Un double-clic sur l'axe nous renvoie à la vue d'origine.

Toutes les modifications sur les données ont un effet immédiat sur le son qui est joué et bien sûr aussi sur la resynthèse.

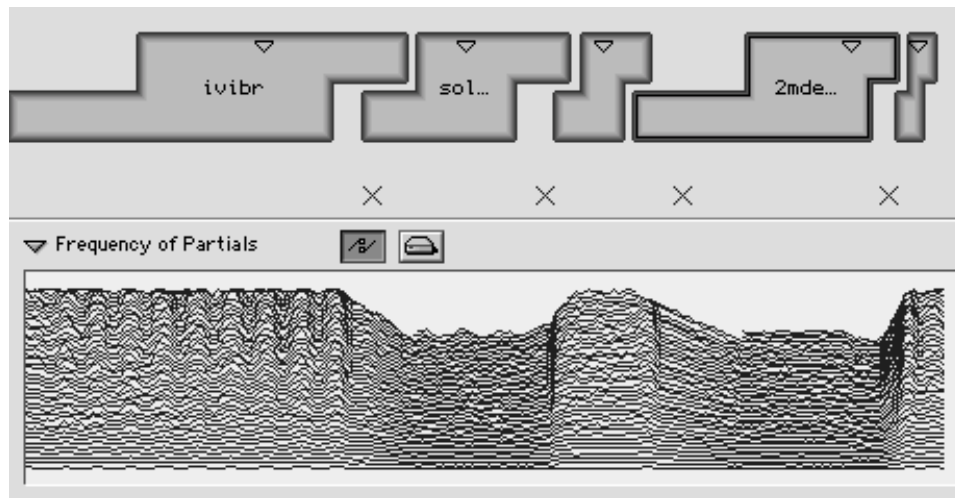
## 2.3. Interpolation entre les segments

L'exemple suivant combine des segments provenant de deux dictionnaires différents. J'ai utilisé les dictionnaires fournis avec le programme afin que chacun puisse suivre les étapes décrites. À partir des dictionnaires *Java* et *Shaku*, des segments ont été placés alternativement dans une nouvelle séquence. Dans le mode de visualisation non-interpolé, on peut constater clairement les différents spectres des deux sources sonores.

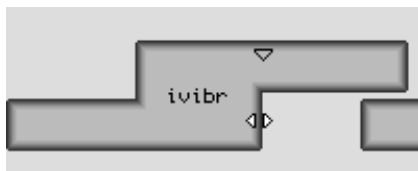


Dans le mode interpolé, on peut observer les glissés produits par l'interpolation.

 Exemple sonore : [java-shaku01](#)

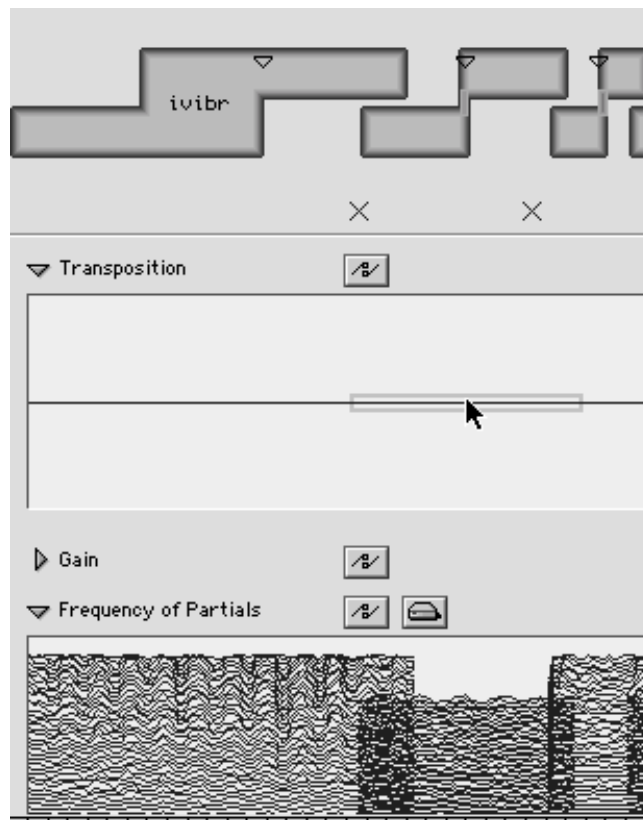


Si on déplace la souris par-dessus le début ou la fin de la zone centrale, le curseur se transforme en une double flèche blanche avec laquelle on peut changer la longueur de cette zone centrale.



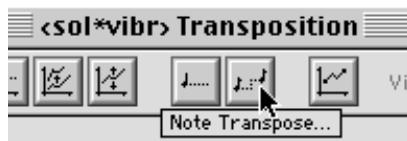
Cela change la durée de l'interpolation entre les deux segments. Comme nous l'avons dit au début, seules les données contenues dans la zone centrale sont jouées dans leur forme originale. En réduisant la longueur de cette zone centrale, on augmente les durées d'interpolation comme dans l'exemple suivant.

 Exemple sonore : [java-shaku02](#)

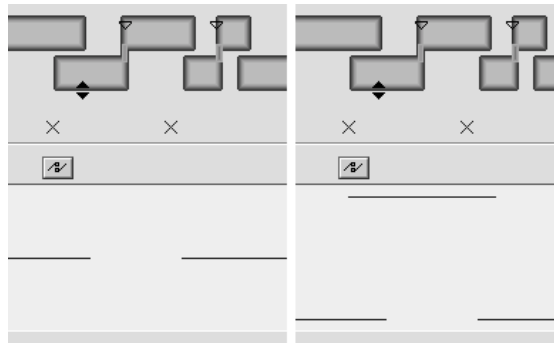


Si on veut éviter les glissés entre les segments, on peut agir sur le paramètre *Transposition*. Comme pour l'édition de la fondamentale, on observe un contour bleu autour du paramètre du segment concerné lorsqu'on déplace la souris au-dessus. Avec un double-clic, on ouvre un éditeur.


Avec la fonction *Note transpose*, nous transposons le segment de 5,5 demi-tons vers le bas, de façon à obtenir approximativement la même hauteur entre le segment précédent et le segment suivant.

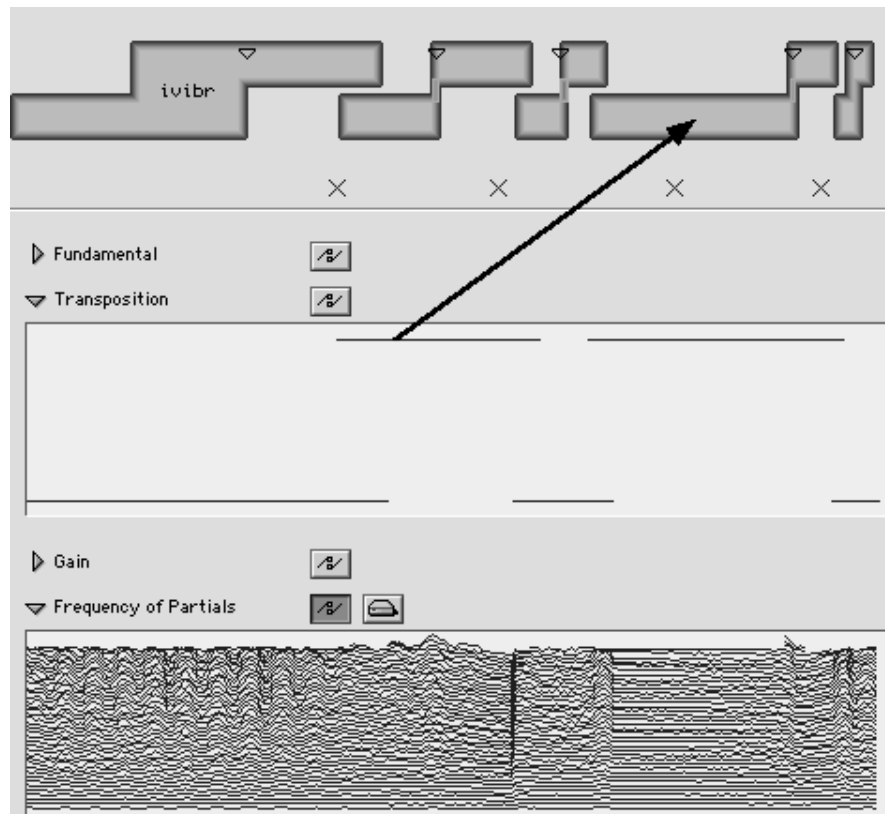


Ensuite, nous déplaçons le pointeur de la souris sur la limite inférieure d'un segment — le pointeur se transforme en une double flèche noire — et nous cliquons une fois pour remettre à l'échelle l'affichage du paramètre édité.

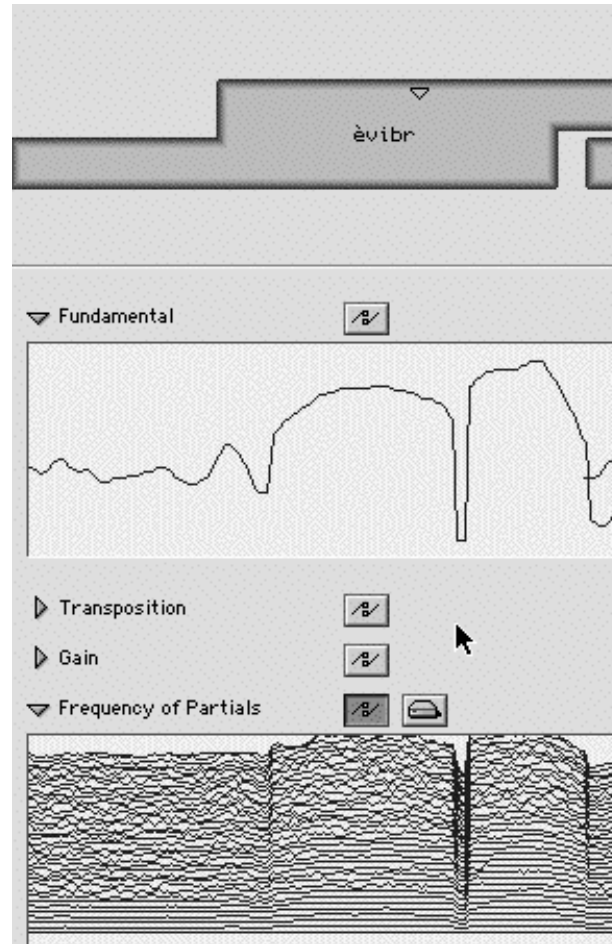


Si on veut réutiliser la même transposition pour le quatrième segment, il suffit de la déplacer directement sur ce segment. Dans le mode interpolé de visualisation des fréquences, on constate qu'il n'y a maintenant plus de glissés entre les segments.

 Exemple sonore : [java-shaku03](#)



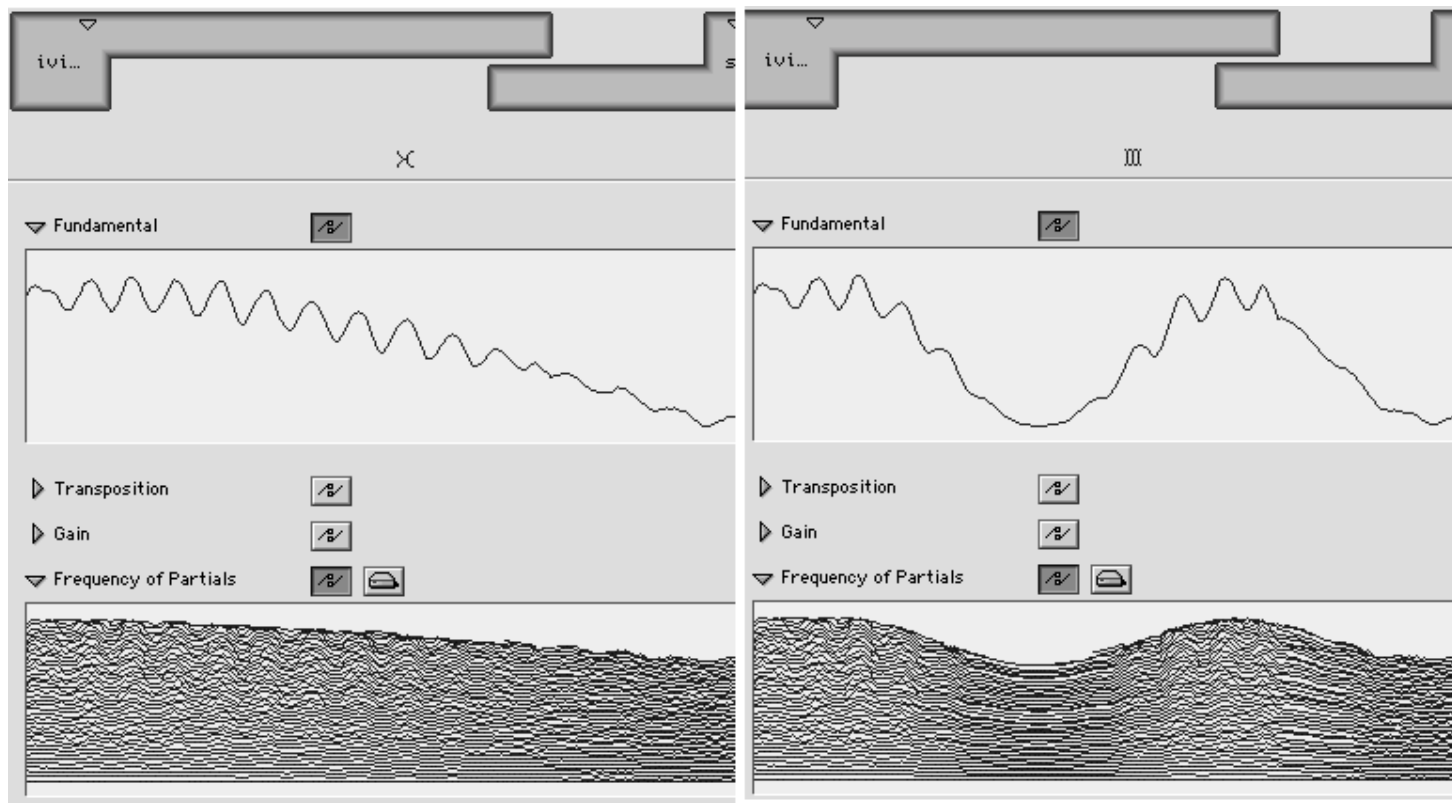
Dans le mode interpolé, les effets des modifications des transpositions et les changements de la fréquence fondamentale sont transmis immédiatement sur l'affichage des partiels.



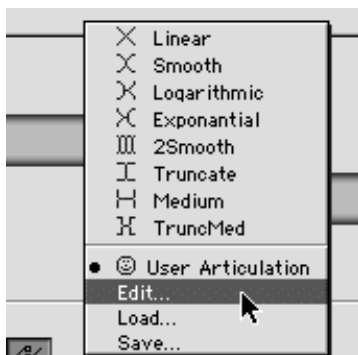
## 2.4. Articulations

Il existe un menu pour contrôler l'articulation de l'interpolation entre deux segments. L'exemple suivant montre la différence entre une interpolation exponentielle (*Exponential*) et une interpolation douce (*2Smooth*). Cette dernière est une sorte de mouvement d'aller et retour entre les données des deux segments.

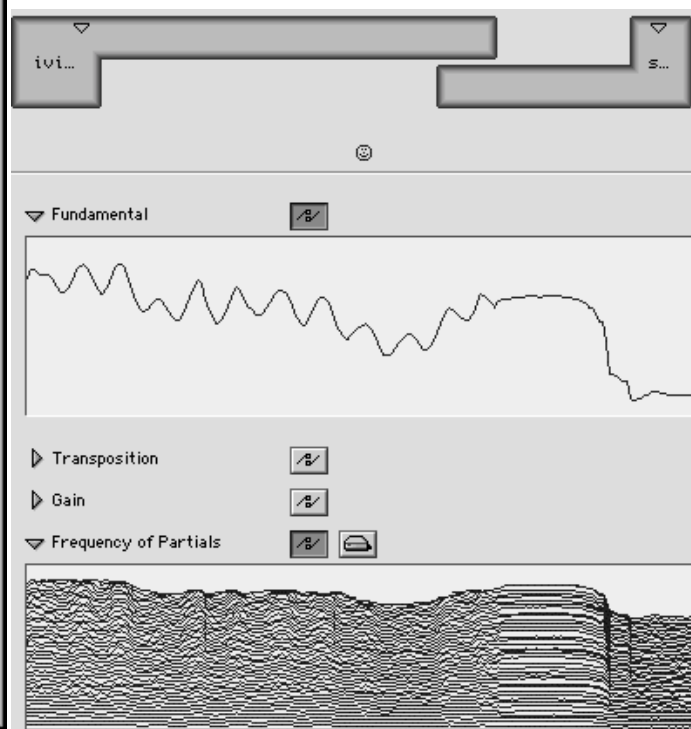
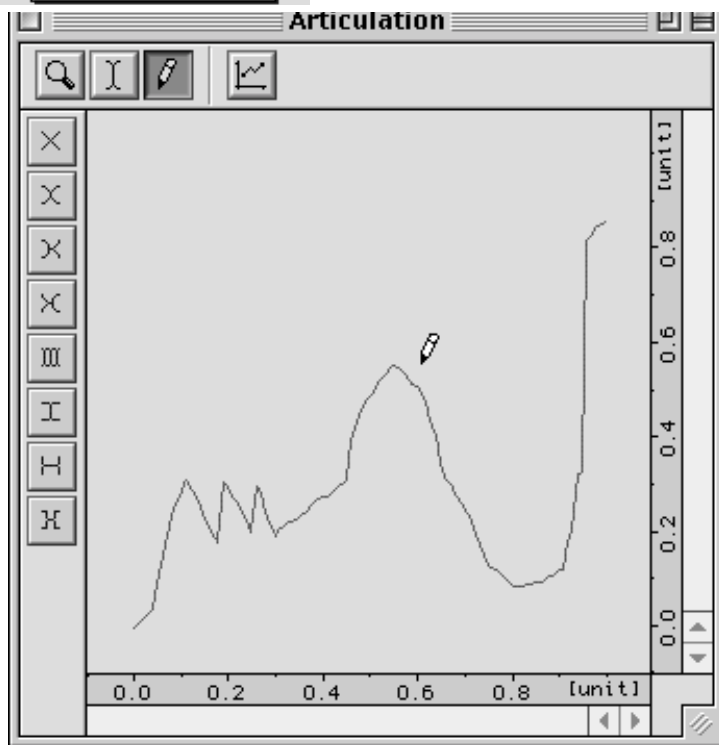
 Exemple sonore : java-shaku04







Il est également possible d'éditer une fonction d'interpolation personnalisée.



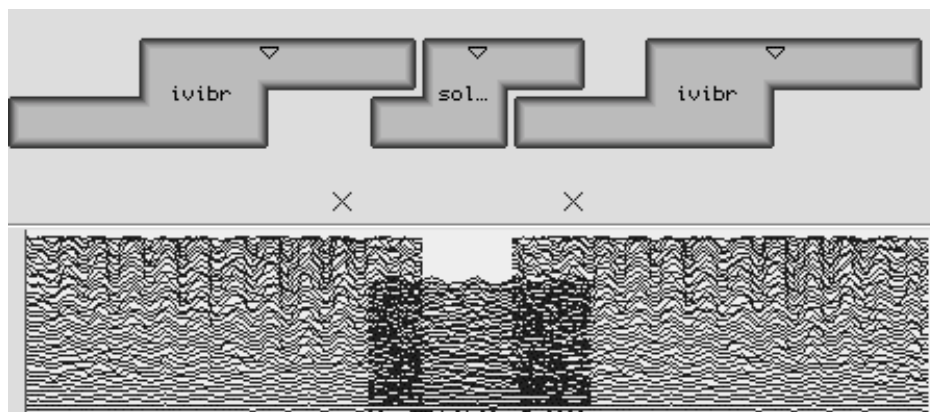
Le résultat de la courbe d'interpolation jouée, comme dans les exemples précédents, sur l'interpolation entre les deux segments de tous les paramètres.

 Exemple sonore : [java-shaku05](#)

## 2.5. Échanges des fréquences entre segments

Cet exemple provient d'une séquence réalisée avec les segments originaux provenant des deux dictionnaires.

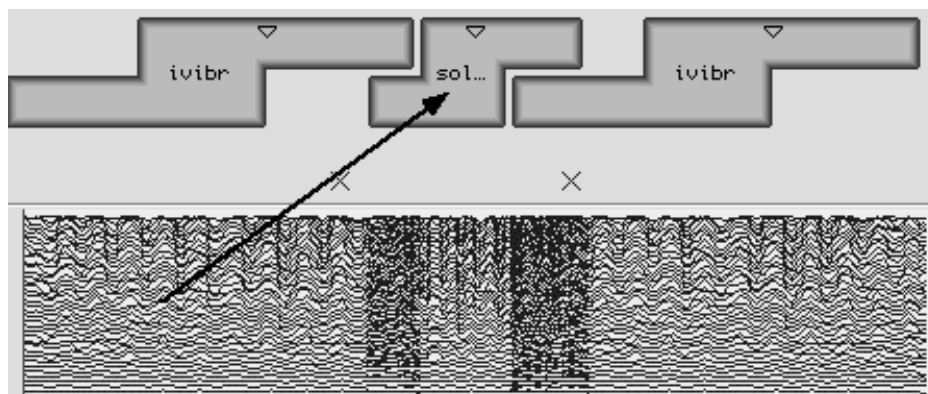
 Exemple sonore : java-shaku06



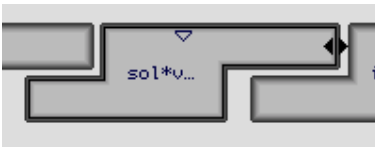
Si maintenant nous déplaçons les fréquences du premier segment de *Java* vers le segment *Shaku* suivant, les fréquences du segment de la voix sont jouées avec les amplitudes du segment de *Shaku*.

De plus, l'axe du temps est compressé puisque le segment de *Shaku* est plus court et donc la longueur des données est adaptée à la taille de ce segment.

 Exemple sonore : java-shaku07



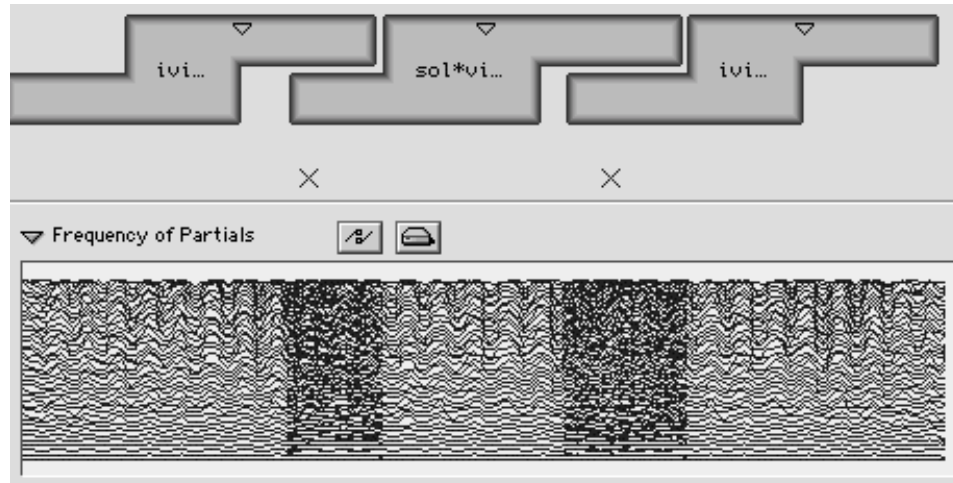
Si on déplace le pointeur de la souris sur le début ou sur la fin d'un segment, il se transforme en une double flèche noire avec laquelle on peut modifier la durée de ce segment.



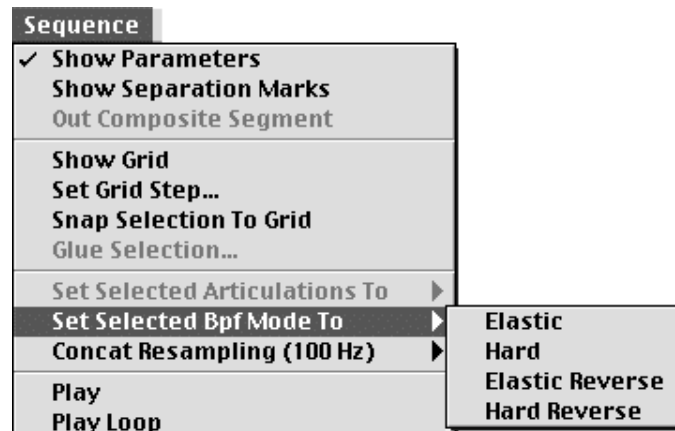
Après avoir étiré le segment du milieu, les données de fréquence de la voix sont jouées à nouveau approximativement à leur vitesse d'origine.

 Exemple sonore : java-shaku08

En suivant ce schéma, toutes sortes de combinaisons sonores peuvent être réalisées. Les amplitudes et les fréquences, ainsi que la fréquence fondamentale peuvent ainsi être combinées.

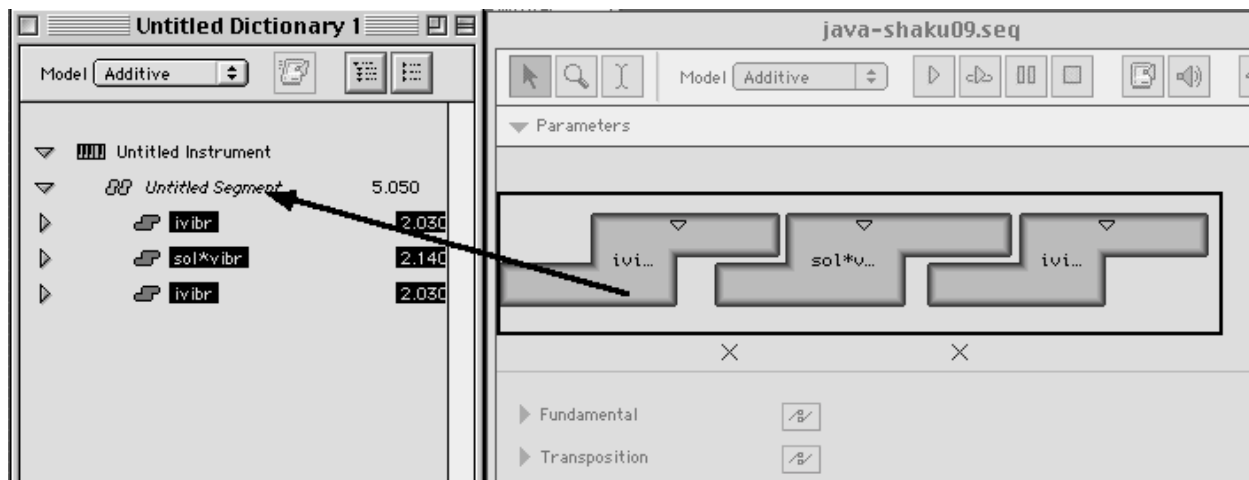


Si on choisit pour les BPFs le mode *Hard* dans le menu *Sequence*, les données ne s'adapteront plus à la durée de la séquence.

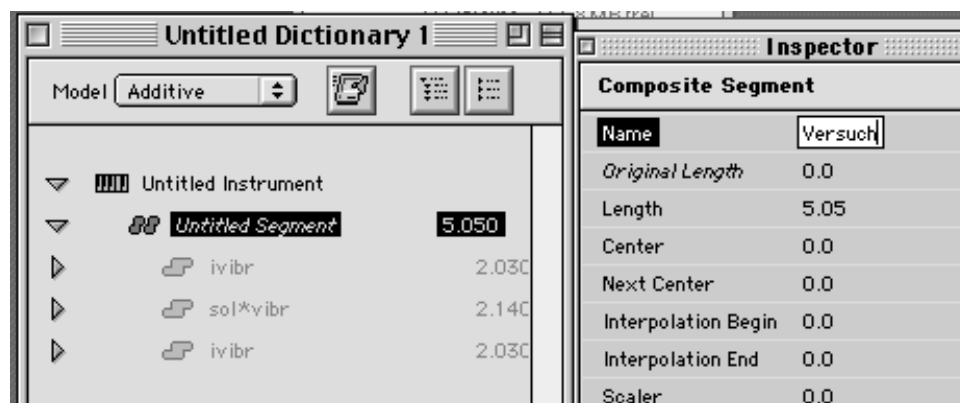


## 2.6. Les segments composites

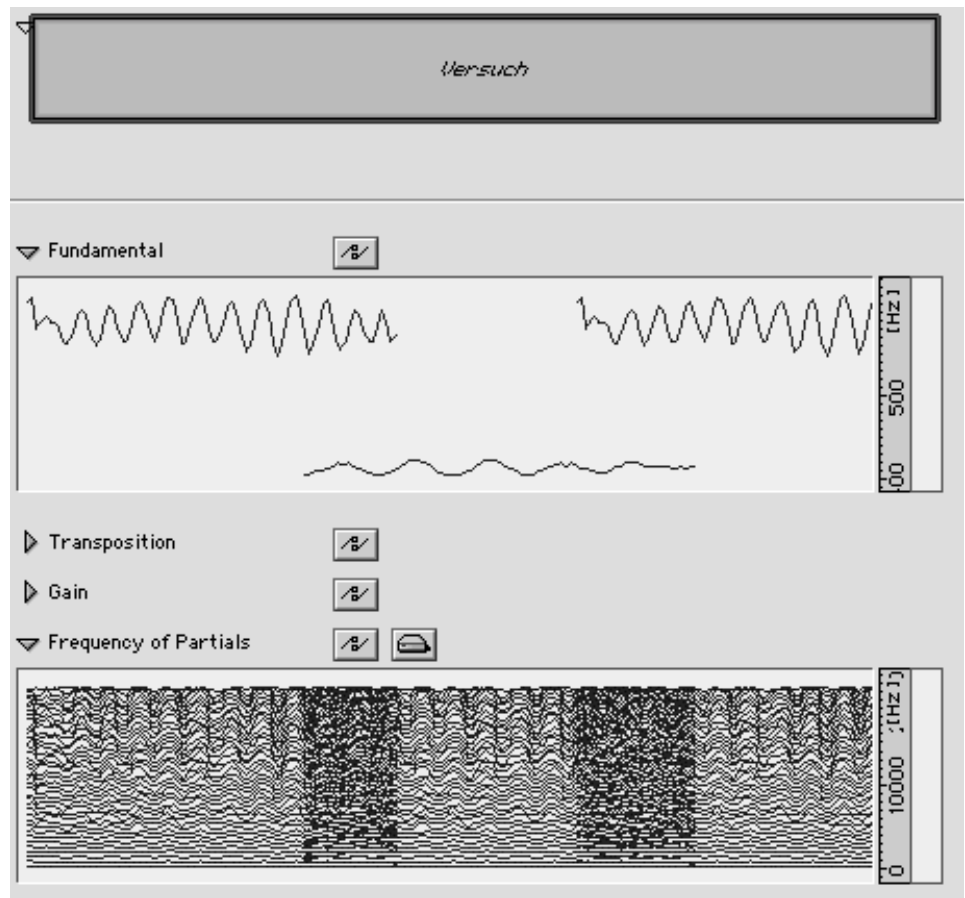
À partir de la séquence précédente, nous allons maintenant créer ce qu'on appelle « une séquence composite ». Nous créons un nouveau dictionnaire (menu *File->New Dictionary*) et dans ce dictionnaire nous créons un nouvel instrument (menu *Dictionary->Create Instrument*) et enfin un segment composite (menu *Dictionary->Create Composite Segment*). Il suffit alors de sélectionner tous les segments de la séquence et de les déplacer dans le segment composite.



Pour changer le nom, on utilise « l'inspecteur » (*Inspector*).

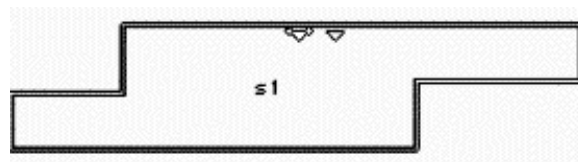


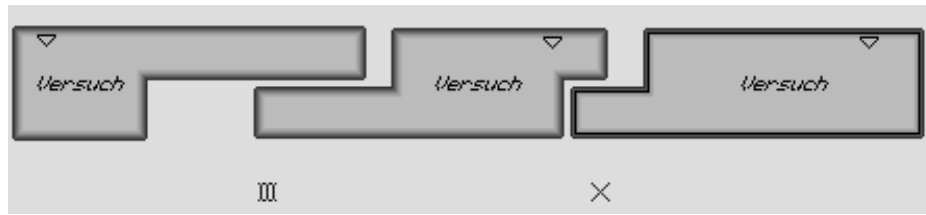
On déplace ce segment composite dans une nouvelle séquence. Il contient les données de trois segments élémentaires. Si nous voulons changer les données dans un segment composite, nous pouvons l'ouvrir avec un double-clic et l'éditer comme une séquence normale. En cliquant sur la flèche supérieure, on retourne sur le segment composite.



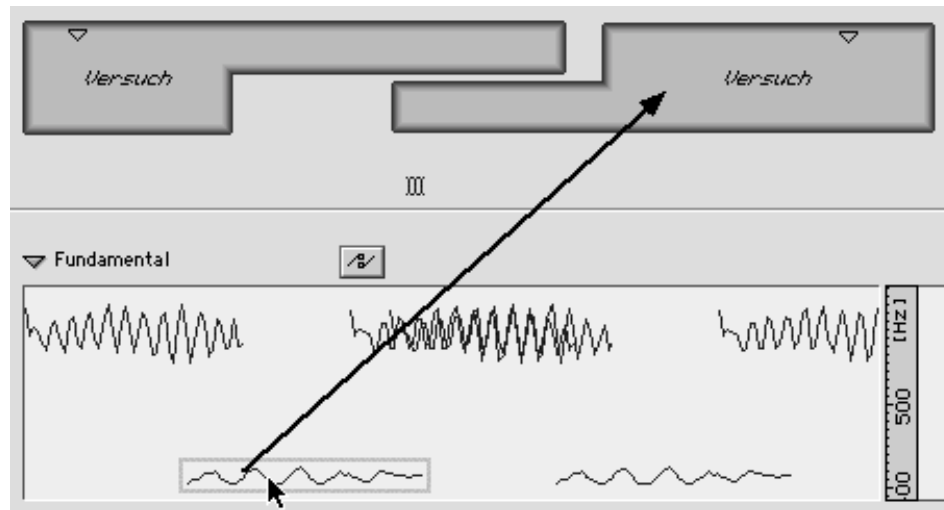
Un avantage des segments composites est qu'ils peuvent à nouveau être utilisés pour construire des séquences. Dans ce cas, les données à l'intérieur d'un segment composite sont vues comme une unité et l'interpolation vers le segment suivant débute à la fin de la zone centrale du segment composite.

Pour changer la taille de la zone centrale, il faut d'abord déplacer le centre en tenant la touche *Alt* enfoncée.

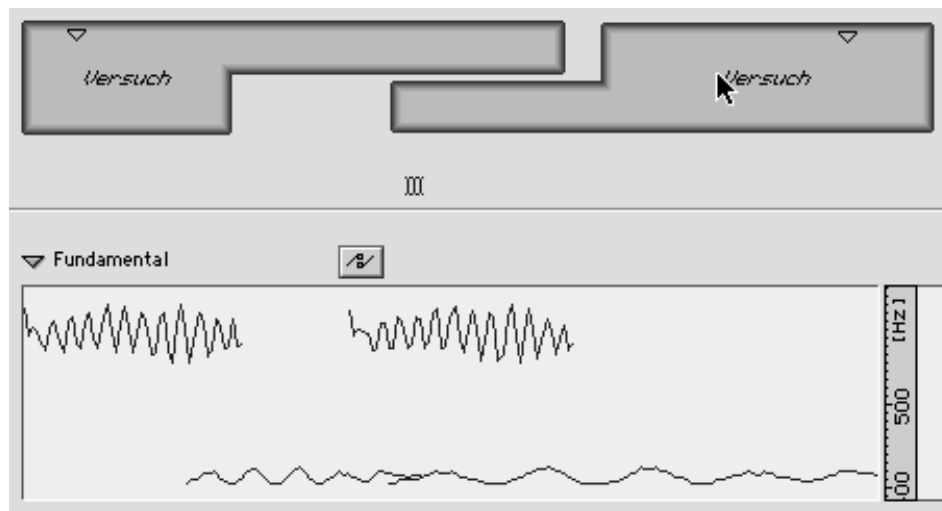




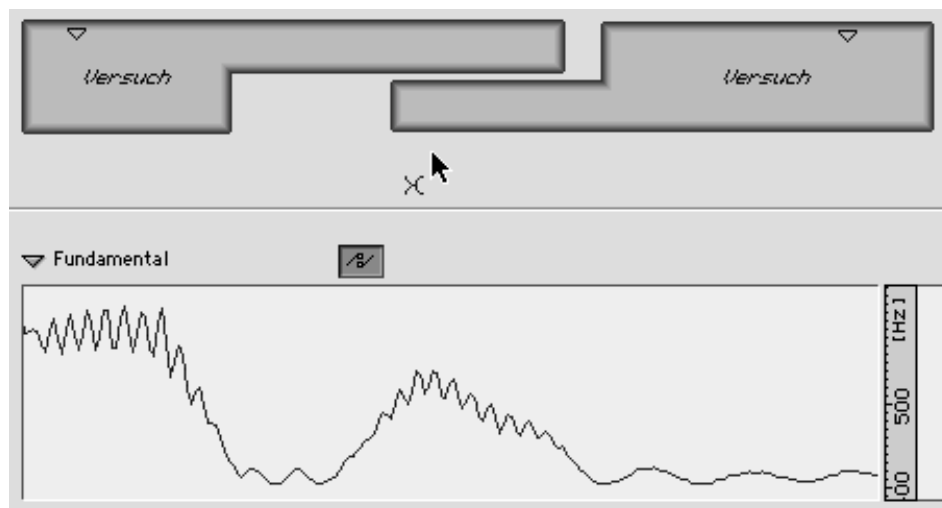
Une autre possibilité consiste à échanger les données qui, dans ce cas, vont affecter toutes les données de ce type (fréquence fondamentale, fréquences des partiels, etc.) contenues dans le segment composite.



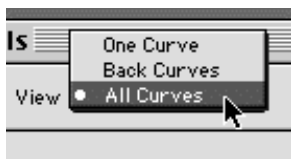
Si, par exemple, on déplace la fréquence fondamentale d'un segment vers un segment composite, celle-ci va remplacer toutes les fréquences fondamentales individuelles.



Ici, la visualisation des données est en mode interpolé et on voit très clairement l'évolution de l'interpolation exponentielle entre les deux segments composites.

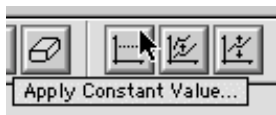


## 2.7. Édition de multi-BPFs



En double-cliquant sur un paramètre, comme les fréquences ou les amplitudes, on ouvre un éditeur similaire aux BPFs simples. Dans une multi-BPFs, il suffit simplement de sélectionner une courbe et c'est elle qui sera affectée par les transformations.

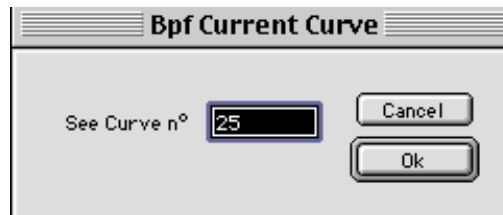
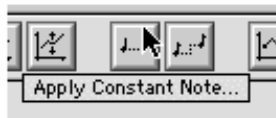
Les transformations suivantes sont disponibles :



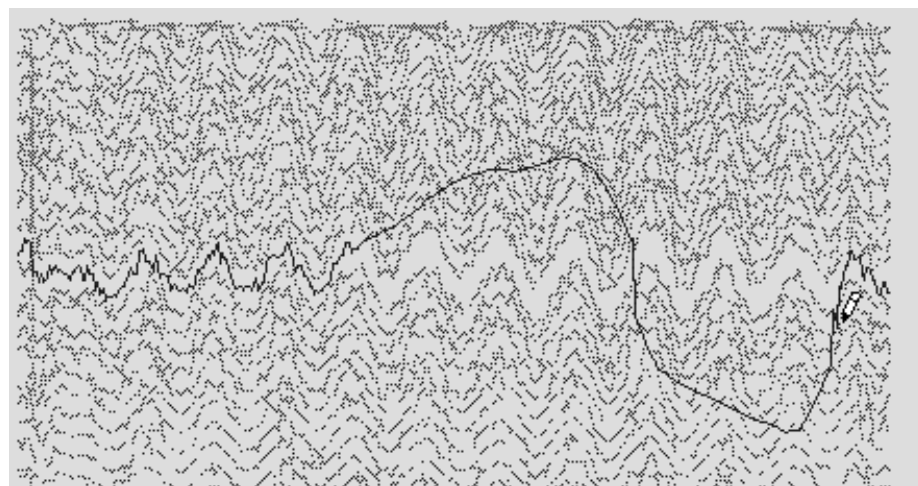
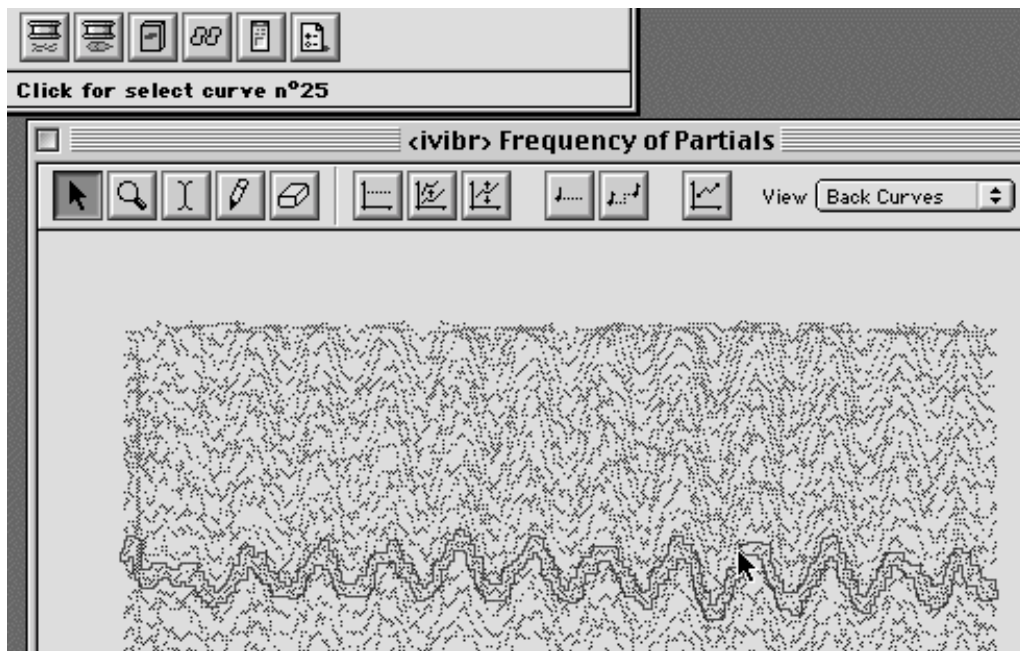
- Affecte une courbe ou l'ensemble des courbes à une valeur fixe,
- Change d'échelle en multipliant les données par un facteur dépendant d'une référence. Cette référence peut être nulle ou être relative : valeur minimale, moyenne ou maximale de la courbe actuelle,
- Ajoute une valeur fixe,
- Permet une conversion de hauteurs de notes en fréquences et leur utilisation dans les courbes,
- Transposition par intervalles.



Pour travailler sur des courbes individuelles dans un éditeur multi-BPFs, on choisit *Back Canvas* (ou *Back Curve*) et on peut alors sélectionner une courbe en cliquant dessus ou en entrant son numéro dans le menu BPF.

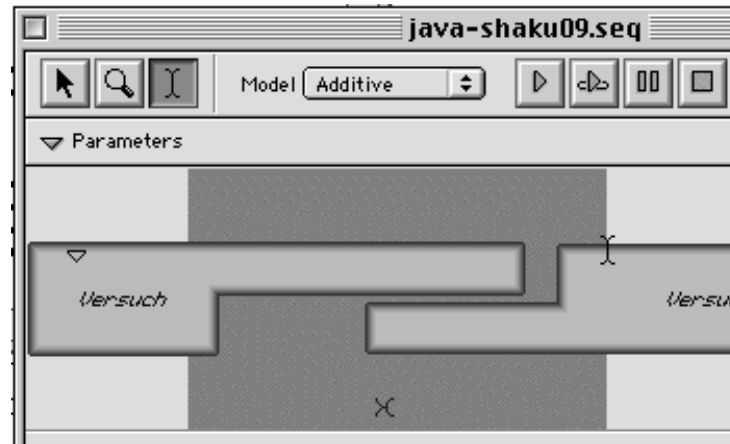






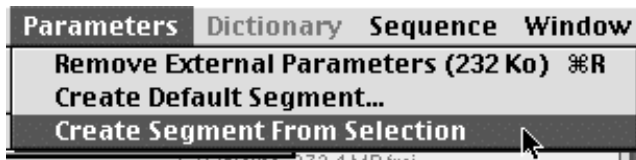
## 2.8. Créer un segment à partir d'une séquence ou d'une partie de séquence

Si on édite une séquence, on peut la définir, elle ou seulement un extrait, comme un nouveau segment. Cela crée alors un nouveau dictionnaire qui ne contient que ce segment. Cette méthode peut être très fructueuse lorsqu'on désire éditer des courbes d'interpolations simples entre des segments très éloignés.



## 2.9. Problèmes connus

Pour convertir des fichiers SDIF en ASCII, on a pour l'instant besoin d'une petite application externe *SDIF/ASCII Converter 1,.0b1*. Cette application peut être téléchargée (<http://www.ircam.fr/forumNet>) ou être envoyée par courrier électronique (66KB).



Il ne faut pas installer le programme trop profondément dans l'arborescence du disque dur, c'est-à-dire dans un nombre trop important de sous-dossiers. En effet, l'adresse des fichiers sur le disque serait trop longue et Diphone pourrait ne pas fonctionner correctement. Si votre structure ressemble à la suivante :

« :Macintosh:Project:Music:Programs:Sound:Ircam:SoundDesign:Diphone »

il est probable que des problèmes vont survenir. Dans ce cas, vous pouvez essayer de déplacer le dossier Diphone à un niveau supérieur :

« :Macintosh:Diphone »

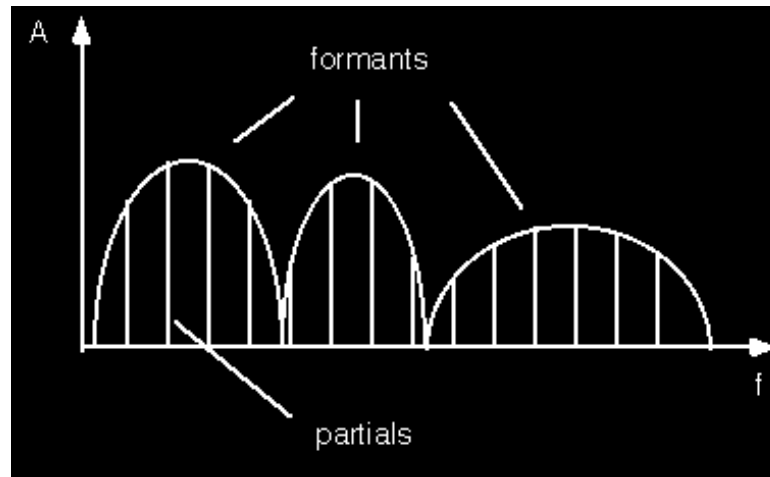
# 3 Analyse par Modèles de Résonance et synthèse avec Chant

La troisième et dernière partie va traiter de façon concise de la question très vaste des Modèles de Résonance et de la synthèse avec Chant. L'analyse par Modèles de Résonance est réalisée avec le programme ResAn (même sans connaissances sur le programme Diphone, il est possible d'obtenir des sons très intéressants avec les Modèles de Résonance).

La synthèse avec Chant peut être séparée en deux domaines qui peuvent respectivement être assimilées à de la synthèse additive et soustractive. Dans le premier cas, le spectre est construit en additionnant des formants. Dans le second cas, les définitions des formants sont utilisées pour filtrer des sons existants. Dans les deux cas, les paramètres des formants peuvent être fournis arbitrairement ou d'après l'analyse de sons existants.

## 3.1. Les formants

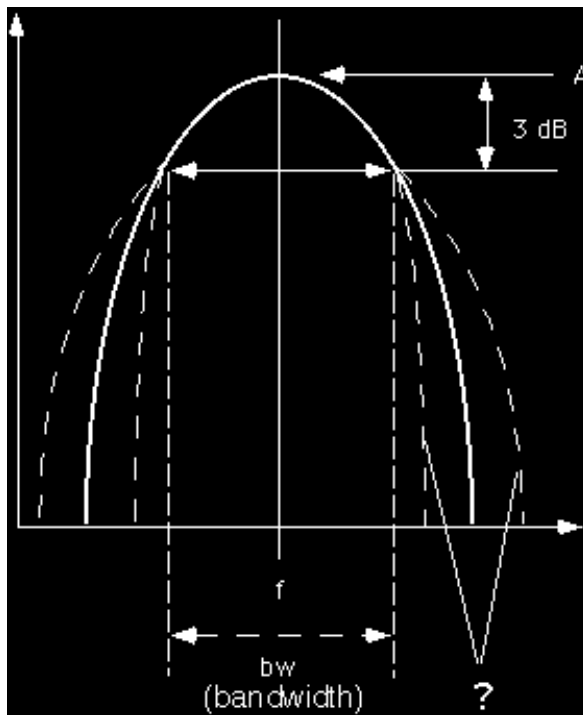
La définition courante d'un formant correspond à une zone dans le spectre qui contient plus d'énergie que les zones de fréquence voisines. C'est en fait un groupe de partiels qui dépassent dans le spectre.



Un formant est décrit par trois paramètres :

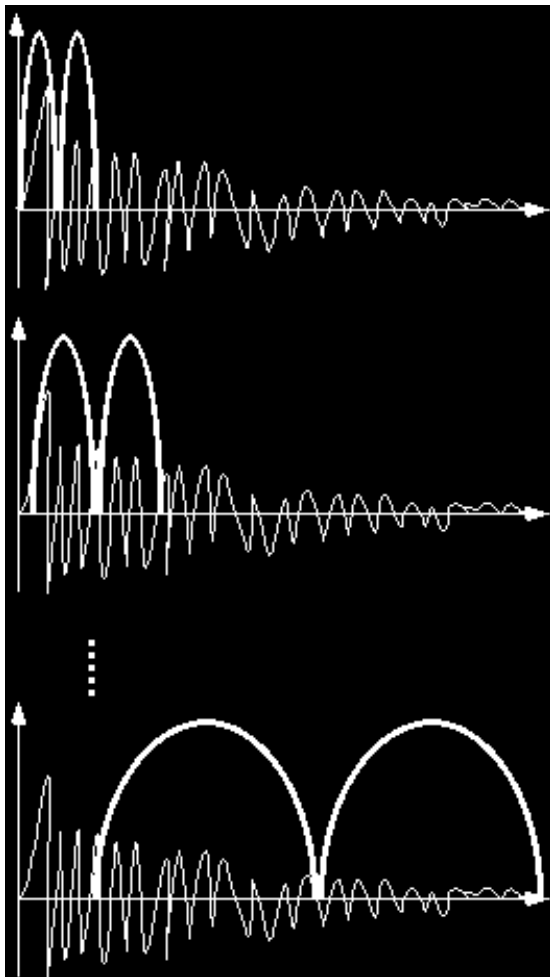
- sa fréquence centrale ( $f$ ),
- son amplitude pour la fréquence centrale ( $a$ ),

- sa largeur de bande (bw) (ouverture du formant à un point où l'amplitude est de 3dB inférieure à l'amplitude maximum — la largeur de bande est exprimée en Hz).



On voit ici que la définition d'un formant est relativement imprécise puisqu'avec ces trois paramètres, il n'est pas possible de définir comment se comporte l'ouverture du spectre dans les régions plus basses.

## 3.2. Analyse par Modèles de Résonance



Ce qu'on appelle des Modèles de Résonance sont des modèles vibratoires qui sont excités par une courte décharge d'énergie puis sont laissés en résonance. Les fréquences restent fixes pendant la résonance. Par exemple, les sons frappés ou pincés appartiennent à ces modèles.

Si on part d'un son enregistré, l'analyse de ces modèles cherche à estimer les qualités de la résonance et à trouver un ensemble de formants avec leurs fréquences, leurs amplitudes et leurs largeurs de bandes. Cet ensemble est fixe et ne doit pas évoluer au cours du temps.

L'analyse est réalisée au cours de plusieurs passages successifs. Elle doit décrire le moment de l'attaque ainsi que celui de la résonance. C'est pourquoi l'analyse utilise deux fenêtres voisines : celle de gauche analyse directement l'attaque et celle de droite la résonance qui lui succède.

Les deux segments de son (fenêtres) sont analysés par une FFT (dont le principe a déjà été décrit dans la première partie de cet article) et on compare ensuite les fréquences (les pics du spectre) obtenues dans les deux fenêtres.

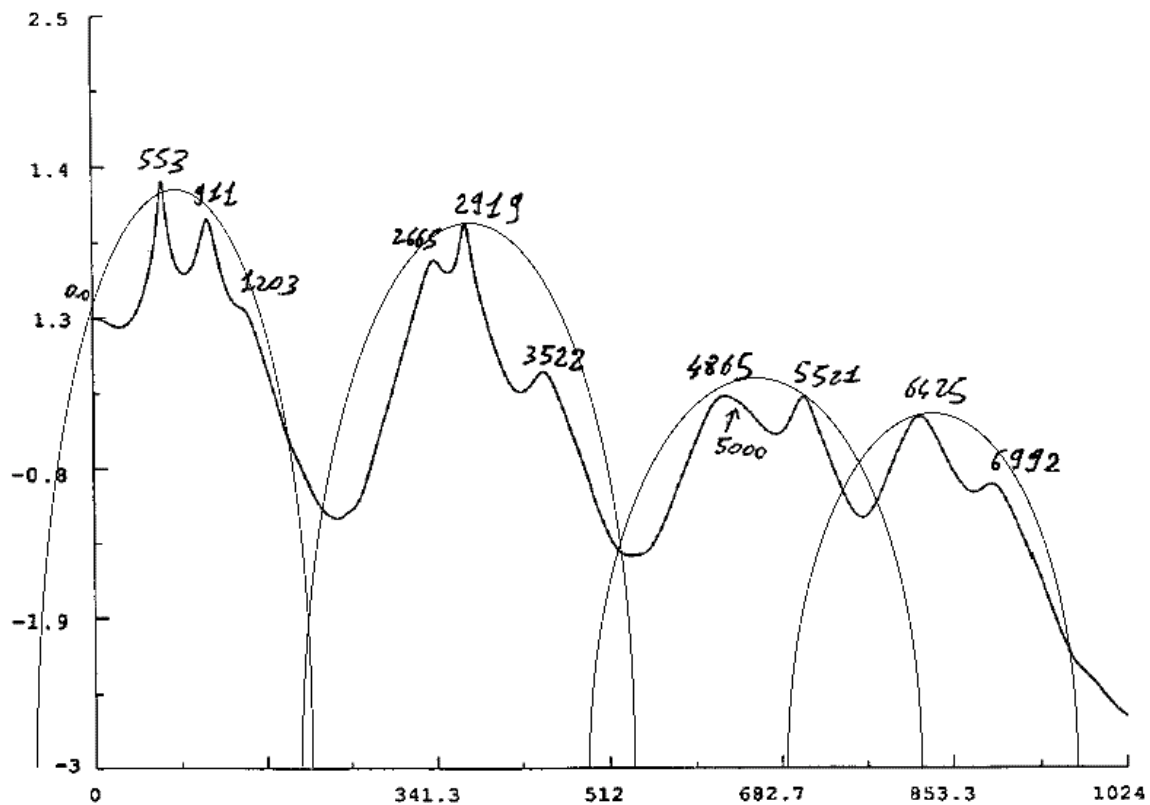
Si une fréquence de résonance apparaît dans les deux fenêtres d'analyse, on en déduit qu'il s'agit bien d'une composante du spectre du modèle analysé. En comparant les amplitudes obtenues dans les deux fenêtres, on peut estimer la durée d'atténuation de cette fréquence.

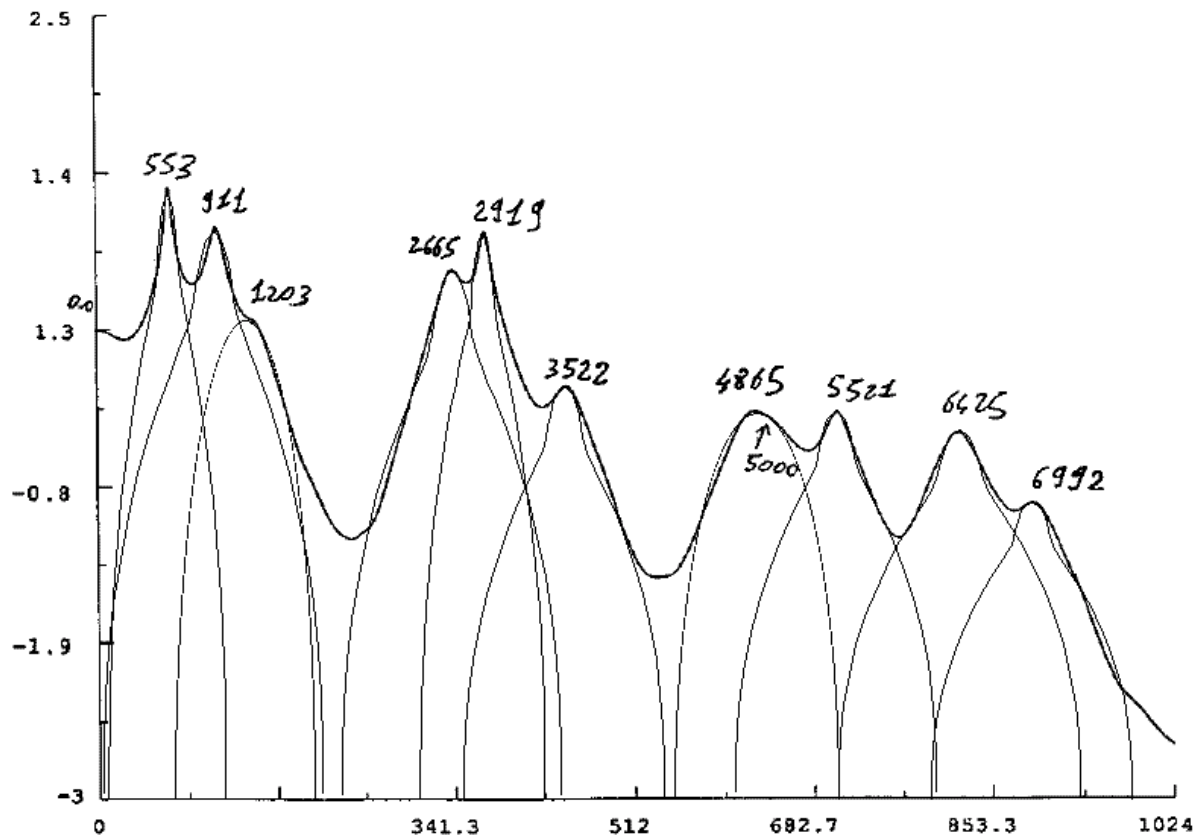
Le premier passage fournit ce qu'on appelle un modèle élémentaire. Pour les passages suivants, les fenêtres sont agrandies et légèrement déplacées de l'attaque vers la résonance.

À la fin de chaque passage, le modèle généré par l'analyse est comparé avec le modèle produit par le passage précédent. Si on trouve des correspondances de fréquences (ou des fréquences vraiment très proches les unes des autres), celles-ci sont conservées dans un nouveau modèle enrichi. Les amplitudes et largeurs de bandes de ces composantes sont calculées en comparant les anciennes valeurs et les nouvelles. Si certaines résonances ont disparu ou si de nouvelles résonances apparaissent, elles restent dans le modèle ou lui sont ajoutées (respectivement).

Du fait que la taille de fenêtre (et la FFT associée) de la première analyse est très petite, on dispose d'une résolution de fréquence assez limitée au moment de l'attaque. Cela donne des formants très larges qui laissent passer une grande quantité d'énergie. Cela correspond au comportement naturel des modèles qui présentent au moment de l'attaque un spectre très riche qui met en vibration chacune des parties des corps vibrants. L'énergie de l'attaque se dissipe très vite et il ne reste rapidement que les résonances du modèle.

En élargissant les fenêtres à chaque passage de l'analyse, on obtient des fréquences de plus en plus précises et un gradation progressive des formants.

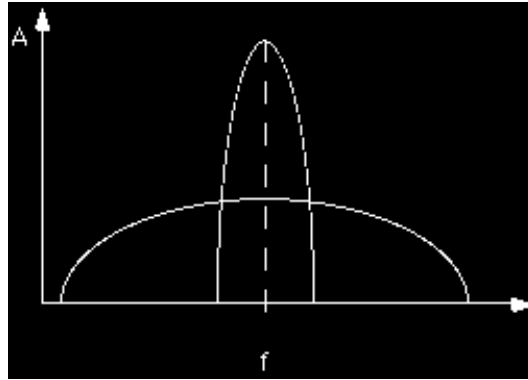




L'analyse d'un son donné avec de petites fenêtres produit un nombre de formants réduit et ceux-ci sont plutôt larges.

L'analyse avec de grandes fenêtres permet le calcul des données des formants avec plus de précision. Des formants larges laissent passer l'énergie et résonnent peu, alors que des formants serrés « prennent plus de temps » pour laisser passer une certaine quantité d'énergie et résonnent donc plus longtemps. Ceci est une explication simplifiée des relations existant entre largeur de bande et durée de résonance. Plus les formants sont serrés, plus ils sonnent longtemps.

Avec les modèles enrichis, il peut arriver qu'une fréquence soit représentée par plusieurs formants ayant différentes amplitudes et largeurs de bande. Ainsi, on peut modéliser les formants larges correspondant à l'attaque et les formants étroits de la résonance qui lui succède.



Tous les paramètres pour la comparaison des modèles élémentaires et enrichis peuvent être contrôlés individuellement (environ 25 paramètres par passage de l'analyse). Les premières tentatives peuvent être réalisées de façon assez simple en utilisant les presets. J'ai tendance à préférer le preset n°6, avec six passages, qui donne de très bons résultats pour une grande variété de sons en entrée.

Après six passages de l'analyse, on trouve dans le dossier « ImpExport:Chant » les six fichiers SDIF qui correspondent aux modèles enrichis des différents passages

Sur la page de dialogue « Synthesis » dans ResAn, on peut resynthétiser les différents passages élémentaires.

On peut entendre dans cet exemple qu'à chaque passage de l'analyse, le spectre des formants est enrichi.

 Exemple sonore : Original: crotale

 Exemple sonore : Resynthesis: crotale.m1

 Exemple sonore : Resynthesis: crotale.m2

 Exemple sonore : Resynthesis: crotale.m6



### 3.3. Création de modèles de résonance « hors du commun »

Les méthodes d'analyse décrites précédemment peuvent être appliquées non seulement à des sons réellement résonants mais également à des séquences sonores longues qui ne correspondent pas à ce type de modèle. Cela produit des sons résonants très surprenants qui correspondent à la somme des formants les plus importants du son analysé.

<i>Original</i>	<i>Resynthèsr</i>
<a href="#">huhn</a>	<a href="#">huhn.m1</a> , <a href="#">huhn.m2</a> , <a href="#">huhn.m5</a>
<a href="#">bulg-2</a>	bulg-2.m5
bulg-frau	bulg-frau.m5
sundanésie-tembang	sundanésie-tembang.m5

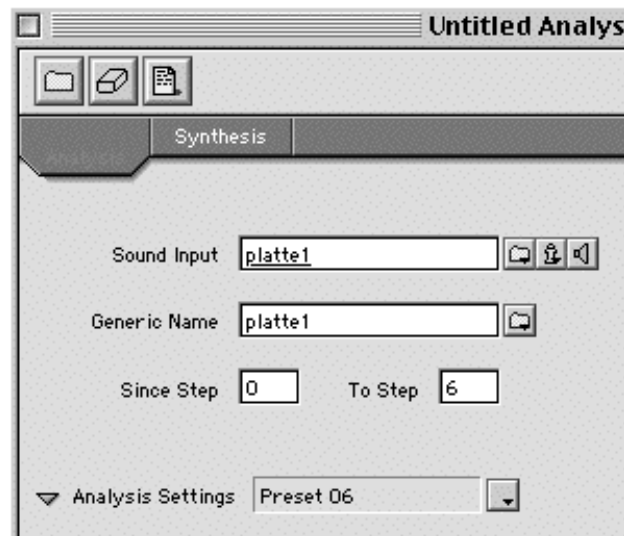
 Exemple sonore : huhn

 Exemple sonore : huhn.m1)

 Exemple sonore : huhn.m2

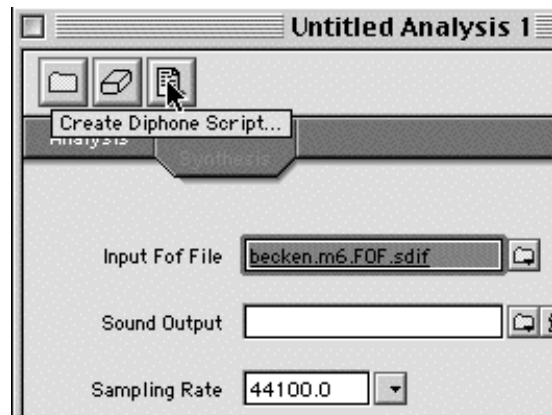
 Exemple sonore : huhn.m5

 Exemple sonore :bulg-2



## 3.4. Dictionnaires

Comme nous l'avons indiqué dans la partie consacrée à l'analyse additive, les données d'analyse doivent être stockées dans ce qu'on appelle des dictionnaires. Ceux-ci contiennent des instruments et des segments qui peuvent être utilisés pour créer de nouvelles séquences.



Si on est satisfait de la resynthèse réalisée avec ResAn, on peut générer le script nécessaire pour la création du dictionnaire (les étapes sont les mêmes que pour AdAn).

On génère un dictionnaire dans Diphone en sélectionnant dans ce cas le modèle Chant.



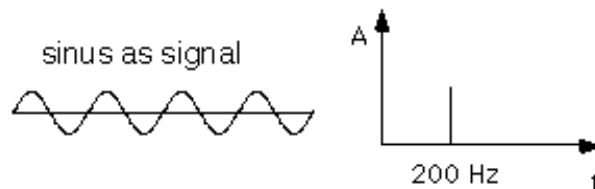
Puisque la définition des formants ne change pas pendant toute la durée des segments, la segmentation des sons sources n'est pas nécessaire. Un segment unique est créé pour chaque son, contenant toutes les données des formants.



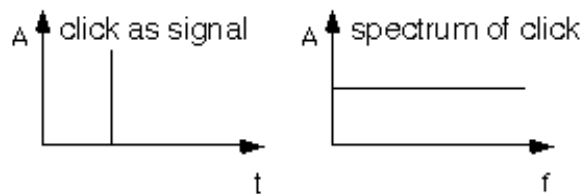
### 3.5. Synthèse par FOF

Avec la synthèse par FOF, les données d'analyse des formants sont utilisées pour générer des formants purement synthétiques. Une FOF (fonction d'onde formantique) est une oscillation sinusoïdale munie d'une enveloppe. Là encore, on voit qu'il y a une relation directe entre le comportement temporel d'un événement sonore et sa définition spectrale (en fréquence). Cela est décrit de façon plus détaillée dans la première partie de cet article sur Diphone. Je vais néanmoins rappeler les principes de base.

Théoriquement, une oscillation sinusoïdale a une définition spectrale correspondant à une fréquence unique.

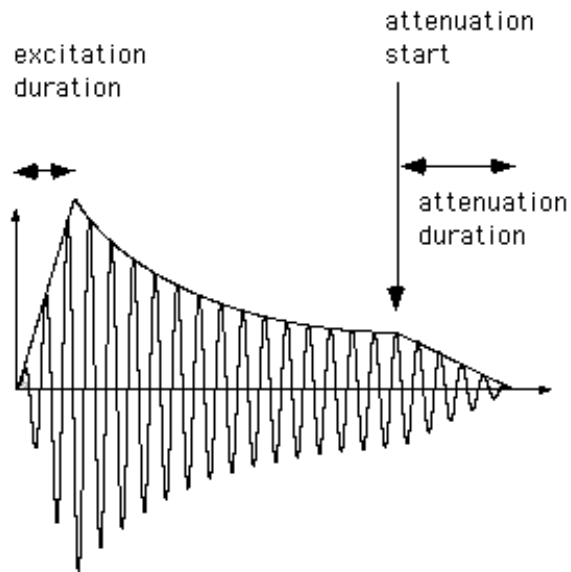


Le son le plus court possible, un clic, a un spectre complètement plat. Une définition en fréquence est impossible puisque le son n'a aucune périodicité et n'est formé que d'un seul échantillon.

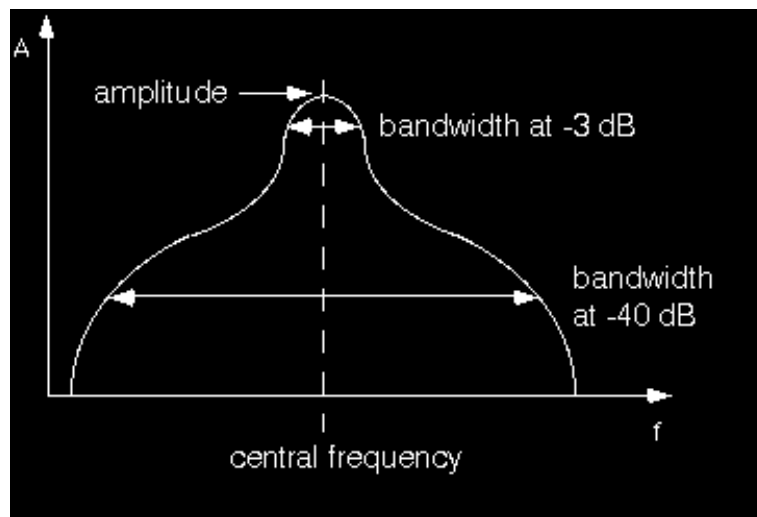


En fait, la définition spectrale précédente d'un sinus n'est valable que si celui-ci a toujours existé et existera toujours. À partir du moment où celui-ci a un début et une fin, c'est qu'on le multiplie par une enveloppe d'amplitude. Cette enveloppe peut être elle-même interprétée comme un signal avec son propre spectre. Du fait de la multiplication des deux signaux (sinus et enveloppe), le résultat est la convolution des deux spectres et la définition simplifiée d'une fréquence unique pour un sinus n'est plus valable.

On peut exploiter ce fait et utiliser la durée d'une onde sinus pour contrôler sa définition spectrale. L'oscillation sinusoïdale d'une FOF est soumise à l'enveloppe suivante : attaque linéaire avec un contrôle de la durée du temps d'excitation, décroissance exponentielle immédiate et enfin, à partir d'une date de début d'atténuation, décroissance linéaire pendant la durée d'atténuation.

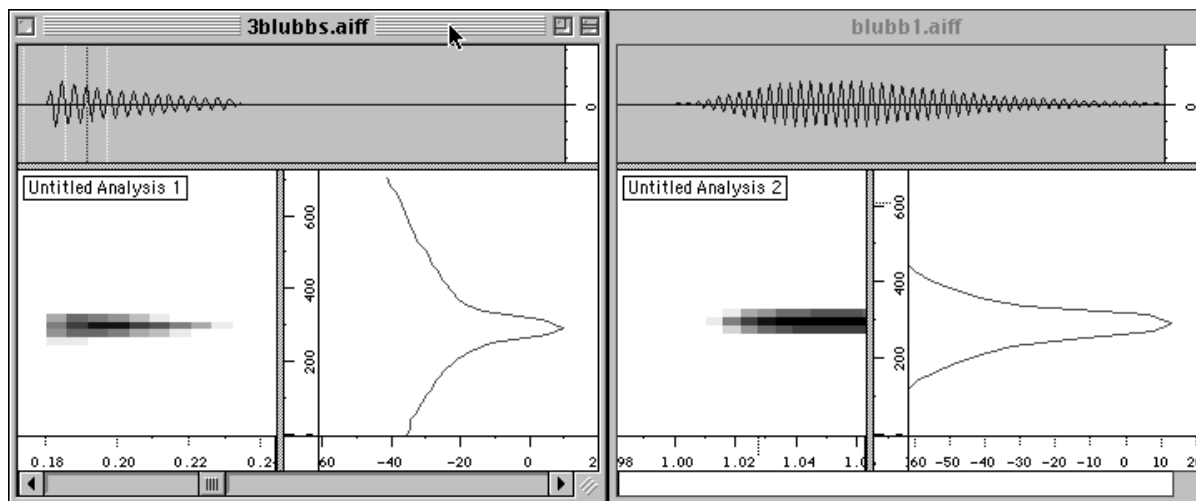


Du point de vue spectral, le signal résultant est un formant. Sa définition spectrale (largeur de bande) dépend de la durée du signal. De plus, il y a un autre paramètre qui est la largeur de bande à  $-40$  dB (relativement à l'amplitude maximale du formant) et qui est liée de façon réciproque au temps d'attaque : plus la « durée d'excitation » est longue, plus la largeur de bande à  $-40$  dB est étroite.



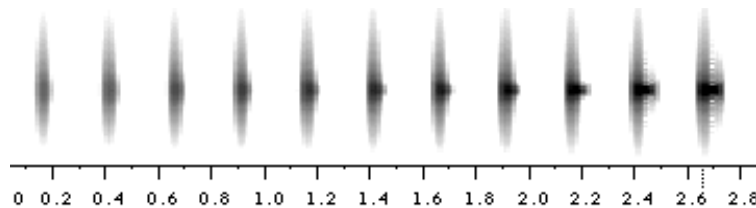
Si on joue une FOF de ce type, on entend un bip court avec un spectre formantique conforme à nos indications.

Dans la figure suivante, on voit le signal avec sa représentation en sonagramme et le formant résultant. La fréquence centrale du formant correspond ici à la fréquence du sinus joué. Pour l'image de droite, la durée d'excitation a été augmentée et on voit très clairement que la largeur de bande à  $-40$  dB est beaucoup plus fine.



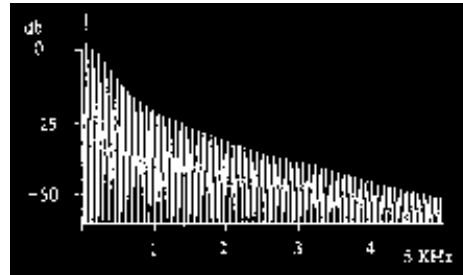
Si on diminue la largeur de bande du formant, sa durée de résonance augmente.

 Exemple sonore : bandwidth

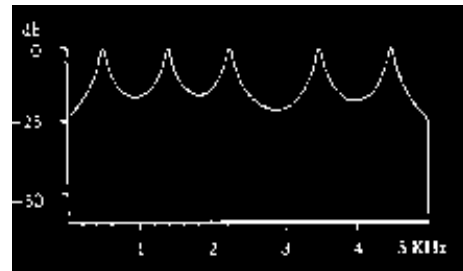


Chant a été développé à la fin des années 1970 à l'Ircam. Il a été initialement conçu pour la synthèse de la voix chantée. Pendant le chant, les cordes vocales génèrent un spectre harmonique riche qui est filtré par le conduit vocal.

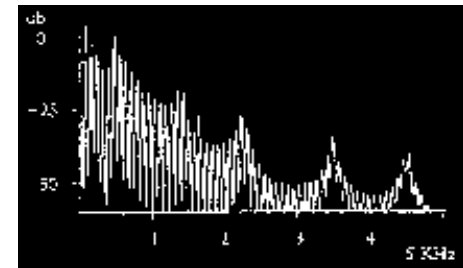
## Spectre harmonique riche



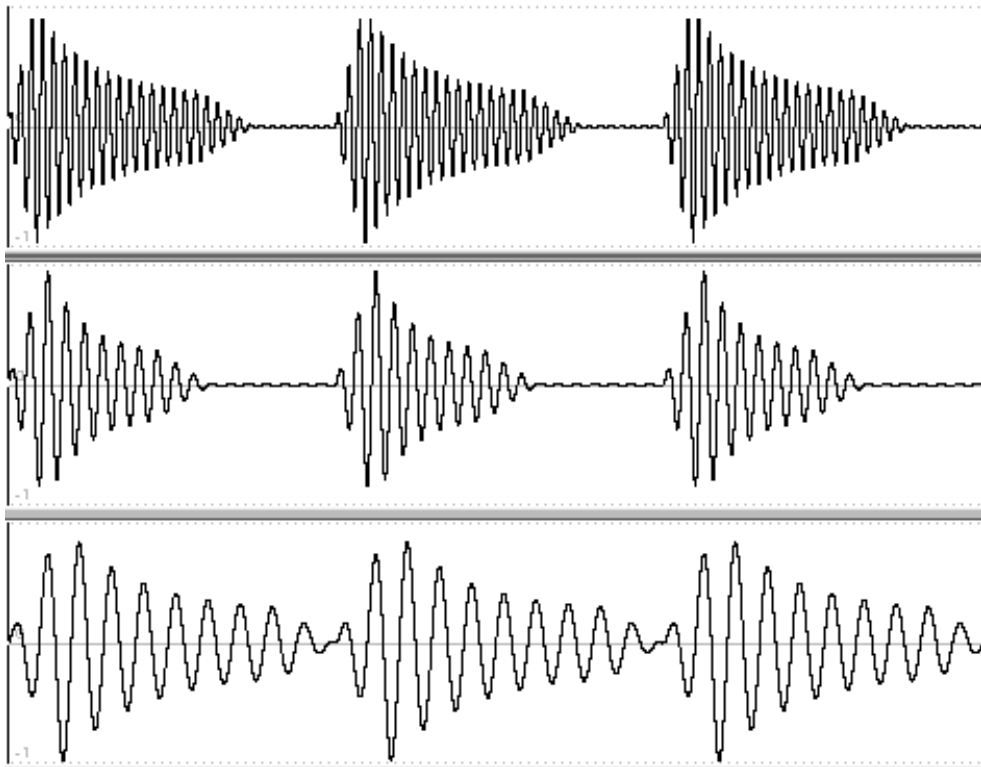
## Filtrage du conduit vocal

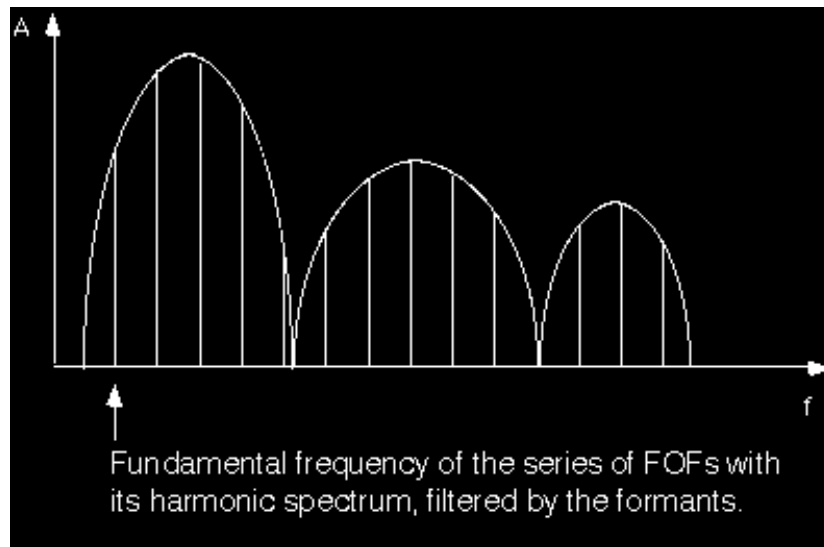


## Spectre résultant



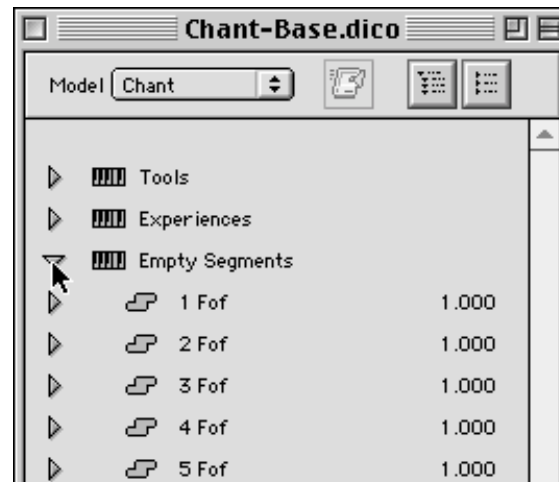
Si on veut générer plusieurs formants avec des FOFs, on doit utiliser plusieurs signaux sinusoïdaux en parallèles qui sont déclenchés de façon synchrone, c'est-à-dire avec la même fréquence fondamentale. Chaque flux de sinus génère un formant.





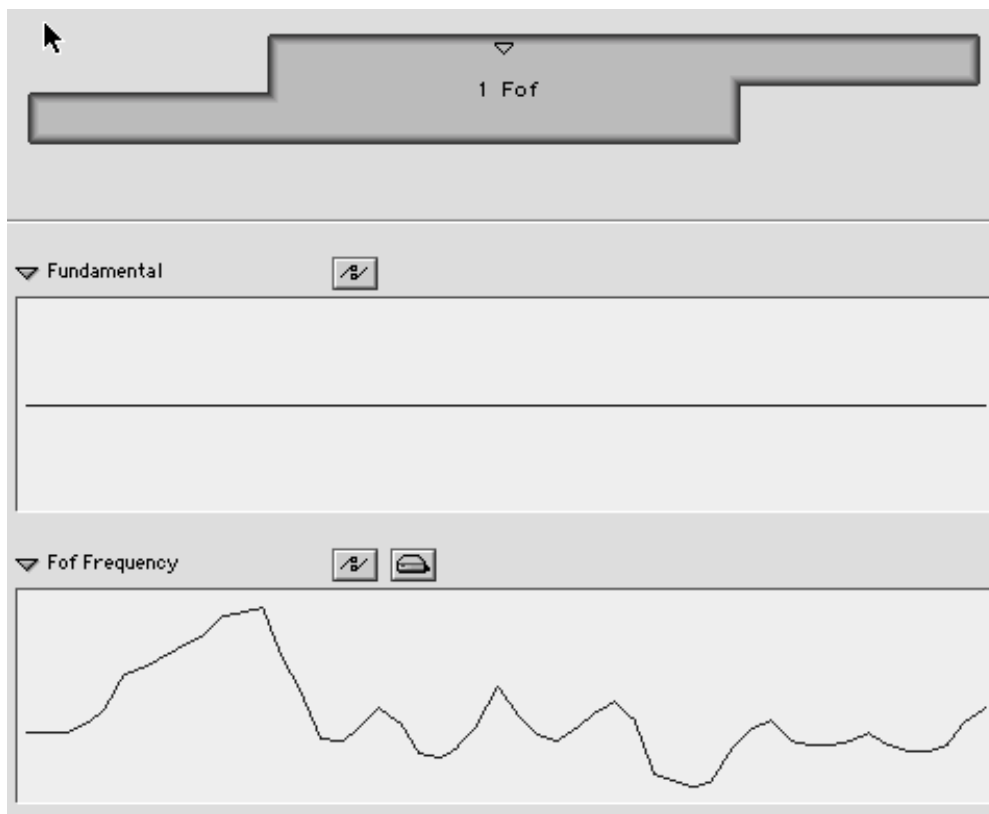
La fréquence fondamentale (la même pour toutes les FOFs) est à la base d'un spectre harmonique qui est filtré par les formants. Chaque train de FOFs peut avoir ses propres réglages d'enveloppe, ce qui peut produire pour les différents formants des largeurs de bandes différentes. Les amplitudes des formants proviennent des amplitudes des signaux sinusoïdaux.

Le dossier de Diphone « *dico&seq* » contient le dictionnaire Chant « Chant-base.dico » qui contient un instrument avec 16 séquences vides. À partir de ces segments par défaut, on peut construire, à partir de rien, des séquences contenant de un à seize formants.





En fait, toutes les manipulations des segments dans une séquence, comme l'articulation, les zones d'interpolation, les séquences composites, sont applicables à la synthèse avec Chant.



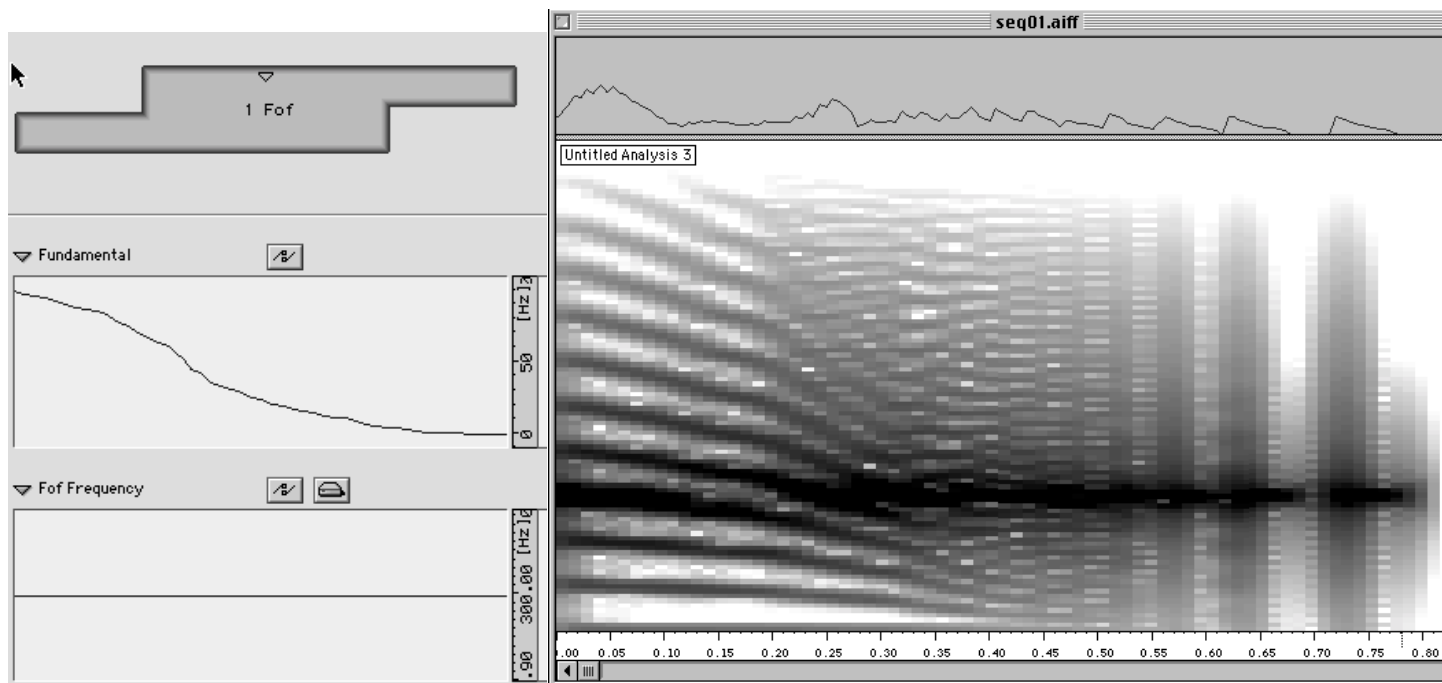
Je donne maintenant une petite série d'exemples sonores qui décrivent l'utilisation des principaux paramètres.

L'exemple sonore « chant-seq01 » est un segment avec une fréquence centrale de 300 Hz. Seule la fréquence fondamentale, c'est-à-dire la fréquence avec laquelle les FOFs individuelles sont déclenchées, descend de 100 Hz à 0 Hz.

En dessous de 16 Hz, on entend le rythme des départs des FOFs individuelles.

 Exemple sonore : chant-seq01

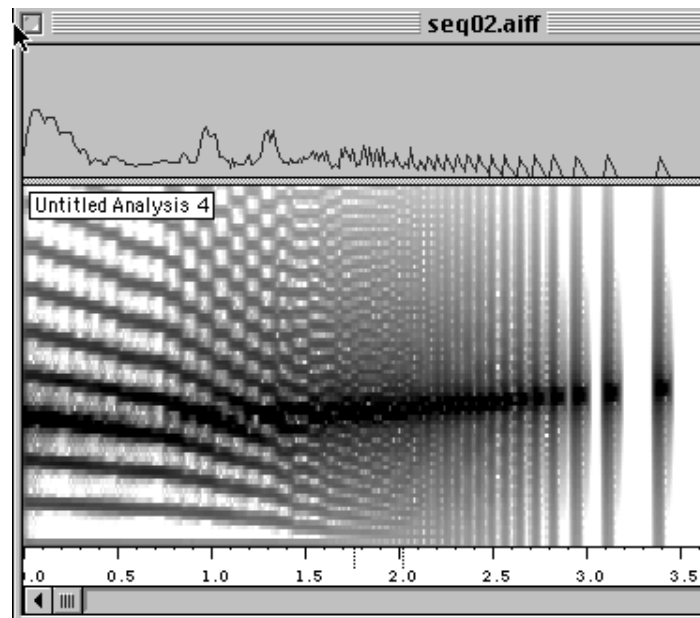
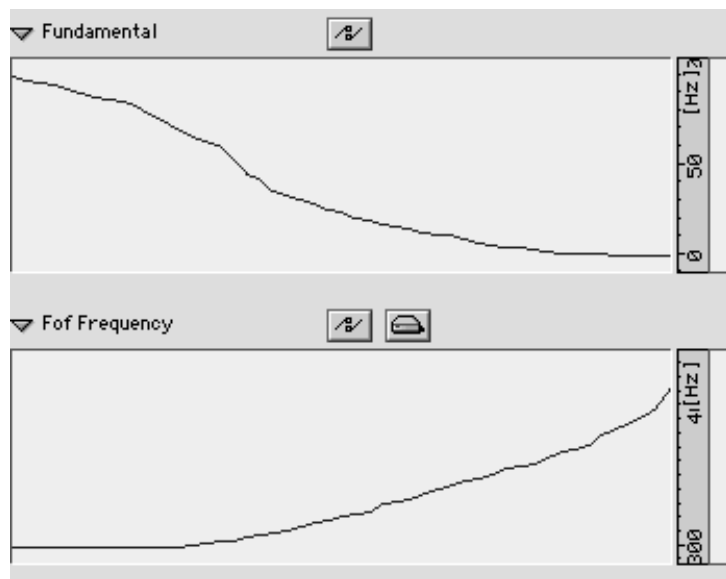
La représentation spectrale montre clairement le spectre harmonique basé sur 100 Hz et qui décrit un glissando descendant. On peut aussi voir nettement la fréquence centrale du formant située vers 300 Hz.



Dans l'exemple « chant-seq02 », la fréquence fondamentale descend comme dans l'exemple précédent.

 Exemple sonore :chant-seq02

Par contre, la fréquence du formant augmente de 300 Hz jusqu'à 400 Hz.



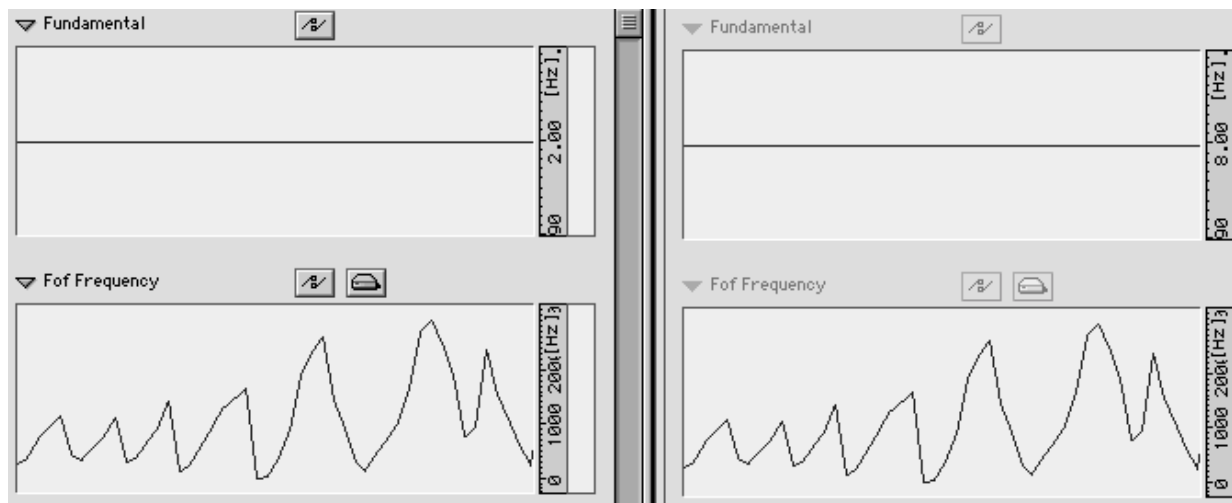
Les changements des paramètres des FOFs ne sont pas continus. En effet, lorsqu'une FOF débute, on lui applique les paramètres courants définis à cet instant. Pendant la durée de vie de la FOF, ces valeurs restent constantes.

L'exemple « chant-seq03 » a la même évolution en fréquence que « chant-seq04 ».

 Exemple sonore : chant-seq03

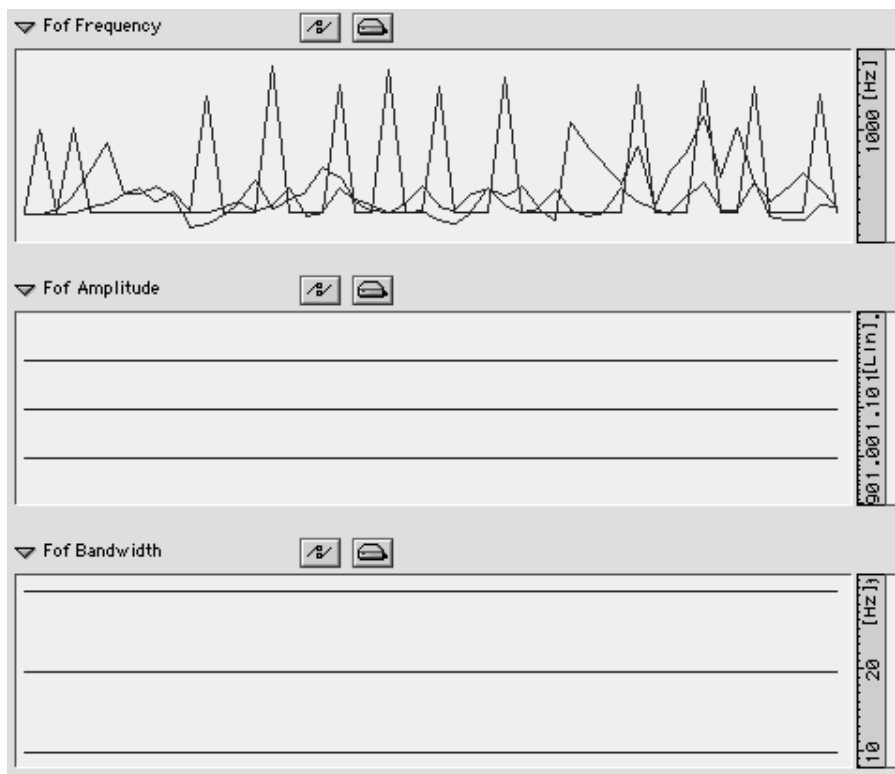
 Exemple sonore : chant-seq04

Mais du fait que la fréquence fondamentale dans « chant-seq03 » est seulement de 2 Hz, les valeurs courantes sont lues à partir de la courbe de fréquence seulement tous les 500 ms. L'exemple « chant-seq04 » a une fréquence de 8 Hz.



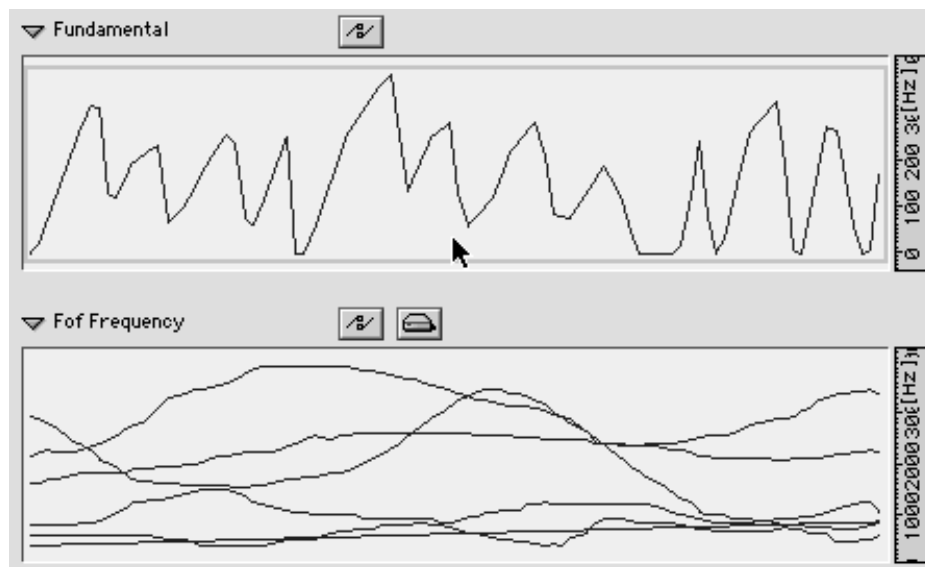
L'exemple « chant-seq05 » est une séquence avec trois formants dont les fréquences varient de façon continue. Les trois amplitudes et largeurs de bandes restent constantes.

 Exemple sonore : chant-seq05



Dans « chant-seq06 », la fréquence fondamentale oscille entre 0 et 500 Hz. Du fait qu'il s'agit d'un segment formé de 6 FOFs, lorsque la fréquence fondamentale est égale à 500 Hz, on déclenche 3000 FOFs. Il en résulte un son très riche, mais cela augmente la durée de calcul en conséquence.

 Exemple sonore : chant-seq06

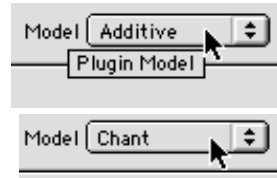


### 3.6. Utilisation de l'analyse additive pour des resynthèse avec Chant

En réduisant à l'extrême les largeurs de bandes d'un formant (0,001 Hz), on construit pratiquement un partiel unique. La série d'exemples sonores qui suit est une comparaison entre la synthèse additive et la synthèse FOF avec Chant.

 Exemple sonore : `java.additive`

L'exemple sonore « `java.additive` » a été produit par la resynthèse additive d'une séquence. Les mêmes données d'analyse peuvent également être utilisées pour définir des formants. Les données de fréquence des partiels deviennent alors les fréquences centrales des formants et les amplitudes des partiels deviennent les amplitudes des formants. Seules les largeurs de bande sont à « inventer ». Quand on travaille avec une séquence contenant des segments additifs, on voit dans la partie supérieure de la fenêtre le modèle « additive » sélectionné.



Si on bascule sur « Chant », les paramètres fréquences et amplitudes sont automatiquement utilisés pour les formants. « `java-as-chantfof` » est la synthèse par FOF avec Chant de la même séquence avec une largeur de bande très fine, égale à 1 Hz.

 Exemple sonore : `java-as-chantfof`

Pour le son « `java-as-chantfof-bw1000` », les largeurs de bande ont été portées jusqu'à 1000 Hz.

 Exemple sonore : `java-as-chantfof-bw1000`

Si on ne laisse pas la fréquence fondamentale suivre la fréquence originale, on ne conserve que les mouvements des formants, ce qui peut produire des sons intéressants. « `java-as-f0-500-bw10` » est la même séquence avec une largeur de bande de 10 Hz, mais la fréquence fondamentale a été choisie égale à 500 Hz et reste constante.

 Exemple sonore : `java-as-f0-500-bw10`

## 3.7. Filtres

Les définitions des formants peuvent contrôler autre chose que des FOFs, elles peuvent être utilisées pour des filtres. On peut filtrer soit du bruit blanc (à partir d'un générateur intégré) ou un fichier son externe.

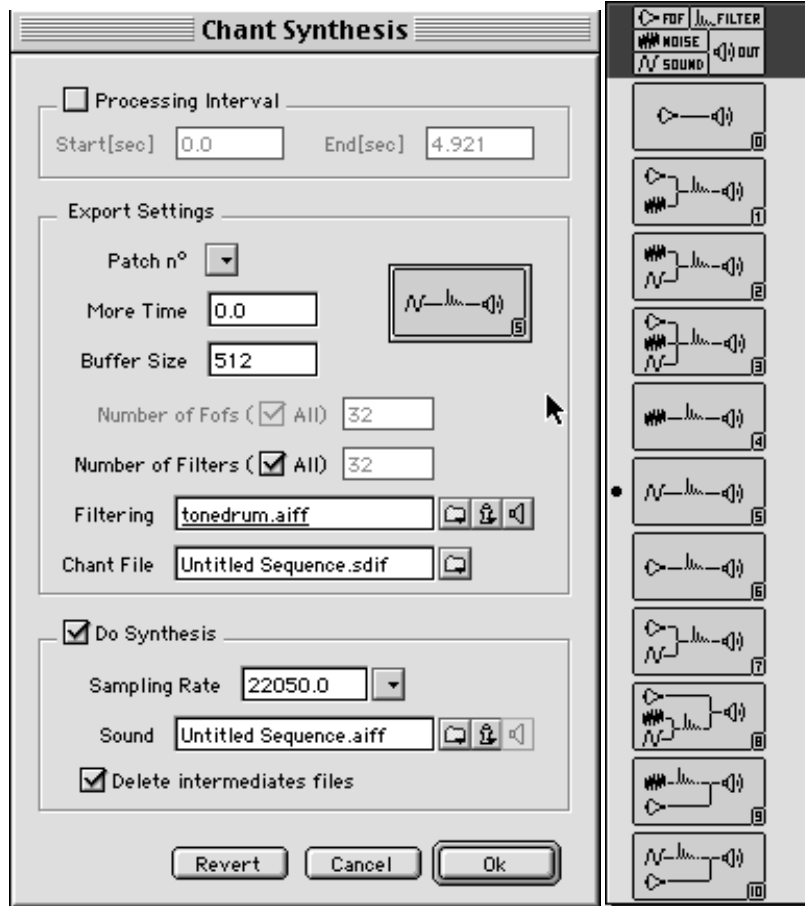
L'exemple « `java-as-filter-noise` » est le résultat du filtrage d'un bruit blanc avec les données formantiques de la séquence précédente. Le fichier-son « `tonedrum` » a été filtré avec les mêmes formants que dans l'exemple « `chant-seq06` ». Contrairement aux FOFs, les variations des paramètres des filtres sont continues ce qui signifie qu'on peut obtenir des glissandis de formants. Le résultat est « `tonedrum.filtered` ».

 Exemple sonore : `java-as-filter-noise`

 Exemple sonore : `tonedrum`

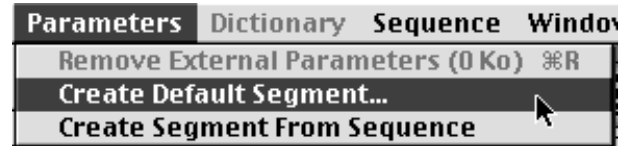
 Exemple sonore : `tonedrum.filtered`



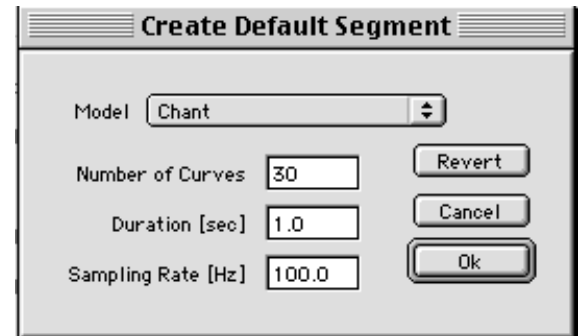


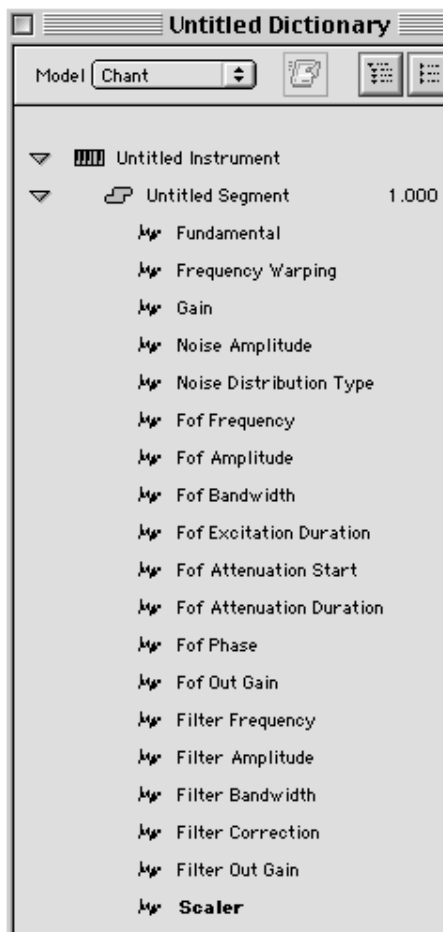
Pendant la resynthèse, on doit spécifier quel patch Chant va utiliser. Jusqu'à présent, nous n'avons utilisé que le premier patch, celui qui connecte directement la synthèse FOF aux « haut-parleurs ». Toutes sortes de combinaisons peuvent être réalisées entre les FOFs, le filtrage de bruits et le filtrage de sons.

Pour utiliser une séquence comme filtre formantique, le segment doit contenir tous les paramètres pour les filtres. Si ce n'est pas le cas, on peut générer ce qu'on appelle un segment par défaut qui contient tous les paramètres pour Chant.



Ce qui est important ici, c'est de sélectionner le modèle Chant pour le segment par défaut qui doit être créé.

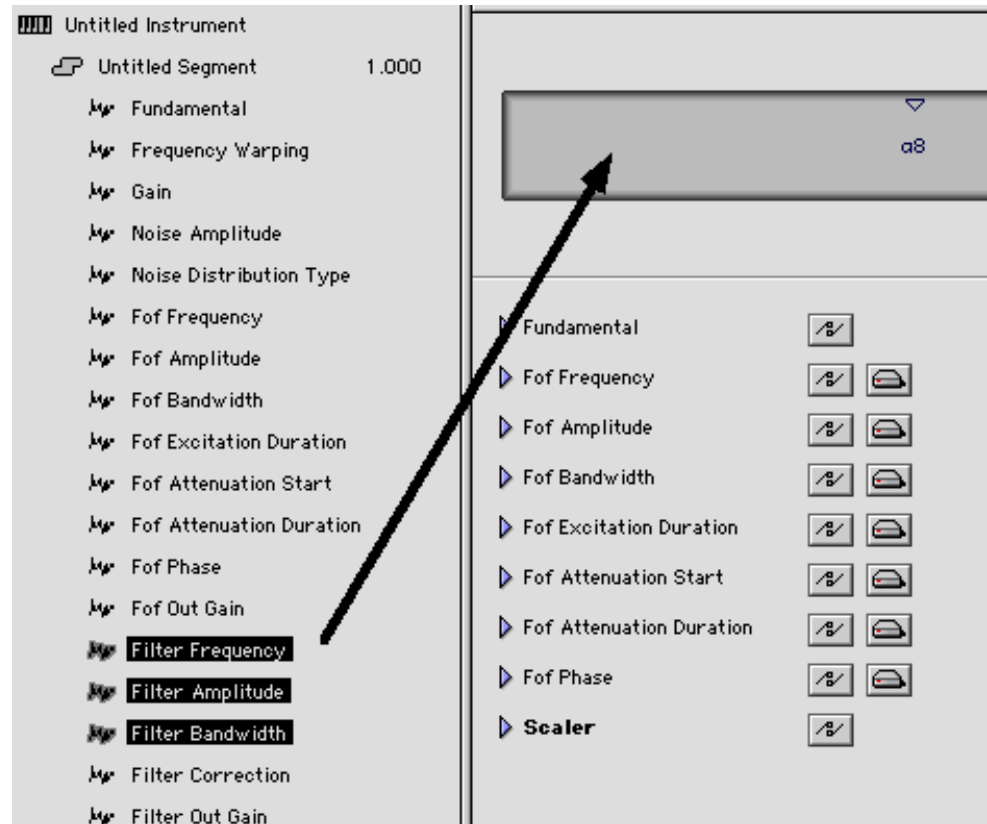




Dans le nouveau dictionnaire qui en résulte, on trouve un segment avec tous les paramètres pour Chant.

Ceux dont on a besoin peuvent alors être sélectionnés et déplacés vers le segment choisi. De cette façon, on peut créer des segments avec toutes les combinaisons possibles de paramètres.

Ceux qui vont être nécessaires dépendent du patch Chant sélectionné.



## 3.8. Conclusion

J'espère qu'avec ce texte en trois chapitres, j'ai pu donner un aperçu du travail qu'on peut réaliser avec Diphone Studio — un programme que je trouve très flexible et qui offre des possibilités de transformations sonores tout à fait uniques. C'est un programme en cours de développement et les versions (en interne) changent parfois d'une semaine sur l'autre. Cela dit, je pense qu'avec les possibilités d'analyses de résonances et additive maintenant disponibles et avec les possibilités d'édition dans Diphone, un grand éventail de moyens extraordinaires pour les transformations sonores est accessible.

Les développements actuels se concentrent sur l'intégration de nouveaux algorithmes de synthèse et de meilleures possibilités d'édition pour les multi-BPFs.

# Liste des exemples sonores

Exemple sonore : 01-floetenmelodie .....	12
Exemple sonore : 02-floetenmelodie.synth .....	17
Exemple sonore : 03-floetenmelodie.noise .....	17
Exemple sonore : 04-floetenmelodie.synth.30Hz .....	18
Exemple sonore : java-shaku01 .....	28
Exemple sonore : java-shaku02 .....	29
Exemple sonore : java-shaku03 .....	31
Exemple sonore : java-shaku04 .....	33
Exemple sonore : java-shaku05 .....	34
Exemple sonore : java-shaku06 .....	35
Exemple sonore : java-shaku07 .....	35
Exemple sonore : java-shaku08 .....	36
Exemple sonore : Original: crotale .....	49
Exemple sonore : Resynthesis: crotale.m1 .....	49
Exemple sonore : Resynthesis: crotale.m2 .....	49
Exemple sonore : Resynthesis: crotale.m6 .....	49
Exemple sonore : huhn .....	50
Exemple sonore : huhn.m1) .....	50
Exemple sonore : huhn.m2 .....	50
Exemple sonore : huhn.m5 .....	50
Exemple sonore : bulg-2 .....	50
Exemple sonore : bandwidth .....	54
Exemple sonore : chant-seq01 .....	59
Exemple sonore : chant-seq02 .....	60
Exemple sonore : chant-seq03 .....	61
Exemple sonore : chant-seq04 .....	61
Exemple sonore : chant-seq05 .....	62
Exemple sonore : chant-seq06 .....	63
Exemple sonore : java.additive .....	64

Exemple sonore : java-as-chantfof .....	64
Exemple sonore : java-as-chantfof-bw1000 .....	64
Exemple sonore : java-as-f0-500-bw10 .....	64
Exemple sonore : java-as-filter-noise .....	65
Exemple sonore : tonedrum .....	65
Exemple sonore : tonedrum.filtered .....	65