Ircam
documentation

- Research reports
- Musical works
- Software

PatchWork

SpData

première édition, mars 1997

IRCAM Centre Georges Pompidou

Ce manuel a été réalisé par Laurent Pottier, et produit sous la responsabilité éditoriale de Marc Battier, département de la Valorisation, Ircam.

PatchWork a été conçu et programmé par Mikael Laurson, Camilo Rueda et Jacques Duthen.

La librairie SpData a été écrite par Xavier Chabot et Laurent Pottier.

Cette documentation correspond à la version 1.0 de la librairie et à la version 2.0 ou ultérieure de PatchWork.

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

Groupe d'utilisateurs IRCAM

L'utilisation de ce programme et de sa documentation est strictement réservée aux membres des groupes d'utilisateurs de logiciels Ircam. Pour tout renseignement supplémentaire, contactez :

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

Tél. 01 44 78 49 62
Fax 01 44 78 15 40
Courrier électronique: bousac@ircam.fr

Veuillez faire parvenir tout commentaire ou suggestion à :

M. Battier
Département de la Valorisation
Ircam
1, Place Stravinsky
F-75004 Paris
France

bam@ircam.fr

http://www.ircam.fr/forumnet

Contenu

1 - Vue d'ensemble 6
Introduction 6
Structure des données 6
Fonctionnalités 12

2 - Référence 22
Interface fichiers 22
Création 28
Transformations 30
Spdata vers Max 43
Etude des données 47
Abstractions 55

3 - Patchs d'exemples 59
Entrées et sorties 59
Création 62
Modifications 64
Max 70
Etude des données 72
SpData vers Csound 77
Données pour AudioSculpt 81

4 - Annexes 82
Annexe 1 : Modes d'interpolations pour le module intpol-model 82
Annexe 2 : Instruments pour le programme Csound 83

5 - Index 87

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

1 Vue d'ensemble

1.1 Introduction

SpData est une librairie du programme PatchWork destinée à permettre la lecture, l'écriture, la création, l'édition, la transfor mation et l'étude de données numériques interprétées comme des analyses spectrales du son. Ces données peuvent provenir d'u programme comme AudioSculpt par exemple.

En raison de la taille considérable des fichiers contenant généralement des données spectrales (en particulier lors d'analyse dynamiques), PatchWork et la librairie SpData utilisent la « programmation par objet » pour la manipulation de ces donnée (voir ci-dessous).

Pour charger la librairie SpData dans PatchWork, il faut choisir le menu *Load-library* et charger le fichier « spdata.lib » présen dans le dossier *spdata.*

Lorsque le chargement est terminé (le symbole ? apparaît à nouveau dans le *Listener*), un menu **SpData** devient disponible dan le menu **User-lib**. Ce menu est divisé en cinq sous-menus.

```
- input/output
- Creation
- Processing
- Max
- Analyses
```

1.2 Structure des données

1.2.1 Les données spectrales

Les données fournies par l'analyse spectrale des sons peuvent être divisées en deux catégories :

- les analyses statiques
- les analyses dynamiques (variant dans le temps)

Dans le premier cas, à l'exception de l'analyse par Modèles de Résonance détaillée plus loin, les données ne contiennent pa d'information concernant le temps. Il s'agit typiquement de listes de valeurs qui sont en général des fréquences, des amplitude de partiels ou d'entités de type similaire et d'autres paramètres qui dépendent de l'analyse choisie.

Dans l'exemple suivant, les données contenues dans le fichier d'analyse (appelé « fichier source ») ont été produites par le pro gramme Audiosculpt et sont organisées par colonnes. La première colonne donne la liste de fréquences des partiels, le second donne les amplitudes et la dernière donne les poids perceptifs de ces partiels obtenus en appliquant les algorithmes de Terhardt.

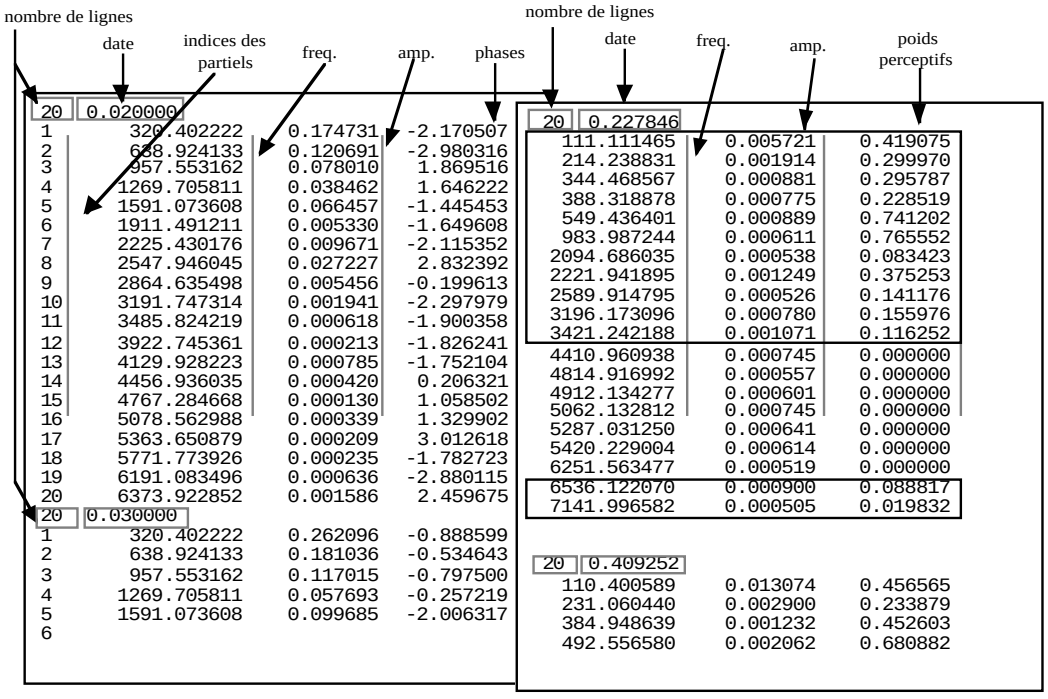
1. Terhardt 1987

frequence	amplitude	poids perceptifs
71.538673	0.002824	0.166112
308.724213	0.226836	0.870254
617.155518	0.155196	0.752622
924.076843	0.106006	0.663114
1230.385376	0.051629	0.462137
1537.142334	0.108144	0.624525
2457.916504	0.006937	0.001581
2618.274902	0.001335	0.000000
2653.756104	0.001014	0.000000
2765.187500	0.009307	0.000000
2791.680420	0.006669	0.000000
3071.094238	0.012088	0.085246
3122.013428	0.001701	0.000000
3378.831787	0.001820	0.000000
3683.535645	0.002130	0.000000
3991.434570	0.000955	0.000000
5829.855957	0.001146	0.232238
6524.287598	0.001302	0.155436
7276.832031	0.000648	0.000000
7306.743164	0.000839	0.000000

Dans le cas d'analyses dynamiques, les données sont une succession d'analyses statiques qui sont décalées dans le temps. Cha que analyse est précédée de sa date (la position dans le temps où l'analyse a été faite dans le son, généralement en secondes) e de sa longueur (le nombre de lignes ou le nombre de partiels présents dans l'analyse).

Deux types de fichiers d'analyses sont présentés dans l'exemple qui suit. A gauche on trouve une analyse réalisée avec la tech nique « Additive » de l'Ircam[1]. Sur la première ligne, on peut lire le nombre de lignes de la première analyse et sa date. Ving lignes plus bas, on peut lire le nombre de lignes de la deuxième analyse et sa date et ainsi de suite. Entre la deuxième et la vingt et-unième ligne, on trouve les informations concernant les partiels, organisées en quatre colonnes : rangs, fréquences, amplitu des et phases des partiels.

1. Garcia, Depalle, Rodet, 1994



Analyse « Additive » **Analyse « Mask »**

L'analyse par Modèles de Résonance[1] est un type d'analyse particulier puisqu'elle intègre des informations temporelles à l'in térieur d'une seule liste de données.

Le principe de modélisation de cette analyse consiste à identifier tous les modes de résonance d'un instrument. Ainsi, on peu décrire la fonction de transfert d'un filtre, ensemble de résonateurs élémentaires, auquel on assimile cet instrument.

L'analyse par Modèles de Résonance est basée sur le principe suivant. Sur une fenêtre extraite au début du son, on effectue un FFT suivie d'une extraction de pics. En comparant cette FFT avec une FFT réalisée sur une fenêtre décalée dans le temps, o peut déceler des résonances, raies spectrales présentes dans les deux analyses, et connaître leurs pentes d'atténuations.

L'opération est répétée avec des tailles de fenêtres croissantes de sorte à augmenter progressivement la résolution en fréquenc des analyses. Les premières analyses permettent ainsi de trouver les résonances à amortissement rapide dans le temps et le analyses suivantes offrent une gamme de résonances de plus en plus étendue, notamment vers les fréquences graves.

Toutes ces informations sont ensuite recoupées pour déterminer une liste de résonances, chacune caractérisée par sa fréquence son amplitude et sa largeur de bande (inversement proportionnelle à sa durée de résonance).

1. Barrière 1987

Cymbalum-C4

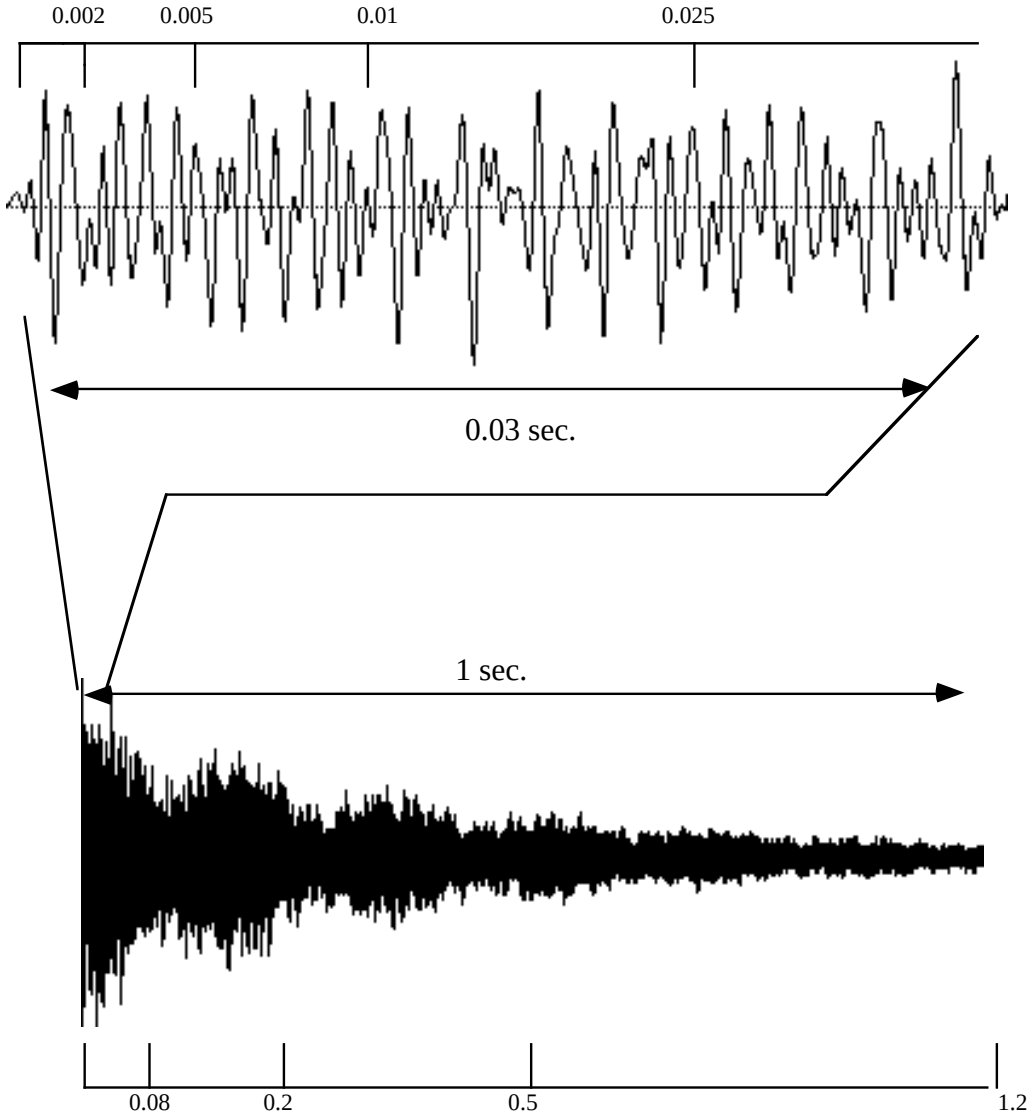
no	freq	ampl	band
1	50.28854	-73.64339	0.01406799
2	59.60196	-73.26854	0.05159099
3	99.00533	-71.45685	0.4662588
4	129.5101	-64.44109	0.1349887
5	135.2107	-64.2648	0.1618868
6	260.5955	-48.26771	0.1494008
7	354.4691	-27.24214	10.87039
8	516.4081	-22.3871	0.1204069
9	1035.153	-36.21174	0.2414675
10	1049.122	-39.79594	0.3320414
11	1557.622	-29.6514	0.2263101
12	1590.699	-22.85688	12.42003
13	2082.687	-30.17866	0.368507
14	2104.975	-36.66444	0.5959811
15	2618.508	-44.40742	0.2159851
16	2653.545	-20.32612	9.63131
17	2667.249	-25.5205	9.906801
18	3020.249	-28.70484	18.61749
19	3116.281	-47.05068	0.2293054
20	3153.267	-38.27651	0.3242503
21	3191.097	-46.84357	0.3104414
22	3596.71	-18.87824	26.1834
23	3667.165	-26.02918	8.63726
24	3709.823	-29.53598	0.46552
25	12638.56	-62.00786	14.75151
26	13089.39	-62.63412	10.11736
27	13118.93	-62.57565	13.45293

L'analyse s'effectue donc par étapes successives en conservant en mémoire les résultats obtenus d'une analyse sur l'autre. Su
les premières analyses, selon la richesse spectrale du son, on débute avec une dizaine de résonances, pour obtenir ensuite un
ou plusieurs centaines de partiels dans les dernières analyses.

Une réduction peut être appliquée sur les données de l'analyse. Elle porte sur des critères d'amplitude, de fréquence ou d'indic
de résonance.

Les données de l'analyse par Modèles de Résonance peuvent être utilisées directement par le synthétiseur Chant. La synthès
en FOF percussives permet de reconstruire le son et de valider ainsi l'analyse. Les paramètres peuvent être utilisés égalemen
pour des synthèses par filtrage, avec le synthétiseur Chant[1] ou d'autres systèmes (Csound[2], Station d'Informatique Musicale d
l'Ircam[3]).

1. 1979 X. Rodet
2. 1989 B. Vercoe
3. 1991 M. Puckette



Les différentes tailles de fenêtre (en msec.) utilisées pour le découpage du son pour une analyse en Modèle de Résonance : 0.002; 0.005; 0.01; 0.025; 0.08; 0.2; 0.5; 1.2

1.2.2 Programmation par objet

Compte tenu de la quantité de données contenue dans ces fichiers, leur traitement dans PatchWork sous forme de listes s'avèr impossible. Il est donc nécessaire de faire appel à la technique de programmation par objets.

Les données sont lues et stockées dans des emplacements de mémoires organisés selon une architecture « objet ». Une premièr classe d'objets, intitulée C-spdata, permet de gérer les analyses statiques, en stockant pour une analyse les fréquences dans u emplacement intitulé « freqs », les amplitudes dans un emplacement intitulé « amps » etc. Onze emplacements ont été conçu pour stocker les données en provenance de divers types d'analyses.

```
(size :initform 0 :initarg :size :accessor size)
(frame :initform 0 :initarg :frame :accessor frame)
(partials :initform () :initarg :partials :accessor partials)
(freqs :initform () :initarg :freqs :accessor freqs)
(amps :initform () :initarg :amps :accessor amps)
(normalized-amps :initform () :accessor normalized-amps)
(bws :initform () :initarg :bws :accessor bws)
(phases :initform () :initarg :phases :accessor phases)
(weights :initform () :initarg :weights :accessor weights)
(resfact :initform () :accessor resfact)
(patch :initform () :accessor patch) ; max patch modules; used by write-msgbox
```

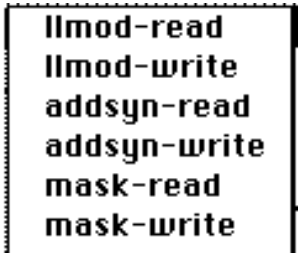
Une deuxième classe sert à mettre en ordre les données d'analyses dynamiques. Cette classe intitulée C-spdata-seq contient un liste d'objets C-spdata correspondant à la liste des différentes analyses statiques ainsi qu'un paramètre indiquant la durée du so analysé.

```
(duration  :initform 0 :initarg :duration :accessor duration)
(spdata  :initform () :initarg :spdata :accessor spdata)
```

Cette technique de programmation par objets permet de manipuler des fonctions plutôt que des valeurs numériques et ainsi d réduire les temps de calcul de façon considérable.

1.2.3 L'interface fichiers

Les fichiers qui contiennent les données doivent être des fichiers textes standards ou des fichiers de données binaires. Actuel lement, trois types de « données-type » peuvent être lues et écrites à l'aide de la librairie SpData: analyses Mask, Peak ou For mants réalisées à l'aide du programme AudioSculpt, analyse Additive de l'Ircam et analyse Modèles de Résonance.



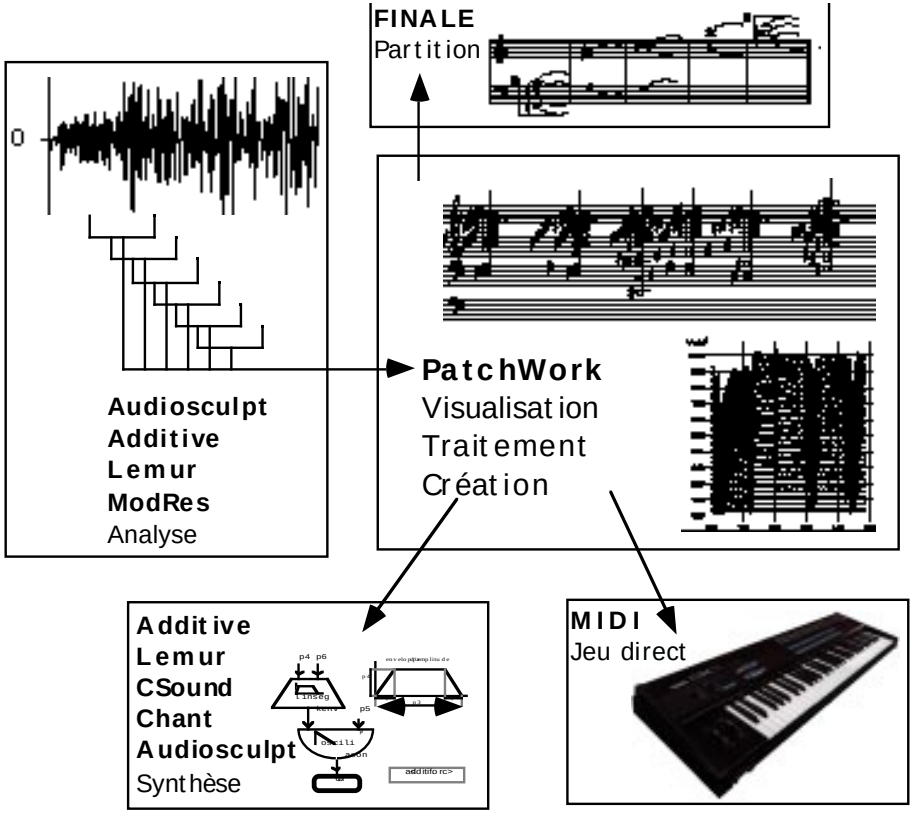
D'autres types de données d'analyses peuvent être lues avec les modules de la librairie SpData à condition qu'ils présentent de données agencées selon une structure compatible avec l'un de ces trois modèles.

Pour chacun de ces trois modèles, il existe des modules spécifiques permettant la lecture des fichiers d'analyse. Ces module produisent un objet C-spdata pour une analyse statique ou un objet C-spdata-seq pour une analyse dynamique.

De même, il est possible d'écrire un fichier au format de ces trois types d'analyses à partir d'objets C-spdata ou C-spdata-seq Ceci peut présenter un intéret lorsque ces techniques d'analyse sont couplées à un système de synthèse spécifique comme c'es le cas pour l'analyse Additive, l'analyse avec le programme Lemur ou l'analyse par Modèles de Résonance. Les données enre gistrées après traitement dans PatchWork peuvent alors donner lieu à la production de sons de synthèse.

En outre, des patchs sont disponibles dans la librairie SpData qui permettent de produire des fichiers de partition (scores) pou la synthèse de sons avec le programme Csound selon diverses techniques, à partir des données contenues dans les objets C spdata-seq. Pour ouvrir ces patchs, la librairie Csound/edit-sco[1] doit avoir été préalablement chargée dans PatchWork. Il es

également possible de traduire les données d'analyse en données Midi pour un jeu Midi en direct depuis PatchWork ou pour l
sauvegarde de fichiers Midi-File[1] pouvant être lus par tout programme compatible avec la norme Midi-File.



1.3 Fonctionnalités

1.3.1 Création de données

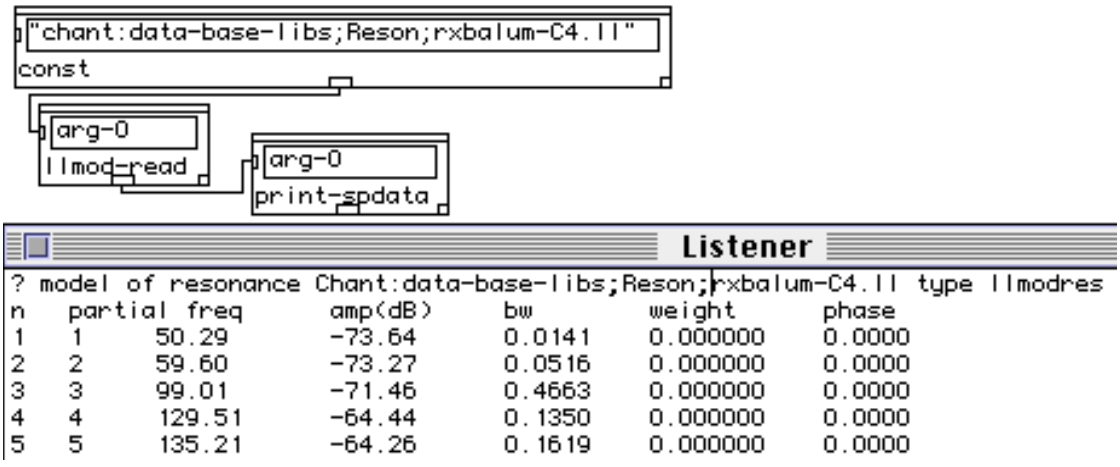
Le module **mk-spdata** permet de construire un objet C-spdata en fournissant des listes de valeurs numériques dans au moin
une de ses entrées. Il est possible de donner à une entrée soit une liste de valeurs, soit une valeur unique qui sera reproduite pou
tous les partiels, soit aucune valeur. Dans ce dernier cas, les partiels prendront les valeurs données par défaut pour le paramètr
considéré. Quand des listes sont données en entrées, elles doivent toutes avoir le même nombre d'éléments.

Le module **mk-spdata-seq** permet de créer un objet C-spdata-seq. Il suffit de lui fournir en entrée une liste de dates, exprimée
généralement en secondes, ainsi qu'une liste d'objets C-spdata de même longueur.

1. 1993 M. Malt & L. Pottier
1. 1993 L. Pottier

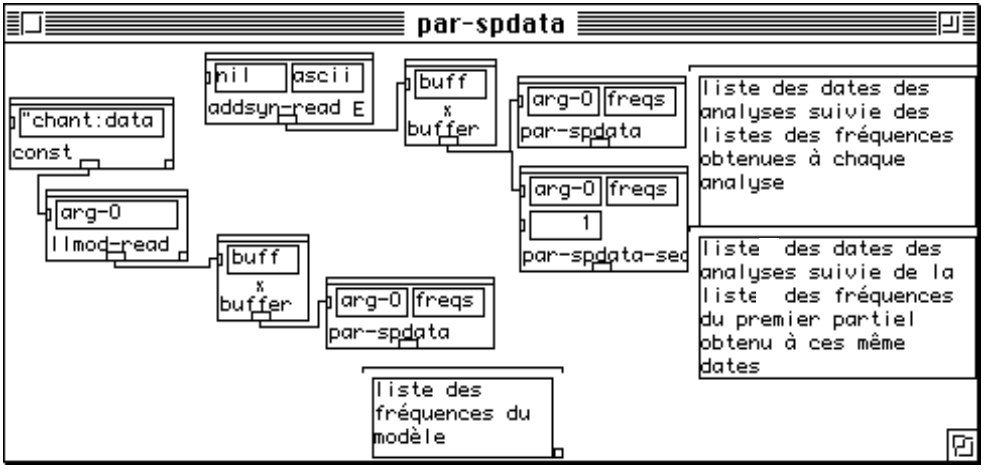
1.3.2 Visualisation des données

Pour visualiser les données contenues dans un objet C-spdata ou C-spdata-seq, il est possible d'utiliser le module **print-spdat** qui va imprimer dans le *Listener* le contenu de toutes les données qu'il contient.

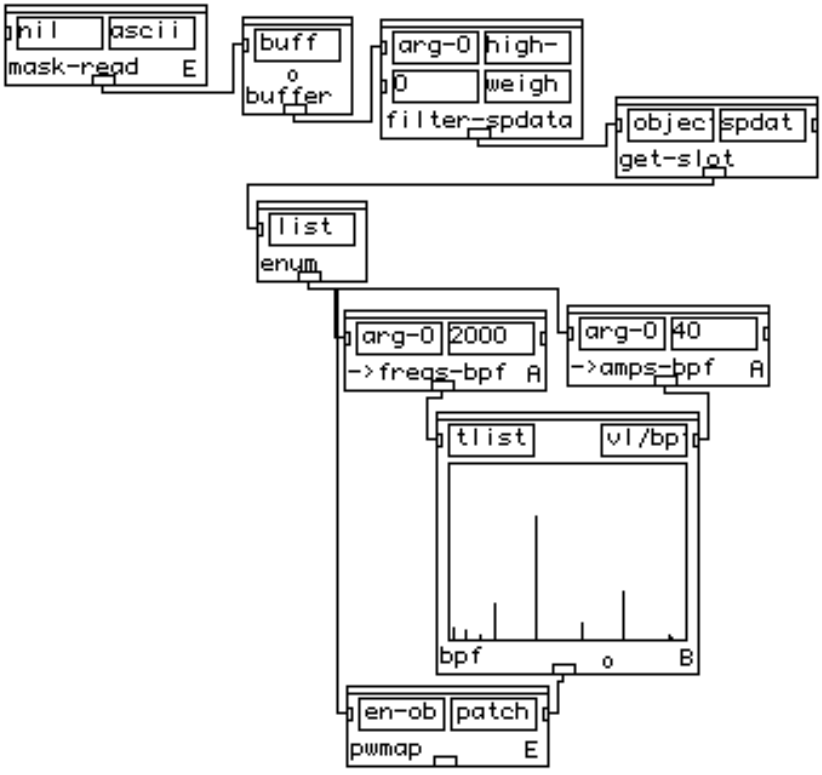


Le module **par-spdata** fournit les données correspondant à la fonction indiquée dans son entrée numéro deux lorsqu'il reçoi un objet C-spdata en entrée et il donne la iste des dates des analyses suivie des listes des données obtenues à chaque analys lorsqu'il reçoit un objet C-spdata-seq en entrée.

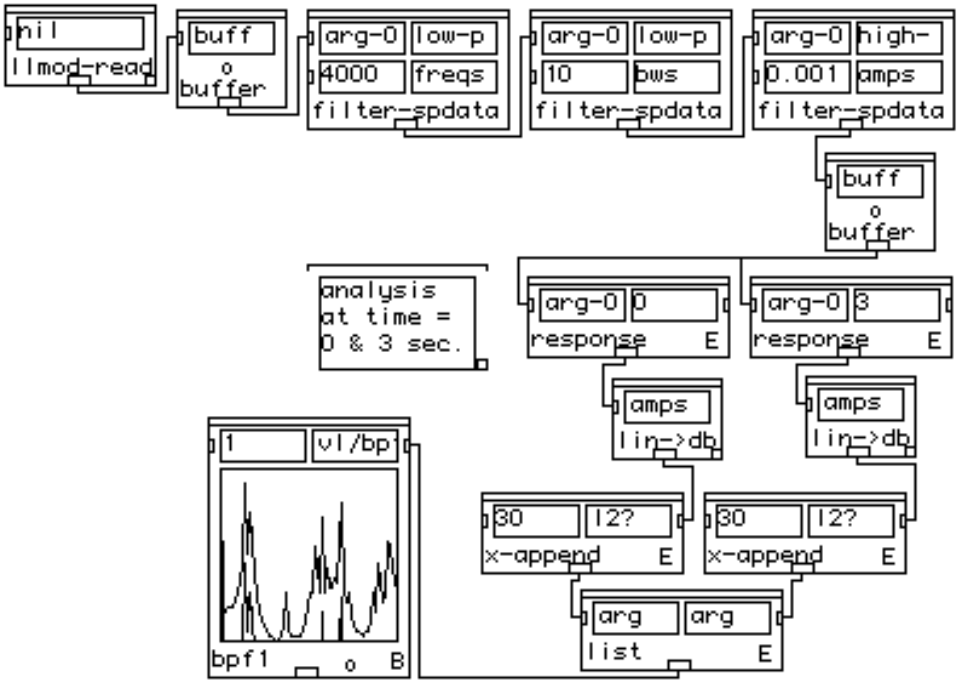
Le module **par-spdata-seq** fournit les données correspondant à la fonction indiquée dans l'entrée numéro deux et pour le partie indiqué par le numéro contenu dans l'entrée numéro trois.



Des abstractions compilées **->freqs-bpf** et **->amps-bpf** permettent de réaliser des graphes représentant les fréquences présente dans les données sous formes de pics.

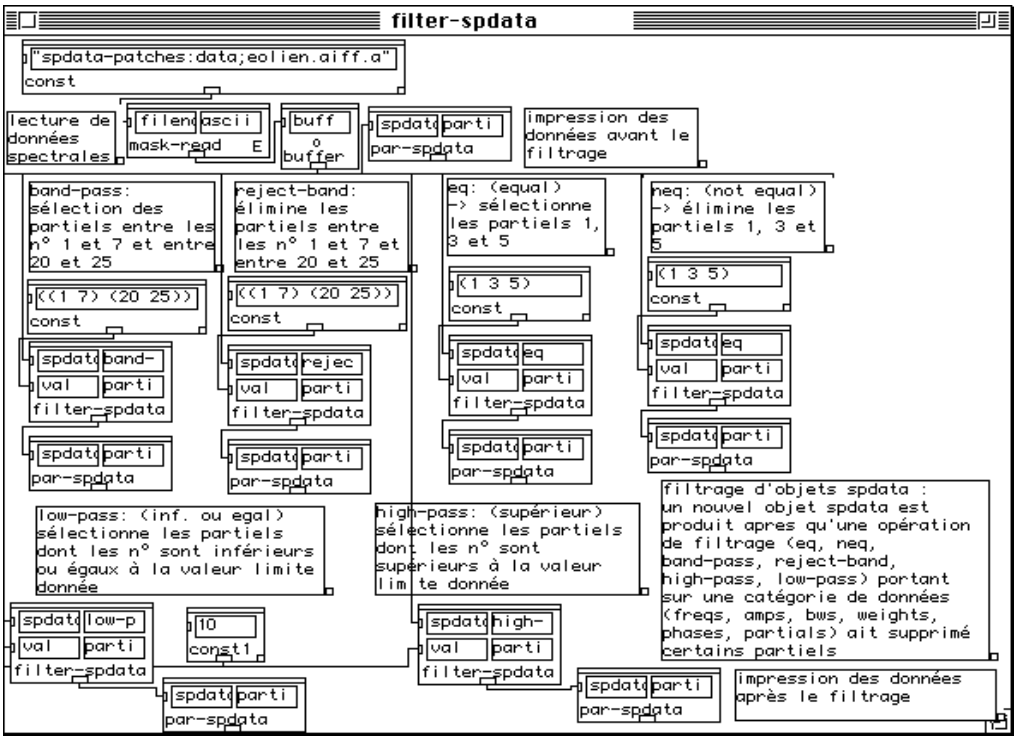


La fonction **response** permet de représenter l'enveloppe spectrale correspondant à des analyses formantiques.



1.3.3 Traitements des données

Le module **filter-spdata** permet de supprimer des partiels à l'intérieur d'un objet C-spdata ou d'objets C-spdata contenus dan un objet C-spdata-seq en appliquant une opération de filtrage (eq, neq, band-pass, reject-band, high-pass, low-pass) portant su une catégorie de données (freqs, amps, bws, weights, phases, partials). Le patch donné en exemple (taper **T** après avoir sélec tionné le module) indique les différentes possibilités offertes en effectuant un filtrage portant par exemple sur les numéros d partiels



Le module **scale-spdata** permet d'effectuer des changements d'échelle sur un paramètre soit en donnant un facteur d'échelle soit en fixant les bornes entre lesquelles elles doivent être réparties.

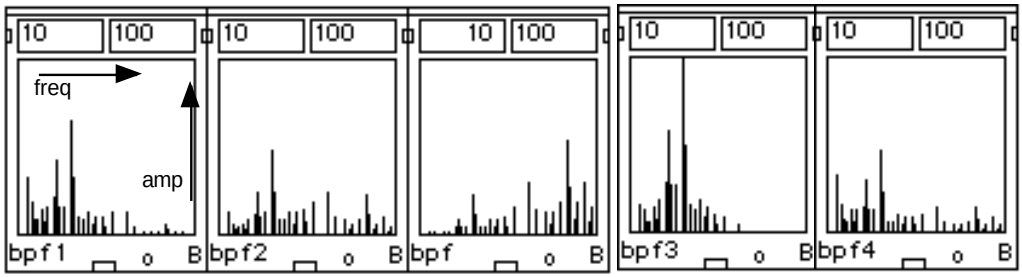
Le module **l-stat-modif** sert à déplacer de façon statistique les zones d'énergie dans le spectre. Il laisse les fréquences inchangées mais agit sur les amplitudes des partiels. Il agit en déplaçant le barycentre des fréquences des partiels.

- Pour des données Additives, le barycentre est égal à :

$$B = \frac{\sum freq_i \,.\, amp_i}{\sum amp_i}$$

- Pour des données Modèles de Résonance, le barycentre est égal à :

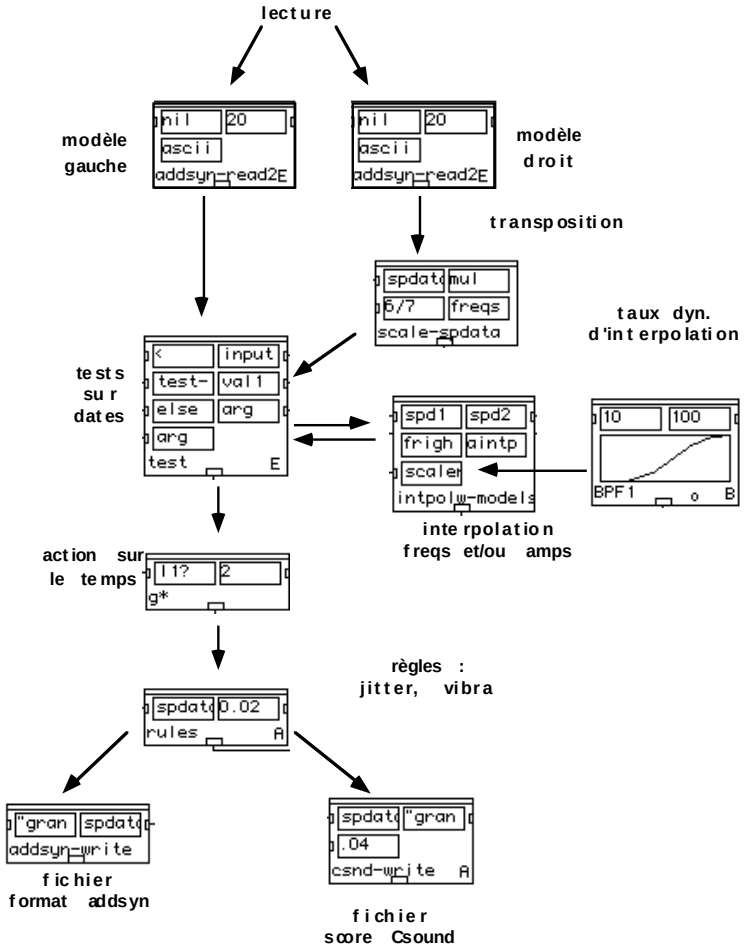
$$B = \frac{\sum \frac{freq_i \,.\, amp_i}{bw_i}}{\sum \frac{amp_i}{bw_i}}$$



original moyenne *1.5 moyenne *2.0 écart *0.6 écart *1.5

Les calculs peuvent être effectués en fréquences ou en midicents suivant l'option choisie dans la quatrième entrée du module.

Les modules **intpol-model** et **interpolw** permettent de créer divers types d'interpolations entre deux objets C-spdata. Le premier module est à appliquer principalement aux analyses par Modèles de Résonance et comporte plusieurs options (flags) détaillées en annexe de ce document. Le deuxième module correspond à des interpolations pour des données provenant d'analyses réalisées principalement avec Additive et permet d'interpoler les fréquences, les amplitudes ou ces deux paramètres en même temps. En combinaison avec un module **pw-map**, il est possible de réaliser des interpolations dynamiques entre deux modèles



Architecture du patch utilisé pour l'interpolation de modèles de sons additifs

1.3.4 L'interface avec Max

Deux fonctions permettent de fabriquer des sous-patchs contenant des données spectrales destinées au programme Max. Ce données vont être écrites sous forme de messages (module **write-msg**) ou sous forme de qlist (module **write-qlist**).

Pour trouver des fonctions plus générales concernant les échanges entre PatchWork et Max, il faut consulter la librairie « PW Max » élaborée par Xavier Chabot.

1.3.5 Etude des données

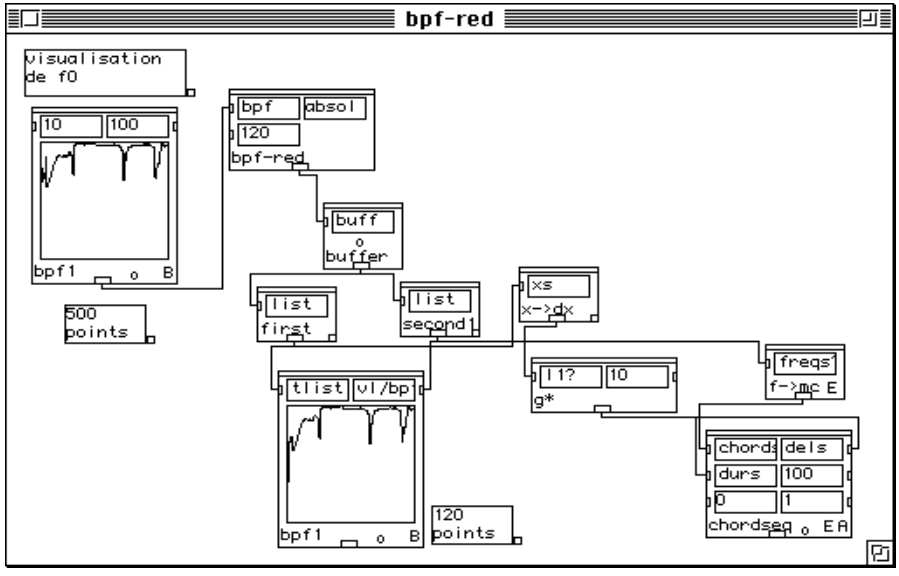
Quelques fonctions additionnelles permettent d'étudier les données d'analyse et d'en extraire quelques paramètres remarqua bles. La fonction **find-funds-datas** donne les séries harmoniques que l'on peut déceler dans une analyse, la fonction **resmoy coef** donne la résonance moyenne (en secondes) d'une analyse en Modèles de Résonance, les fonctions **l-moy** et **l-ecart** donnen respectivement le barycentre et la dispersion des fréquences des partiels des analyses et enfin la fonction **matrix-response** per met de connaître l'enveloppe spectrale d'une série de formants d'après leurs largeurs de bandes.

1.3.6 Liaisons avec Csound

Les données contenues dans les objets C-spdata et C-spdata-seq peuvent être des données d'analyses de sons ou des donnée obtenues par des processus algorithmiques. Dans les deux cas, il est possible de les utiliser pour réaliser des synthèses de sons en particulier avec le programme Csound.

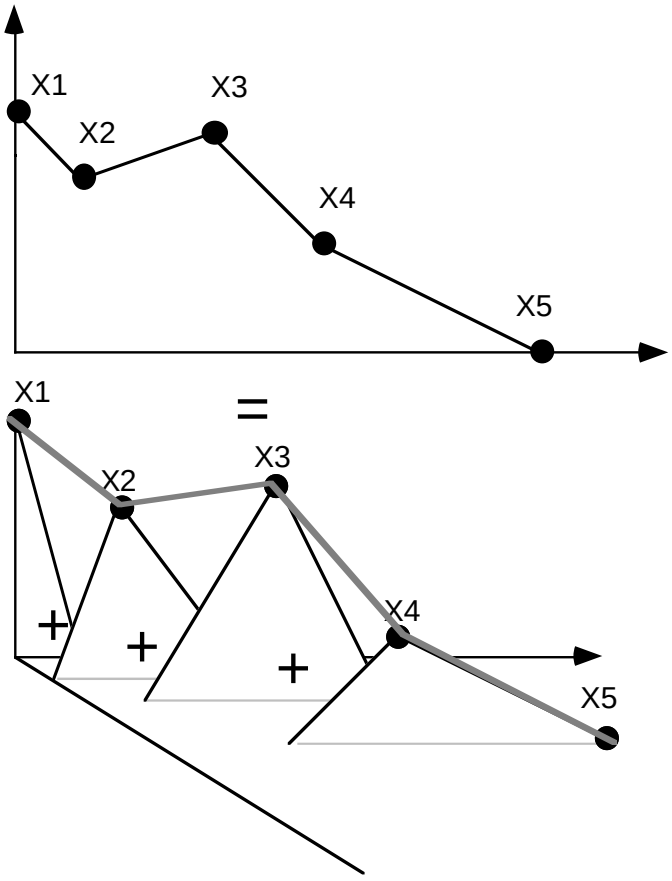
Typiquement, ces données peuvent être utilisées pour décrire des paramètres pilotant des oscillateurs, en synthèse additive, ou des filtres, en synthèse soustractive. Deux démarches sont proposées dans ce paragraphe.

La première consiste à construire des fonctions par segments donnant l'évolution des paramètres « fréquence » et « amplitude » en fonction du temps (voir le patch « Exemples:rd-fmt->csndtabs.pw »). Cette démarche permet de gérer dans Csound d'autres contrôles sur le son, comme des modulations, de façon indépendantes de ces enveloppes. Le programme Csound ne permet pas de créer des enveloppes de dimensions illimitées. Dans sa version standard, il est limité à une trentaine de points. Dans la librairie Utils de PW, automatiquement chargée avec la librairie SpData, on peut trouver un module, **bpf-red**, qui permet de réduire le nombre de points d'une BPF en tentant d'en modifier le moins possible l'allure.



Réduction du nombre de points contenus dans une BPF (de 500 à 120 points)

Le deuxième démarche utilisée est une démarche pointilliste, qui consiste à synthétiser séparément chaque analyse sous forme d'un « grain sonore » (voir patch « Exemples:m-add-rd3 »). Les analyses vont donc permettre la synthèse de grains qui vont se recouvrir pour produire un son continu.



Décomposition d'une courbe par segments en motifs élémentaires

L'instrument « granul-r.orc » qui est décrit en annexe permet de réaliser la synthèse des sons dont le score est produit par cett technique.

Dans le cas de la synthèse additive, en respectant les phases pour les enchaînements des grains, on peut reconstruire le son ori ginal avec une très bonne qualité.

Pour des analyses Mask ou Peak, pour lesquelles les analyses sont généralement beaucoup plus espacées et pour lesquelles le partiels ne correspondent pas aux même indices d'une analyse sur l'autre, deux cas peuvent se présenter. Lorsque les analyse sont très espacées dans le temps, il est possible de réaliser un son de synthèse de qualité sans avoir à se soucier des problème de phases (voir le patch « Exemples:mask-rd.pw ») en utilisant l'orchestre créé pour Csound « granul-j.orc » (cité en annexe) Lorsque les analyses sont rapprochées, il se produit des effets de modulation d'amplitude qui détériorent la qualité sonore d son synthétisé. Il est alors indispensable de procéder à une recherche de trajets de partiels présents dans l'analyse. Deux tech niques peuvent être envisagées: la plus simple consiste à diviser l'axe des fréquences en un certain nombre de zones et à ne relie des partiels que s'ils sont dans la même zone d'une analyse sur l'autre (voir patch « Exemples:rd-mask-creat-bpf). L'autre tech nique consiste à employer des fonctions utilisant des probabilités (chaîne de Markov par exemple) pour découvrir quels son les partiels qui ont le plus de chance de se succéder d'une analyse sur l'autre. Cette deuxième méthode mise au point par l'équip de X. Rodet n'est pas implémentée dans SpData.

Les recherches en cours pour le développement de la librairie SpData portent sur des fonctions destinées

- au calcul automatiques de trajets de partiels pour des analyses Mask ou Peak ;
- à l'étude et la séparation des phases stables et des phases transitoires pour ces même analyses ;
- à la transfomation de l'analyse de la fondamentale en données Midi ;
- au portage des règles du synthétiseur Chant dans la librairie SpData ;
- à l'intégration de la lecture et l'écriture de données d'analyse réalisées avec le programme Lemur sur Macintosh.

2 Référence

2.1 Interface fichiers

2.1.1 llmod-read

Syntaxe

(spdata::llmod-read *filename*)

Entrées

filename chaîne de caractères

Sortie

spdata

Lit les données contenues dans le fichier *filename* représentant des paramètres formantiques ou de résonances et produit un ob jet C-spdata de type modèle de résonance.

Les données doivent être contenues dans un fichier de type texte (généralement pourvu de l'extension « .ll ») au format produi par le langage LISP: les fréquences, les amplitudes et les largeurs de bande sont stockées dans trois listes qui sont, lorsqu'elle lues, sont interprétées comme des variables globales. Ces listes doivent être de même tailles :

```
(setq l-freqf '( n n n n n))
(setq l-amplf '(n n n n n n))
(setq l-bandf '(n n n n n))
```

2.1.2 llmod-write

Syntaxe

(spdata::llmod-write *filename spdata*)

Entrées

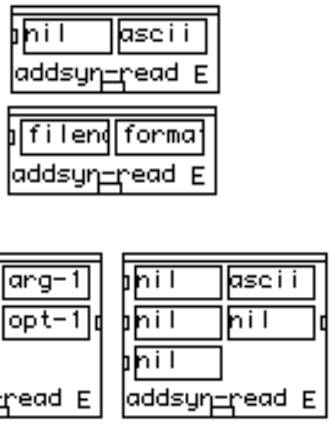
filename	chaîne de caractères
spdata	object C-spdata

Sortie

nil

Ecrit un fichier « texte » au format Modèles de Résonance à partir d'un objet C-spdata.

2.1.3 addsyn-read



Syntaxe

(spdata::addsyn-read *filename format &optional beg end npart*)

Entrées

filename chaîne de caractères
format menu (ascii ou bin))))
optionnel :
beg nombre entier ou flottant
end nombre entier ou flottant
npart nombre entier

Sortie

spdata-seq

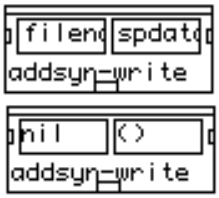
Lit un fichier contenant des données provenant de l'analyse Additive (généralement pourvu de l'extension « .format ») et produi un objet C-spdata-seq. Les données doivent être agencées en quatre colonnes (indices, fréquences, amplitudes et phases) et pré cédées de deux nombres indiquant le nombre de partiels et la date de l'analyse courante.

```
3  0
1          440                  0.1                0.000
2          882                  0.02               0.000
3         1321                  0.01               0.000

4  0.1
1          440                  0.1                0.001
2          881                  0.02               0.002
3         1317                  0.01               0.001
4         1755                  0.006              0.020
```

Le nombre *npart* indique le nombre maximum de partiels que l'on veut lire dans chaque analyse.

2.1.4 addsyn-write



Syntaxe

(spdata::addsyn-write *filename spdata)*

Entrées

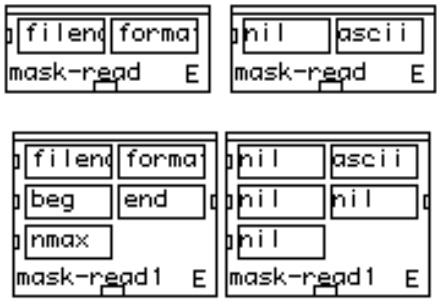
filename	chaîne de caractères
spdata	object C-spdata-seq

Sortie

nil

Ecrit dans un fichier « texte » les données au format de l'analyse Additive à partir d'un objet C-spdata-seq.

2.1.5 mask-read



Syntaxe

(spdata::mask-read *filename format &optional beg end nmax*)

Entrées

filename	chaîne de caractères
format	menu (ascii ou bin))))
optionnel:	
beg	nombre entier ou flottant
end	nombre entier ou flottant
nmax	nombre entier

Sortie

spdata-seq

Lit un fichier contenant des données provenant de l'analyse Mask (réalisée avec le programme AudioSculpt) et produit un obje
C-spdata-seq. Les données doivent être agencées en trois colonnes (fréquences, amplitudes et poids perceptifs) et précédées d
deux nombres indiquant le nombre de partiels et la date de l'analyse courante.

```
3  0
440             0.10.525
625             0.020.000
1261            0.010.000

4  0.1
380             0.030.193
440             0.10.525
625             0.020.000
1261            0.010.000
```

2.1.6 mask-write

Syntaxe

(spdata::mask-write *filename spdata*)

Entrées

filename	chaîne de caractères
spdata	object C-spdata-seq

Sortie

nil

Ecrit dans un fichier « texte » les données au format de l'analyse Mask à partir d'un objet C-spdata-seq.

2.2 Création

2.2.1 mk-spdata

Syntaxe

(spdata::mk-spdata *freqs amps phases partials weights bws)*

Entrées

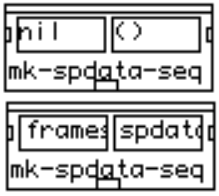
freqs	liste ou nombre entier ou flottant
amps	liste ou nombre entier ou flottant
phases	liste ou nombre entier ou flottant
partials	liste ou nombre entier ou flottant
weights	liste ou nombre entier ou flottant (init. 60)
bws	liste ou nombre entier ou flottant (init. 1)

Sortie

spdata

Crée un objet C-spdata ou une liste d'objets C-spdata. Les entrées doivent être des listes simples si l'on veut créer un objet C spdata ou des listes de listes pour créer une série d'objets C-spdata. Au moins une des entrées doit comporter une liste. Les autre entrées peuvent ne contenir qu'un nombre qui sera alors répété pour tous les partiels de l'objet C-spdata.

2.2.2 mk-spdata-seq



Syntaxe

(spdata:mk-spdata-seq*frames spdata)*

Entrées

frames	liste de nombres
spdata	liste d'objets C-spdata

Sortie

spdata-seq

Crée un objet « C-spdata-seq à partir d'une liste d'objets C-spdata et d'une liste de frames (en secondes).

2.3 Transformations

2.3.1 print-spdata

Syntaxe

(spdata::print-spdata *spdata*)

Entrées

spdata objet C-spdata

Sortie

nil

Imprime le contenu d'un objet C-spdata en colonnes dans le Listener.

2.3.2 par-spdata

Syntaxe

(spdata:par-spdata *spdata slot*)

Entrées

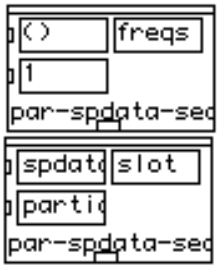
spdata	objet C-spdata ou C-spdata-seq
slot	menu (freqs amps bws partials weights phases size)

Sortie

liste

Donne les données de l'objet C-spdata correspondant à la fonction indiquée. Si l'entrée est un objet C-spdata-seq, le résustat es la liste des frames suivie des listes des données correspondant à la fonction indiquée.

2.3.3 par-spdata-seq



Syntaxe

(spdata::par-spdata-seq *spdata-seq slot partials*)

Entrées

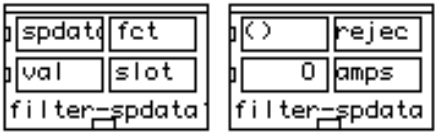
spdata	objet C-spdata ou C-spdata-seq
slot	menu (freqs amps bws partials weights phases size)
partials	liste ou nombre entier

Sortie

liste

Donne les données de l'objet C-spdata-seq correspondant à la fonction indiquée (*slot*) et au partiel ou à la liste des partiels in diqués (*partials*). Le format est le suivant : ((dates ..) (valeurs..)) pour un partiel et (((dates ..) (valeurs..)) ((dates ..) (valeurs..) ...) pour une liste de partiels.

2.3.4 filter-spdata



Syntaxe

(spdata::spdata:filter-spdata *spdata fct val slot*)

Entrées

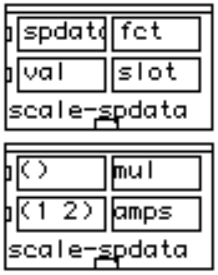
spdata	objet C-spdata ou C-spdata-seq
fct	menu (band-pass low-pass high-pass reject-band eq neq)
val	liste ou nombre entier ou flottant
slot	menu (freqs amps bws partials weights phases size)

Sortie

objet C-spdata ou C-spdata-seq

Crée un nouvel objet spdata ne comportant plus que les partiels satisfaisant au test.

2.3.5 scale-spdata



Syntaxe

(spdata::scale-spdata *spdata fct val slot)*

Entrées

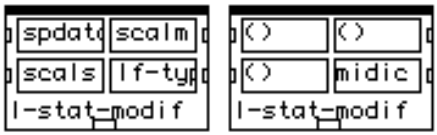
spdata	objet C-spdata ou C-spdata-seq
fct	menu (mul max min-max)
val	liste ou nombre entier ou flottant
slot	menu (freqs amps bws partials weights phases size)

Sortie

objet C-spdata ou C-spdata-seq

Crée un nouvel objet spdata dont les valeurs des fonctions sélectionnées ont été remises à l'échelle.

2.3.6 l-stat-modif



Syntaxe

(spdata::l-stat-modif *spdata scalm scals lf-typ*)

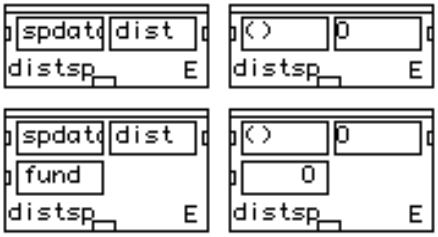
Entrées

spdata	objet C-spdata ou C-spdata-seq
scalm	nombre flottant
scals	nombre flottant
lf-typ	menu (midic hz)

Sortie

Calcule la moyenne et la dispersion de l'objet C-spdata puis effectue une transformation « gaussienne » en multipliant l moyenne par *scalm* et la dispersion par *scals*. Si *lf-typ* est en Hz, tous les calculs (moyenne écart-type) se font en Hz, sinon e midicents.

2.3.7 distsp



Syntaxe

(spdata::distsp *spdata dist &optional fund*)

Entrées

spdata	objet C-spdata ou C-spdata-seq
dist	nombre flottant
fund	nombre entier ou flottant

Sortie

objet C-spdata ou C-spdata-seq

Modifie les fréquences de l'objet C-spdata en appliquant à chaque harmonique un coefft de distorsion.

Si *dist* est égal à 1, il n'y a pas de distorsion.

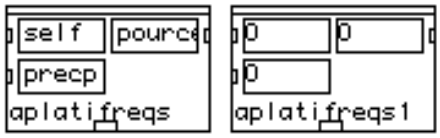
Si l'entrée *fund* est sélectionnée et différente de nil, **distsp** calcule des harmoniques avec une distorsion sans relation avec le fréquences des partiels mais calculée par rapport à cette fondamentale selon les indices des partiel.

$f_n = fund * n^{dist}$

$f_n = (f_{ni} / n) * n^{dist}$

avec fn fréquence du nième partiel
fni fréquence initiale du nième partiel
n indice partiel

2.3.8 aplatifreqs



Syntaxe

(spdata::aplatifreqs *spdata pourcent precp*)

Entrées

spdata	objet C-spdata-seq
pourcent	nombre flottant
precp	nombre entier

Sortie

objet C-spdata-seq

Cette fonction diminue les écarts de fréquences de chaque partiel en fonction des frames en les rapprochant d'un centre (le mod des fréquences). *Pourcent* est le taux de rapprochement (entre 0 et 100 en général, si *pourcent* = 0 toutes les fréquences resten égales). *Prec* est la précision utilisée pour trouver le centre (nombre de décimales pour calculer le mode, par exemple 0).

Exemple

```
;(aplatifreqs2 '(1 5 2 4 5 1 2 5 4 4 ) 10.0 0)
--> (4.6 5.0 4.7 4.9 5.0 4.6 4.7 5.0 4.9 4.9)
```

2.3.9 xtrajts-chds

Syntaxe

(spdata::xtrajts-chds *Lchds seuil*)

Entrées

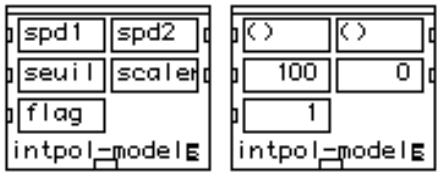
Lchds	Liste d'objet « C-chords »
seuil	nombre en midicents

Sortie

Liste

xtrajts-chds construit des listes de listes. Chaque liste est une liste double et correspond à l'évolution d'un partiel donné suiv au cours du temps : liste de hauteurs (en midics) et liste d'amplitudes associées (velocités). *Seuil* donne le glissé maximum (e midic) autorisé pour une hauteur, entre deux analyses consécutives.

2.3.10 intpol-models



Syntaxe

(spdata::intpol-models *spd1 spd2 seuil scaler flag &optional th-type mod-type*)

Entrées

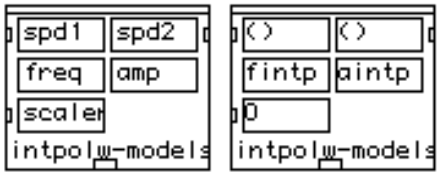
spd1	objet C-spdata ou C-spdata-seq
spd2	objet C-spdata ou C-spdata-seq
seuil	nombre entier
scaler	nombre flottant
flag	nombre entier entre 0 et 7
th-type	menu (midic Hz)
mod-type	menu (midic Hz)

Sortie

Liste

Donne un nouveau modèle interpolé à partir de deux modèles de départ selon la méthode d'interpolation des Modèles de Réso nance (voir en Annexe 1). Les frames du modèle résultant sont calculées selon celles du modèle gauche.

2.3.11 intpolw-models



Syntaxe

(spdata::intpolw-models *spd1 spd2 freq amp scaler*)

Entrées

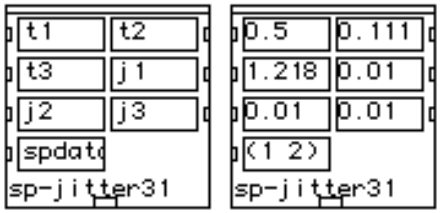
spd1	objet C-spdata ou C-spdata-seq
spd2	objet C-spdata ou C-spdata-seq
freq	menu (fintpol fleft fright)
amp	menu (aintpol aleft aright)
scaler	nombre flottant

Sortie

objet C-spdata ou C-spdata-seq

Donne un nouveau modèle de type « Analyse Additive » interpolé à partir de deux modèles de ce type également. Il est possibl
d'interpoler les amplitudes (*amp* = aintpol), les fréquences (*freq* = fintpol), ou d'échanger les fréquences et les amplitudes (*fre*
= fleft et *amp* = aright par ex.) des deux modèles. Les frames du modèle résultant sont calculées selon celles du modèle gauche

2.3.12 sp-jitter3



Syntaxe

(spdata::sp-jitter3 *t1 t2 t3 j1 j2 j3 spdata-seq*)

Entrées

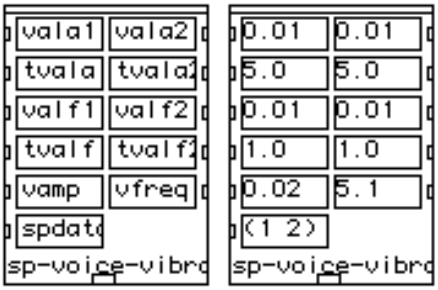
t1	nombre flottant
t2	nombre flottant
t3	nombre flottant
j1	nombre flottant
j2	nombre flottant
j3	nombre flottant
spdata-seq	objet C-spdata-seq

Sortie

objet C-spdata-seq

Crée de multiples variations aléatoires sur les fréquences des partiels. Les fréquences de ces variations sont égales à 1/t1, 1/t2 et 1/t3 et leurs amplitudes valent j1, j2, j3 en pourcentage de la fréquence du partiel.

2.3.13 sp-voice-vibra



Syntaxe

(spdata::sp-voice-vibra *vala1 vala2 tvala1 tvala2 valf1 valf2 tvalf1 tvalf2 vamp vfreq spdata-seq*)

Entrées

vala1	nombre flottant
vala2	nombre flottant
tvala1	nombre flottant
tvala2	nombre flottant
valf1	nombre flottant
valf2	nombre flottant
tvalf1	nombre flottant
tvalf2	nombre flottant
vamp	nombre flottant
vfreq	nombre flottant
spdata-seq	Liste

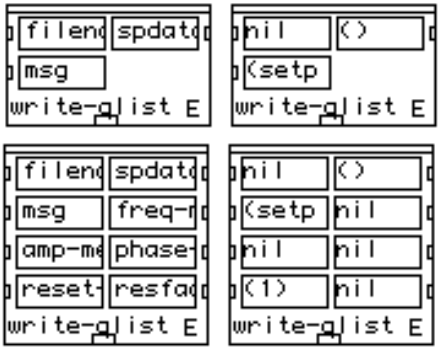
Sortie

Liste

Crée sur les fréquences des partiels des variations sinusoïdales périodiques doublées de microvariations aléatoires, de fréquen ces respectives *vfreq*, 1/*tala1*, 1/*tala2*, 1/*tala3* d'amplitude respectives *vamp vala1*, *vala2*, *vala3* .

2.4 Spdata vers Max

2.4.1 write-qlist



Syntaxe

(spdata::write-qlist *filename spdata msg &optional freq-method amp-method phase-method reset-flag resfact-flag*)

Entrées

filename	chaîne de caractères
spdata	objet C-spdata-seq
msg	chaîne de caractères
freq-method	liste
amp-method	liste
phase-method	liste
reset-flag	liste
resfact-flag	liste

Sortie

objet C-spdata-seq

Ecrit une qlist pour le programme Max au format indiqué par *msg*.

msg doit être de la forme :

'(<msg> {n}{<slot-name}*)

<msg> est le nom d'une variable globale contenu dans la qlist qui est envoyée avec les données Spdata. Une ligne dans la qlis est du type :

msg slotvalue slotvalue ...

Par exemple, le message '(setpartials partials freqs) produit les lignes suivantes :

setpartials 1 100.0 ;setpartials 2 120.0; etc...

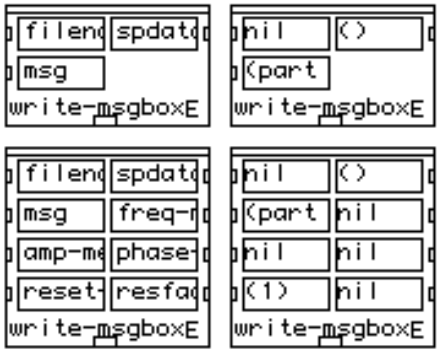
Si n est indiqué après <msg>, un indice sera accolé au nom du message:

'(setpartial n freqs amps) produit les lignes suivantes :

`setpartial1 100.0 0.1;setpartial2 120.0 0.8;` etc....

Quand *reset flag* est sur true (valeur par défaut), les valeurs pour réinitialiser le spectre sont ajoutées avec la valeur de temps 1 (time tag).Quand une qlist est utilisée avec le module nqlist-del, les messages <rewind, next> vont déclencher le spectre e l'arrêter au temps -1 (time tag). Pour réinitialiser les données, il suffit d'envoyer le message <next> à la qlist.

2.4.2 write-msg-box



Syntaxe

(spdata::write-msg-box *filename spdata msg &optional freq-method amp-method phase-method reset-flag resfact-flag*)

Entrées

filename	chaîne de caractères
spdata	objet C-spdata-seq
msg	chaîne de caractères
freq-method	liste
amp-method	liste
phase-method	liste
reset-flag	liste
resfact-flag	liste

Sortie

objet C-spdata-seq

Ecrit un sub-patch pour Max contenant les données spectrales dans des messages.

msg doit être sous la forme :

'(<msg> {n}{<slot-name}*)

<msg> est le nom d'une variable globale contenu dans le message qui est envoyée avec les données Spdata. Une ligne dans le message est du type :

msg slotvalue slotvalue ...;

Par exemple, le message '(setpartials partials freqs) produit les lignes suivantes :

setpartials 1 100.0 ;setpartials 2 120.0; etc...

Si n est indiqué après <msg>, un indice sera accolé au nom du message:

'(setpartial n freqs amps) produit les lignes suivantes :

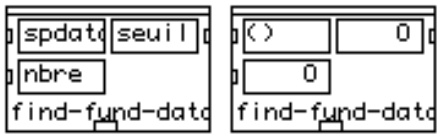
setpartial1 100.0 0.1;setpartial2 120.0 0.8; etc....

Pour ce module, l'option *reset-flag* n'est pas utilisée. Deux boites de messages la remplacent :

la première attribue les données et la seconde réinitialise les amplitudes à zéro.
Les données sont envoyées par la sortie du module en tant que messages.

2.5 Etude des données

2.5.1 find-fund-data



Syntaxe

(spdata::find-fund-data *spdata seuil nombre*)

Entrées

spdata	objet C-spdata
seuil	objet C-spdata-seq
nombre	objet C-spdata-seq

Sortie

Liste

Fonction qui recherche les séries de partiels qui sont harmoniques à travers l'ensemble des fréquences de l'objet C-spdata. Une série est trouvée si elle comporte au moins *nombre* fréquences multiples. Une fréquence est considérée comme multiple d'une autre si elle est égale à :

fn = n*f0 +- seuil.

Chaque série harmonique est donnée dans une liste avec un poids relatif (somme des amplitudes de ses partiels).

2.5.2 get-py/px

Syntaxe

(spdata::get-py/px *Ldevals valsref datas*)

Entrées

Ldevals	Liste
valsref	Liste
datas	Liste

Sortie

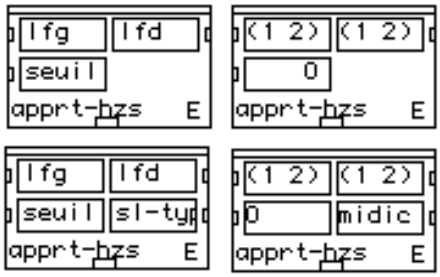
Liste

Cherche les valeurs de *Ldevals* dans *valsref* et donne les valeurs aux mêmes positions dans *datas*.

On suppose que la liste*valsref* et la liste *datas* ont la même structure et la même taille.

On suppose que toutes les valeurs de *Ldevals* existent dans la liste *valsref*

2.5.3 apprt-Hzs



Syntaxe

(spdata::apprt-Hzs *Lfg Lfd seuil &optional sl-type*)

Entrées

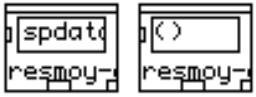
Lfg	Liste
Lfd	Liste
seuil	Liste
sl-type	menu (midic Hz)

Sortie

Liste

Fournit une liste de paires de fréquences après intersection des deux listes de fréquences données en entrée. Les valeurs de fré quences entre la liste gauche et la liste droite qui sont proches sont appariées (pour des distances inférieures à *seuil*).

2.5.4 resmoy-coef



Syntaxe

(spdata::resmoy-coef *spdata*)

Entrées

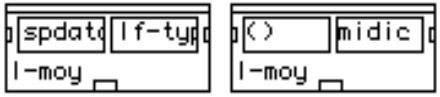
spdata objet C-spdata

Sortie

nombre flottant

Calcule pour un objet C-spdata correspondant à un modèle de résonance, son coefficient de durée de résonance (approximativement en secondes)

2.5.5 l-moy



Syntaxe

(spdata::l-moy *spdata lf-type*)

Entrées

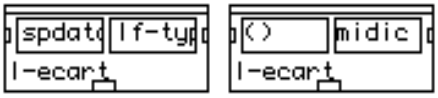
spdata	objet C-spdata
lf-type	menu (midic Hz)

Sortie

nombre flottant

Calcule la fréquence moyenne d'un objet C-spdata. Les largeurs de bande pondèrent les amplitudes (en 1/bw). Si « midic » es sélectionné comme type (*sl-type*), le résultat est en midic et le calcul est fait sur les hauteurs en midics des partiels

2.5.6 l-ecart



Syntaxe

(spdata::l-ecart *spdata lf-type*)

Entrées

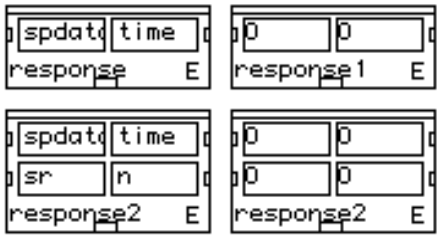
spdata	objet C-spdata
lf-type	menu (midic Hz)

Sortie

nombre flottant

Calcule la dispersion des fréquences d'un objet C-spdata. Les largeurs de bande pondèrent les amplitudes (en 1/bw). S « midic » est sélectionné comme type (*sl-type*), le résultat est en midic et le calcul est fait sur les hauteurs en midics des partiels

2.5.7 response



Syntaxe

(spdata::get-py/px *time &optional sr n*)

Entrées

spdata	objet C-spdata
time	nombre entier ou flottant
sr	nombre entier
n	nombre entier

Sortie

Liste

Calcule l'enveloppe spectrale d'un objet C-spdata dont les partiels sont des résonances correspondant à des filtres du second ordre. Le résultat est une liste d'amplitudes correspondant aux fréquences fi :

fi=(i/N)*(SR/2)

2.5.8 matrixresponse

Syntaxe

(spdata::matrixresponse *spdata dur filename &optional SR N M amp*)

Entrées

spdata	objet C-spdata
dur	nombre entier ou flottant
filename	chaîne de caractères
sr	nombre entier
n	nombre entier
m	nombre entier
amp	menu (lin db)

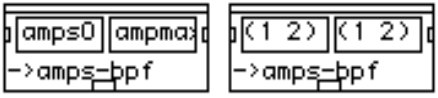
Sortie

Liste

Calcule une série d'enveloppes spectrales correspondant à l'objet C-spdata qui doit être un Modèle de Résonance. Ces envelop pes spectrales donnent l'atténuation des résonances en fonction du temps (sur la durée *dur*). Ces données sont sauvegardées dan un fichier *filename* pour être lues, par exemple, par un tableur.

2.6 Abstractions

2.6.1 ->amps-bpf



Syntaxe

(->amps-bpf *spdata ampmax*)

Entrées

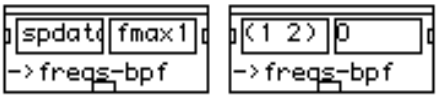
spdata	objet C-spdata
ampmax	nombre entier ou flottant

Sortie

Liste

Formate les données d'amplitudes contenues dans l'objet C-spdata pour permettre leur affichage dans une BPF. La sortie de ce module doit entrer dans l'entrée numéro deux d'un objet BPF. *ampmax* donne l'échelle de l'axe des ordonnées. Les amplitudes sont multipliées par 1000 avant d'être converties en nombres entiers dans la BPF.

2.6.2 ->freqs-bpf



Syntaxe

(->freqs-bpf *spdata fmax*)

Entrées

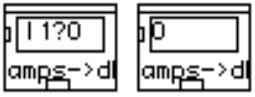
spdata	objet C-spdata
fmax	nombre entier ou flottant

Sortie

Liste

Formate les données de fréquences contenues dans l'objet C-spdata pour permettre leur affichage dans une BPF. La sortie de ce module doit entrer dans l'entrée numéro un d'un objet BPF. *fmax* donne l'échelle de l'axe des abscisses.

2.6.3 amps->db



Syntaxe

(spdata::amps->db *amps*)

Entrées

amps Liste

Sortie

Liste

Convertit la liste des amplitudes fournie par le module **->amp-bpf** en amplitudes en décibels. Ce module est destiné à être bran ché dans l'entrée numéro deux d'une BPF pour l'affichage du spectre en décibels.

2.6.4 sp-frames

Syntaxe

(sp-frames *spdata-seq*)

Entrées

spdata-seq objet C-spdata-seq

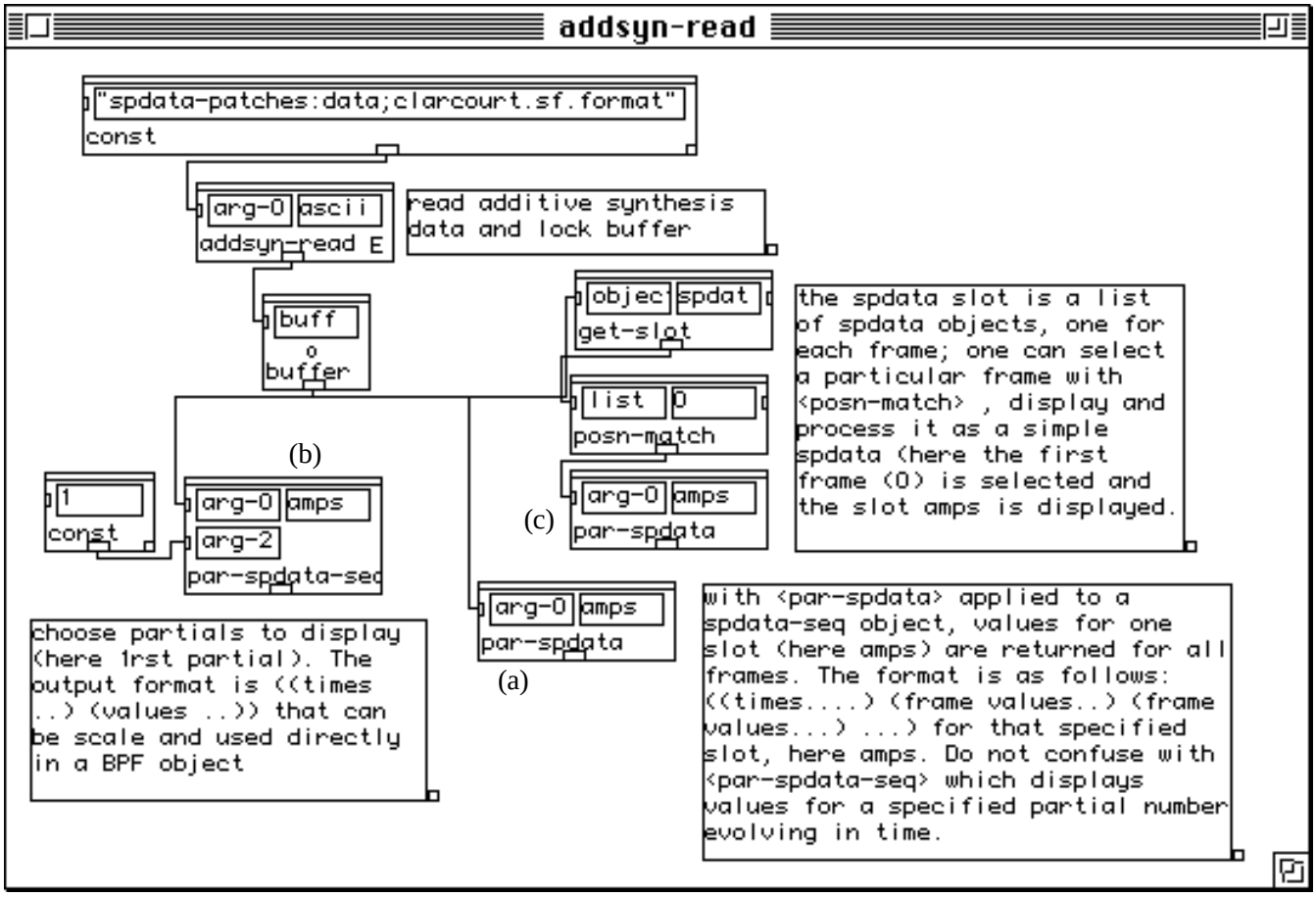
Sortie

Liste

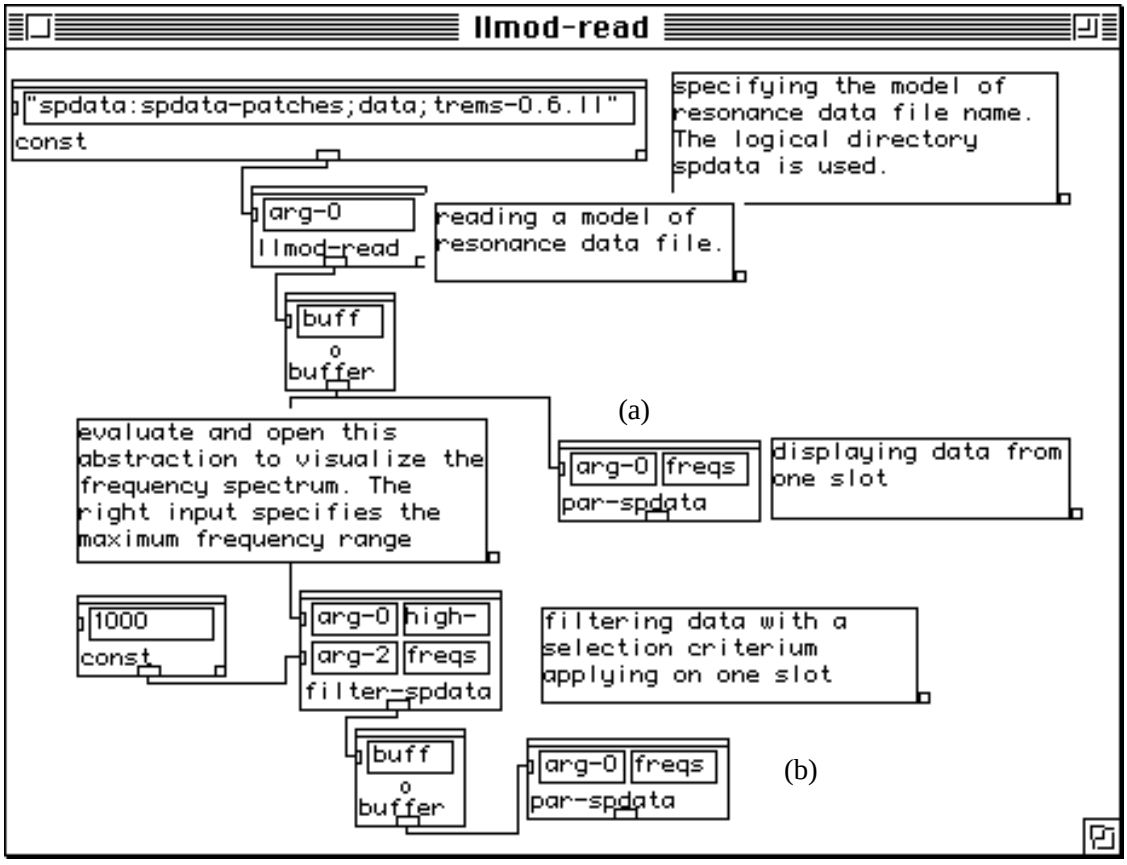
Donne la liste des frames (en sec.) des objets C-spdata contenus dans l'objet **C-spdata-seq**.

3 Patchs d'exemples

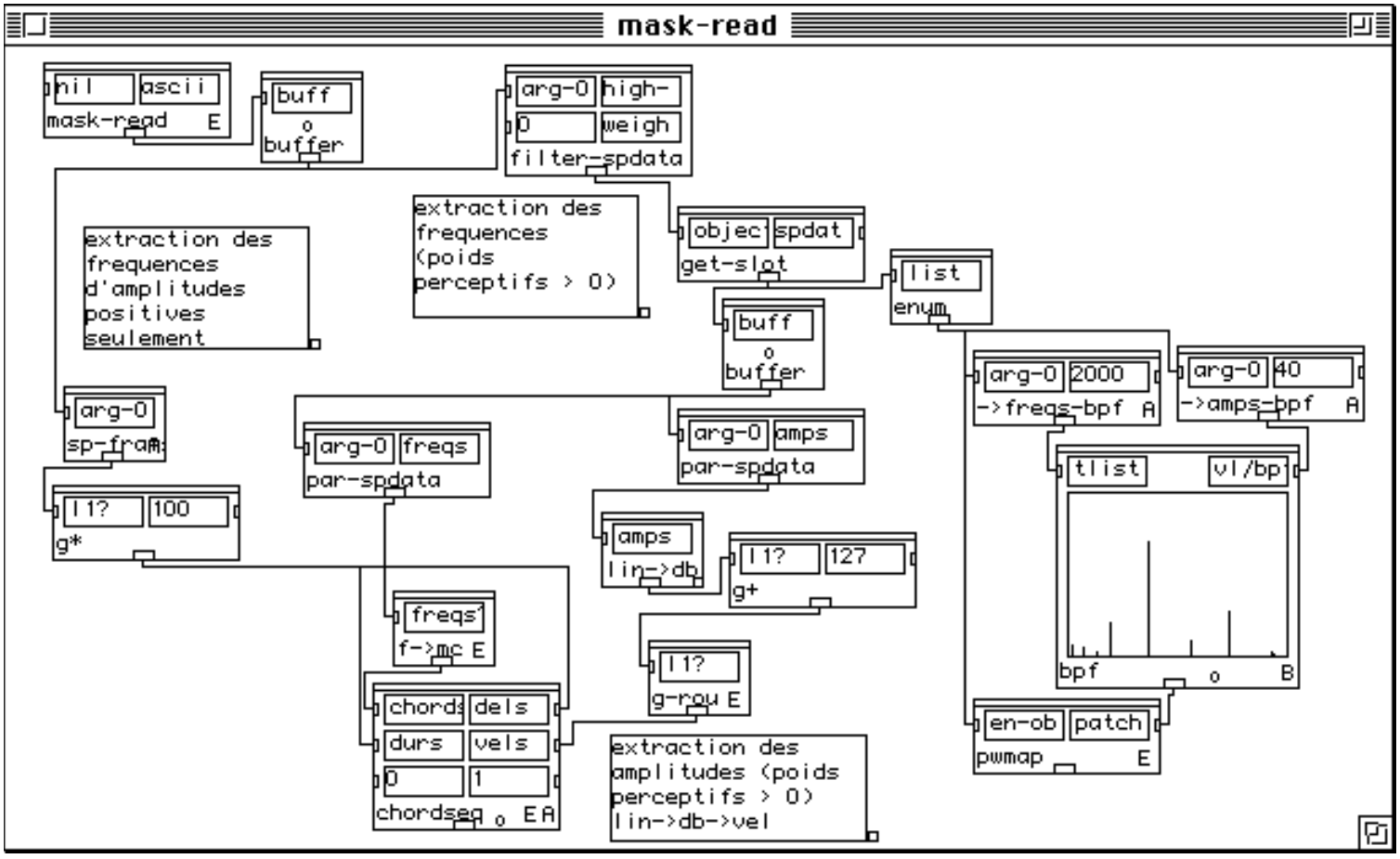
3.1 Entrées et sorties



Patch pour la lecture d'un fichier de données au format "Additive". Ici le patch permet la visualisation des amplitudes des partiels soi dans leur ensemble (a), soit sur l'évolution du premier partiel (b), soit pour les partiels de la première analyse réalisée dans le temps (c).

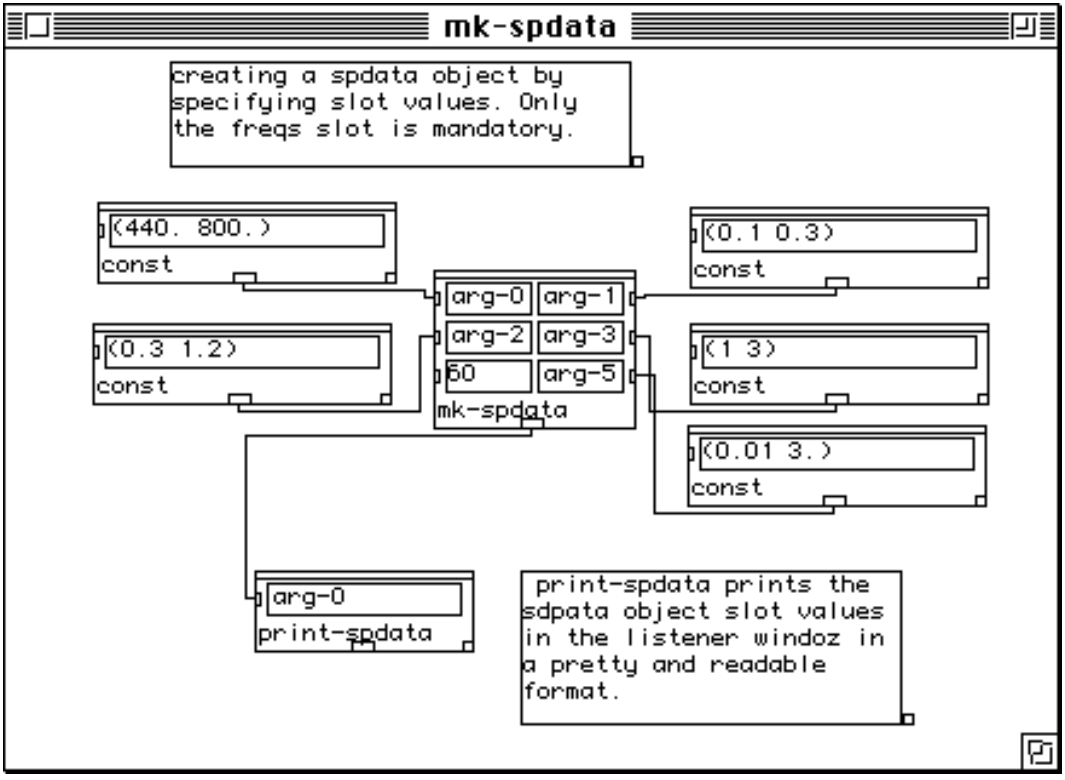


Patch pour la lecture d'un fichier de données au format "Modèle de Résonance". Le module **par-spdata** permet l'affichage des fréquences du modèles (a). Dans la partie inférieure du patch (b), les fréquences inférieures à 1000 Hz seront affichées seulement.

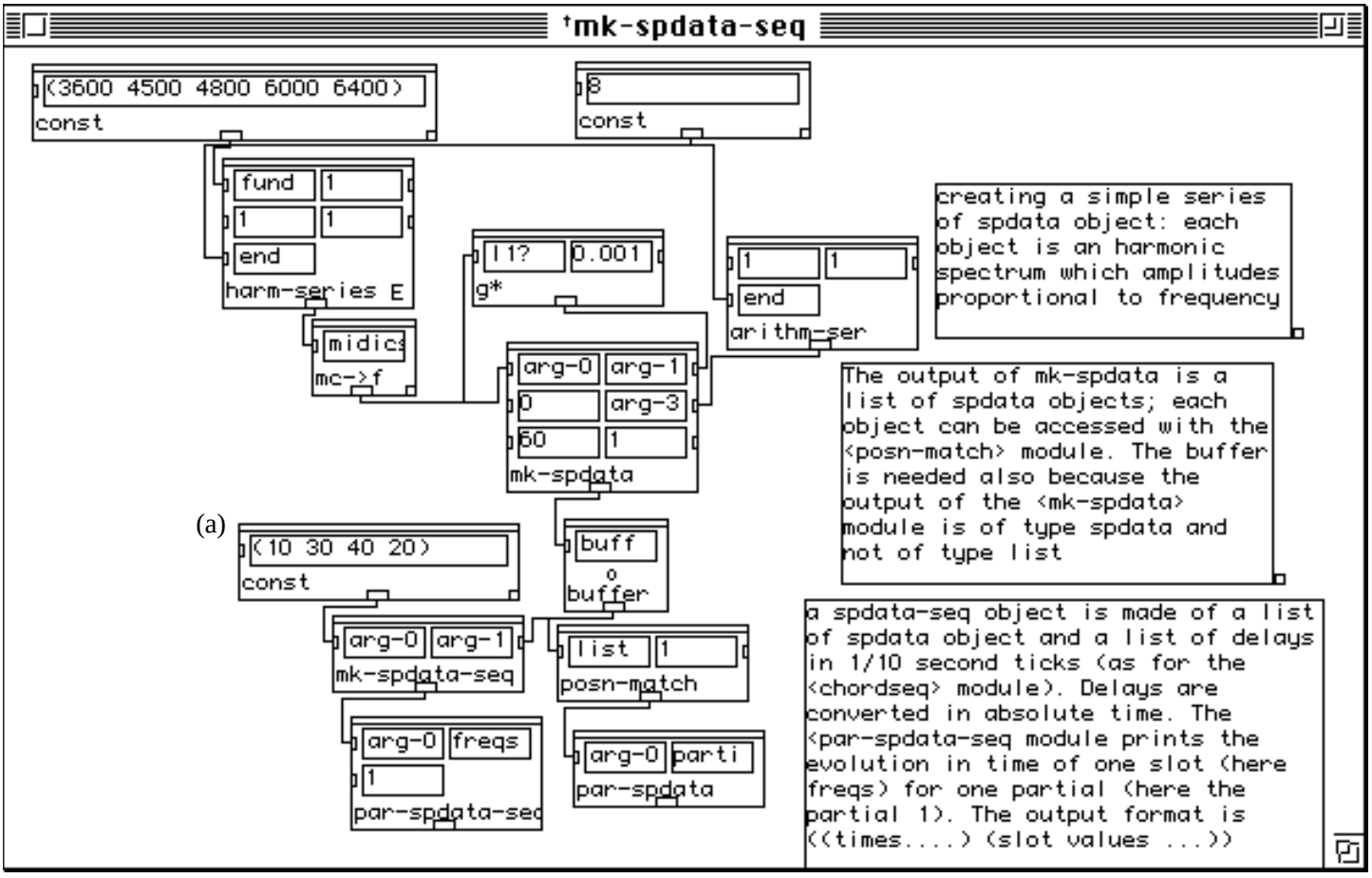


Patch destiné à la lecture d'un fichier d'analyse contenant des données au format "Mask" réalisées avec le programme AudioSculpt
Après filtrage, ces données sont affichées soit dans un éditeur d'accords avec le module **chordseq**, soit sous forme spectrale dans une
BPF.

3.2 Création

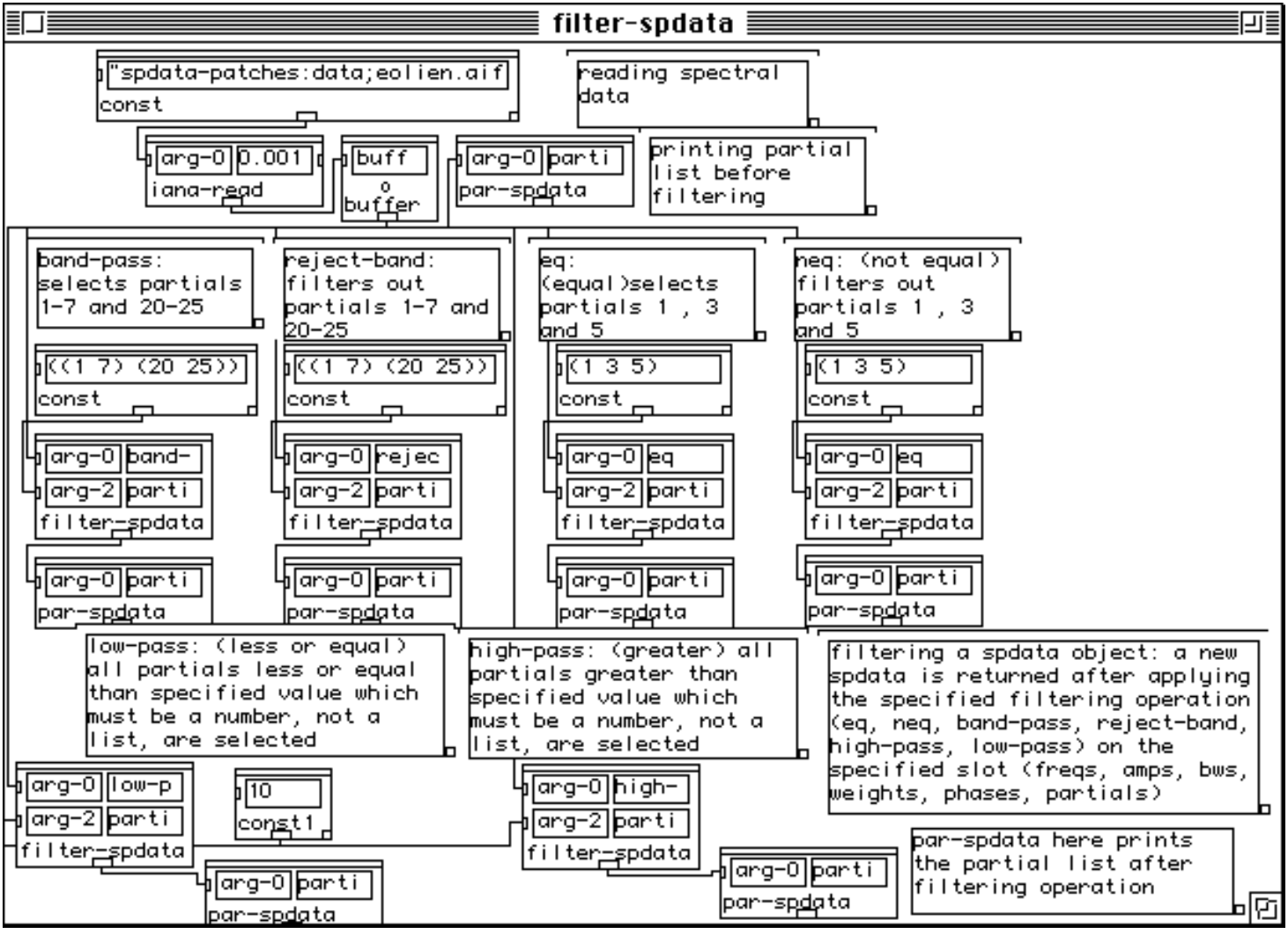


Patch pour la création d'un objet C-spdata en fournissant des listes de valeurs pour les différents « slots ». Au moins l'un des slots doit recevoir une liste.

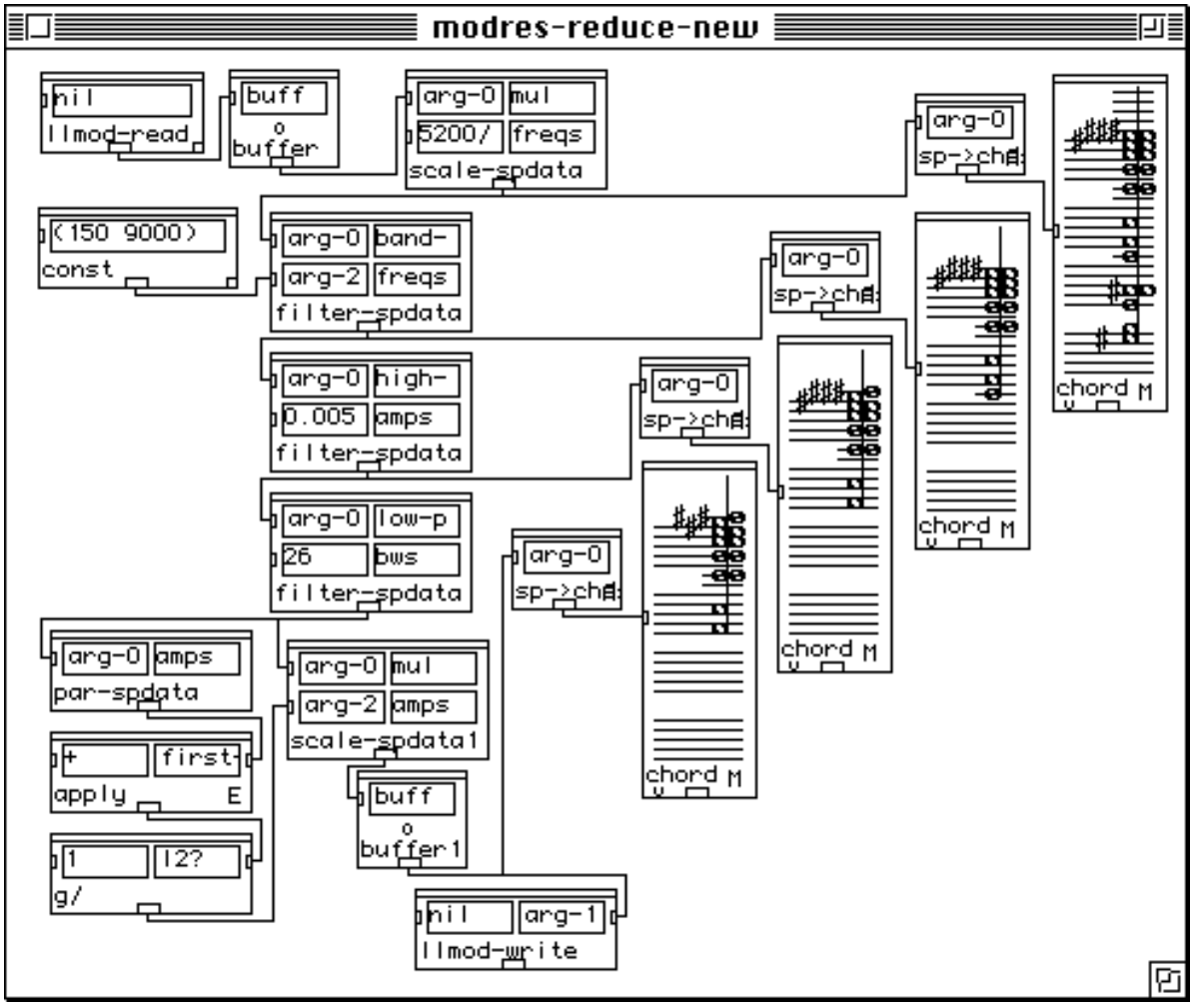


Patch pour la création d'un objet C-spdata-seq formé d'une liste d'objets C-spdata créés à partir d'une suite de séries harmoniques. La liste (a) donne les dates des différents objets C-spdata.

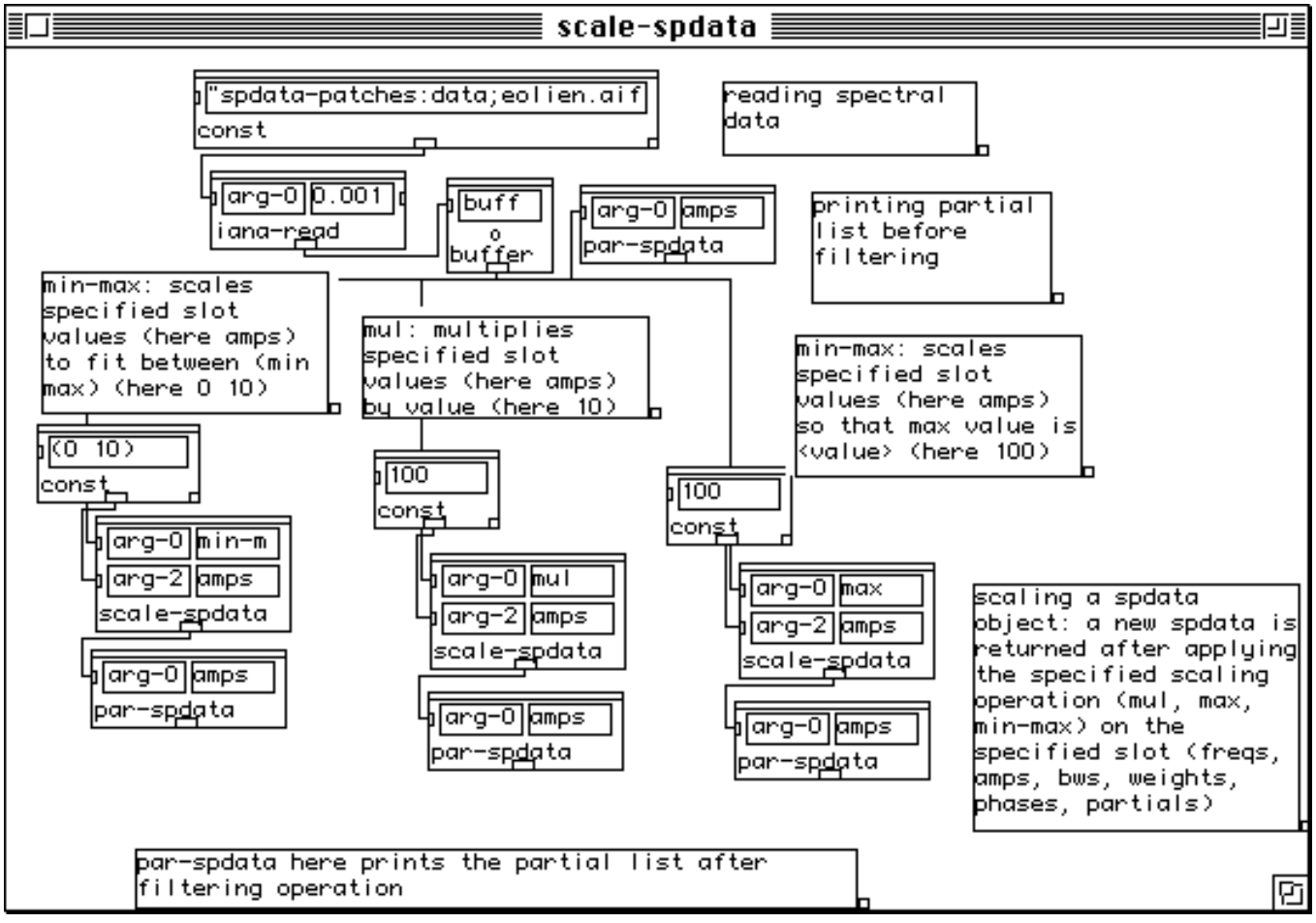
3.3 Modifications



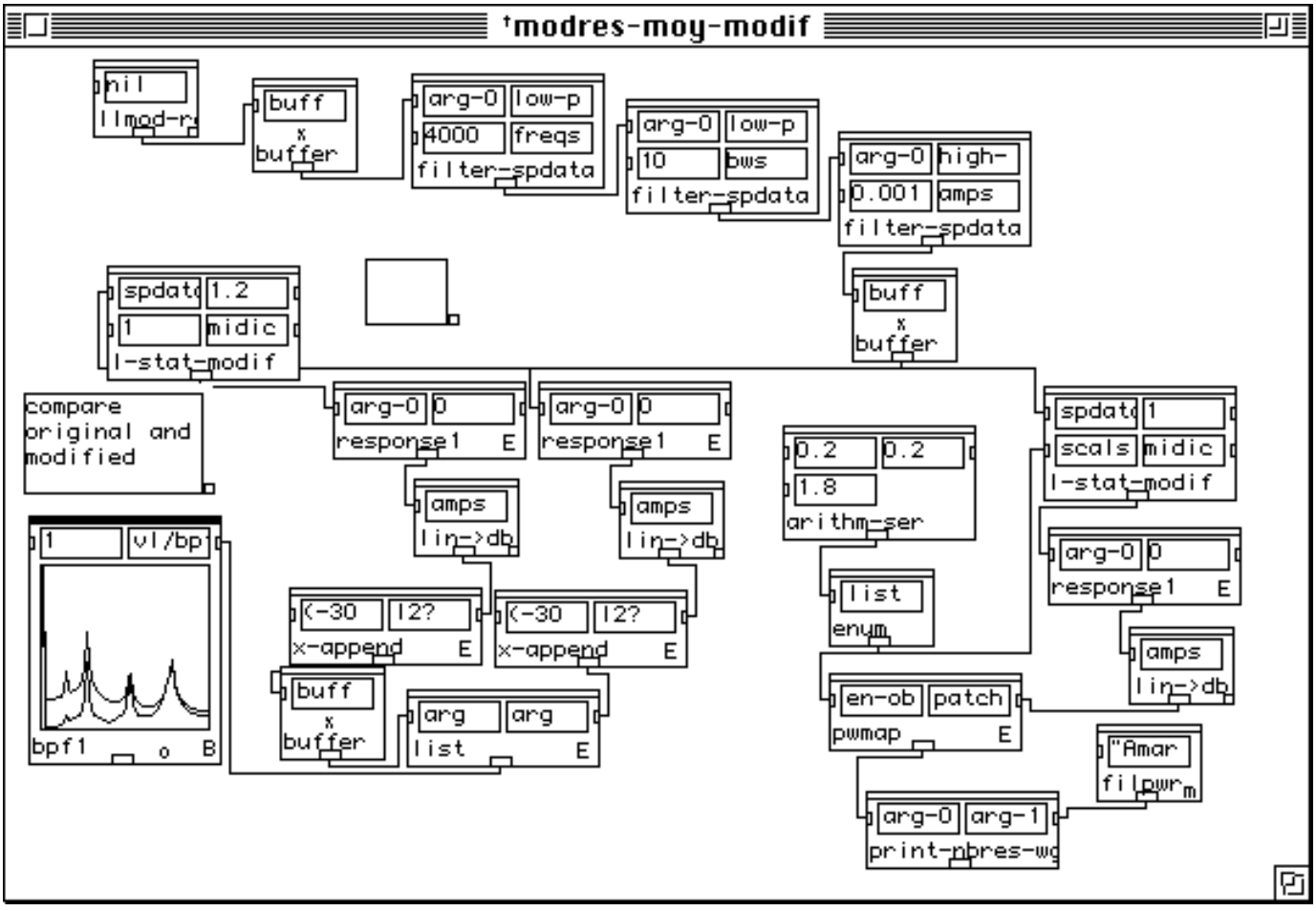
Patch illustrant les différentes options pour le filtrage d'objets C-spdata ou C-spdata-seq. Les nouveaux objets produits par le module **filter-spdata** ne contiennent plus que les partiels satisfaisant aux conditions énoncées dans le filtre.



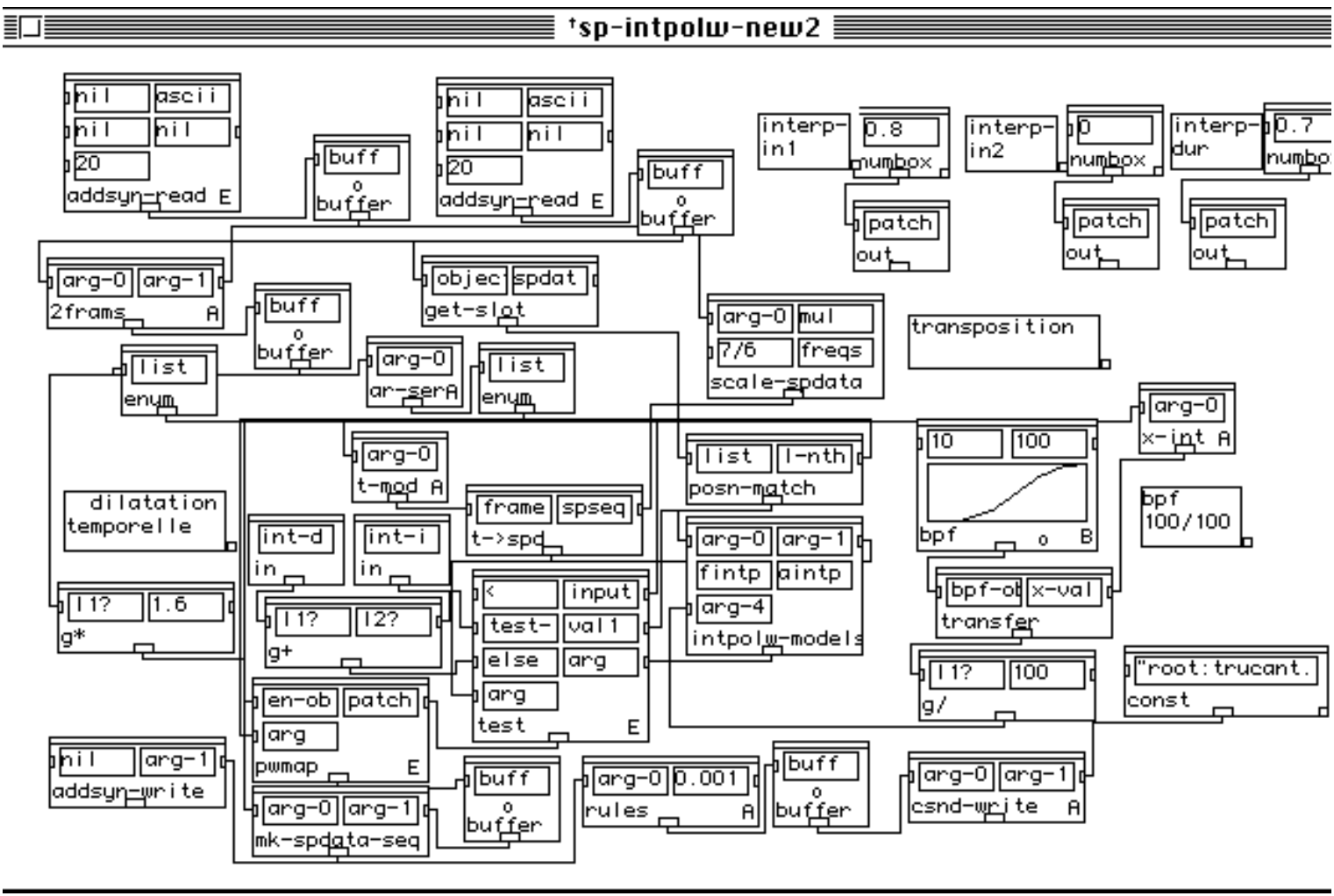
Patch permettant de réduire progressivement la quantité de données contenue dans un objet C-spdata après un filtrage éliminant les fréquences inférieures à 150 Hz et supérieures à 9000 Hz, puis un filtrage supprimant les partiels d'amplitudes inférieures à la valeur 0,005 et un autre éliminant les partiels de largeurs de bande supérieures à 26 Hz. Les amplitudes des partiels sont ensuite normalisées en utilisant le module **scale-spdata**.



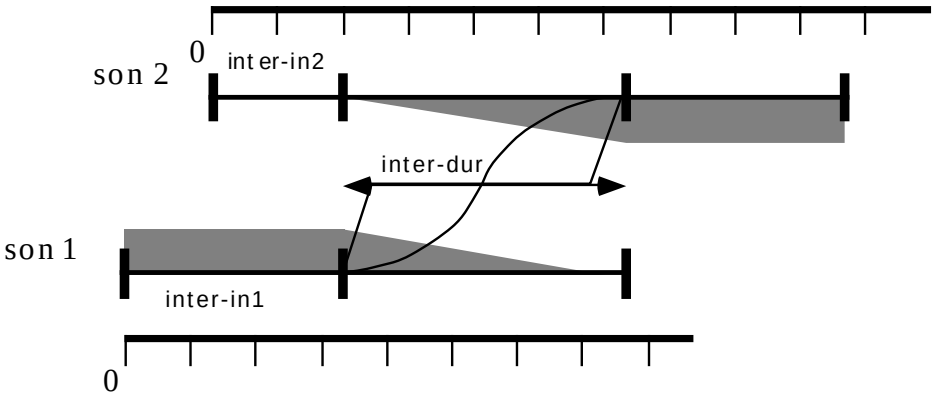
Patch illustrant les différentes options pour l'utilisation du module **scale-spdata**. Les nouveaux objets produits par ce module subissent une remise à l'échelle des slots indiqués.

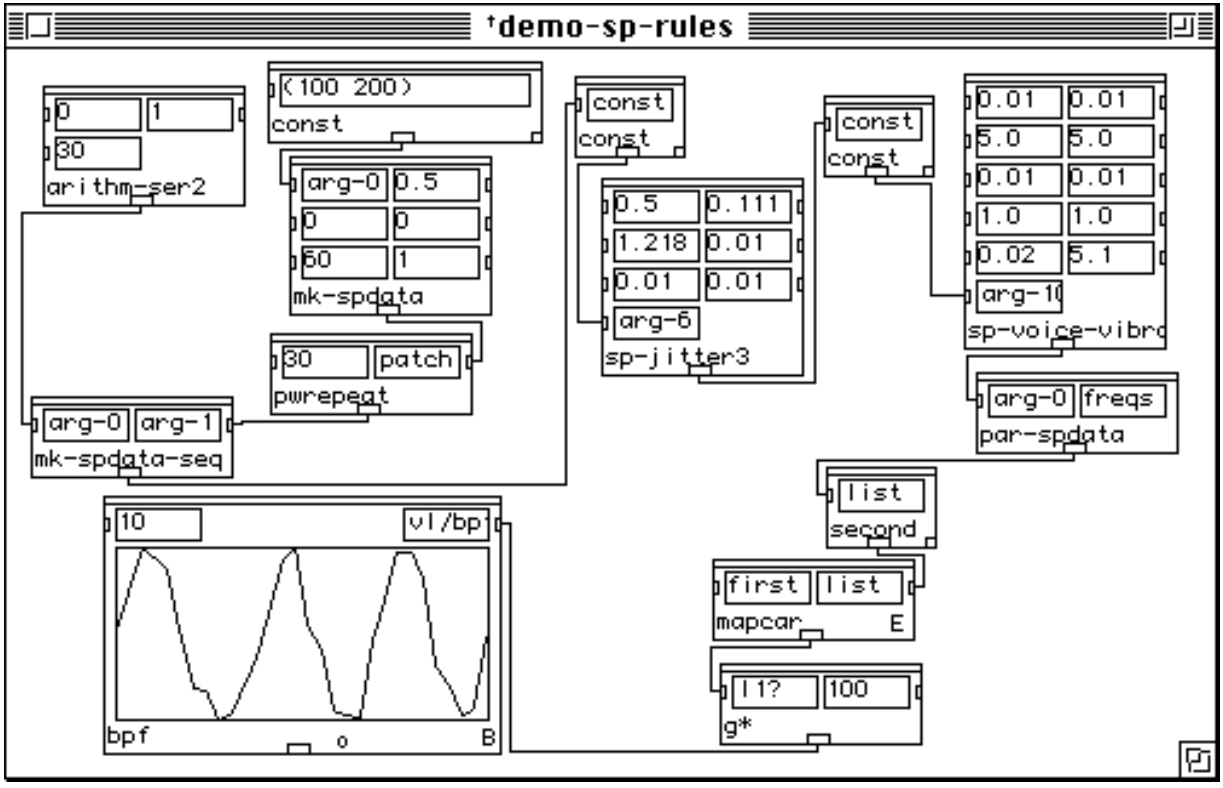


Patch utilisant le module **l-stat-modif** permettant de modifier la répartition d'énergie dans le spectre des fréquences. Ici, les fréquences grave ont été atténuées. Un nouvel objet a été produit dont la fréquence moyenne est multipliée par un facteur 1,2 par rapport à l'original.



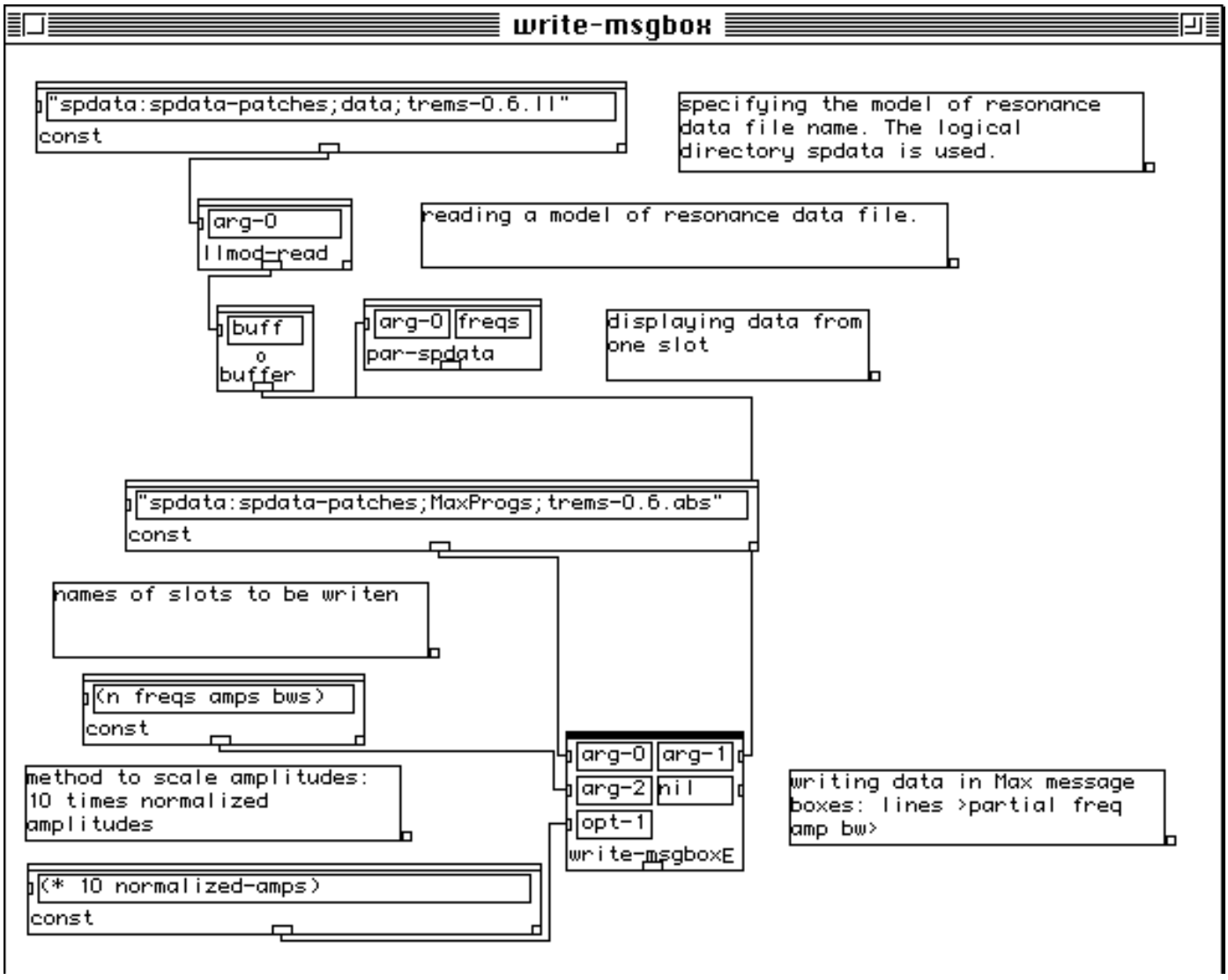
Patch pour l'interpolation des données entre deux analyses "Additives". Les valeurs "inter-in1", "inter-in2" et "inter-dur" indiquen respectivement la durée du son 1 avant que ne commence son interpolation, la durée du son 2 avant que ne commence son interpolation et la durée de l'interpolation entre le son 1 et le son 2. La librairie "Csound/Edit-sco" doit être chargée pour pouvoir ouvrir ce patch qui permet l'écriture d'un score pour le programme CSound (abstraction **csnd-write**).



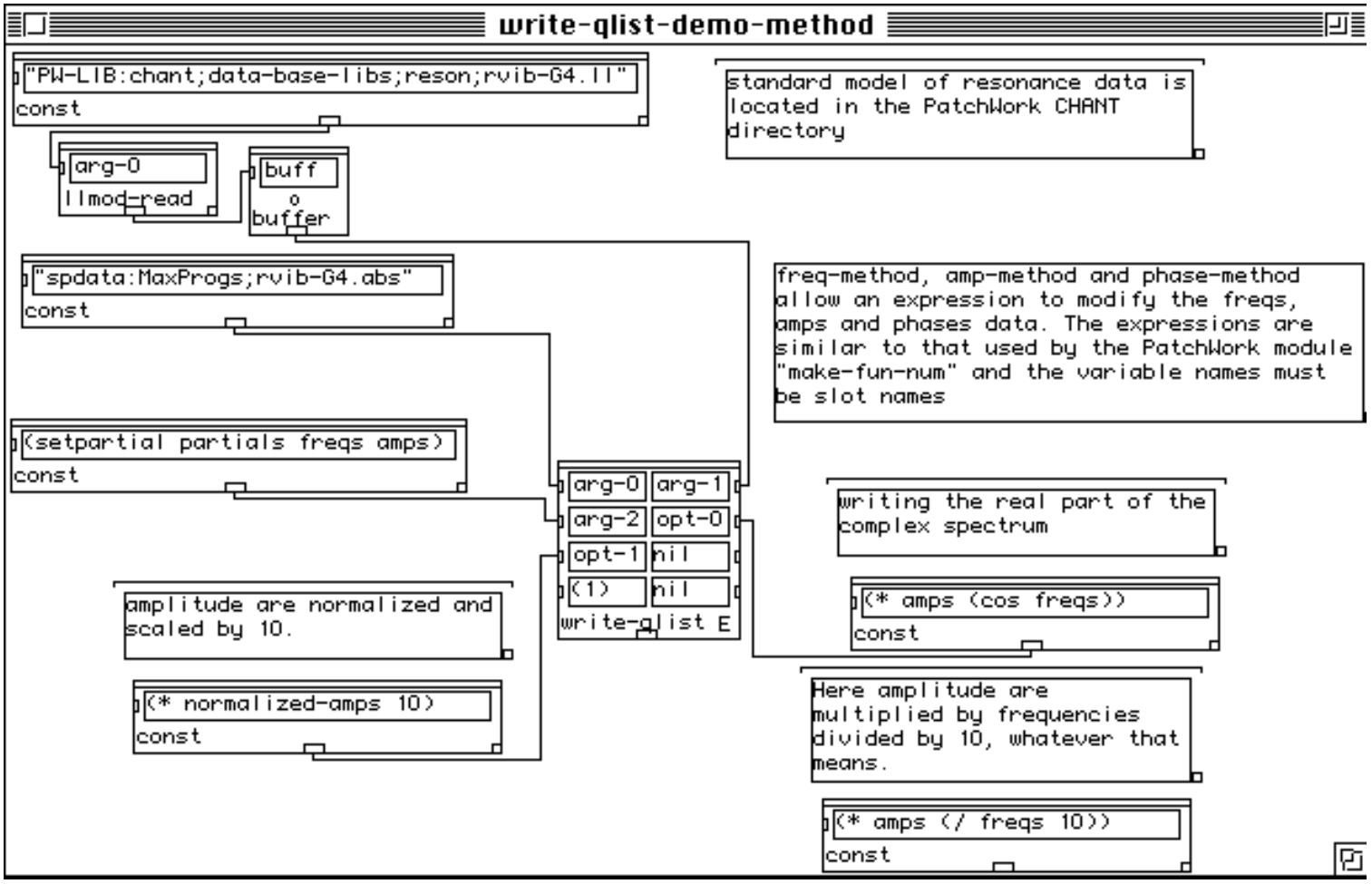


Patch permettant la modification des fréquences des partiels sous l'effet d'un vibrato et d'un triple jitter.

3.4 Max

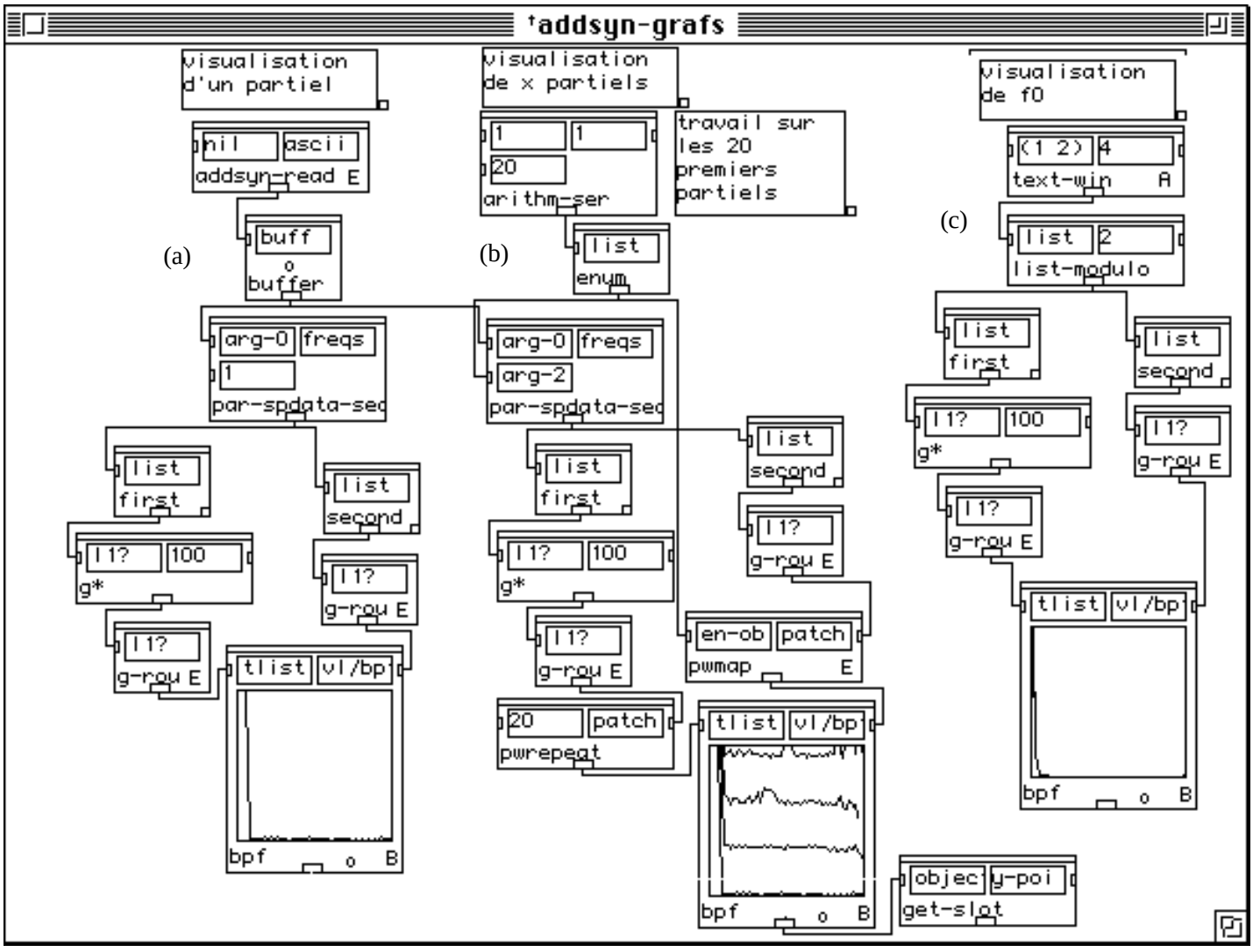


Patch pour la création d'un sous-patch pour la Station Musicale de l'Ircam. Ce sous-patch contient les données assignées à des variables dans des modules de messages

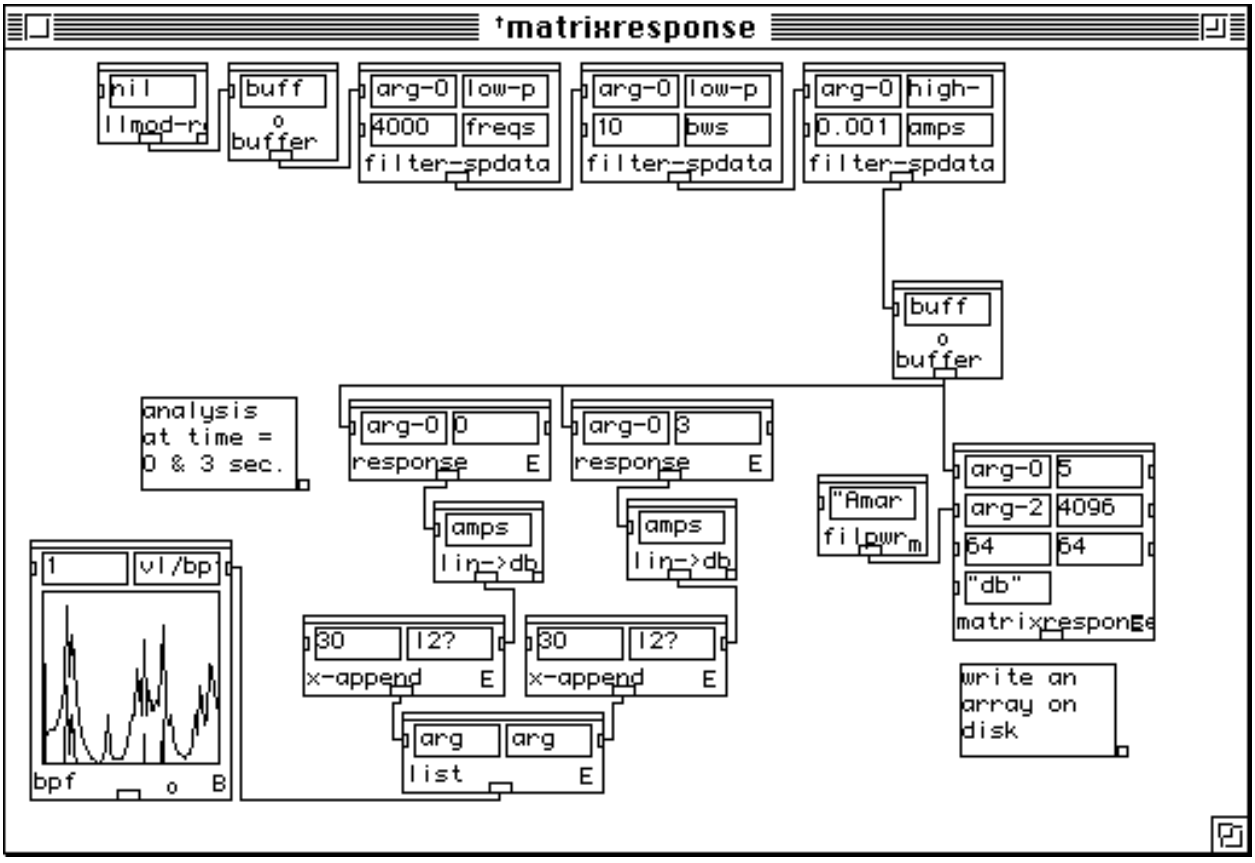


Patch pour la création d'un sous-patch pour la Station Musicale de l'Ircam. Ce sous-patch contient les données assignées à des variables dans des modules **qlist**

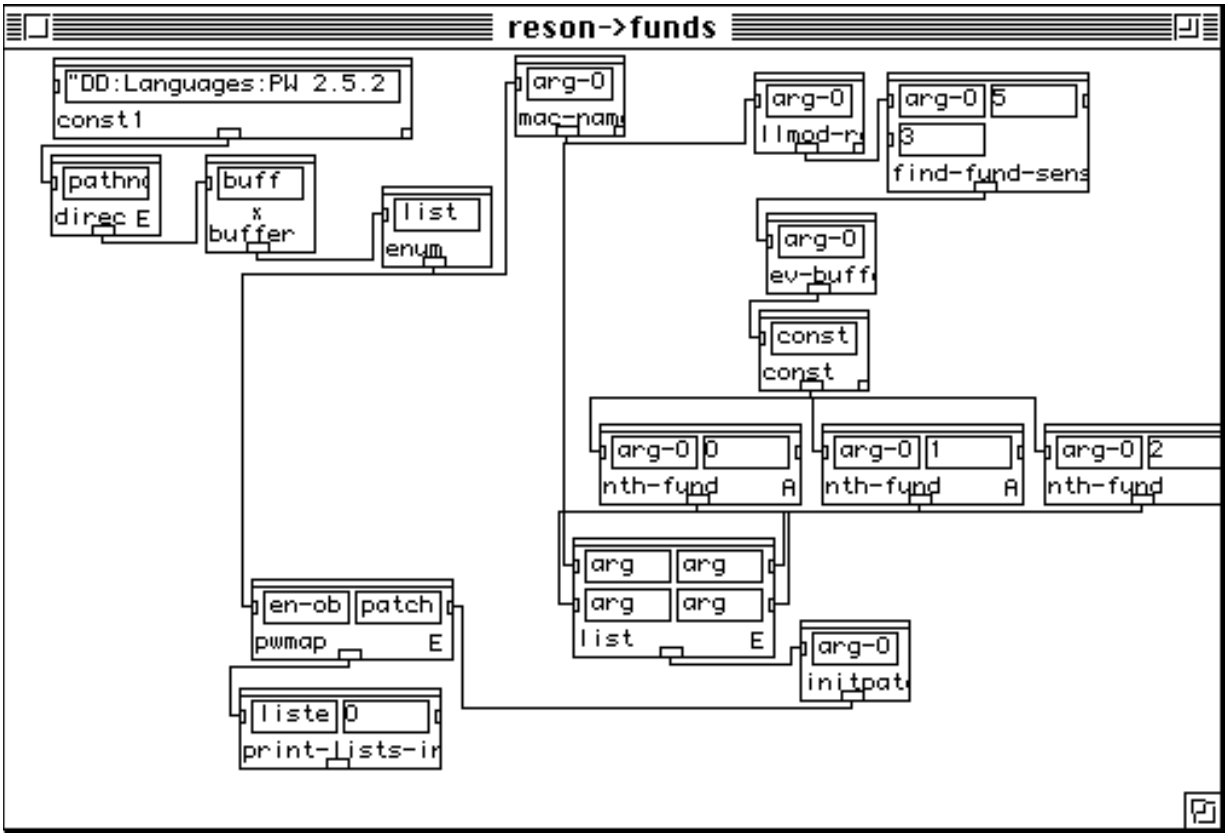
3.5 Etude des données



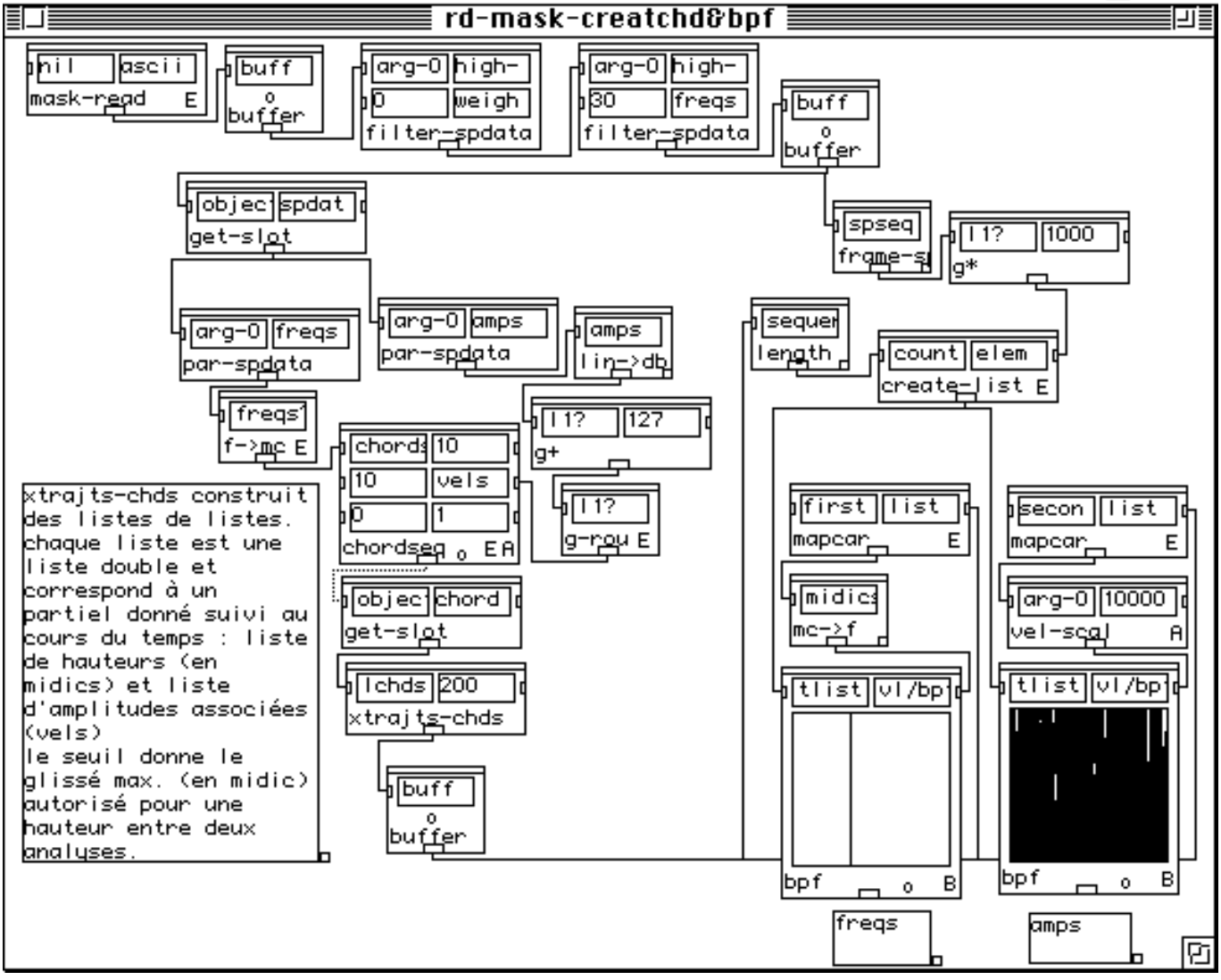
Patch pour la visualisation des données dans des BPF pour un fichier d'analyse "Additive". La branche (a) permet de visualiser le premier partiel, la branche (b) permet de visualiser les 20 premiers partiels et la branche (c) permet de visualiser la fondamentale qui aura été chargée à l'aide du module **text-win**.

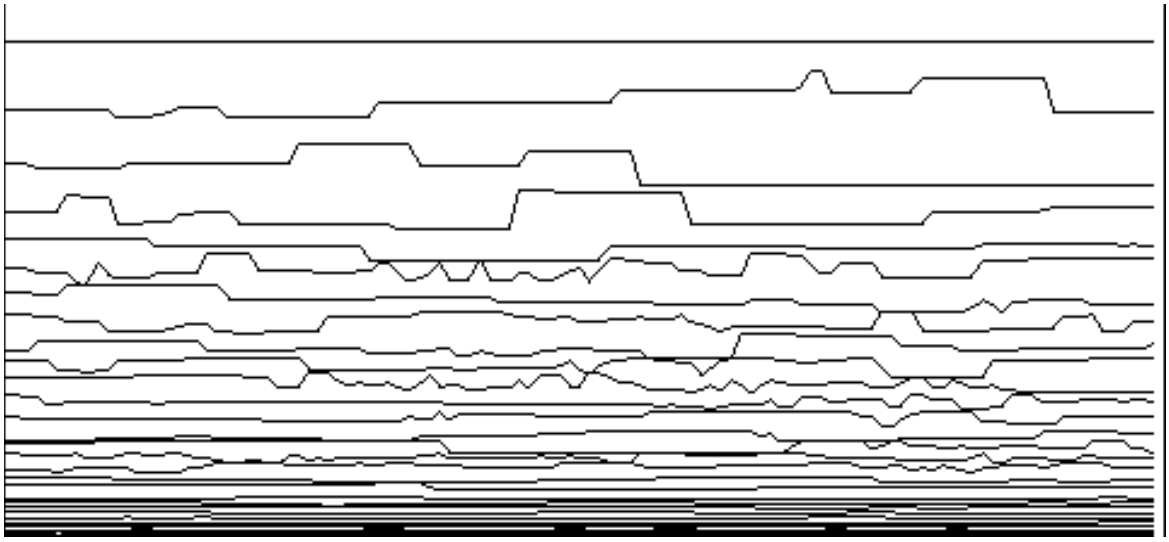


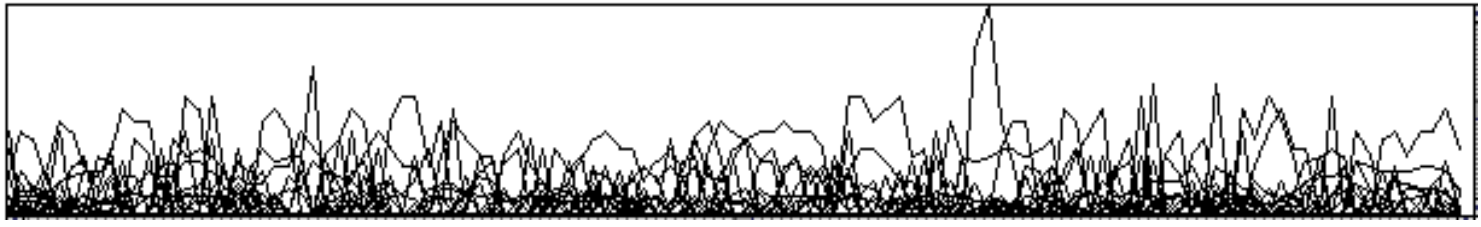
Patch permettant d'évaluer l'enveloppe spectrale d'un Modèle de Résonance et son évolution dans le temps.



Patch pour la recherche des séries harmoniques contenues dans un ensemble de fichiers contenant des données de type "Modèle de Résonance".

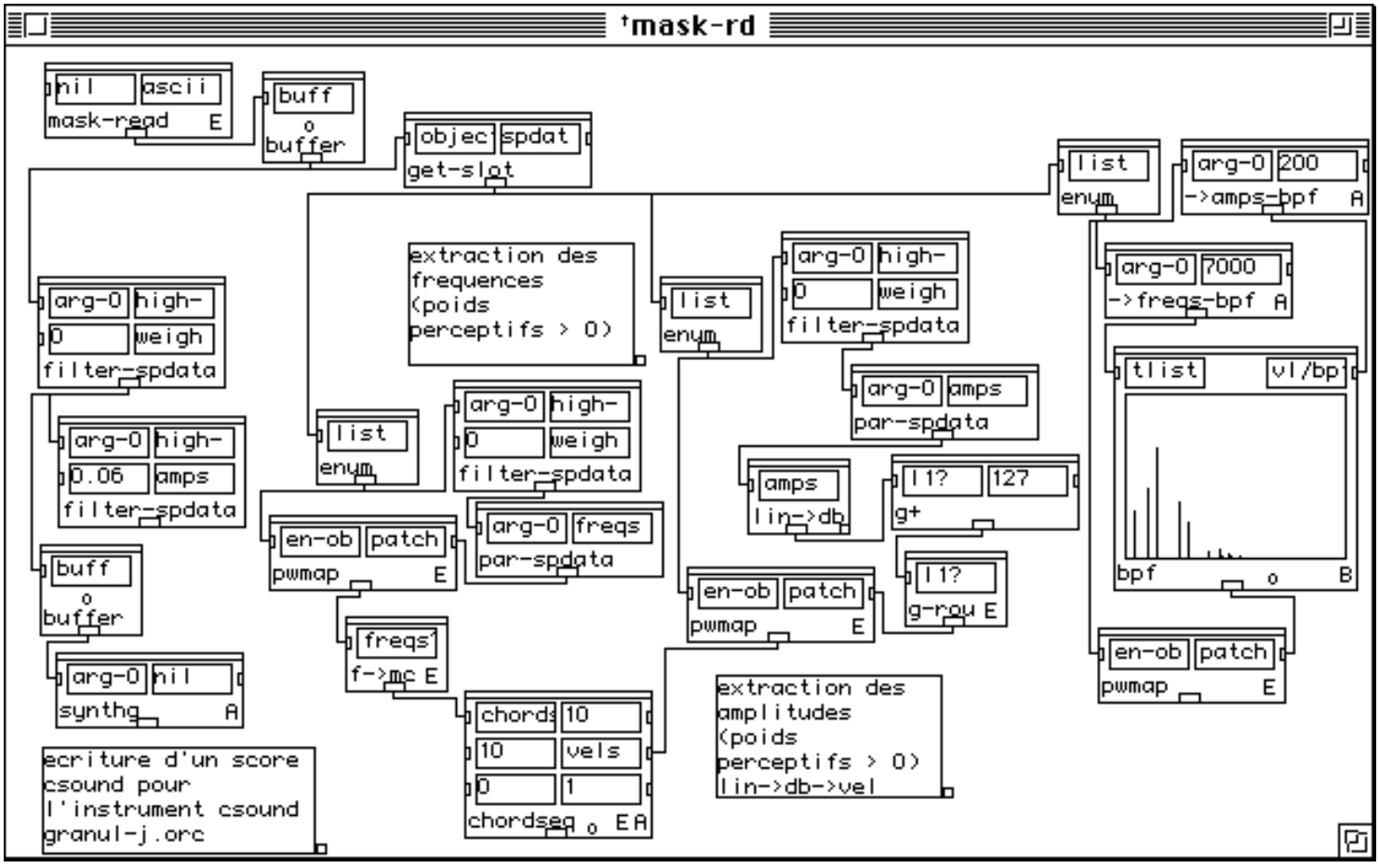

rd-mask-creatchd&bpf
nil
ascii
mask-read
E
buff
o
buffer
arg-0
high-
0
weigh
filter-spdata
arg-0
high-
30
freqs
filter-spdata
buff
o
buffer
objec
spdat
get-slot
spseq
frame-s
l1?
1000
g*
arg-0
freqs
par-spdata
arg-0
amps
par-spdata
amps
lin->db
sequer
length
count
elem
create-list
E
freqs
f->mc
E
chords
10
10
vels
0
1
chordseq
o
EA
l1?
127
g+
l1?
g-rou
E
xtrajts-chds construit des listes de listes. chaque liste est une liste double et correspond à un partiel donné suivi au cours du temps : liste de hauteurs (en midics) et liste d'amplitudes associées (vels)
le seuil donne le glissé max. (en midic) autorisé pour une hauteur entre deux analyses.
first
list
mapcar
E
secon
list
mapcar
E
midics
mc->f
arg-0
10000
vel-scal
A
objec
chord
get-slot
lchds
200
xtrajts-chds
tlist
vl/bp
tlist
vl/bp
buff
o
buffer
bpf
o
B
bpf
o
B
freqs
amps



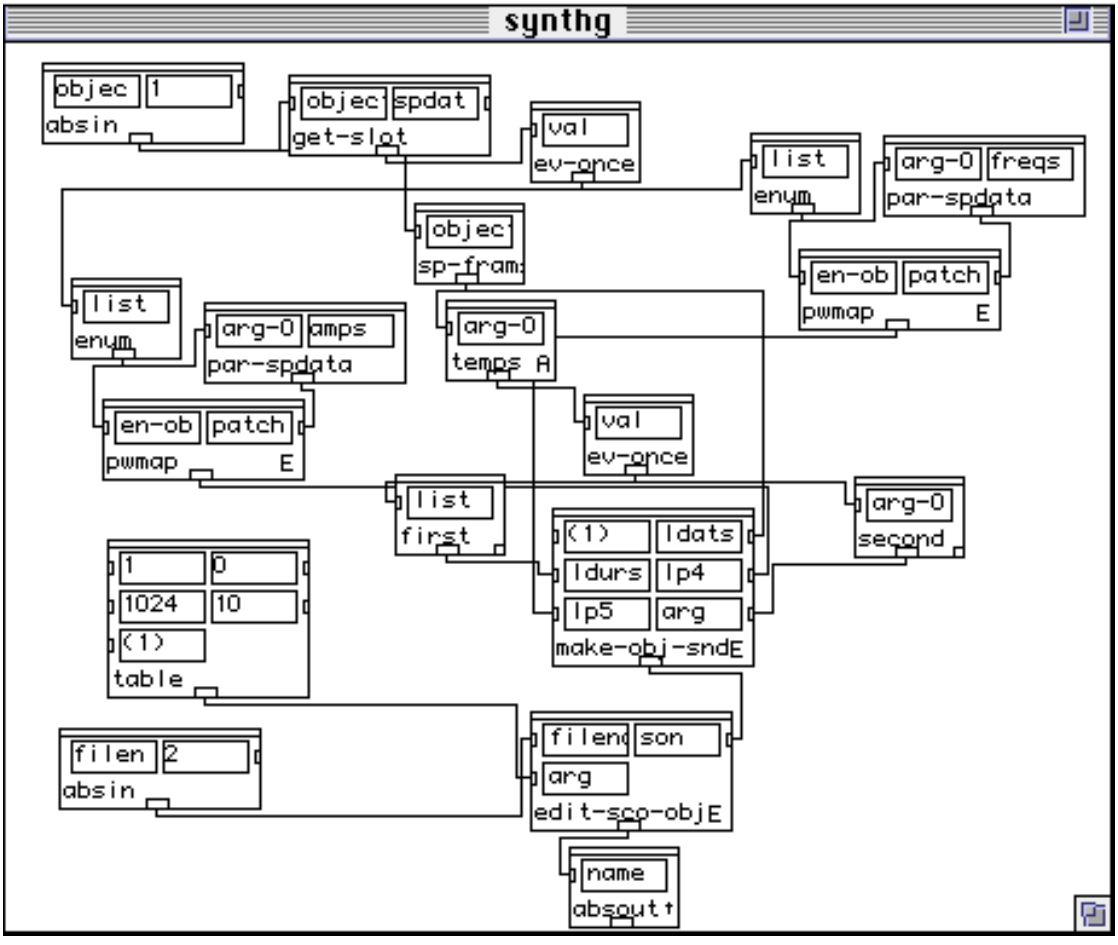


Patch pour la construction de BPF donnant des trajets de fréquences et d'amplitudes de partiels calculés à partir de données de type "Mask". Ces données peuvent alors être utilisées pour des synthèse additives ou soustractives.

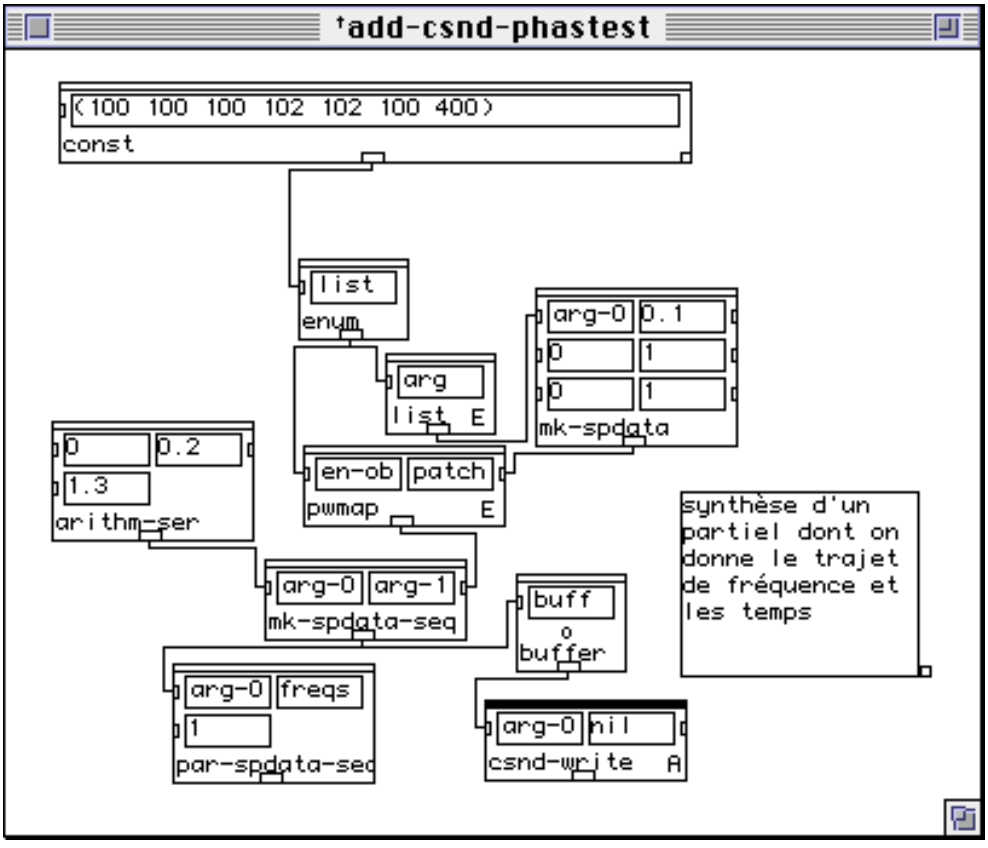
3.6 SpData vers Csound



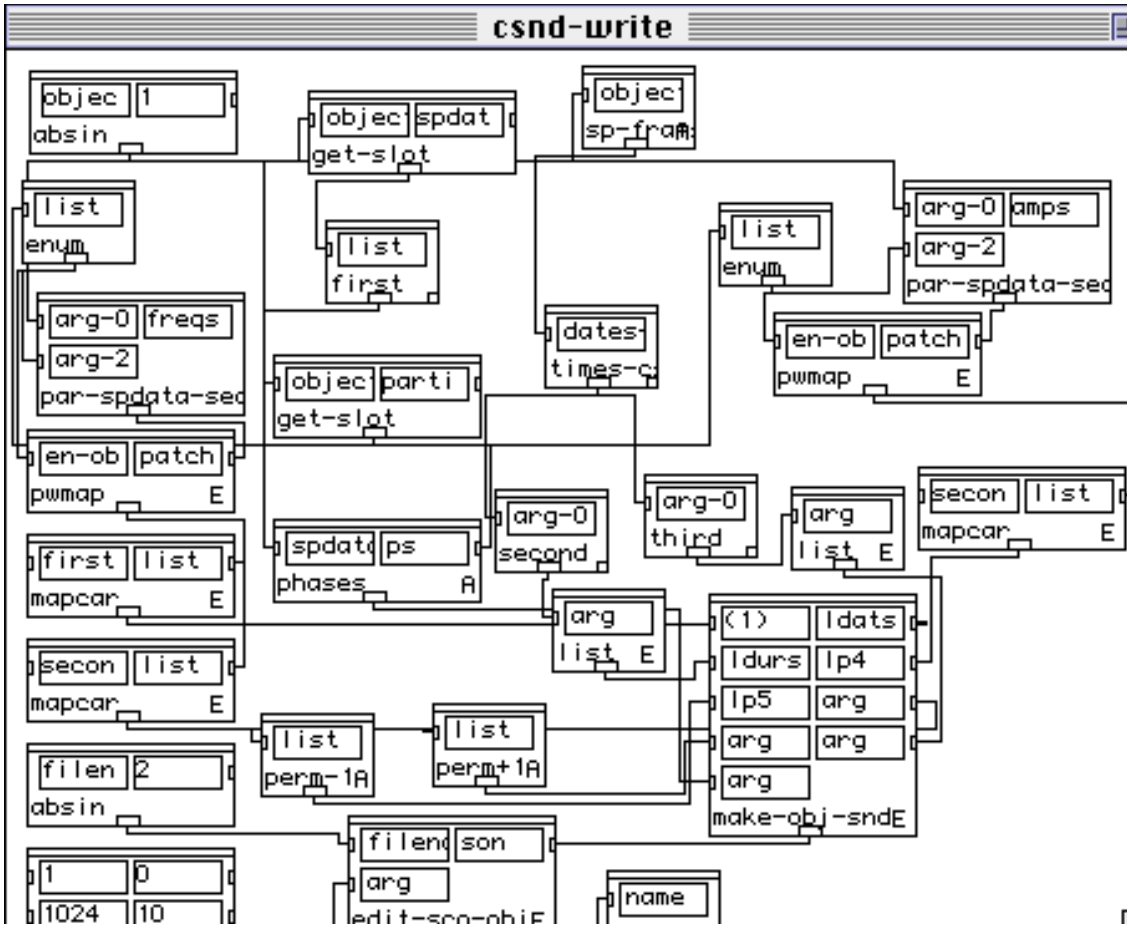
Patch permettant l'affichage des données contenues dans un fichir d'analyse de type "Mask". L'affichage peut se faire dans un module **chordseq** ou dans une BPF. L'abstraction **synthg** permet la production d'un fichier de partition (score) pour le programme Csound. Ce score peut être utilisé avec l'orchestre Csound "granul-j.orc" pour réaliser un son de synthèse permettant le valider la cohérence de l'analyse.



Détail de l'abstraction **synthg** permettant de produire un fichier de partition (score) pour l'orchestre "granul-j.orc" à partir de données d'analyse de type "Mask". Pour une synthèse de qualité, il est important que les analyses n'aient pas été trop proches dans le temps. Des valeurs du pas d'avancement de l'analyse supérieures à 0,1 seconde sont souhaitables.

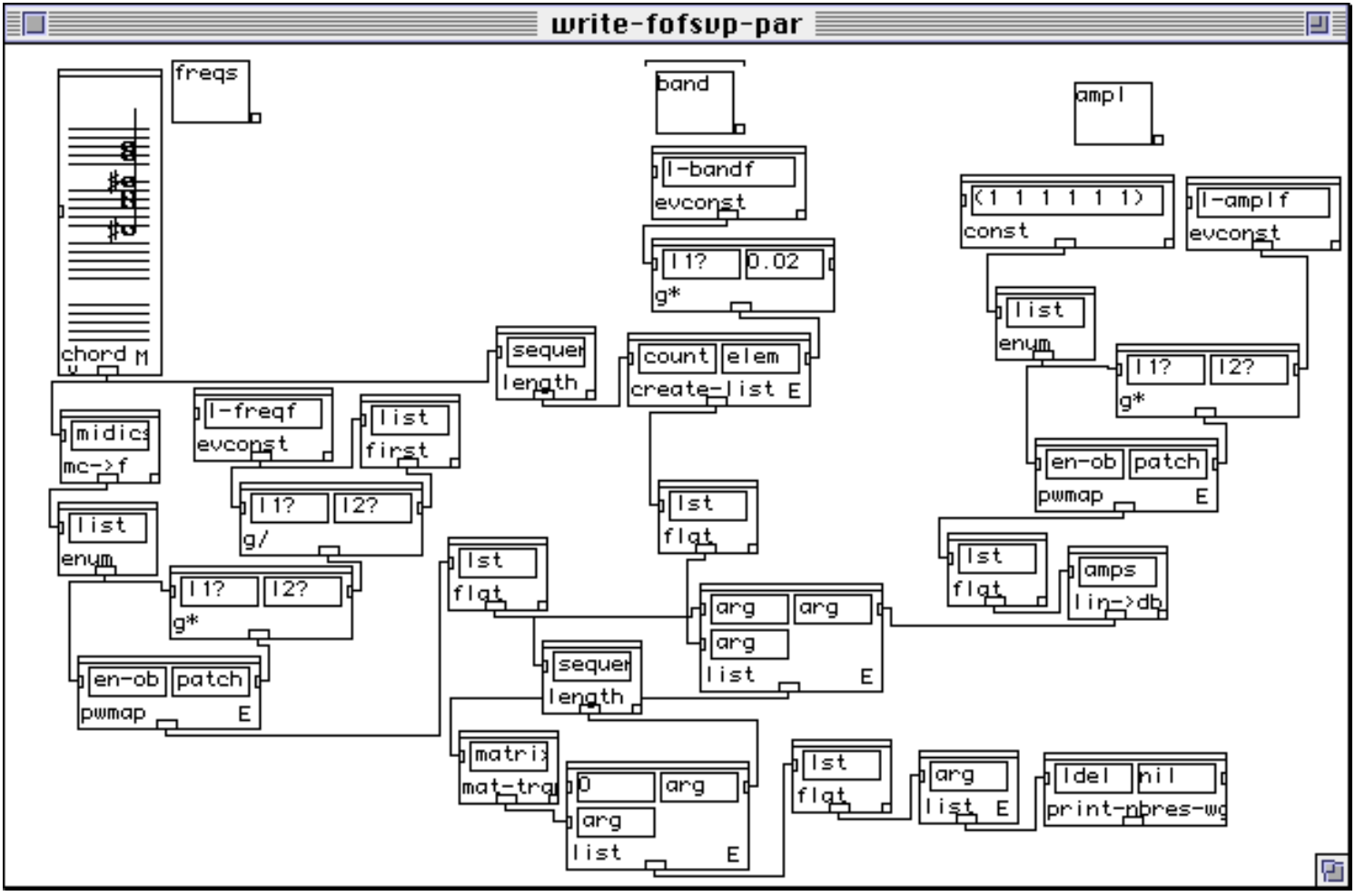


Patch permettant de créer un score pour le programme Csound pour réaliser une synthèse "Additive" avec l'orchestre "granul-r.orc" L'abstraction **csnd-write** permet la production du fichier de partition (score).



Détail de l'abstraction **csnd-write** permettant de produire un fichier de partition (score) pour l'orchestre "granul-r.orc" à partir de données d'analyse de type "Additive".

3.7 Données pour AudioSculpt



Patch pour la production d'un fichier de données numériques agencées en colonnes destinées à être utilisées pour un filtrage formantique avec le programme AudioSculpt.

4 Annexes

4.1 Annexe 1 : Modes d'interpolations pour le module intpol-model

Les différents cas de figures sont résumés par un paramètre: *flag* qui peut prendre 8 valeurs différentes:

• flag = 0: synthèse sans interpolation (modèle G seul).

• flag = 1: interpolation entre les modèles G et D contrôlée par le paramètre scaler de type "intersection + mixage" :

-le mixage correspond au simple mélange des résonances des deux instruments; le paramètre scaler étant utilisé pour contrôle les amplitudes respectives des deux modèles.

- l'intersection correspond à une interpolation des résonances des modèles gauche et droit qui sont suffisamment proches en fréquences (appariement lorsque la différence de hauteurs ne dépasse pas le paramètre seuil (en midicents). Les résonances trop éloignées ne sont pas utilisées.

• flag = 2: interpolation par mixage.

• flag = 3: interpolation par intersection seule.

• flag = 4: type 1 + interpolation de la "durée globale de la résonance" des instruments :

la qualité plus ou moins résonante d'un modèle a été définie par les concepteurs des outils des Modèles de Résonance par un indice. Celui-ci est calculé pour chacun des modèles à interpoler et sert à modifier les largeurs de bandes des résonances de deux modèles pour produire une interpolation de la résonance:

$$\text{indice de résonance globale} = \sqrt{\frac{\sum_i \frac{amp_i}{(bw_i)^2}}{\sum_i amp_i}}$$

• flag = 5: type 2 + interpolation de la "durée globale de la résonance" des instruments.

• flag = 6: Pour une balance (scaler) variant de 0 à 0,5 , on passe de l'instrument G original à l'instrument gauche avec la qualité résonante de l'instrument droit puis pour une balance variant entre 0,5 et 1, on passe de l'instrument droit à l'instrument droit avec la qualité résonante de l'instrument gauche.

• flag = 7: Pour une balance variant entre 0 et 1, on passe successivement par les étapes suivantes: instrument gauche -> instrument gauche avec qualité résonante de l'instrument droit -> instrument droit -> instrument droit avec qualité résonante de l'instrument gauche -> instrument gauche

4.2 Annexe 2 : Instruments pour le programme Csound

```
;-------------------------------------------------------------
              ;(ORCHESTRE) add-tab.orc

; synthèse additive avec tables
; pour trajets de fréquences et d'amplitudes

;p1 =ins
;p2 =date
;p3 = dur
;p4 = amp globale
;p5 = n° table d'env d'amplitude
;p6 = n° table freq globale
;-------------------------------------------------------------
   sr = 44100
   kr = 4410
   ksmps = 10
   nchnls = 1
;-------------------------------------------------------------
instr 1

kgliss1  oscil1i 0,1,p3,p6
kenvamp  oscil1i 0,ampdb(p4),p3,p5
ason     oscili  kenvamp,kgliss1,1
out      ason
endin
;-------------------------------------------------------------
```

Score-type :

```
;
;-------------ondes-----------------
f1 0 4096 10 1
;-----------enveloppes---------------
f10 0 4096 7 0 596  1 500   .6  2500   .2 450   0 50   0
;-------------porta------------------
f20 0 4096 -7 220 4000   440 96   220
;-----------------------------------

;-- par. gaux
 i1 0 3
;-- amp (dB)
 90
;-- n° des tables
 10 20 1
e
```

```
;-------------------------------------------------------------
                ;(ORCHESTRE) granul-r.orc

  ;p1 =ins
  ;p2 =date
  ;p3 = dur
  ;p4 = amp max
  ;p5 = frequence1
  ;p6 = frequence2
  ;p7 = frequence3
  ;p8 = dur avant amp max
  ;p9 = phase
  ;
     sr = 44100
     kr = 4410
     ksmps = 10
     nchnls = 1
  ;
  instr 1

  kenvamp       linseg    0, p8, p4, p3-p8, 0
  kenvfreq      linseg    p5, p8, p6, p3-p8, p7
  ason          oscili   32000*kenvamp, kenvfreq, 1, p9
  out           ason
  endin
  ;-------------------------------------------------------------

  Score-type :

  ;-------------------------------------------------------------
  ; onde sinus
  f 1 0 1024 10 1
  ;  ins    dat     dur     amp    freq1   freq2  freq3  att    phase
  i   1     0       0.8     10     200     202    202    0.4    0
  e
  ;
```

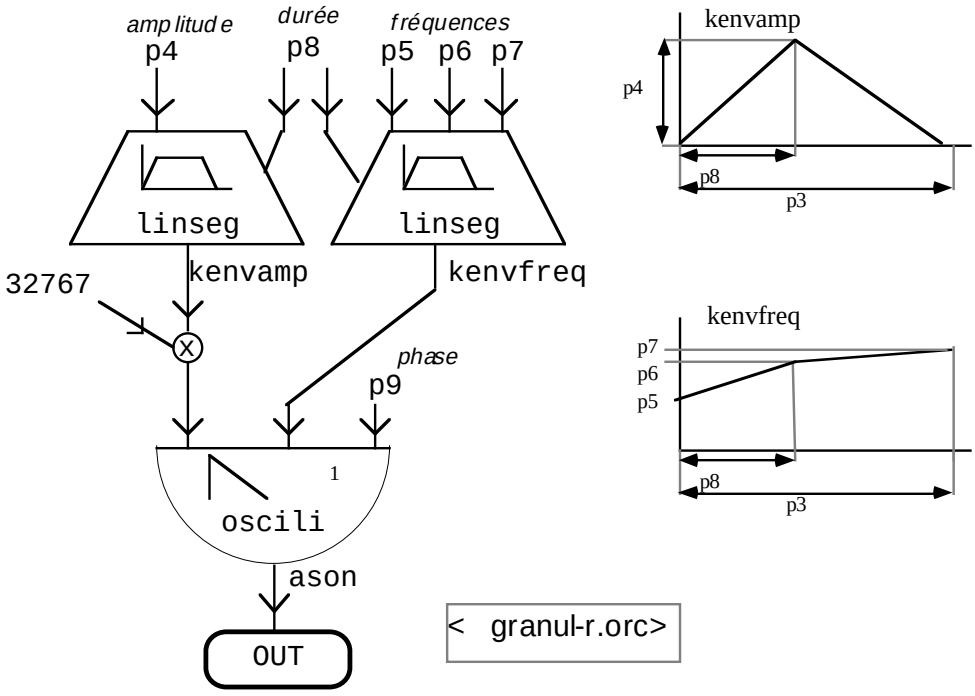


Schéma représentant l'instrument granul-r.orc pour Csound

```
;-------------------------------------------------------------
                    ;(ORCHESTRE) granul-j.orc


  ;p1 =ins
  ;p2 =date
  ;p3 = dur
  ;p4 = amp max
  ;p5 = frequence
  ;p6 = dur avant amp max
  ;
     sr = 44100
     kr = 44100
     ksmps = 1
     nchnls = 1
  ;
       instr 1

  kenvamp       linseg0, p6, p4, p3-p6, 0
  ason          oscili32000*kenvamp, p5,1
  out           ason
                endin
  ;-------------------------------------------------------------

  Score-type :

  ;-------------------------------------------------------------
  ; onde sinus
  f 1 0 1024 10 1
  ;  ins    dat     dur     amp    freq    att
  i   1     0       0.8     10     200     0.4
  e
  ;
```

Index

A

Additive 7, 11, 16, 24, 25, 40, 59, 68, 72, 79
addsyn-read 24
addsyn-write 25
add-tab.orc 83
amps 15
amps->db 57
amps-bpf 13, 55
analyse 11
aplatifreqs 37
apprt-Hzs 49
AudioSculpt 6, 11, 26, 61, 81

B

Barrière J.-B. 8
BPF 19, 55, 56, 57, 61, 72, 76, 77
bpf-red 19
bws 15

C

Chabot X. 2, 18
Chaîne de Markov 20
Chant 9, 21
chordseq 61, 77
Création 28, 62
csnd-write 68, 79, 80
Csound 9, 11, 19, 68, 77, 79, 83
Csound/Edit-sco 68
Csound/edit-sco 11
C-spdata 10, 11, 12, 17, 19, 47, 55, 58, 62
C-spdata-se 58
C-spdata-seq 11, 19

D

Décibel 57
distsp 36
Duthen J. 2

F

FFT 8
filter-spdata 15, 64
Filtrage 64, 65
  band-pass 15
  eq 15
  formantique 81
  high-pass 15
  low-pass 15
  neq 15
  reject-band 15
find-fund-data 47
find-funds-datas 18
flag 82
FOF 9
Formants 11, 81
freqs 15
freqs-bpf 13, 56

G

get-py/px 48
granul-j.orc 20, 77, 78, 86
granul-r.orc 20, 79, 80, 84, 85

I

ilter-spdata 33
Interface 22
Interpolation 17, 39, 68, 82
interpolw 17
intpol-model 17, 82
intpol-models 39
intpolw-models 40

L

Laurson M. 2
l-ecart 18, 52
Lemur 11, 21
llmod-read 22
llmod-write 23
l-moy 18, 51
Load-library 6
l-stat-modif 16, 35, 67

M

Malt M. 12
Mask 11, 20, 21, 26, 27, 61, 76, 78
mask-read 26
mask-write 27
matrix-response 18
matrixresponse 54
Max 18, 43, 45, 70
Midi 12, 21
mk-spdata 12, 28
mk-spdata-seq 12, 29
Modèles de Résonance 6, 8, 9, 11, 16, 23, 39, 50, 54, 60, 73, 74
Modèles de résonance 22
Modifications 64

P

par-spdata 13, 31, 60
par-spdata-seq 13, 32
partials 15
Peak 11, 20, 21
phases 15
Pottier L. 2, 12
print-spdata 13, 30
Programmation par objet 10
Puckette M. 9
pw-map 17
PW-Max 18

Q

qlist 18, 43, 71

R

resmoy-coef 18, 50
response 14, 53
Rodet X. 9, 20
Rueda C. 2

S

scale-spdata 16, 34, 65, 66
Score 11
sp-frames 58
sp-jitter3 41
sp-voice-vibra 42
Station d'Informatique Musicale de l'Ircam 9
Station Musicale de l'Ircam 70, 71
synthg 77, 78

T

Terhardt 6
text-win 72
Transformations 30

U

User-lib 6

V

Vercoe B. 9

W

weights 15
write-msg 18
write-msg-box 45
write-qlist 18, 43