Ircam documentation

- *Research reports*
- *Musical works*
- *Software*

OpenMusic

Om2Csound

Bibliothèque de modules de génération de scores pour Csound

version 1

Première édition, avril 1999

IRCAM Centre Georges Pompidou

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

OpenMusic a été conçu et programmé par Carlos Agon et Gerard Assayag.

La bibliothèque Om2Csound a été écrite par Karim Haddad, Mikhail Malt et Laurent Pottier.

Cette documentation correspond à la version 1.0 de la bibliothèque et à la version 2.0.1 ou ultérieure de OpenMusic.

Première édition de la documentation, avril1999.

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
Courrier électronique: bam@ircam.fr
http://www.ircam.fr/forumnet

Contenu

Introduction

La bibliothèque **Om2Csound** est un « portage » de la bibliothèque **Csound/Edit-sco**[1] de Patchwork pour l'environnement d'OpenMusic. Cette bibliothèque est destinée à préparer les « scores » de Csound. Elle permet la création de paramètres et de tables, graphiquement (avec les outils graphiques de OpenMusic comme les **Bpf**s), d'une manière algorithmique, à travers la notation traditionnelle, à partir de fichiers d'analyse ou tout simplement de fichiers Midi. Il est entendu que l'utilisateur possède des notions de base de Csound. Nous invitons le lecteur à consulter le manuel de Barry Vercoe[2] pour de plus amples renseignements sur Csound ; le manuel de Csond se trouve sur le CD-Rom de distribution du Forum, dans le dossier :
`Technologies:Csound docs & examples:CSound 68K & PPC.`

[1] **Csound/Edit-sco** est une bibliothèque pour la génération de « scores » de Csound écrite par Laurent Pottier et Mikhail Malt.

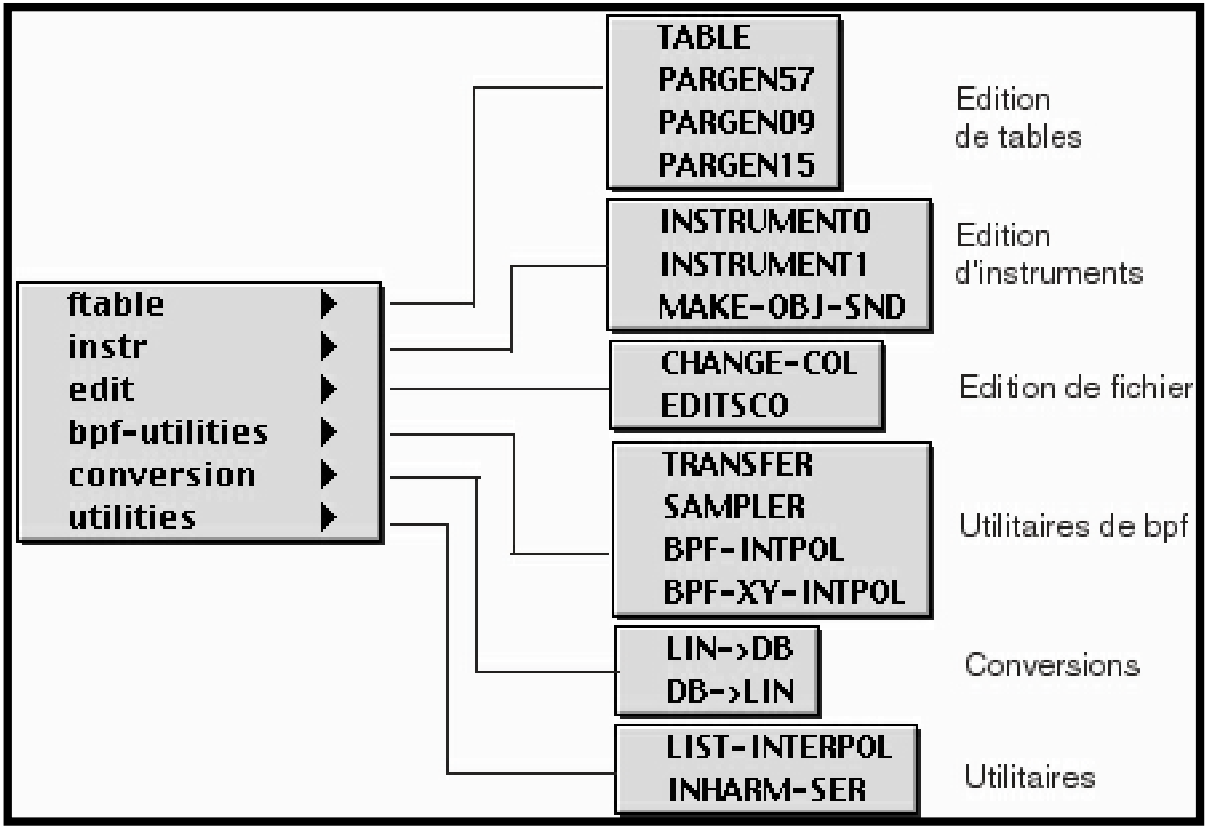
[2] **Csound - A Manual for the Audio Processing System and Supporting Programms with Tutorials**, Barry Vercoe, Media Lab, MIT.

Structure de la bibliothèque Om2Csound

Csound nécessite deux fichiers : « orchestra » et « score », le premier étant pour la description sous forme d'algorithme des instruments de synthèse, le second est une partition (une succession d'ordres) fournissant les paramètres nécessaires à l'exécution de la synthèse.

Dans cette livraison[3] de la bibliothèque, on trouvera la section **score** qui ne traite que le fichier « score ». Cette section est organisée en 6 sous-menus décrits plus bas.

[3] La partie « orchestra » est en cours de développement et sera disponible ultérieurement.


ftable
instr
edit
bpf-utilities
conversion
utilities
TABLE
PARGEN57
PARGEN09
PARGEN15
Edition
de tables
INSTRUMENT0
INSTRUMENT1
MAKE-OBJ-SND
Edition
d'instruments
CHANGE-COL
EDITSCO
Edition de fichier
TRANSFER
SAMPLER
BPF-INTPOL
BPF-XY-INTPOL
Utilitaires de bpf
LIN->DB
DB->LIN
Conversions
LIST-INTERPOL
INHARM-SER
Utilitaires

–

Structure générale du Score de Csound

Le fichier score présente trois parties majeures : les *tables*, les *notes* et la déclaration finale du score. La déclaration de chacune de ces parties est caractérisée par un *opcode* (un caractère réservé) : **f** pour les *tables,* **i** pour les *notes,* et **e** pour la fin du score.

Les tables

La déclaration de tables définie par "**f** " en début de ligne, est suivie par différents paramètres symbolisés par "**p**" et un numéro d'ordre. Les quatre premiers paramètres sont communs à toutes les GEN subroutines que l'on peut trouver dans Csound, les autres étant spécifiques à chacune d'elles.

f p1 p2 p3 p4…, pn

p1 numéro de table
p2 moment (onset) de la disponibilité de la table
p3 taille de la table
p4 numéro de la GEN subroutine
…
les autres paramètres dépendent de p4 (i.e., de la Gen utilisée).

Les notes

Les déclarations de notes commencent par la lettre "**i** " (comme instrument) et sont précédées de différents paramètres. Les trois premiers étant d'ordre général et doivent figurer dans tout score ; quant aux autres, ils dépendent de l'instrument auquel ils s'adressent.

i p1 p2 p3 …, pn

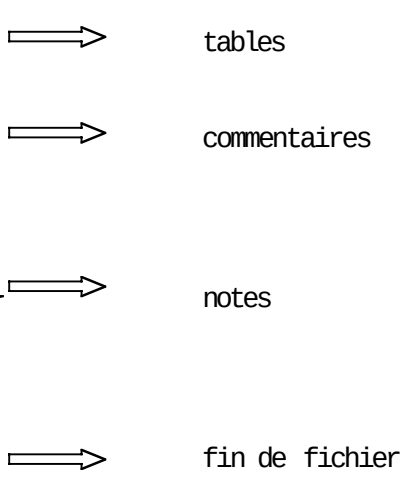
p1 numéro de l'instrument de l'orchestre
p2 temps d'attaque (onset)
p3 durées
…
autres paramètres.

La déclaration e

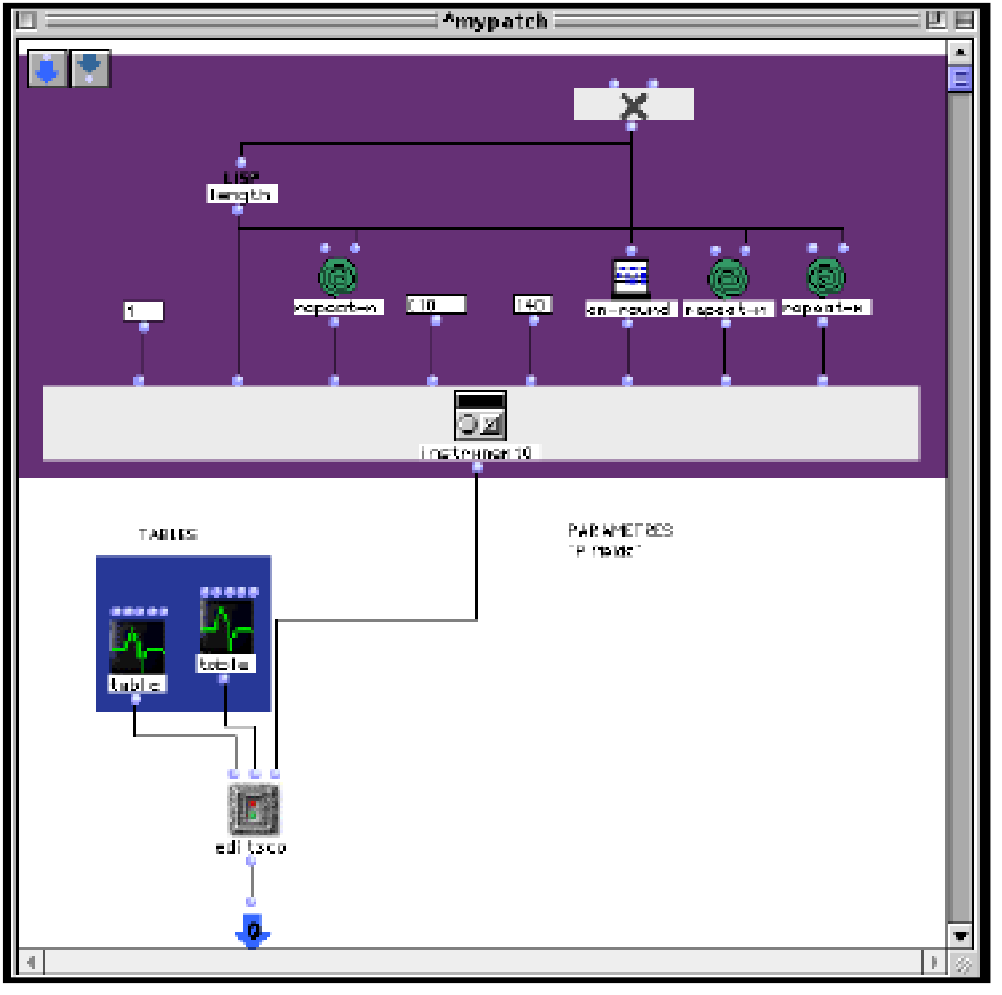
Elle indique la fin du fichier score.

Exemple d'un score

```
f1      0       512     9       1       1       0
f2      0       513     5       512     512     1

;p1     p2      p3      p4      p5      p6      p7

i1      0       10      4000    273     1       2
i1      0       7.5000  2000    455     1       2
i1      0       4.5000  2000    576     1       2
i1      0       6.5000  1500    648     1       2
i1      0       4       1501    864     1       2

e
```



Même exemple sous forme de patch :



Modules du menu Score

ftable

table

formatage de données pour les fonctions

entrées

table	(number)	numéro de table (p1)
ttab	(number)	temps en secondes pour déclencher la table (p2)
points	(number)	nombre de points dans la table (p3)
gen	(number)	numéro de la GEN subroutine (p4)
pargen	(list)	liste contenant les paramètres auxiliaires (p5, p6, p7,…) spécifiques à la GEN subroutine.

Le module **table** est utilisé pour générer les paramètres des fonctions de Csound. Ces tables sont des données destinées aux GEN subroutines. Selon la GEN utilisée, l'entrée *pargen* nécessite un format particulier (voir le manuel de Csound). Ce format peut aussi être entré via les modules « pargen » décrits plus loin. Ce module ne concerne donc que les déclarations de type « f ».

Ex :

```
OM->((f 1 0 4096 10 1 2))
```

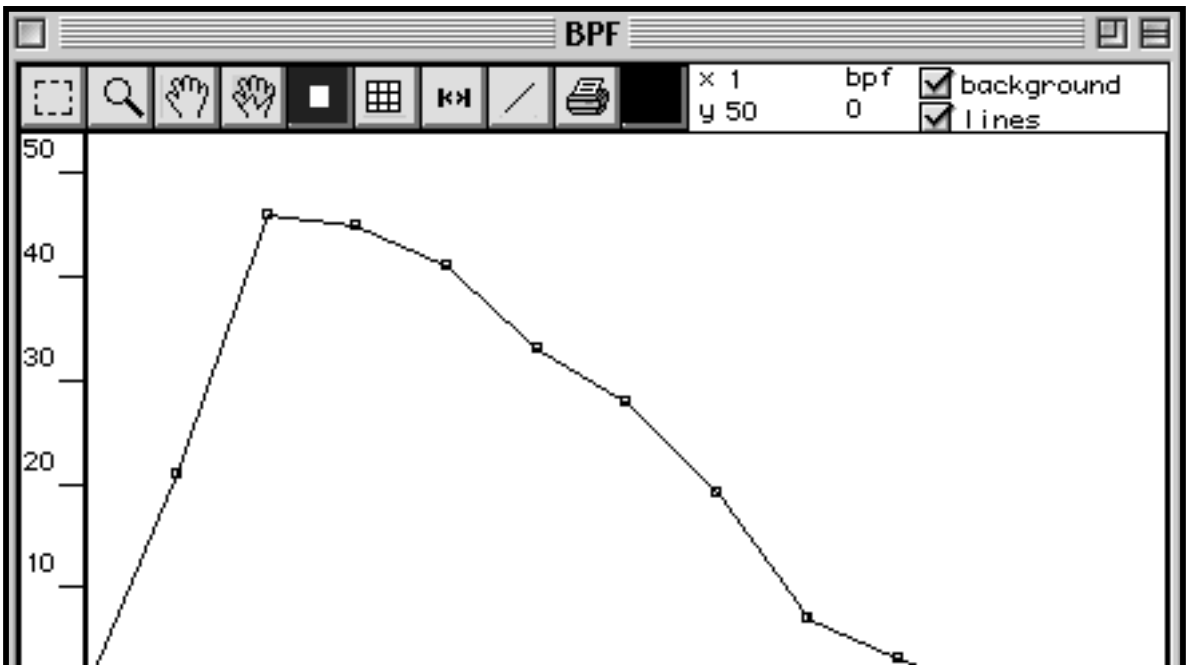
pargen57

formatage de données pour les GEN 5 et 7

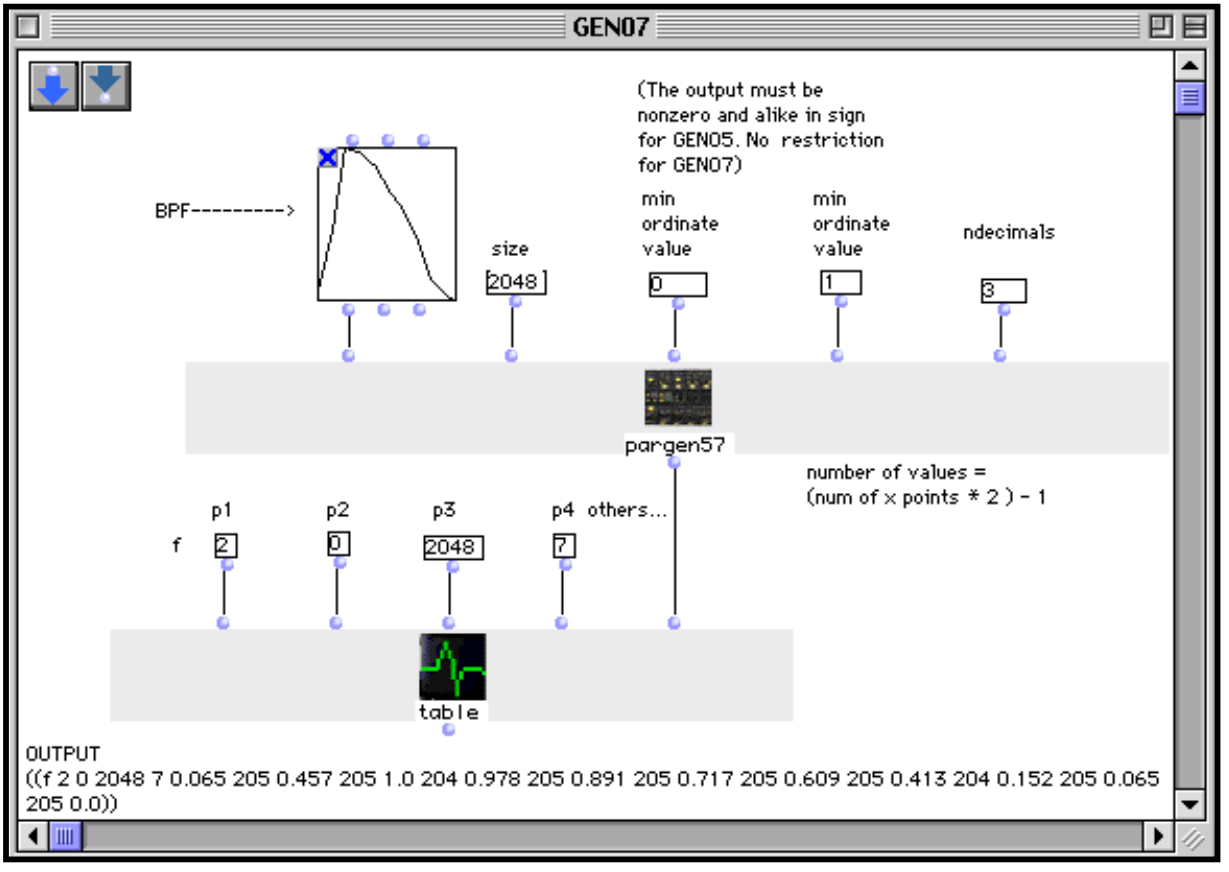
entrées

bpf	(bpf)	un objet bpf
points	(number)	nombre de points de la table (p3)
y-min	(number)	valeur minimale des y
y-max	(number)	valeur maximale des y
ndec	(number)	nombre des décimales

Ces subroutines construisent des segments de courbes exponentielles (GEN05) ou des lignes droites (GEN07). Utilisées en général pour les enveloppes d'amplitudes, nous allons voir un exemple d'enveloppe dessinée dans une bpf pour la GEN07.



Prenons une bpf et dessinons 9 points en s'assurant que les points du début et de la fin sont bien ancrés au point 0 des ordonnées.



Une fois l'éditeur bpf verrouillé, connectons-le au module pargen57. Choisissons la taille p3 de la table qui devrait être semblable à celui de la troisième entrée de l'objet **table.** Pour les valeurs des ordonnées <*ymin*> et <*ymax*>, nous laisserons celles par défaut (dans le cas où l'on utilisera la GEN05, il faudrait choisir entre les valeurs positives ou négatives tout en évitant le zéro qui est proscrit). On gardera aussi la valeur 3 indiquant le nombre de chiffres après la virgule.

Le module **pargen57** sera connecté à son tour au module **table** comme indiqué dans l'exemple ci-dessus. On n'oubliera pas de préciser la valeur p4 de la table qui indique quelle Gen subroutine on utilise.

Ce module peut être aussi utilisé pour la GEN08.

pargen09

formatage de données pour la GEN09

entrées

pn	(t)	proportions des harmoniques par rapport au premier
str	(t)	intensités relatives des harmoniques
phs	(t)	valeurs des phases
npart	(number)	nombre des décimales pour les nouveaux paramètres

La GEN09 construit une forme d'onde qui est le résultat de l'addition de plusieurs sinus. Les sinus peuvent être en rapport harmonique ou non, en phase ou pas, à la différence de la GEN10 qui ne peut avoir que des rapports harmoniques et en phase.

Le format de sortie du module est une liste comportant :

- le rapport harmonique *pn* (en nombre entier ou fractionnaire si ce rapport est inharmonique)
- le poids relatif des harmoniques *strn*.
- le rapport de phase exprimé en degrés ; pour une valeur de phase identique garder la valeur zéro par défaut.
-

Ces trois paramètres seront répétés pour chaque harmonique désirée.

(p1 str1 phs1 p2 str2 phs2… pn strn phsn)

(voir plus loin à propos du module **sampler** un exemple de patch pour la génération de 24 partiels pour ce module).

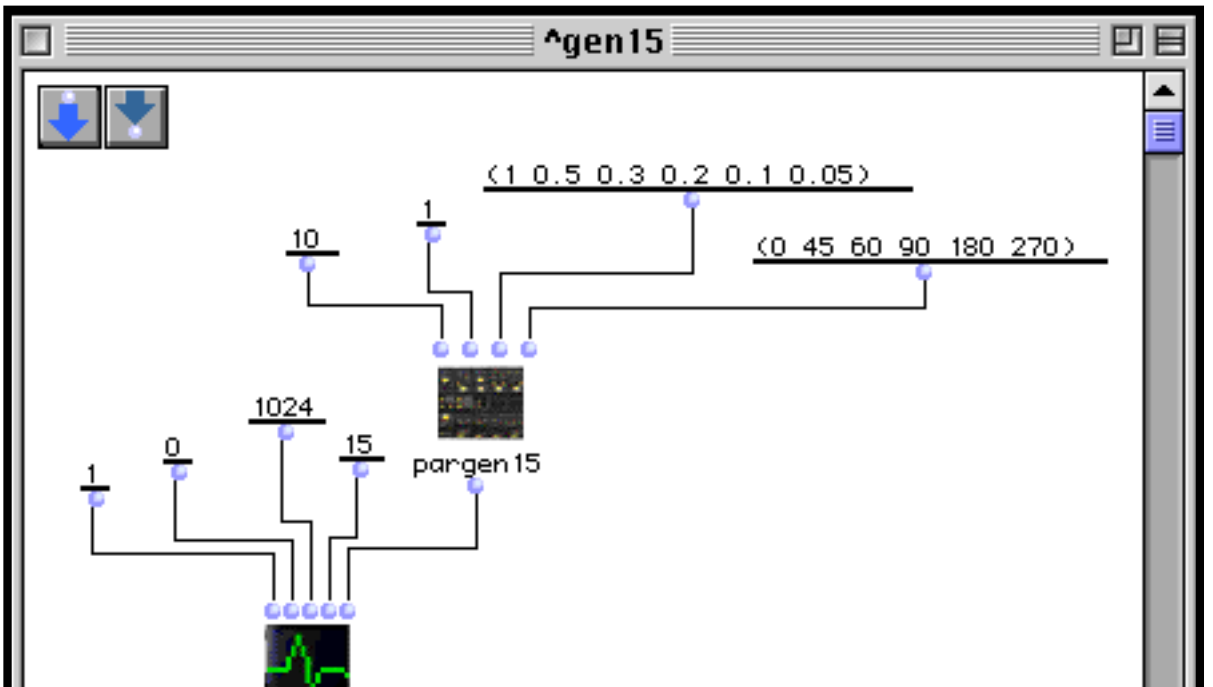
pargen15

formatage de données pour la GEN15

entrées

xint	(number)	limite inférieure de l'intervalle définissant le polynôme (-xint)
xamp	(number)	limite supérieure de l'intervalle définissant le polynôme (+xint)
ho	(list)	une liste d'intensités relatives des harmoniques
phs	(list)	une liste de phases

Le module **pargen15** est utilisé pour générer une table utilisant la GEN15 qui crée deux tables de polynômes.



instr

instrument0

formatage de données pour les notes

entrées

instr	(number)	numéro de l'instrument
onset	(list)	la date de chaque évènement (en secs.)
durs	(list)	les durées en secondes
add lst	(list)	paramètres supplémentaires (entrées extensibles)

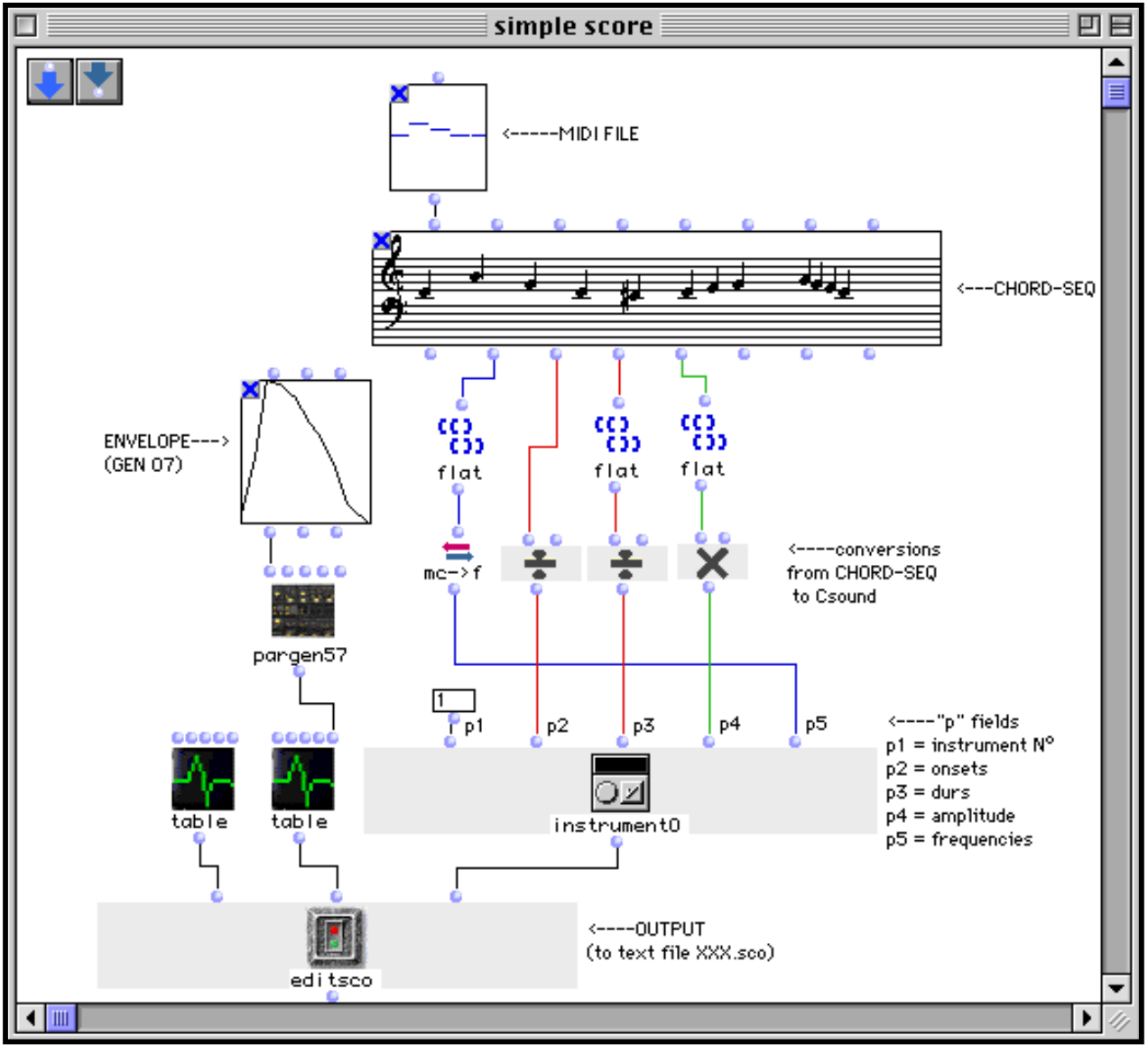
Le module **instrument0** est un module extensible qui génère un score pour être imprimé à partir de la boîte **editsco** et ne concerne que les déclarations de données de type « i ».

Dans l'exemple ci-dessous, nous avons pris à partir d'un fichier Midi une ligne mélodique (sans accords) que nous entrons dans une **chord-seq** par l'entrée <object>. Ainsi nous obtenons un objet contenant l'information nécessaire à la réalisation d'un score pour Csound : les hauteurs, exprimées en midicents, les onsets et les durées en millisecondes et la vélocité exprimée linéairement de 0 à 100.

La deuxième étape consiste à convertir ces données : les midicents en Hz, les onsets et les durées en secondes (les diviser par 1000).

On entre ces données converties dans le module **instrument0** selon l'ordre des paramètres, sachant que le premier paramètre p1 est réservé au nom de l'instrument, le deuxième paramètre p2 aux « onsets » et le troisième pramètre p3 aux durées, le reste étant relatif à l'instrument donné.

La sortie du module **instrument0** est reliée au module **editsco** qui formatera le texte pour Csound.



```
Score :
f1      0 4096 10 1
f2      0 2048 7  0.0650 205 0.4570 205 1.0000 204 0.9780 205 0.8910 205      0.7170
        205     0.6090 205      0.4130 204      0.1520 410      0.0000
;p1     p2      p3      p4      p5
i1      0.0000 1.0420 10000  293.6648
i1      1.0420 1.0460 10000  440.0000
i1      2.0880 1.0410 10000  349.2282
i1      3.1290 1.0460 10000  293.6648
i1      4.1750 1.0420 10000  277.1826
i1      5.2170 0.5210 10000  293.6648
i1      5.7380 0.5240 10000  329.6276
i1      6.2620 1.3050 10000  349.2282
i1      7.5670 0.2580 10000  391.9954
i1      7.8250 0.2630 10000  349.2282
i1      8.0880 0.2620 10000  329.6276
i1      8.3500 0.6620 10000  293.6648
e
```

instrument1

formatage de données pour les notes

entrées

instr	(number)	numéro de l'instrument
onset	(list)	la date de chaque événement (en secondes) si ce paramètre est un atome et non une liste, toutes les notes seront jouées simultanément
durs	(list)	les durées en secondes
p4	(list)	paramètre 4
p4	(list)	paramètre 5
others	(list)	paramètres supplémentaires donnés sous forme de liste ou de liste de liste

Comme le précédent, le module **instrument1** génère des données pour les déclarations de type « i ». On l'utilise dans le cas où les paramètres p4 et p5 sont constants et suivis par un grand nombre de paramètres qui seront entrés comme liste ou liste de liste dans l'entrée *others*. Ce module est utile notamment dans le cas de filtrage. Cependant, on peut toujours avoir recours au module **instrument0** qui est d'ordre plus général.

make-obj-snd

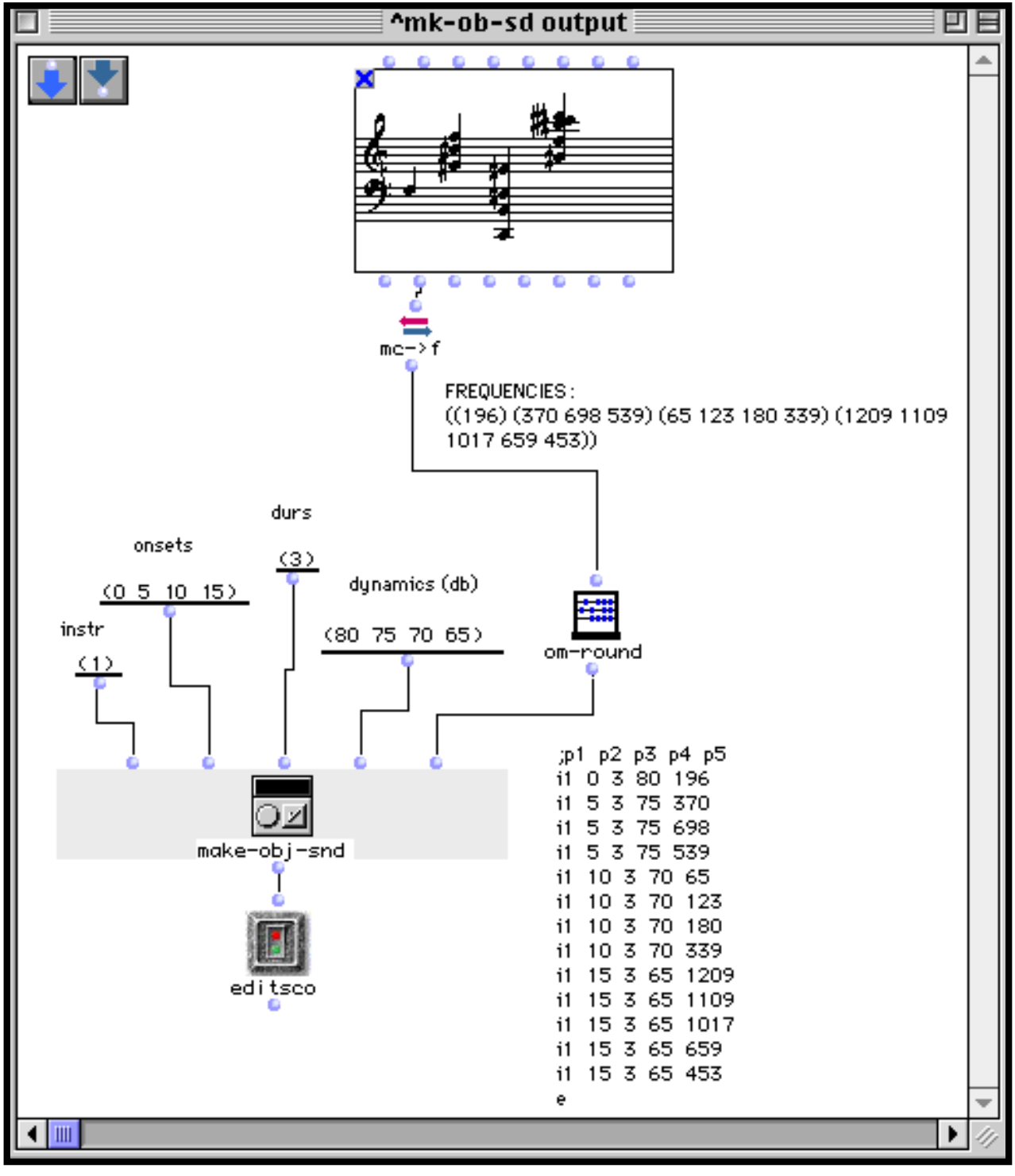
formatage de données pour les notes

entrées

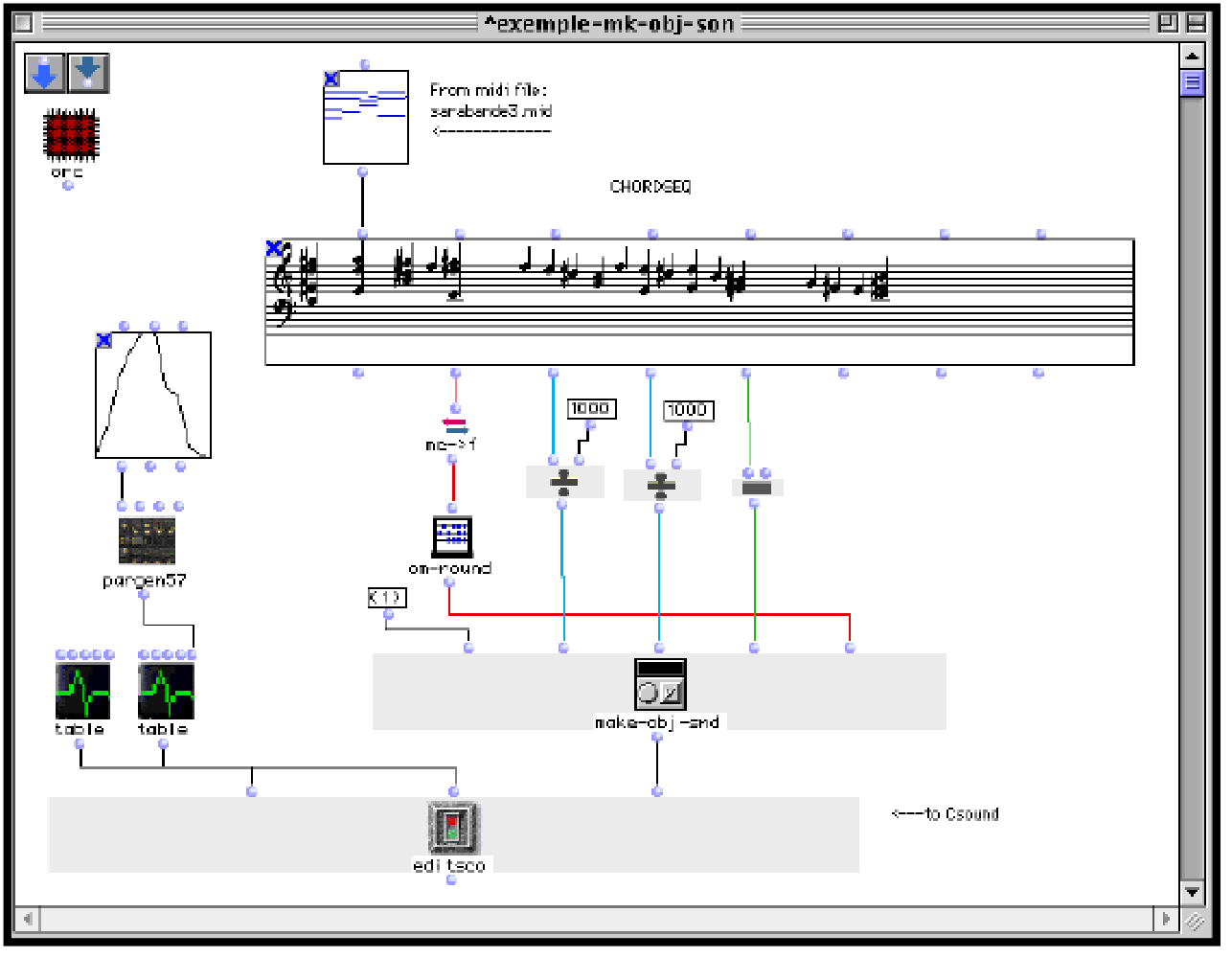
lins	(list)	numéro de l'instrument
lonsets	(list)	liste des onsets (en secondes)
ldurs	(list)	liste des durées (en secondes)
lp4	(list)	paramètre p4
lp5	(list)	paramètre p5 (réservé aux notes)
args	(list)	paramètres supplémentaires

make-obj-snd est un module extensible qui retourne un objet CLOS contenant toutes les informations nécessaires aux déclarations de type « i ». Ce module a été spécialement conçu pour éditer les notes et plus particulièrement les accords d'une manière similaire au module **chordseq**. Les notes seront saisies sous forme d'une liste de listes où les listes représentent des accords à n-sons. Cette liste devra être saisie dans <*lp5* > qui déterminera le format des autres paramètres. Si ceux-ci n'ont pas le même format, i.e. s'ils sont sous forme de liste simple, les valeurs contenues seront répétées selon la longueur de chaque liste de la liste entrée en <*lp5* >.

Afin d'illustrer ceci, prenons 4 accords de densités différentes (1 3 4 5) issus d'une **chord-seq**. Nous obtenons une liste de listes '((196) (370 698 539) (65 123 180 339) (1209 1109 1017 659 453)). L'entrée <*lonsets* > prend une liste de 4 valeurs '(0 5 10 15) pour la date des évènements. Dans <*ldurs* > on entre une valeur unique pour obtenir 4 accords de durée identique. <*lp4* > représentant les dynamiques, nous entrons une liste simple de quatre valeurs décroissantes '(80 75 70 65). Le « score » obtenu affiche des valeurs cohérentes pour chaque accord et cela par rapport aux données globales.


^mk-ob-sd output
mc->f
FREQUENCIES:
((196) (370 698 539) (65 123 180 339) (1209 1109 1017 659 453))
durs
(3)
onsets
(0 5 10 15)
dynamics (db)
(80 75 70 65)
instr
(1)
om-round
make-obj-snd
editsco
;p1 p2 p3 p4 p5
i1 0 3 80 196
i1 5 3 75 370
i1 5 3 75 698
i1 5 3 75 539
i1 10 3 70 65
i1 10 3 70 123
i1 10 3 70 180
i1 10 3 70 339
i1 15 3 65 1209
i1 15 3 65 1109
i1 15 3 65 1017
i1 15 3 65 659
i1 15 3 65 453
e

Dans l'exemple ci-dessous on a extrait à partir d'un fichier Midi les données concernant les notes et les durées en passant par un **chordseq** (notez la connexion entre les deux qui se fait à travers l'entrée <*object* > de **chordseq**). Les notes sont récupérées par la sortie <*lmidic* > du **chordseq** comme liste de liste en midicents :




```
OM->((5900 6600 7400 7800) (6400 7400 7900) (6900 7300 7800) (7600) (7800 6200
7400) (7600) (7400) (7300) (7100 6700) (7600) (7400 6400) (7300) (7400 6500)
(7100) (7000 6600) (6700) (6600) (6400) (6200 6600 7100))
```

qui seront convertis en Hz :

```
OM->((247 370 587 740) (330 587 784) (440 554 740) (659) (740 294 587) (659)
(587) (554) (494 392) (659) (587 330) (554) (587 349) (494) (466 370) (392)
(370) (330) (294 370 494))
```

Les accords seront sous le format d'une liste à n notes comme par exemple `(247 370 587 740)`. Les notes jouées seules seront aussi dans une liste à un élément seulement comme `(659)`. Ceci permet le formatage des autres paramètres comme celui des onsets (p2) qui représente une liste simple. Dans le cas où on entre une liste simple dans <*lmidic* > on obtiendra une succession de notes simples.

edit

Editsco

édition du score

entrées

args	(list)	prend en entrée les sorties des modules **instrument0, instrument1** et **table** **make-obj-snd.**
		Ce module est extensible. Il est possible aussi
		d'inclure des commentaires (;) ou autres « opcodes » comme (s), (a) ou (t), et ceci toujours sous forme de liste (entre parenthèses).

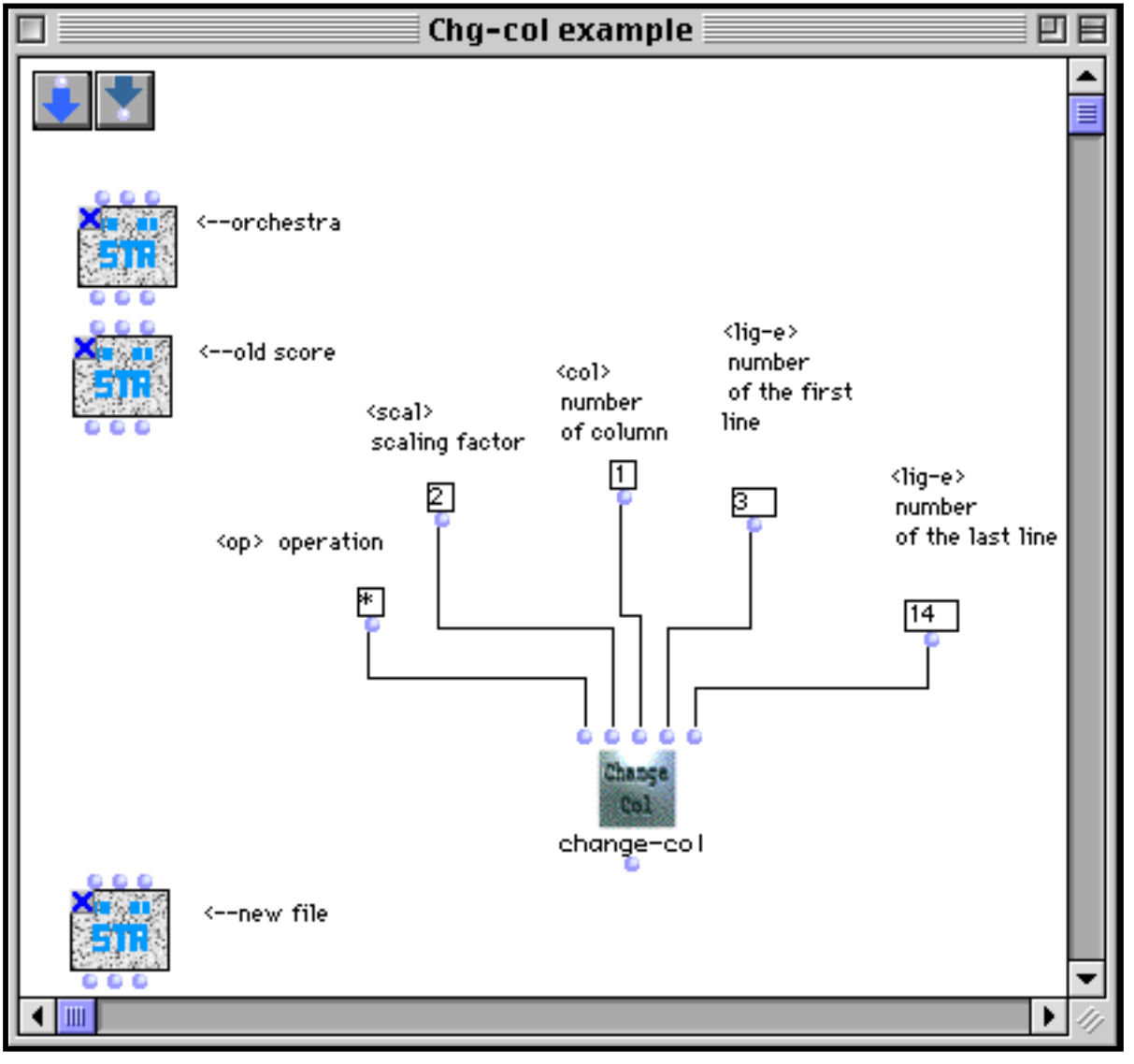
Change-col

modification du fichier texte

entrées

op	(symbol)	nature de l'opération arithmétique
scal	(number)	facteur
col	(number)	numéro de la colonne (1 = p2)
lig-b	(number)	numéro de la première ligne
lig-e	(number)	numéro de la dernière ligne

Dans cet exemple nous modifions le fichier score de l'exemple « simple score » en multipliant le temps d'attaque de chaque évènement (les onsets) par 2.



Remarquons qu'à l'entrée *<col >* la valeur 1 est assignée, étant donné que les onsets se trouvent dans le champs de paramètres de l'instrument en p2. A L'entrée *<lig-e >*, on a assigné la valeur 3 car nous commençons à modifier à partir de la ligne 3 du fichier, les deux premières lignes ne comportant que des tables.

Fichier « simple score.sco » :

```
f1    0     4096  10    1
f2    0     2048  7     0.0000      205   0.4570      205   1.0000
      204   0.9780      205   0.8910      205   0.7170      205
      0.6090      205
      0.4130      204   0.1520      410   0.0000

;p1   p2          p3          p4          p5

i1    0.0000      1.0420      80.0000     293.6648
i1    1.0420      1.0460      80.0000     440.0000
i1    2.0880      1.0410      80.0000     349.2282
i1    3.1290      1.0460      80.0000     293.6648
i1    4.1750      1.0420      80.0000     277.1826
i1    5.2170      0.5210      80.0000     293.6648
i1    5.7380      0.5240      80.0000     329.6276
i1    6.2620      1.3050      80.0000     349.2282
i1    7.5670      0.2580      80.0000     391.9954
i1    7.8250      0.2630      80.0000     349.2282
i1    8.0880      0.2620      80.0000     329.6276
i1    8.3500      0.6620      80.0000     293.6648
e
```

Le même fichier modifié :

```
f1    0     4096  10    1
f2    0     2048  7     0     205   0.457 205   1     204   0.978 205
      0.891 205   0.717 205   0.609 205   0.413 204   0.152 410   0


;p1   p2          p3          p4          p5
i1    0           1.042       80          293.6648
i1    2.084       1.046       80          440
i1    4.176       1.041       80          349.2282
i1    6.258       1.046       80          293.6648
i1    8.35        1.042       80          277.1826
i1    10.434      0.521       80          293.6648
i1    11.476      0.524       80          329.6276
i1    12.524      1.305       80          349.2282
i1    15.134      0.258       80          391.9954
i1    15.65       0.263       80          349.2282
i1    16.176      0.262       80          329.6276
i1    16.7        0.662       80          293.6648
e
```

bpf-utilities

transfer

fonction de transfert

entrées

bpf	(bpf)	prend une bpf en entrée
x-val	(t)	un nombre ou une liste de valeurs de x

Le module **transfer** retourne les valeurs des y correspondants à <*x-val* >

sampler

échantillonnage de bpf

entrées

bpf	(bpf)	prend en entrée une bpf
mode	(menu)	mode de mise à l'echelle (**xfact, max, sum**)
nsamp	(number)	nombre d'échantillons
xmin	(number)	valeur minimale de x
xmax	(number)	valeur maximale de x
oper	(number)	facteur de mise à l'échelle des y
ndec	(number)	nombre des décimales

menu

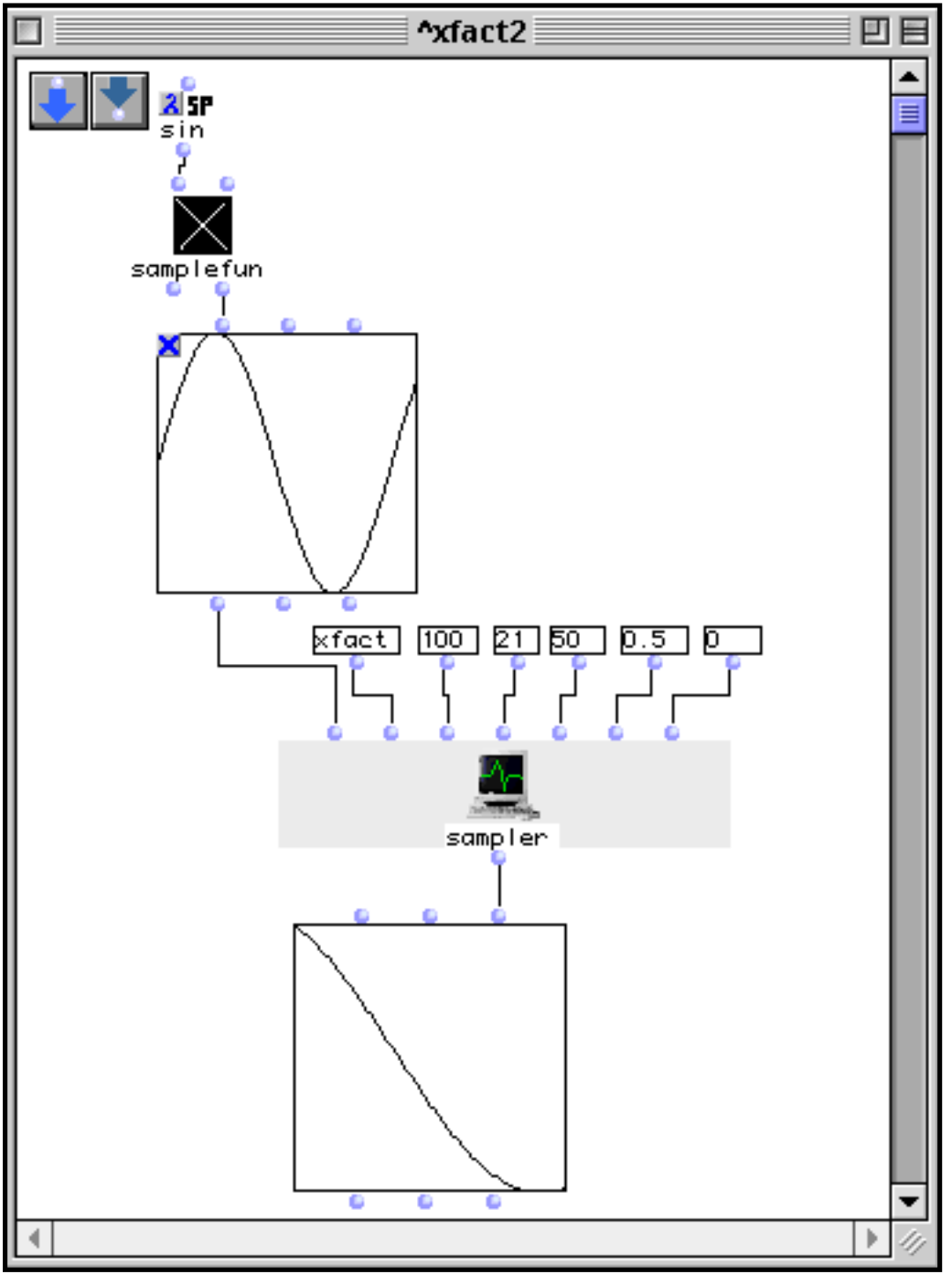
xfact	les échantillons y seront multipliés par la valeur <*oper*>
max	les échantillons y seront mis à l'échelle tel que la valeur maximale est égale à <*oper*>
sum	les échantillons y seront mis à l'échelle tel que leur somme est égale à <*oper*>

Ce module sert à échantillonner une **bpf.** Trois modes d'échantillonnage sont prévus.

xfact

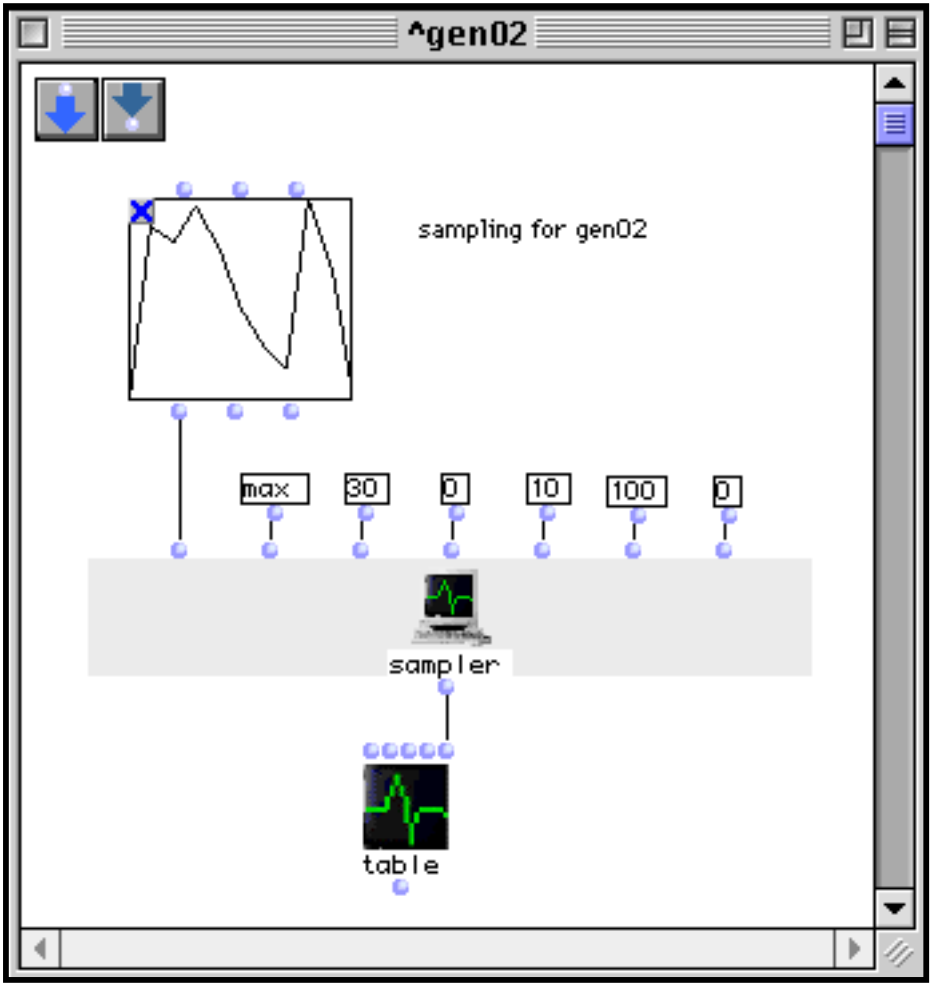
Quand le mode *xfact* est activé, l'échantillonnage s'effectue avec un facteur multipliant les y par <*oper* >.

Dans l'exemple ci-dessous nous avons généré un sinus en utilisant le module **sample-fun**. L'échantillonnage s'effectue sur 100 points allant de x=21 à x=50 avec un facteur multiplicatif de 0.5 réduisant les valeurs y de moitié.


^xfact2
sin
samplefun
xfact
100
21
50
0.5
0
sampler

max

Le mode *max* permet l'échantillonnage avec une mise à l'échelle des y sur un plafond (maximum) <*oper*>. Ceci est utile quand on veut charger des valeurs ne dépassant pas une certaine limite dans la GEN02 qui sert en général de réservoir de données numériques.



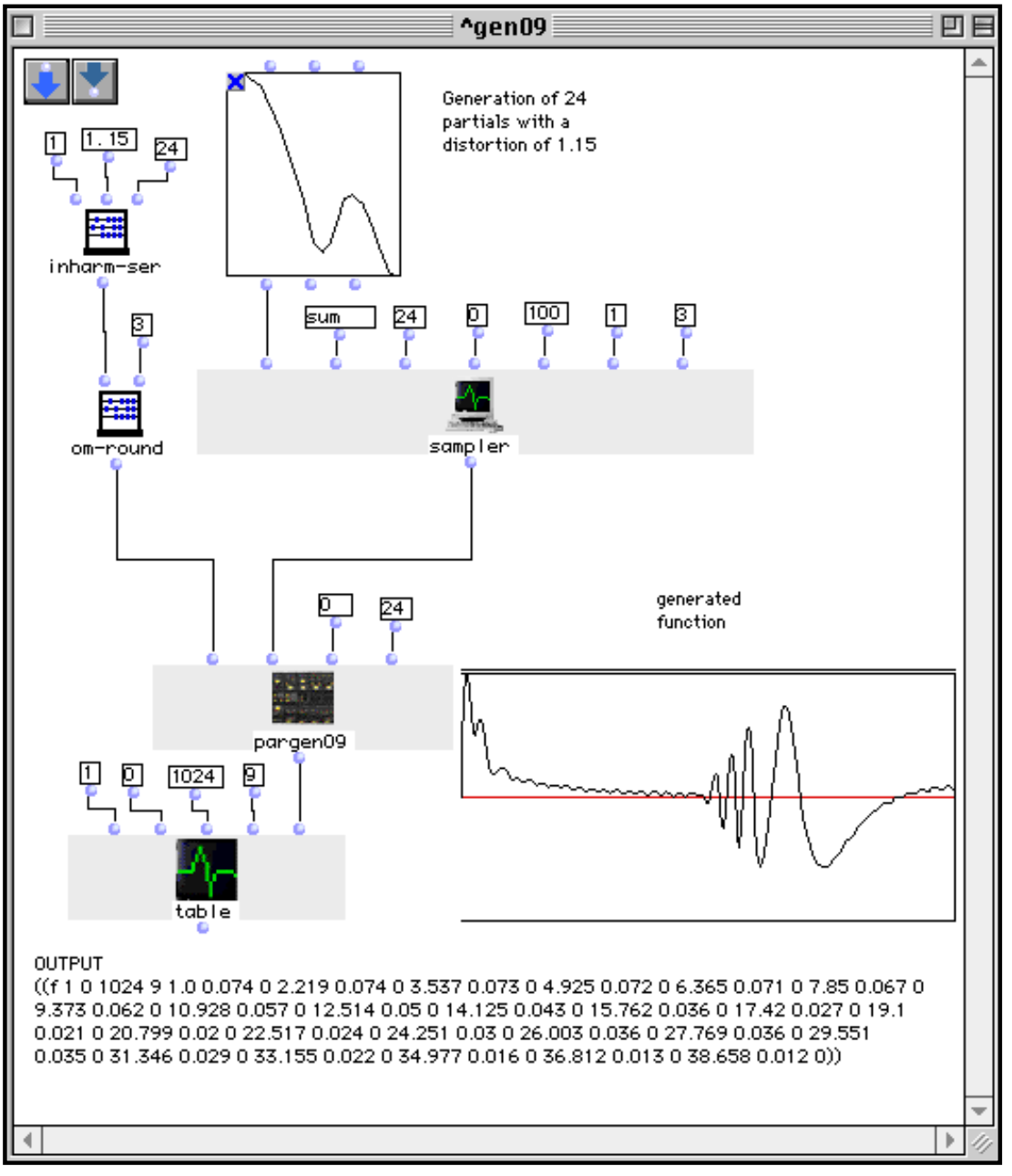
sum

Le mode *sum* est destiné à échantillonner les y tel que leurs somme soit égale à <*oper*>. Nous présentons un exemple d'échantillonnage pour la GEN09 :

Dans cet exemple, nous utiliserons deux modules pour spécifier d'une part le rang de 24 partiels et leur rapport qui est rendu inharmonique à l'aide du module **inharm-ser** décrit plus loin, et d'autre part, le rapport d'intensité relatif de ces partiels par une **bpf** (ces deux données sont destinées aux deux premières entrées de **pargen09)**.

Le module **sampler** est utilisé ici en mode *sum*. Dans l'entrée <*nsamp* > nous entrons 24 afin d'obtenir 24 échantillons. Dans les deux entrées qui suivent, <*xmin* > et <*xmax* > nous déterminons la section d'échantillonnage par rapport à l'axe des x, qui ici va de 0 à 100. Ces paramètres sont importants car ils dépendent directement des données de la **bpf**. Il faut s'assurer que les segments dessinés ou entrés numériquement couvrent bien cet espace sur l'axe des x afin d'éviter un échantillonnage de valeurs nulles.

Dans l'entrée <*oper*> nous avons mis la valeur 1 car nous voulons obtenir les proportions des intensités relatives qui sont représentés sur l'axe des y. Ces valeurs devant être des nombres fractionnaires nous entrons 3 dans <*ndec* > afin de réduire le nombre de chiffres après la virgule et de ne pas charger inutilement le fichier.


^gen09
Generation of 24
partials with a
distortion of 1.15
1
1.15
24
inharm-ser
3
om-round
sum
24
0
100
1
3
sampler
0
24
pargen09
generated
function
1
0
1024
9
table
OUTPUT
((f 1 0 1024 9 1.0 0.074 0 2.219 0.074 0 3.537 0.073 0 4.925 0.072 0 6.365 0.071 0 7.85 0.067 0
9.373 0.062 0 10.928 0.057 0 12.514 0.05 0 14.125 0.043 0 15.762 0.036 0 17.42 0.027 0 19.1
0.021 0 20.799 0.02 0 22.517 0.024 0 24.251 0.03 0 26.003 0.036 0 27.769 0.036 0 29.551
0.035 0 31.346 0.029 0 33.155 0.022 0 34.977 0.016 0 36.812 0.013 0 38.658 0.012 0))

bpf-interpol

interpolation de deux bpfs

entrées

bpfa	(bpf)	bpf 1
bpfb	(bpf)	bpf 2
nvals	(number)	nombre de résultats voulus (2 étant la valeur par défaut qui donne les 2 bpfs sans interpolation)
curve	(number)	courbe d'interpolation (0 = linéaire)

Le module **bpf-intpol** retourne une liste de listes de valeurs y qui est le résultat de l'interpolation de deux bpfs ayant les mêmes valeurs x.

bpf-yx-interpol

interpolation de deux bpfs

entrées

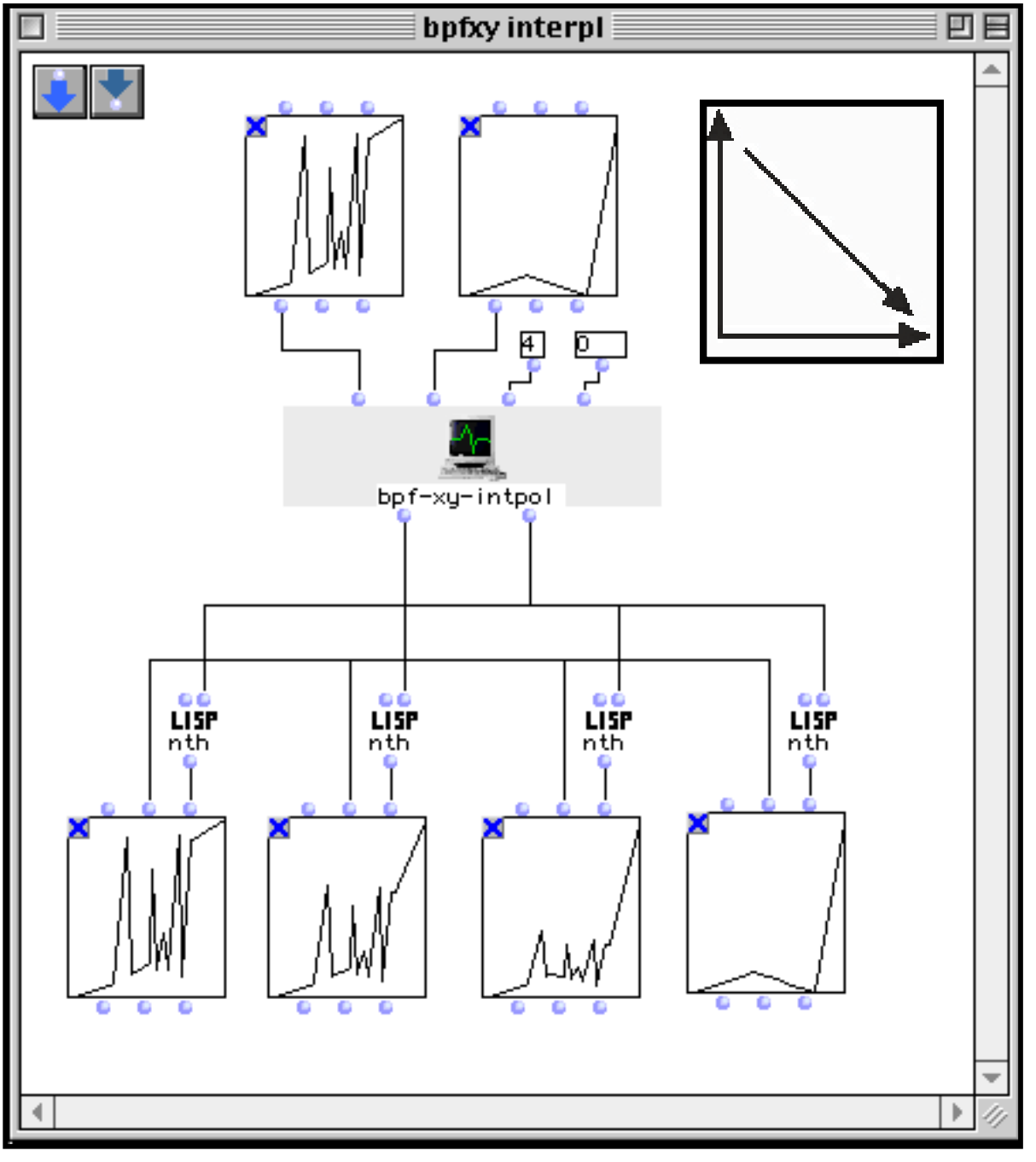
bpfa	(bpf)	bpf 1
bpfb	(bpf)	bpf 2
nvals	(number)	nombre de résultats voulus (2 étant la valeur par défaut qui donne les 2 bpfs sans interpolation)
curve	(number)	courbe

sorties

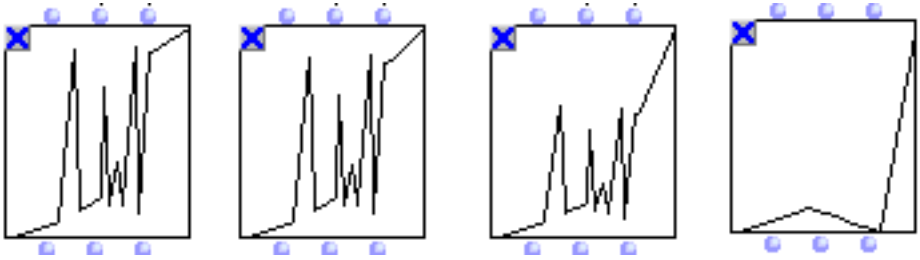
sortie1	(list)	donne une liste des x communs aux deux bpfs
sortie2	(list)	donne une liste de liste des y interpolés correspondants aux x de la première sortie

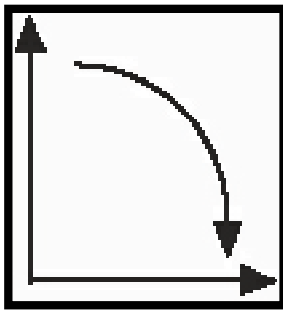
Le module **bpf-xy-intpol** retourne une liste de listes de valeurs y qui est le résultat de l'interpolation de deux bpfs ainsi qu'une liste de valeurs des x. A la différence du module **bpf-intpol,** ce module calcule une nouvelle liste des x résultant de cette interpolation. Ci-dessous un exemple d'interpolation avec différentes courbes d'interpolation.

Interpolation linéaire (curve = 0) :

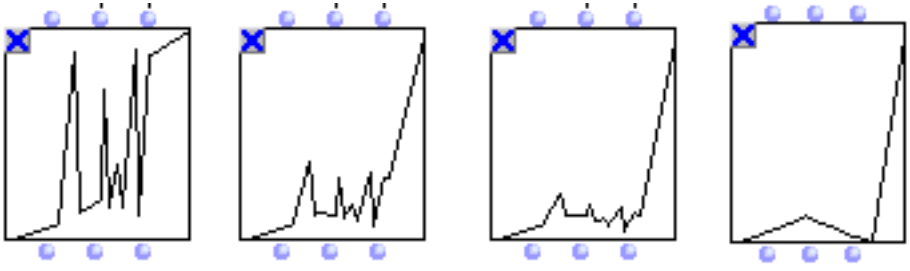


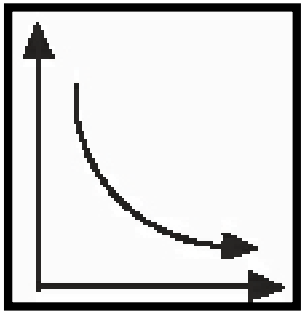
Courbe positive (curve = 1) :





Courbe négative (curve = -1) :





conversion

lin->db

conversion d’amplitude linéaires vers décibels

entrées

amps	(t)	nombre ou liste de valeur d’amplitude linèaire

db->lin

conversion d’amplitude - décibels vers linéaires

entrées

amps	(t)	nombre ou liste de valeur d’amplitude en décibels

utilities

list-interpol

interpolation de deux listes

entrées

list1	(list)	liste 1 (doit être une liste simple)
list2	(list)	liste 2 (doit être une liste simple)
nvals	(number)	nombre de résultats voulus (2 étant la valeur par défaut qui donne les 2 listes sans interpolation)
curve	(number)	courbe d'interpolation (0 = linéaire)

ex :

```
(om::list-interpol '(100 75 50 25) '(25 50 75 100) 6 0)
```

```
-   ((100.0 75.0 50.0 25.0) (85.0 70.0 55.0 40.0) (70.0 65.0 60.0 55.0)
(55.0 60.0 65.0 70.0) (40.0 55.0 70.0 85.0) (25.0 50.0 75.0 100.0))
```

inharm-ser

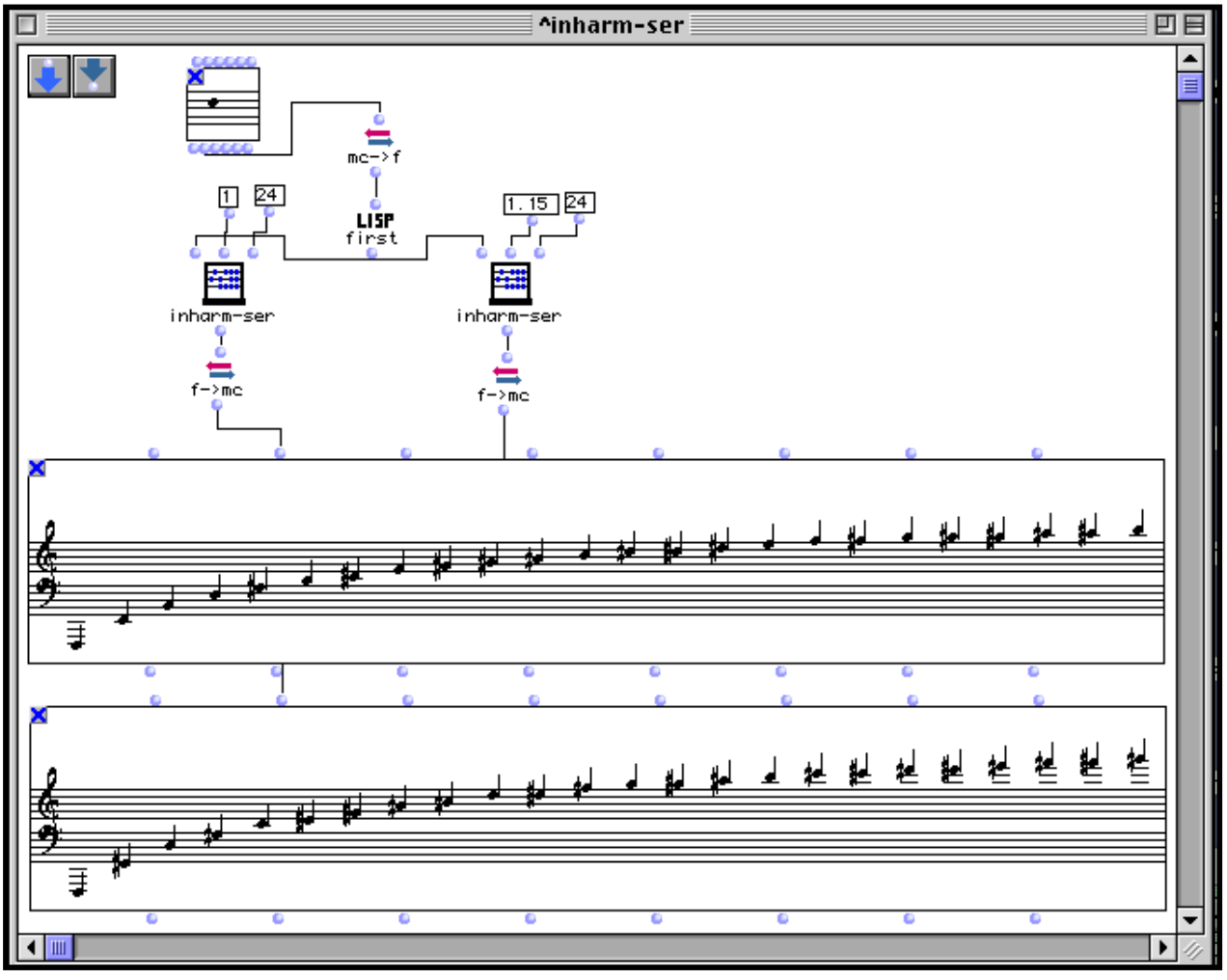
distorsion de série harmonique

entrées

begin	(number)	fréquence fondamentale
distortion	(number)	taux de distortion (1 = pas de distortion)
npartls	(number)	nombre de partiels voulu

Le module **inharm-ser** produit une liste de n partiels à partir de <*begin*> selon la formule partiel = begin * rang^dist".

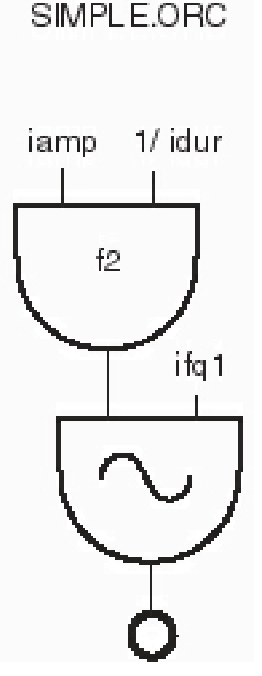
Dans l'exemple ci-dessous nous montrons une génération d'une série harmonique (*distortion* = 1) et d'une série inharmonique (*distortion* = 1.15) de 24 partiels.


^inharm-ser
mc->f
1
24
LISP
first
1.15
24
inharm-ser
inharm-ser
f->mc
f->mc

Annexe

L'orchestre « simple.orc » utilisé dans les exemples :

```
sr= 44100
kr= 441
ksmps= 100
nchnls= 1

instr 1
iamp = ampdb(p4)
idur = p3
ifq1 = p5
a2 oscili iamp,1/idur,2
a1 oscili a2,ifq1,1
   out a1
endin
```



Bibliographie

DE POLI, G. « A Tutorial on Digital Sound Synthesis Techniques. », *Computer Music Journal* 7(4), 1983. Reprint in C.Roads, ed.. *The Music Machine*. MIT Press,, 1989.

DODGE, C., et T.A. Jerse, *Computer Music: Synthesis, Composition, and Performance*, Schirmer Books, 1985.

GATHER, J.P, *Amsterdam catalogue of Csound computer Instruments*, Internet version-(exists in book version), © J.-P. Gather, 1995

MATHEWS, M.V., *The Technology of Computer Music*, MIT Press, 1969.

MOORE, F.R., *Elements of Computer Music*, Prentice-Hall, 1990.

RISSET, J.-C., *Introductory Catalogue of Computer Synthesized Sounds*,, Murray Hill, N.J.: Bell Telephone Laboratories. 1969.

SAMSON, P., « A General-Purpose Digital Synthesizer », Journal of the Audio Engineering Society 28(3), 1978, 106-113.

Index