Contrôle de l'objet *mTDelHarmo16~* [1]

Alain Bonardi

Description générale de l'objet mTDelHarmo16~

L'objet *mTDelHarmo16~* est un ensemble de 16 traitements, chacun d'entre eux incluant une ligne à retard et un harmonizer par effet doppler (approche temporelle grâce à un délai variable) avec une possibilité de réinjection vers n'importe quelle ligne. Il a été développé en langage Faust (cf. le code Faust mTDelHarmo16.dsp).

Le schéma général du traitement est le suivant (figure 1) :

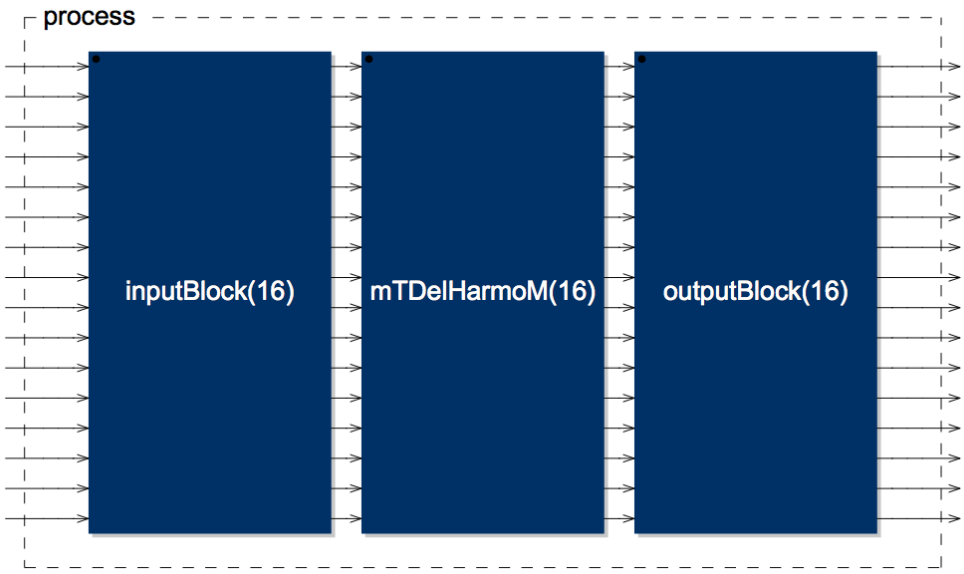

Figure 1. Schéma général de traitement de l'objet *mTDelHarmo16~*.

Les blocs *inputBlock* et *outputBlock* sont des ensembles de 16 gains linéaires permettant de varier les niveaux en entrée et en sortie. Nous mettrons l'accent sur la description du bloc *mTDelHarmoM*, dont la structure est indiquée sur le bloc-diagramme suivant (figure 2). Il est composé de seulement deux blocs : *fdToMatrixBlock* et *DelHarmoBlock*. Le premier (*fdToMatrixBlock*) prend en charge la réinjection, permettant d'envoyer la sortie de n'importe quelle ligne vers les entrées de n'importe quelle combinaison de lignes grâce à une matrice interne. Le second (*DelHarmoBlock*) gère les retards et harmonizers qui sont organisés en deux blocs séparés[2] (cf. figure 3).

[1] A partir de la version 1.2, le module est proposé avec 8, 12 ou 16 lignes de traitement. Les principes de fonctionnement sont exactement les mêmes, seul le nombre de lignes diffère. Les codes Faust sont les fichiers mTDelHarmo8.dsp, mTDelHarmo12.dsp, mTDelHarmo16.dsp. Les objets générés que ce soit pour Max ou PureData sont mTDelHarmo8~, mTDelHarmo12~, mTDelHarmo16~. De plus, toutes les lignes à retard sont maintenant limitées à un retard maximal de 21 secondes.

[2] Le bloc *DelBlock* comporte 16 lignes à retard, mais possède 32 entrées pour permettre la réinjection (chaque entrée est la somme du signal entrant et de la réinjection).

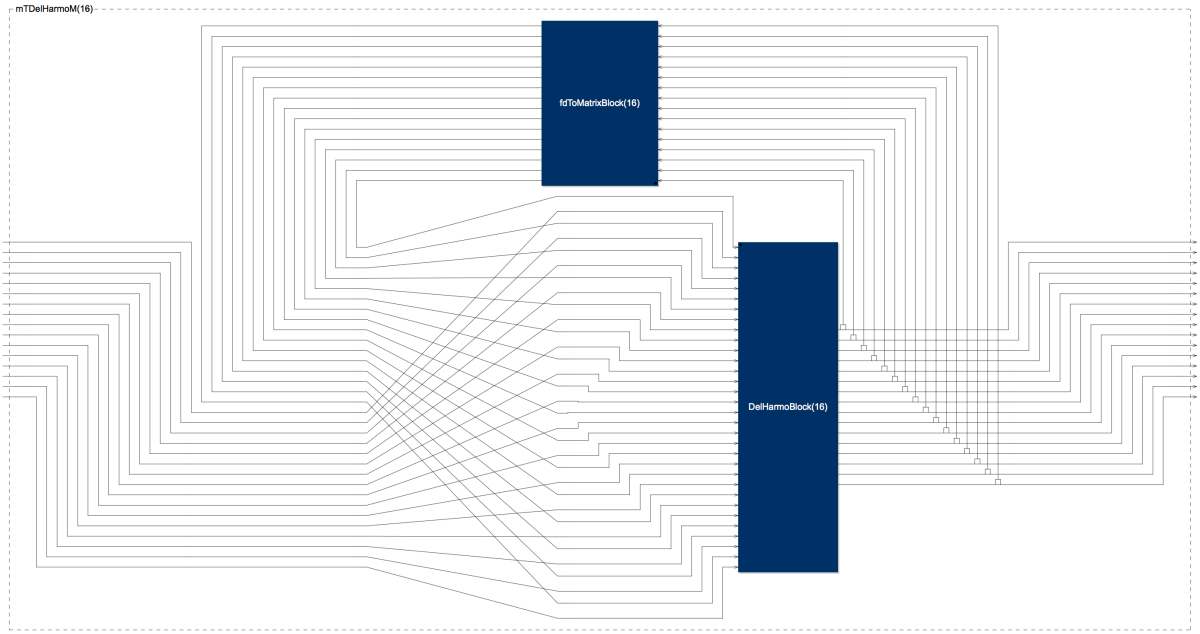

Figure 2. Le bloc-diagramme *mTDelHarmoM*.

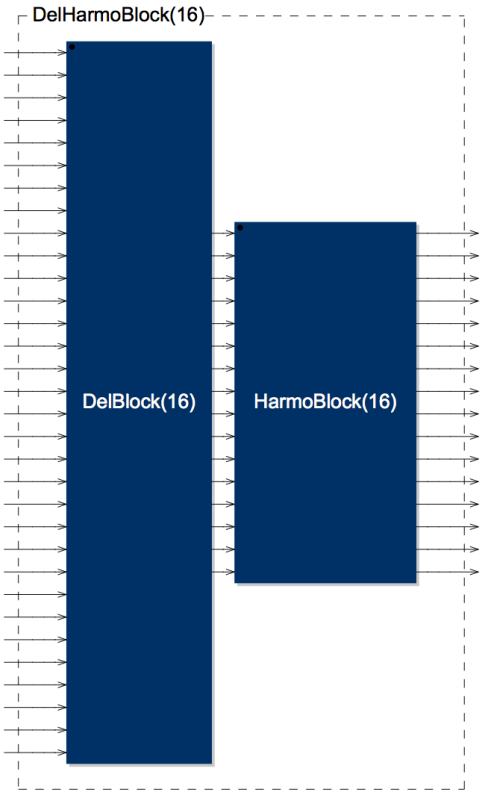

Figure 3. Les deux blocs composant le bloc *DelHarmoBlock* : *DelBlock* et *HarmoBlock*.

Chaque ligne à retard est un double délai avec recouvrement (*overlapping*) permettant de modifier la durée du délai sans clic (figure 4).

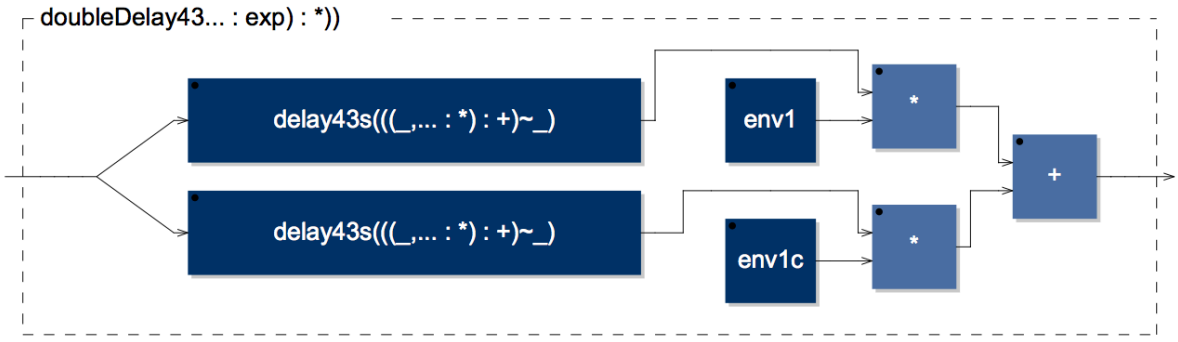

Figure 4. Une ligne à retard constituée à partir d'un double délai avec recouvrement.

Chaque harmonizer comporte quatre délais variables (principe de l'effet Doppler) avec recouvrement (figure 5).

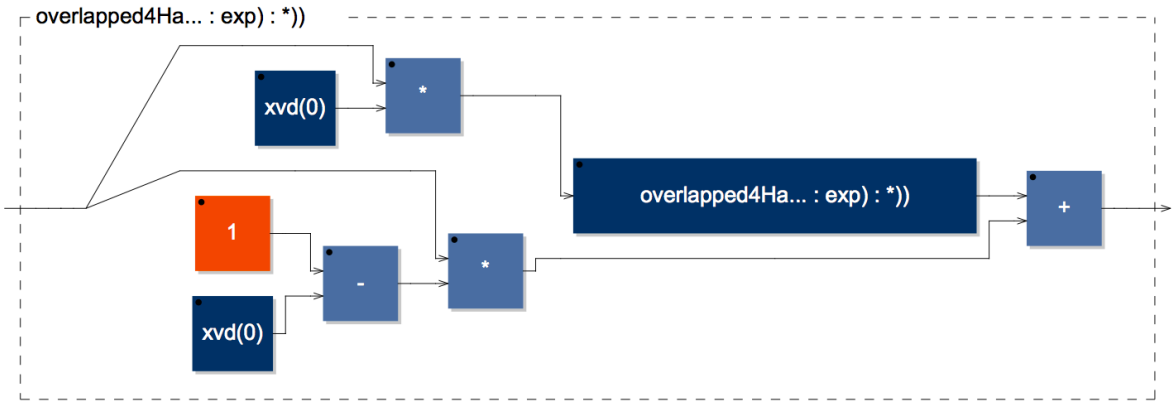

Figure 5. La structure des harmonizers fondée sur 4 délais variables avec recouvrement.

Pour chaque processus {ligne à retard + harmonizer}, nous avons les paramètres de contrôle suivants :

- la **durée** du délai en millisecondes ;
- la valeur de la **réinjection**, comprise entre 0 et 0.99 ;
- la valeur du paramètre **xvd** (effeX versus delay), qui établit la répartition entre l'harmonizer value (1) et le délai (0) ;
- la **transposition** en midicents (comprise entre -2400 et +2400 midicents) ;
- le gain linéaire en entrée (compris entre 0 et 1) ;
- le gain linéaire en sortie (compris entre 0 and 4).

De plus, il existe des contrôles globaux appliqués à l'ensemble des 16 lignes :

- le facteur d'**étirement des délais** : ce facteur multiplie toutes les durées de délai. S'il vaut 1, pas de changement. S'il vaut 2, les durées de délai sont multipliées par 2, etc. La valeur minimale est 0.01, la valeur maximale vaut 10 [3].
- la facteur d'**étirement des transpositions** : ce facteur multiplie toutes les transpositions (exprimées en midicents). S'il vaut 1, pas de changement. S'il vaut 2,

[3] Les durées des délais sont limitées à 21 secondes.

les transpositions sont multipliées par 2, etc. S'il vaut -1, les transpositions sont inversées. La valeur minimale est -10 et la valeur maximale 10 [4].

- la **durée d'interpolation** : il s'agit de la durée globale d'interpolation (en milli-secondes) entre les valeurs courantes des paramètres et de nouvelles valeurs. Cette durée est comprise entre 20 et 5000 milli-secondes.
- la **largeur de fenêtre d'harmonizer** : ce paramètre global contribue à transformer le timbre du son transposé. Le paramètre est un nombre décimal variant entre 0.0 et 127.0 Plus le son d'origine est grave, plus la valeur devrait être grande, et réciproquement. Mais les valeurs doivent être testées, permettant différents résultats allant de transpositions fidèles à d'autres altérant le timbre.

Enfin, la réinjection est contrôlée grâce à une matrice qui permet d'envoyer n'importe quelle sortie vers n'importe quelle combinaison d'entrées. Cette matrice interne comporte 16 x 16 = 256 toggles (chacun valant 1 ou 0).

Contrôle direct de l'objet mTDelHarmo16~ [Max et PureData]

La première et la plus simple manière de faire consiste à contrôler l'objet *mTDelHarmo16~* en envoyant des messages à sa première entrée. Ceci est possible en Max et en PureData.
1) Concernant l'environnement Max, le patch maxhelp de l'objet montre (dans un important message sur la droite), comment mettre à jour les valeurs de contrôle de l'objet *mTDelHarmo16~* (figure 6).

En Max, les messages utilisent les adresses suivantes :

- les durées sont indiquées par *d00*, *d01*, …, jusqu'à *d15*
- les valeurs de réinjection sont indiquées par *fd00*, *fd01*, … jusqu'à *fd15*
- les valeurs xvd sont indiquées par *xvd00*, *xvd01*, …, jusqu'à *xvd15*
- les valeurs des gains en entrée sont indiquées par *inp00*, *inp01*, …, jusqu'à *inp15*
- les valeurs des gains en sortie sont indiquées par *out00*, *out01*, …, jusqu'à *out15*
- l'étirement des délais est indiqué par *dStretch*
- l'étirement des transpositions est indiqué par *hStretch*
- la largeur de fenêtre d'harmonizer est indiquée par *hWin*
- la durée d'interpolation est indiquée par *smoothDuration*

Pour la matrice de réinjection, la convention est quelque peu différente. Les différentes cellules sont repérées par:

- sur la première ligne : *r000* *r001* … *r015*
- sur la seconde ligne : *r016* *r017* *r031*
- sur la troisième ligne : *r032* *r033* *r047*
- …
- sur la 16ème ligne : *r240* *r241* *r255*

[4] Les transpositions restent comprises entre -2400 et 2400 midicents.

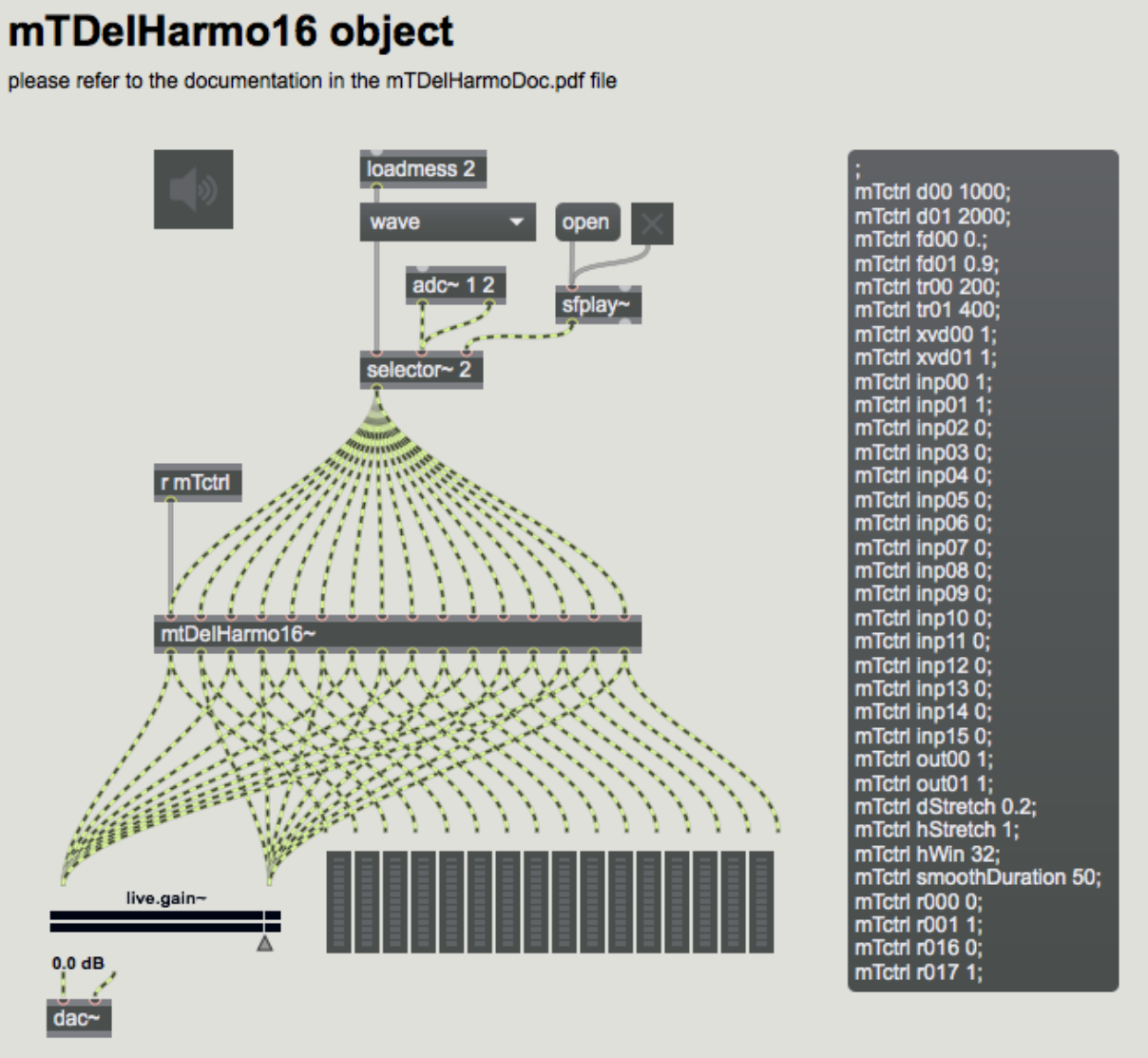

Figure 6. Le patch maxhelp de l'objet *mTDelHarmo16~*.

2) Avec le logiciel PureData, le système de messages fonctionne de la même manière mais les adresses utilisées sont différentes. La figure 7 montre le patch de help de l'objet *mTDelHarmo16~*.

En PureData, les messages utilisent les adresses suivantes:

- les durées sont indiquées par *d-0*, *d-1*, *d-2*, …, *d-9*, *d10* jusqu'à *d15*
- les valeurs de réinjection sont indiquées par *fd-0*, *fd-1*, …, *fd-9*, *fd10* jusqu'à *fd15*
- les valeurs xvd sont indiquées par *xvd-0*, *xvd-1*, …, *xvd-9*, *xvd10* jusqu'à *xvd15*
- les valeurs des gains en entrée sont indiquées par *inp-0*, *inp-1*, …, *inp-9*, *inp10* jusqu'à *inp15*
- les valeurs des gains en sortie sont indiquées par *out-0*, *out-1*, …, *out-9*, *out10* jusqu'à *out15*
- l'étirement des transpositions est indiqué par *dStretch*
- la largeur de fenêtre d'harmonizer est indiquée par *hStretch*
- la largeur de fenêtre d'harmonizer est indiquée par *hWin*
- la durée d'interpolation est indiquée par *smoothDuration*

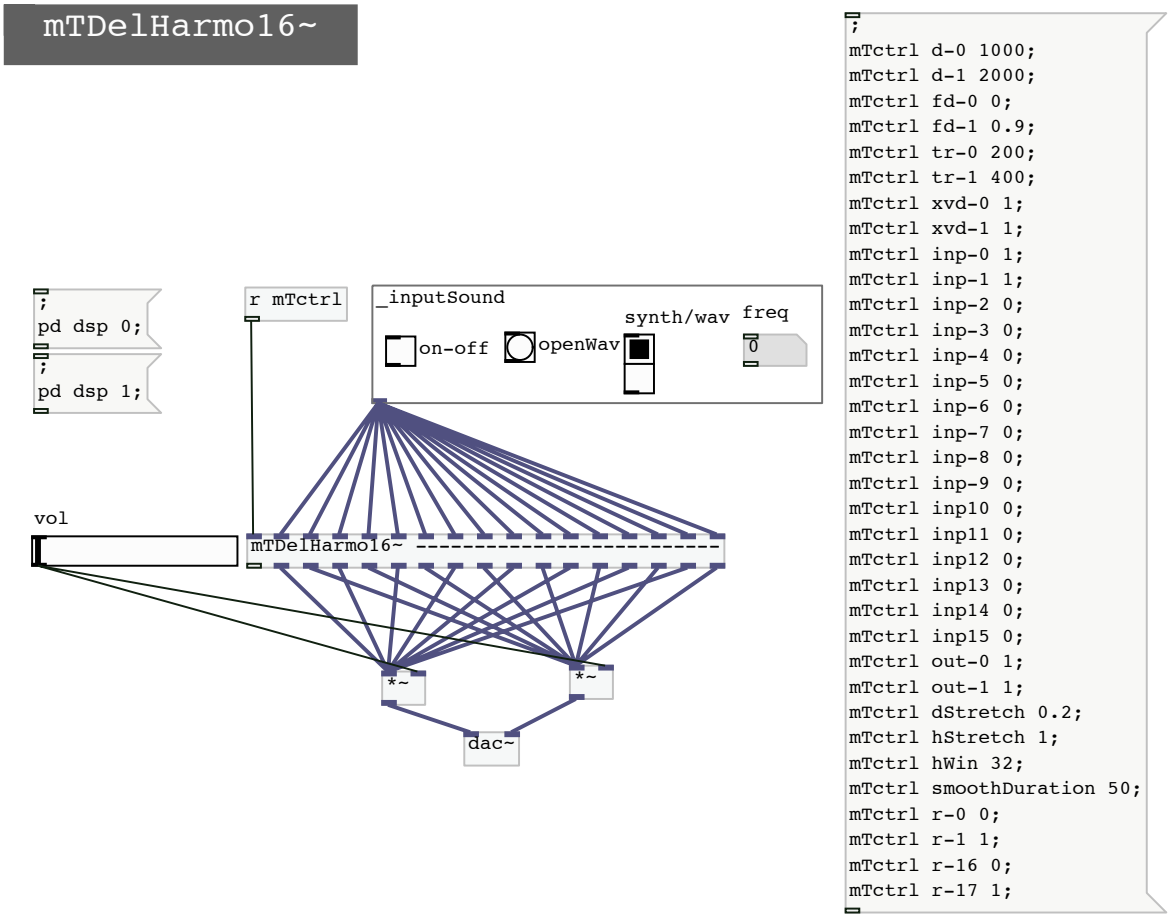

Figure 7. Le patch d'aide en PureData de l'objet *mTDelHarmo16~*.

Pour la matrice de réinjection, les différentes cellules sont indiquées par :

- sur la première ligne : *r-0* *r-1* … *r-15*
- sur la seconde ligne : *r-16* *r-17* *r-31*
- sur la troisième ligne : *r-32* *r-33* *r-47*
- …
- sur la 16ème ligne : *r240* *r241* *r255*

Pour les index 0 à 99: *r-0*, *r-1*, …, *r-99*
Pour les indices 100 à 255 : *r100*, *r101*, …, *r255*

Contrôle de l'objet *mTDelHarmo16~* via un programme Javascript [Max]

La deuxième manière de contrôler l'objet *mTDelHarmo16~* en Max consiste à utiliser un programme Javascript qui gère des messages de haut niveau qui sont plus faciles et concis à implémenter. Ce programme Javascript implémenté en Max est *mTDelHarmo16MessageHandler.js*. Il est encapsulé dans une abstraction *mTDelHarmo16Controller.maxpat* conçue pour être utilisée dans un bpatcher en Max (cf. figure 8).

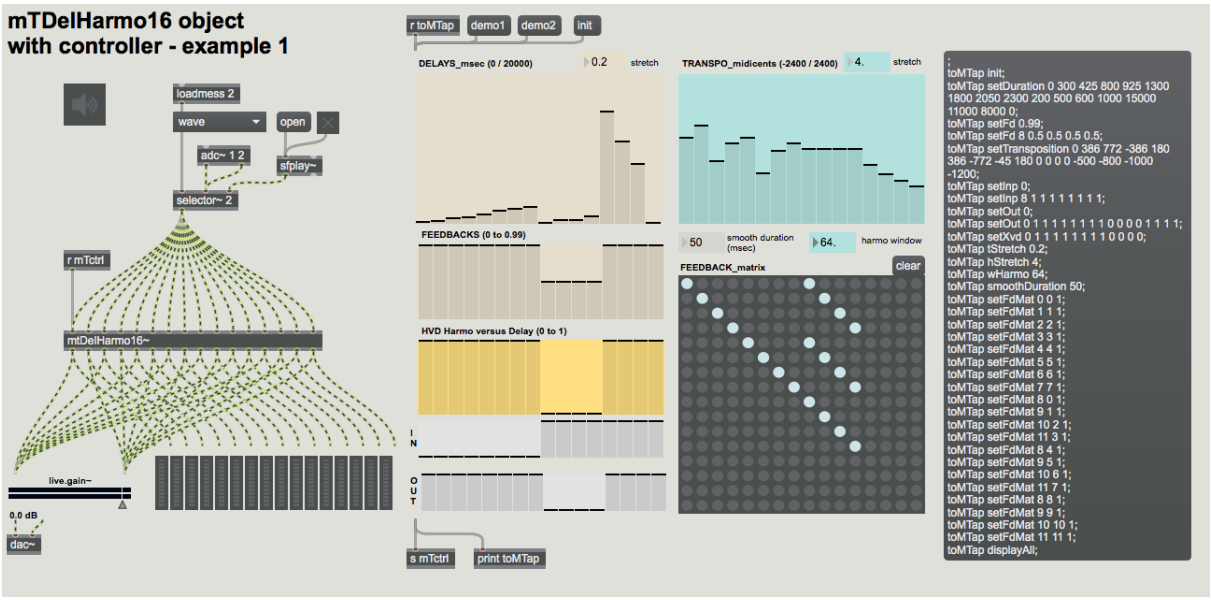

Figure 8. L'abstraction *mTDelHamo16Controller* encapsulée dans un bpatcher (centre de la figure).

Durées des délais

setDuration(list)

Ce message met à jour les valeurs des durées et les affiche sur l'interface.

S'il y a un seul argument, il s'agit de la valeur commune à toutes les durées.
S'il y a plusieurs arguments : le 1er élément de la liste est l'index de départ de la liste (i), les autres éléments de la liste sont les durées, d0, d1, etc.
La durée # i est alors mise à d0, la durée # (i+1) est mise à d1, etc.

Exemples

- avec un seul argument : setDuration(100) met toutes les durées à 100
- avec un index et une seule valeur : setDuration(1, 200) met dur[1] à 200
- avec un index et plusieurs valeurs : setDuration(3, 300, 100, 500) met dur[3] à 300, dur[4] à 100 et dur[5] à 500.

outputAllDurations()

Ce message envoie à la sortie 1 toutes les durées précédées par un message *mDelToDisplay*.

Valeurs des réinjections

setFd(list)

Ce message met à jour les valeurs des réinjections et les affiche sur l'interface.

S'il y a un seul argument, il s'agit de la valeur commune à toutes les réinjections.
S'il y a plusieurs arguments : le 1er élément de la liste est l'index de départ de la liste (i), les autres éléments de la liste sont les réinjections, fd0, fd1, etc.
La réinjection # i est alors mise à fd0, la réinjection # (i+1) est mise à fd1, etc.

Exemples

- avec un seul argument : setFd(0.6) met toutes les réinjections à 0.6
- avec un index et une seule valeur : setFd(1, 0.2) met fdbk[1] à 0.2
- avec un index et plusieurs valeurs : setFd(3, 0.3, 0.1, 0.5) met fdbk[3] à 0.3, fdbk[4] à 0.1 et fdbk[5] à 0.5.

outputAllFds()
Ce message envoie à la sortie 1 toutes les réinjections précédées par un message *mFdToDisplay*.

Valeurs des transpositions

setTransposition(list)
Ce message met à jour les valeurs des transpositions et les affiche sur l'interface.

S'il y a un seul argument, il s'agit de la valeur commune à toutes les transpositions.
S'il y a plusieurs arguments : le 1er élément de la liste est l'index de départ de la liste (i), les autres éléments de la liste sont les transpositions, tr0, tr1, etc.
La transposition # i est alors mise à tr0, la transposition # (i+1) est mise à tr1, etc.

Exemples

- avec un seul argument : setTransposition(250) met toutes les transpositions à 250
- avec un index et une seule valeur : setTransposition(1, 200) met tra[1] à 200
- avec un index et plusieurs valeurs : setTransposition(3, 300, 100, 500) met tra[3] à 300, tra[4] à 100 et tra[5] à 500.

outputAllTranspositions()
Ce message envoie à la sortie 1 toutes les transpositions précédées par un message *mTraToDisplay*.

Valeurs Xvd (Xvd = effeX versus delay)

setXvd(list)
Ce message met à jour les valeurs des xvd et les affiche sur l'interface.

S'il y a un seul argument, il s'agit de la valeur commune à toutes les valeurs xvd.
S'il y a plusieurs arguments : le 1er élément de la liste est l'index de départ de la liste (i), les autres éléments de la liste sont les valeurs xvd, xvd0, xvd1, etc.
La valeur xvd # i est alors mise à xvd0, la valeur xvd # (i+1) est mise à xvd1, etc.

Exemples

- avec un seul argument : setXvd(0.5) met toutes les valeurs des xvd à 0.5
- avec un index et une seule valeur : setXvd(1, 0.2) met xvd[1] à 0.2
- avec un index et plusieurs valeurs : setXvd(3, 0.3, 0.1, 0.5) met xvd[3] à 0.3, xvd[4] à 0.1 et xvd[5] à 0.5.

outputAllXvds()
Ce message envoie à la sortie 1 toutes les valeurs xvd précédées par un message *mXvdToDisplay*.

Input values

setInp(list)
Ce message met à jour les valeurs des gains en entrée (inputs) et les affiche sur l'interface.

S'il y a un seul argument, il s'agit de la valeur commune à tous les inputs.
S'il y a plusieurs arguments : le 1er élément de la liste est l'index de départ de la liste (i), les autres éléments de la liste sont les inputs, inp0, inp1, etc.
La valeur inp # i est alors mise à inp0, la valeur inp # (i+1) est mise à inp1, etc.

Exemples
- avec un seul argument : setInp(0.5) met tous les inputs à 0.5
- avec un index et une seule valeur : setInp(1, 0.2) met inp[1] à 0.2
- avec un index et plusieurs valeurs : setInp(3, 0.3, 0.1, 0.5) met inp[3] à 0.3, inp[4] à 0.1 et inp[5] à 0.5.

outputAllInps()
Ce message envoie à la sortie 1 toutes les valeurs d'inputs précédées par un message *mInpToDisplay*.

Output values

setOut(list)
Ce message met à jour les valeurs des gains en sortie (outputs) et les affiche sur l'interface.

S'il y a un seul argument, il s'agit de la valeur commune à tous les outputs.
S'il y a plusieurs arguments : le 1er élément de la liste est l'index de départ de la liste (i), les autres éléments de la liste sont les outputs, out0, out1, etc.
La valeur out # i est alors mise à out0, la valeur out # (i+1) est mise à out1, etc.

Exemples
- avec un seul argument : setOut(0.5) met toutes les sorties à 0.5
- avec un index et une seule valeur : setOut(1, 0.2) met out[1] à 0.2
- avec un index et plusieurs valeurs : setOut(3, 0.3, 0.1, 0.5) met out[3] à 0.3, out[4] à 0.1 et out[5] à 0.5.

outputAllOuts()
Ce message envoie à la sortie 1 toutes les valeurs d'outputs précédées par un message *mOutToDisplay*.

Paramètres globaux

tStretch(s)
Met l'étirement des durées à la valeur s.

hStretch(s)
Met l'étirement des transpositions à la valeur s.

wHarmo(x)
Met la largeur de la fenêtre d'harmonizer à la valeur x.

smoothDuration(x)
Met la durée d'interpolation à la valeur x.

Matrice de routing pour le feedback

setFdMatCell(x, y, val)
Met la valeur de la cellule de coordonnées (x, y) à la valeur val (mais n'envoie pas la valeur à l'affichage).

setFdMat(x, y, val)
Met la valeur de la cellule de coordonnées (x, y) à la valeur val et affiche la valeur val en utilisant un objet *matrixctrl* de Max.

setFdMatList(list)
Met à jour les valeurs de routing pour le feedback et les affiche sur l'interface.

S'il y a un seul argument, il s'agit de la valeur commune à toutes les valeurs de routing pour le feedback.
S'il y a plusieurs arguments, alors le nombre d'arguments doit être un multiple de 3, car ils sont donnés sous la forme de triplets avec les coordonnées (x, y) de la cellule et sa valeur.

Exemples

- avec un seul argument: setFdMatList(1) met toutes les cellules de la matrice à 1.
- avec des triplets d'arguments : setFdMatList(4, 5, 1, 3, 9, 1) met les 2 cellules de coordonnées (4, 5) et (3, 9) à 1.

Contrôle de l'objet mTDelHarmo16~ en utilisant une interface PureData

Le patch *mTDelHarmo16.pd* patch (figure 9) contient à la fois l'objet *mTDelHarmo16~* et l'interface pour contrôler l'objet avec des objets d'interface PureData (sliders, checkboxes, number boxes).

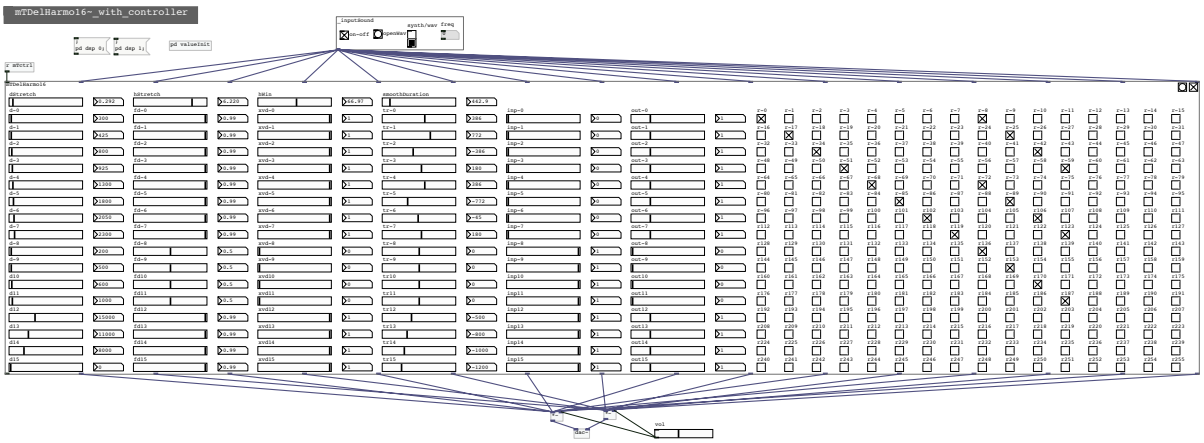

Figure 9. Le patch *mTDelHarmo16.pd*.

Au-dessus de chaque contrôleur figure son nom. Ces contrôleurs peuvent être adressés par des messages comme indiqué figure 10.

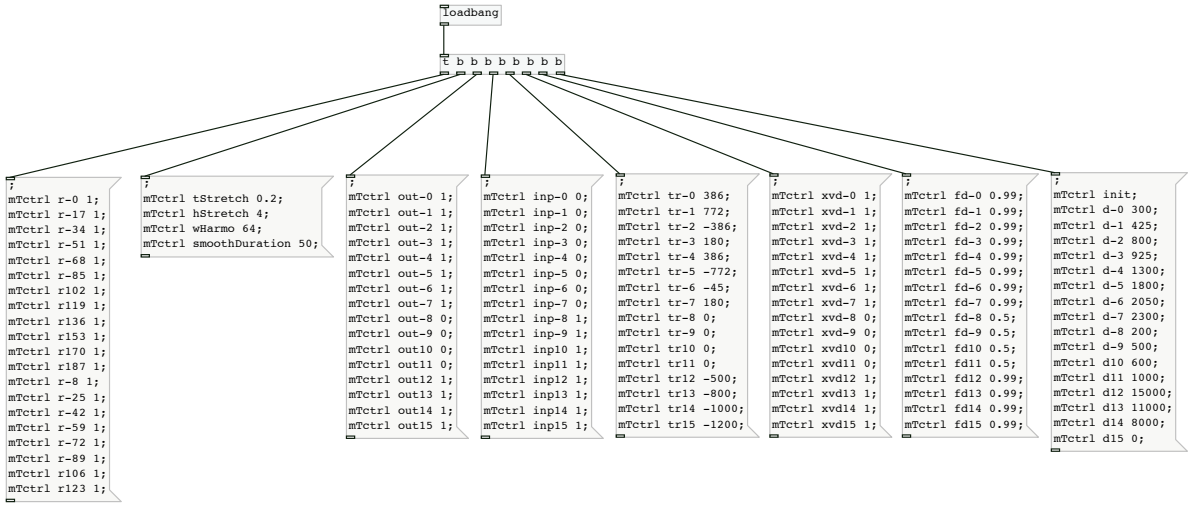

Figure 10. Exemple de messages contrôlant l'objet *mTDelHarmo16~* grâce à l'interface *mTDelHarmo16.pd*.