# Processament Audio a Temps Real

**Professors: Sergi Jordà i Helena Cuesta**

## Generació d'ones, filtres i envolupants

En aquesta document, expliquem pas a pas, diverses maneres d'obtenir formes d'ona diferents a la ona sinusoïdal, estudiarem envolupants i filtres i mirarem de fer un seqüenciador de passos molt bàsic.

### Exercici 1. Dent de serra

Hem vist l'objecte **`osc~`** que genera una ona sinusoïdal. Amb l'objecte **`phasor~`** podem crear ones dent de serra, però amb la peculiaritat de que en lloc de variar entre -1 i +1, ho fan entre 0 i 1. Adaptar l'objecte **`phasor~`** per obtenir un dent de serra entre -1 i +1.

**Tip**: multiplicar per 2 el senyal i restar 1, o bé restar 0,5 i multiplicar per 2.

### Exercici 2. Ona quadrada (mètode 1)

Quan multipliquem una ona sinusoïdal per una amplitud >1 és produirà distorsió. Fent servir l'objecte **`[clip~ -1 1]`** després de multiplicar i distorsionar la senyal, podrem "retallar la part sobrant", i quan més gran sigui el factor multiplicatiu, més quadrada serà la ona resultant. Construir el patch i visualitzar la ona resultant.

El fet d'utilitzar **`[clip~ -1 1]`** ens permetrà a més a més, mitjançant una nova multiplicació, aplicar un control d'amplitud posterior a aquesta distorsió. Feu-ho.

### Exercici 3. Ona quadrada (mètode 2)

Estudiar la documentació del objecte **`expr~`** i utilitzar-lo per construir una ona quadrada a partir de una ona sinusoïdal (podeu mirar també la documentació del manual de Pure Data FLOSS - inclòs en el Moodle de la assignatura - pp. 72-73).

Apliqueu aquest mètode per obtenir dues formes d'ona addicionals.

### Exercici 4. Ones varies a partir de síntesi additiva (Fourier)

Les ones que hem construït fins ara tenen infinits harmònics. Per evitar l'aliasing i obtenir ones més properes a les ones generades de forma analògica, podem aplicar el teorema de Fourier i utilitzar el missatge **`"sinesum"`** (que omple un array de mostres a partir de la amplitud de varis harmònics) i l'objecte **`tabosc4~`** (que reprodueix un array com si fos un cicle de un oscil·lador) que ens ofereix Pd.

L'objecte **`tabosc4~`** permet reproduir qualsevol array de mostres, interpretant aquest array com un únic cicle. Com a entrada **`tabosc4~`**, admet un control de freqüència de forma anàloga a **`osc~.`** La limitació de **`tabosc4~`** és que els arrays utilitzats han de tenir un tamany igual a "potencia de 2 + 3". Si utilitzem menys de 15 harmònics, 512 + 3 pot ser un tamany suficient.

Consultar l'ajuda de l'objecte **tabosc4~.**
Mirar el manual FLOSS pp. 85-88 per l'us de sinesum.
Consultar la Wikipedia, per obtenir els coeficients dels harmònics de una ona quadrada.

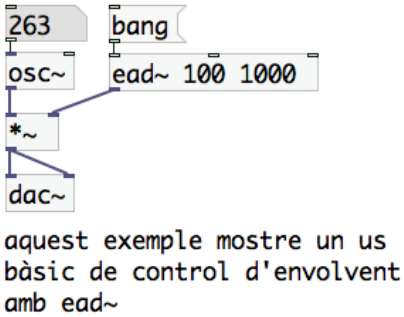
$$x_{\text{square}}(t) = \frac{4}{\pi}\sum_{k=1}^{\infty}\frac{\sin\left(2\pi(2k-1)ft\right)}{(2k-1)}$$
$$= \frac{4}{\pi}\left(\sin(2\pi ft) + \frac{1}{3}\sin(6\pi ft) + \frac{1}{5}\sin(10\pi ft) + \cdots\right)$$

Si la suma d'arguments fos superior a 1, després de crear la taula amb **"sinesum"**, l'haurem de normalitzar amb el missatge **"normalize"** *(; nom_taula normalize)*

## Exercici 5. Envolupants

Les ones que hem construït sonen indefinidament. Per fer sons d'amplitud variable en el temps, utilitzarem envolupants.

Hi han varis objectes de Pd que ens permeten treballar amb envolupants (**line~, vline~, Line~, eadsr~**, etc.), però si volem treballar amb sons percussius que tinguin sols *attack* i *decay*, podem utilitzar l'objecte **ead~** (*envelope with attack and decay*) que admet 2 paràmetres (*attack* i *decay* en ms).



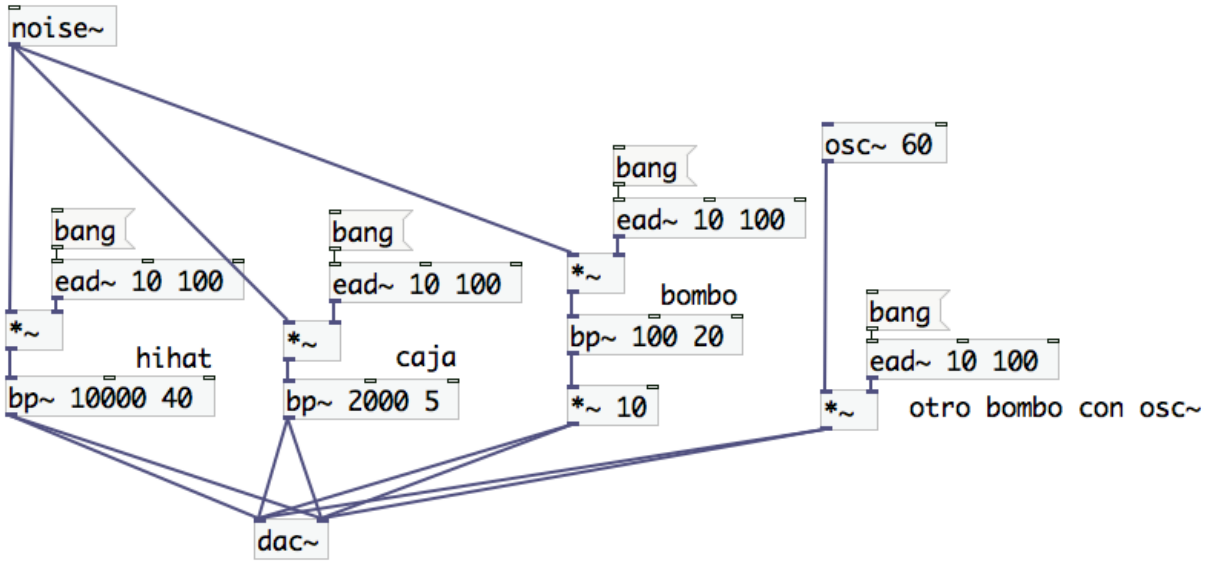

1. Visualitzar la forma de la envelopant, dibuixant la seva sortida en un array.
2. Crear un pulsació periòdica amb els generadors d'ones quadrades anteriors, i els objectes **ead~** i **metro.** Buscar unes freqüències (del oscil·lador) i unes envelopants que produeixen "bons sons de baix".
3. Fer el mateix amb l'oscil·lador de soroll blanc **noise~** per obtenir sons de percussió.

## Exercici 6. Filtres (1)

A Pd hi han varis objectes per filtrar sons. Els més senzills son:
**lop~** (passa baix), **hip~** (passa alt), **bp~** (passa banda).

En particular, **bp~** admet 2 arguments: freqüència de ressonància (en Hz) en la Q del filtre.

El següent patch mostra un exemple molt senzill per simular uns quant sons percussius bàsics, a partir de soroll blanc filtrat.




Experimentar amb diferents valors de **ead~** i de **bp~** per obtenir diferents sons de percussió.

Construir una senzilla caixa de ritmes de 8 passos, afegint:

- Un objecte **metro**
- Un comptador
- Un operador modulo (%)
- Un objecte **[sel 0 1 2 3 4 5 6 7]**

Per la primera part (comptador, etc ), podeu inspirar-vos del exercici 8.

Hi han moltes formes (i varies opcions de cara a l'usuari) de dissenyar aquesta caixa de ritmes; probablement la més senzilla sigui una caixa de ritmes monofònica (sols un so a la vegada), on per cada un dels passos, l'usuari pugui escollir un dels sons de percussió disponible o cap (ie. silenci). Si voleu fer-le més sofisticada, podeu fer-la polifònica, on l'usuari podrà escollir per cada pas, qualsevol combinació de sons de percussió (o cap).

## Exercici 7. Filtres (2)

Pd també inclou filtres més sofisticats. Per exemple **vcf~, moog~, reson~** o **resofilt~** permeten controlar la freqüència de ressonància amb una senyal d'àudio, el que pot resultar molt útil quan volem obtenir timbres que variïn en el temps.

Experimenteu amb diferents ones (quadrades, serra, etc.) i aquests diferents filtres, controlant també la freqüència de ressonància amb una envelopant (**ead~**) per obtenir sons de baix més electrònics.

## Exercici 8. Seqüenciador de baix

Estudiar i comentar l'exemple de la figura. A partir de l'exemple construir un seqüenciador de baix bàsic, aplicant algun dels filtres de l'exercici anterior. Estudiar la documentació de random, i afegir al patch un generador de seqüencies aleatori.