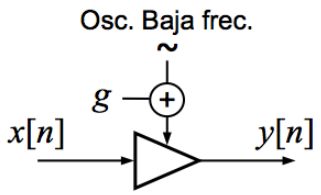
Práctica 7: Procesadores digitales del sonido

Esta práctica se realizará en dos sesiones presenciales y trabajo no presencial. En la primera sesión se trabajará en procesadores que modifican la dinámica y la altura. En esta segunda sesión estudiaremos procesadores que modifican tiempo, espacio y timbre. Los implementaremos en Csound siguiendo los esquemas vistos en teoría.

Procesadores de dinámica

Trémolo

En el trémolo, la envolvente de amplitud se modula con frecuencias de modulación entre 0,1 y 10 Hz. Para aplicar este efecto sobre un sonido en un fichero, se carga el sonido y se le aplica una modulación sinusoidal a su amplitud, multiplicando sus muestras por una ganancia $g[n]$ que varíe periódicamente. La señal de modulación, creada por un oscilador de baja frecuencia (*low frequency oscilator*, que se implementa con el operador **lfo**), de amplitud A_{mod}, se suma a una ganancia constante, g, de manera que $g[n]$ oscilará en baja frecuencia (menos de 10 veces por segundo) entre $g - A_{mod}$ y $g + A_{mod}$.



Por ejemplo, una ganancia g = 1, dejaría el sonido inalterado. Si se le aplica una modulación de amplitud A_{mod} = 0,3, con una frecuencia f_{mod} = 4 Hz, la ganancia variable $g[n]$ oscilaría entre 0,7 y 1,3 y lo haría 4 veces por segundo, produciendo el trémolo.

Compresores y limitadores

La compresión de la dinámica se suele especificar en dBFS, con el cero en el máximo, por lo que el umbral de compresión será negativo. Este umbral debe convertirse en valor PCM, para que esté en las mismas unidades que la señal. Para ello, Θ_C (en dBFS) debe convertirse en θ_c (PCM) $= A_{max} \cdot 10^{\left(\frac{\Theta_C}{20}\right)}$ (esto es lo que hace **ampdbfs(•)**). Para ficheros de 16 bits, la amplitud máxima es $A_{\max} = 2^{15} = 32768$ PCM.

Además, la compresión se debe hacer sobre los valores de la señal directamente, que están en PCM. Para ello $y[n] = x[n] \times g[n]$ con una ganancia que será

$$g[n] = \begin{cases} 1 & \text{si } \hat{x} \le \theta_C \\ \left(\frac{\hat{x}}{\theta_C}\right)^{\frac{1}{R_C}-1} & \text{si } \hat{x} > \theta_C \end{cases} .$$

En Csound $v = a^b$ puede escribirse de dos maneras:

```
v =    a^b
```

o bien `v pow a,b` (faltaría el prefijo de tipo)

La estimación de la amplitud de la señal, $\hat{x}$, se puede hacer de varias maneras. La más sencilla posible es usar el valor absoluto de la muestra a procesar: $\hat{x} = |x[n]|$, aunque veremos que da problemas, para los cuales se propondrá una solución.

Expansores y puertas

El concepto teórico de estos procesadores es idéntico al de los anteriores. La única diferencia en su implementación es que estos actúan si $\hat{x}$ es menor que el umbral y, en su operativa, que los umbrales de expansión Θ_E y de puerta de ruido Θ_P suelen ser de valores mucho menores que el de los compresores y limitadores.

Puerta controlada mediante envolvente RMS

Estimar la amplitud de la onda como $\hat{x} = |x[n]|$ es un criterio muy inestable. Por ejemplo, en el caso de la puerta de ruido, cada vez que la onda pasa cerca de cero, el procesador actúa y luego deja de actuar al alejarse del cero en cada ciclo. Y esto puede suceder cientos o miles de veces por segundo. Si $|x[n]| > \theta_p$ y $|x[n+1]| < \theta_p$ la muestra n quedará inalterada y la n+1 será puerteada (= 0). Esto crea ruidos inaceptables. Para suavizar la aplicación de la puerta, basta con cambiar el estimador de amplitud por la envolvente RMS aplicada en ventanas de M muestras: $\hat{x} = \text{RMS}(x[n+m]|_{m=0}^{M-1})$.

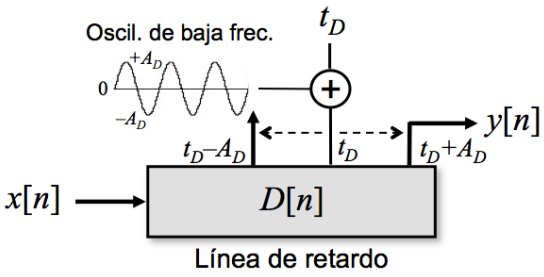
Ducking

El *ducking* consiste en un compresor cuya ganancia no está controlada por la amplitud de la señal a procesar sino por la de otra que se quiere superponer a ella, normalmente una voz que se superpone a una música o a un ambiente. Este efecto se usa continuamente en emisoras de radio y en sonorización de cine o vídeo. Se propondrá un ejercicio para ver este efecto, aplicando la tasa de compresión R_C a una señal cuando la envolvente RMS de una voz supere el umbral.

Procesadores de altura

Vibrato

El vibrato modula en baja frecuencia la altura. Para aplicar un vibrato a un sonido existente, se puede simular un efecto Doppler oscilante mediante una línea de retardo con un t_D que varíe según un oscilador de baja frecuencia. Cuando el t_D se va acortando es como si la señal "se acercara" al oyente (cada vez llega antes) lo que causa un aumento de su frecuencia y lo contrario cuando t_D aumenta. La f_{mod} (en Hz) controlará cuántas veces por segundo se produce la modulación y la amplitud A_D (en segundos) controlará la desviación respecto a la frecuencia sin alterar, por tanto, la "profundidad" del vibrato.



Para implementar este retardo variable en Csound se debe usar el operador **vdelay**, que es una línea de retardo variable que aplica a la señal de entrada un retardo especificado en ms. Su tercer parámetro es una constante que el operador necesita para reservar memoria para la línea de retardo, según el máximo tiempo de retardo posible, que es $t_D + A_D$ ms. Si el máximo retardo fueran 2 s, a 44.100 muestras/s necesitará reservar memoria para almacenar, como máximo, 88.200 muestras en la línea de retardo.

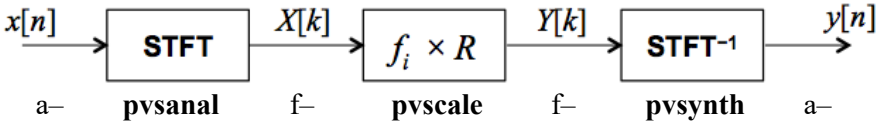
Transposición

Trabajaremos con la transposición tonal (subir o bajar la altura de una señal un número entero de semitonos). Hay diversas técnicas para transponer una altura en un intervalo dado. Lo vamos a implementar mediante la técnica del escalado de frecuencias en el espectro: leeremos la señal del fichero, calcularemos su espectrograma, desplazaremos sus frecuencias según el valor de una relación de intervalo y reconstruiremos la señal desde el espectro modificado.

Para operar en el dominio de las frecuencias, Csound nos ofrece un nuevo tipo de variables que son las que empiezan con una **f–**, que son espectrogramas, y que son creadas y utilizadas por una serie de operadores que comienzan con el prefijo **pvs-** (hay más de 40 de estos). En esta práctica vamos a usar tres de estos operadores:

- ✓ **pvsanal**: calcula un espectrograma **f–** de una señal mono con una ventana y posible solapamiento.
- ✓ **pvscale**: aplica un factor de escala a las frecuencias de un espectrograma **f–**.
- ✓ **pvsynth**: reconstruye una señal a partir de un espectrograma **f–**.

El esquema visto en teoría es directamente aplicable a estas tres fases del procesador:



De estos operadores, el primero calcula la transformada de Fourier a corto plazo (STFT) y necesita que se le indiquen sus parámetros de cálculo: tamaño, tipo de la ventana y el solapamiento:

```
fespectro  pvsanal  asenal, ifftsize, isolapa, ivensize, iventipo
```

donde `ifftsize` – Tamaño de la transformada de Fourier en muestras. Mejor que sea potencia de 2.
`isolapa` – Salto de la ventana en muestras para el cálculo del espectrograma, p.ej. *ifftsize*/4.
`ivensize` – Longitud de la ventana en muestras. Normalmente = *ifftsize*.
`iventipo` – Forma de la ventana. Por ej.: 0 = Hamming, 1 = von Hann. Se recomienda esta segunda.

Armonizador

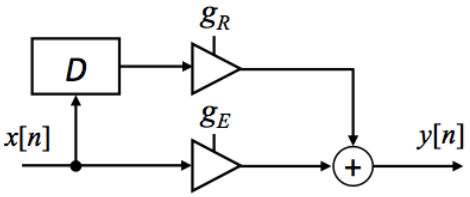
El armonizador es un transpositor tonal múltiple que convierte una señal monofónica en un acorde polifónico de varias voces, utilizando la misma técnica usada en el ejercicio de transposición por escalado de frecuencias. El acorde estará formado por la propia señal monofónica a la que se sumarán 2 transposiciones, con relaciones de frecuencias R_A y R_B, según se indica en el correspondiente esquema visto en teoría.

Procesadores de tiempo

Eco simple

Se implementa con un filtro peine FIR con un número de muestras de retardo tal que $t_D > 50$ ms. Esto permitirá separar la percepción del original y de la copia retardada. La ecuación en diferencias es la siguiente:

$$y[n] = g_E \cdot x[n] + g_R \cdot x[n–D]$$



Los coeficientes para la señal de entrada, g_E, y para la copia retardada, g_R, controlan los volúmenes respectivos de ambas ondas. Al oírse cada copia por separado no es necesario que la suma de ambos coeficientes sea la unidad: $g_E + g_R \neq 1$.

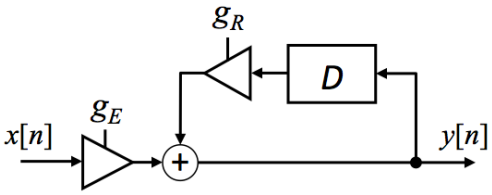
Una particularidad de este tipo de procesadores es que, al dilatar la duración del sonido, los instrumentos deberán estar activos durante más tiempo que lo que dure la señal de entrada. Como mínimo deberán estarlo durante su duración + t_D, aunque habrá casos particulares que se indicarán en su momento.

Eco múltiple incontable

Estos ecos se implementan mediante filtros peine IIR, con este diagrama y ecuación:

$$y[n] = g_E \cdot x[n] + g_R \cdot y[n–D]$$

Cuanto mayor sea D (t_D) mayor será la distancia entre las copias y cuanto mayor sea g_R más lenta será la caída de volumen de los ecos. Tampoco precisa que $g_E + g_R = 1$.

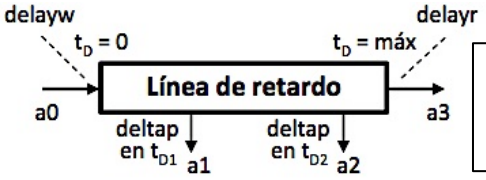


Como ya se ha hecho en la práctica anterior, la realimentación en Csound se consigue poniendo la salida a la derecha de un operador, como su entrada, antes de aparecer en una parte izquierda como salida de otra operación previa. Este instrumento es prácticamente igual que el correspondiente filtro peine IIR de la parte no presencial de la práctica de filtros. Puedes usar aquel proyecto como referencia.

Eco múltiple contable (multi-tap)

Este tipo de eco se consigue mediante el uso de una línea de retardo con múltiples, pero pocos, puntos de acceso. Csound crea este tipo de líneas con el operador **delayr**, que las dimensiona en segundos de duración del retardo, escribe señales en la línea con **delayw** y puede leer desde cualquier posición temporal de ella mediante **deltap**. El uso de estos operadores en Csound permite implementar líneas de retardo con accesos múltiples (multi-tap delays) pero es poco intuitivo.

Por ejemplo, para una situación como la de la figura, el orden de los operadores es el que se escribe es el de este recuadro (que es el inverso a lo que se podría pensar):




```
a3 delayr 2.0 ;Crea línea de retardo y lee a3 al final
a2 deltap 1.0 ;Lee a2 de la línea con 1 s de retardo
a1 deltap 0.5 ;Lee a1 de la línea con 0.5 s de retardo
   delayw a0  ;Escribe la señal a0 al principio
```

Procesadores de espacio

Para procesar el espacio, podemos modificar el volumen, el panorama, filtrar altas frecuencias o crear reverberaciones artificiales. Esto último se hace mediante la *auralización*, que consiste en hacer la convolución con una respuesta al impulso real digitalizada de un espacio acústico concreto.

Reverberadores por auralización

El concepto teórico de estos procesadores es muy sencillo y también lo es su implementación. Se trata de aprovechar que los coeficientes de un filtro FIR son su respuesta al impulso (RI), $a_n = h[n]$. Por tanto, si conseguimos la respuesta al impulso de un recinto, tendremos los coeficientes del filtro que producirá sobre la señal de entrada el mismo efecto que si suena en ese recinto.

La implementación en Csound es igualmente sencilla, pues disponemos del operador **pconvolve** que hace la convolución de la señal de entrada *ain* como si fuera un filtro FIR con la $h[n]$ suministrada mediante un fichero WAV:

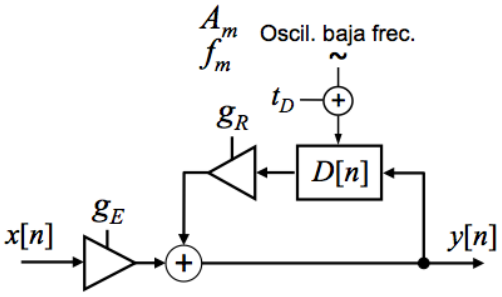
```
aoutL, aoutR  pconvolve  ain, "una_RI.wav", iNmuestras
```

La salida del operador es una señal estéreo que contendrá la reverberación por convolución. Hay que indicarle el tamaño de la ventana, en número de muestras, para trocear la $h[n]$.

Procesadores de timbre

Flanger

El *flanger* se crea con un peine IIR con tiempo de retardo, habitualmente entre 1 y 15 ms, que es modulado cíclicamente mediante un oscilador de baja frecuencia (ver figura). Así, los picos en el espectro cambian de frecuencias cíclicamente. En el esquema, un oscilador de baja frecuencia genera una señal de amplitud A_m y frecuencia f_m que, al sumarla al tiempo de retardo medio, t_D, oscilará entre $t_D - A_m$ y $t_D + A_m$. Como el retardo es variable, habrá que usar **vdelay**.



El diseño se parece bastante al peine IIR para ecos incontables, pero con retardo variable y mucho más corto que en aquel caso.

Wah-wah

El wah-wah consiste en uno o más filtros pasa-banda (2 en cascada en el caso del de la figura) de f_c variable y de banda estrecha (del orden de $Q \geq 2$). Su salida se mezcla con la señal directa. Para implementar los pasa-banda en Csound se pueden usar 2 operadores **reson**, siendo la salida del primero la entrada del segundo.