DRC-FIR v3.2.1

http://drc-fir.sourceforge.net/doc/drc.html#htoc26

traducción / extracto, por Rafax che.es/uniforo
versión 0.1

4 Program description and operation

DRC needs at least 1 second of the impulse response of your **room** and audio system at the **listening position**, separated for each channel. The impulse response should be provided in raw format, either in signed 16 bit format or using 32/64 bit IEEE floating point samples.

DRC includes some command line tools to do accurate time aligned impulse response measurement, see section 4.4 for further details.

4.1 Filter generation procedure. Cómo trabaja DRC-FIR:

Un entorno acústico típico es un sistema de fase NO mínima, de modo que en teoría no puede ser invertida para obtener una compensación perfecta. Además, es un sistema lineal diferente para cada posición de escucha...

Es muy compleja la generación de un filtro que proporciona una buena compensación de magnitud y fase de la respuesta de frecuencia del sonido directo, un buen control de la magnitud de la respuesta de frecuencia del campo estacionario y una sensibilidad aceptable en la posición de escucha. Ese es el goal de DRC, para ello las etapas son:

1. Enventanado inicial y normalización de la respuesta de impulso de entrada.
2. Compensación de micrófono opcional.
3. Limitación inicial de dips para evitar inestabilidades numéricas durante deconvolución homomórfica.
4. Descomposición en componentes de fase mínima y de exceso de fase utilizando deconvolución homomórfica.
5. Prefiltrado de la componente de fase mínima con ventanas dependientes de la frecuencia.
6. Limitación de la respuesta de frecuencia de los dips en la componente de fase mínimo para evitar inestabilidades numéricas durante la etapa de inversión.
7. Prefiltrado de la componente de fase en exceso con ventanas dependientes de la frecuencia.
8. La normalización y la convolución del componente de fase mínima preprocesada y del componente de exceso de fase (opcional partir de la versión 2.0.0).
9. Inversión del Impulso de respuesta a través de técnicas de mínimos cuadrados o deconvolución rápida.
10. Cómputo opcional de una respuesta objetivo psicoacústico basado en la respuesta envolvente de magnitud de la respuesta de impulso corregida.
11. Limitador de picos de respuesta de frecuencia para evitar la sobrecarga de los altavoces y amplificación.

12. Truncamiento del Ringing con ventanas dependientes de la frecuencia para eliminar cualquier exceso de Ringing no deseado causada por la etapa de inversión y la etapa de limitación de picos.
13. Postfiltering para eliminar bandas incorregibles (subsónica y ultrasónica) para proporcionar la respuesta de frecuencia final.
14. Generación opcional de una versión de fase mínima del filtro de corrección.
15. Convolución final opcional del impulso de entrada con el filtro de corrección propuesto, para pruebas.

Casi todas estas etapas tienen parámetros opcionales y la capacidad de sacar resultados intermedios.

El autor de DRC ha conseguido que su equipo HiFi se parezca a un sistema de monitores de estudio ubicados en una sala altamente amortiguada. Además tiene un subwoofer de altas prestaciones con su respuesta extendida hasta el rango infrasónico. Para ver ejemplos de los resultados obtenidos ir al appendix A.

4.2 Frequency dependent windowing

Es una de las operaciones más comunes en DRC. Se benefica del hecho de que en una sala, la sensibilidad de la función de transferencia de la sala respecto de la posición de escucha es brutalmente dependiente de la longitud de onda. Entonces la criticidad de la posición de escucha aumenta con la frecuencia...

El efecto lateral es que debemos reducir la corrección en frecuencias altas (longitudes de onda cortas).

Esto además tiene algunas implicaciones psicoacústicas, porque nuestro sistema de audición está concebido para tener en cuenta exactamente el mismo comportamiento de sala, por lo que su comportamiento sigue reglas similares.

DRC implementa frequency dependent windowing con dos tipos de procedimiento:

1.- Band windowing (enventanado de banda): simply filters the input signals into logarithmically spaced adjacent bands and applies different windows to them, then summing the resulting signals together to get the output windowed impulse response

2.- Sliding lowpass linear time variant filtering (correderas de paso bajo variable en el filtrado lineal temporal): uses a time varying lowpass filter, with a cut-off frequency that decreases with the window length.

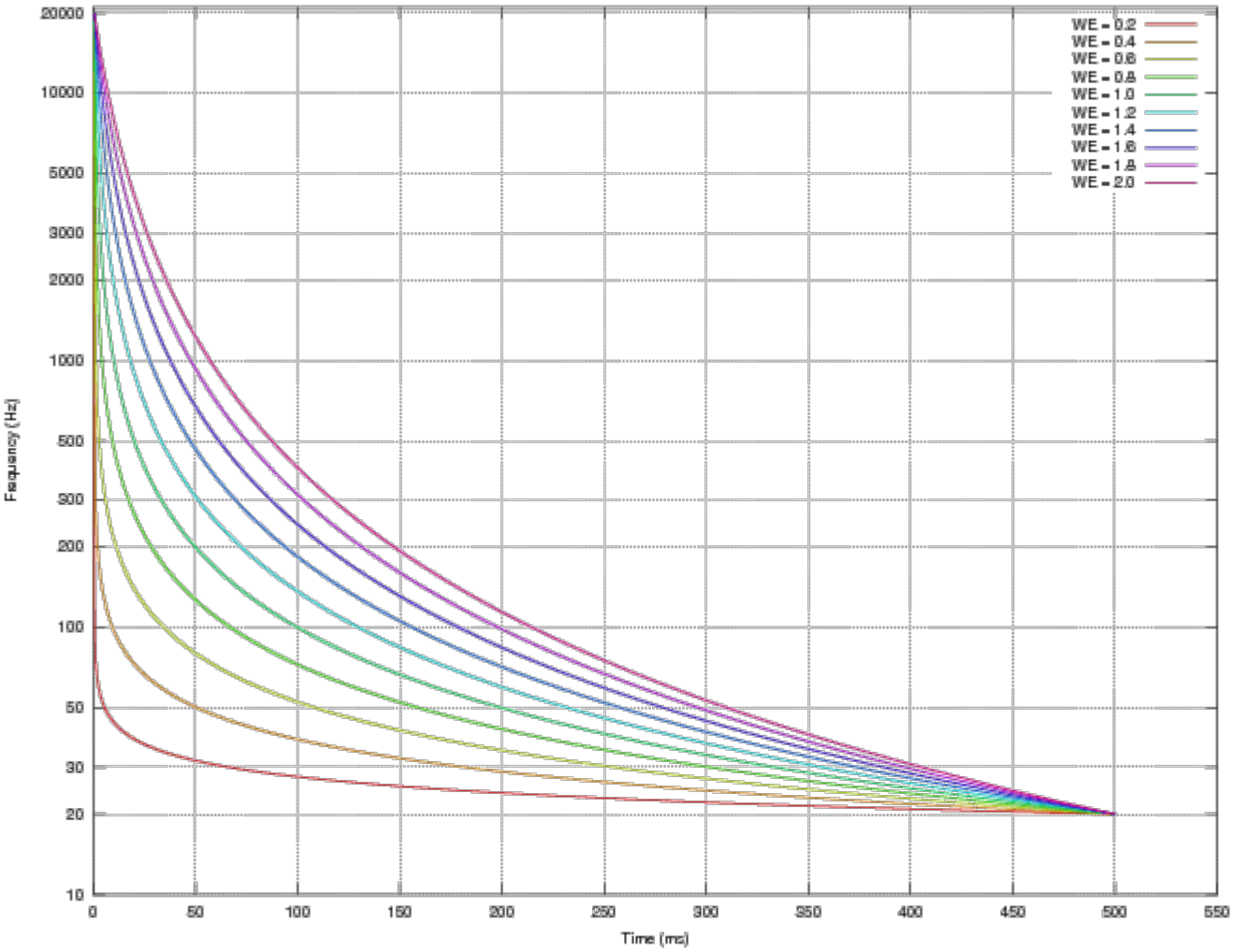
El segundo procedimiento da resultados con menos errores y es más flexible..

Los parámetros básicos son:

1.- La ventana inferior, la que se aplica a la frecuencia más baja del rango de frecuencias.

2.- La ventana superior, la que se aplica a la frecuencia más alta del rango de frecuencias.

3.- El exponente de la window, que conecta las ventanas inferior y superior con una función paramétrica que evoluciona inversamente a la frecuencia. ver section 6.4.8



La figura muestra el típico conjunto de curvas de prefiltering aplicadas al impulso de respuesta de sala en la configuración **normal.drc**. Para este fichero el valor por defecto de WE (window exponent) = 1.0. La parte de la respuesta impulso que es preservada y además por tanto corregida, es la que queda por debajo de las curvas. La parte restante del plano tiempo-frecuencia es simplemente dejado fuera de ventana e ignorado en la corrección.

Podemos ver que solo una fracción del plano tiempo-frecuencia es corregida por DRC. Esta fracción define prácticamente los límites de aplicación de DRC. Por encima de estos límites aparecerán artifactos con pequeños cambios en la posición de escucha.

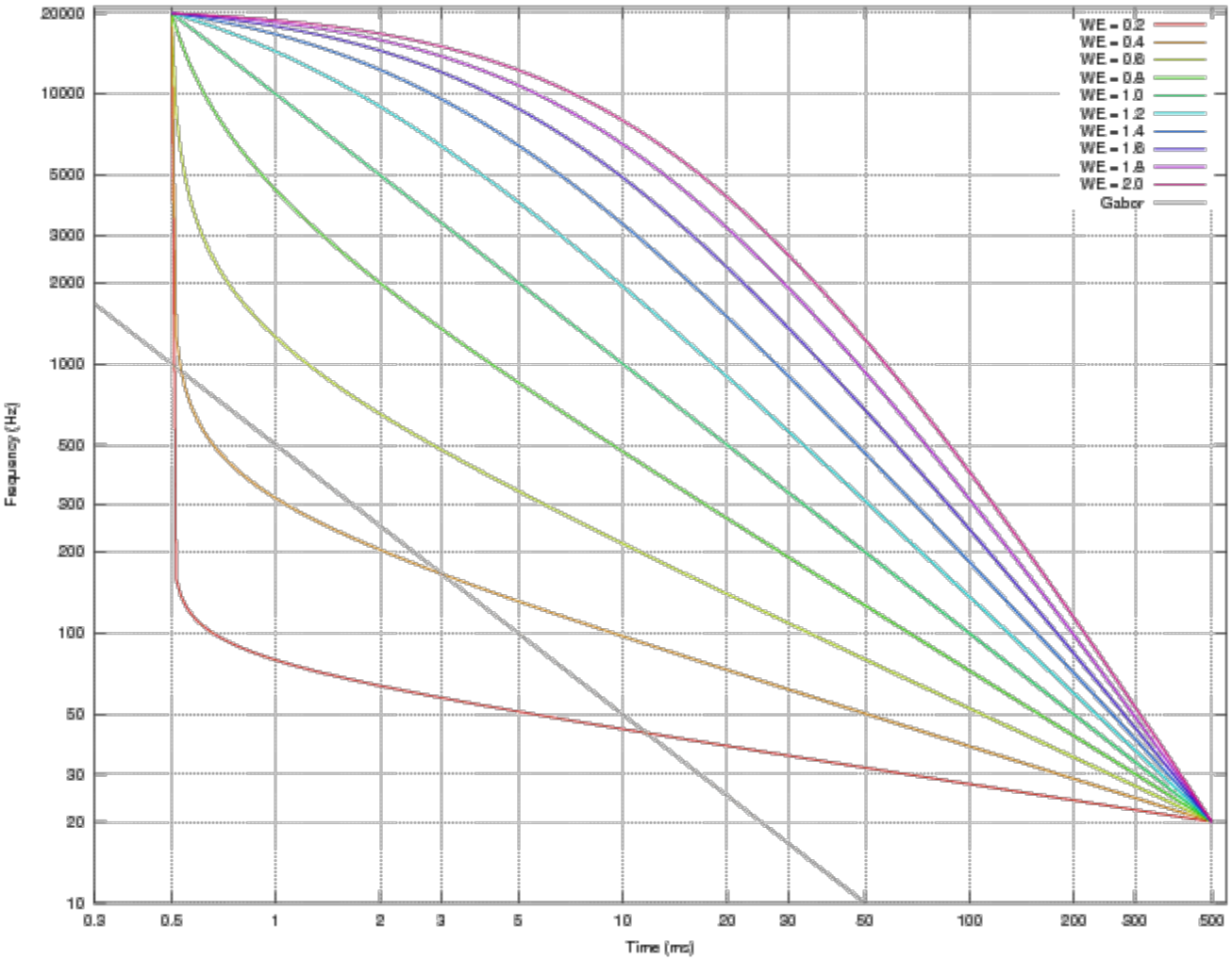
Si pintamos el eje de frecuancias lineal, el área de trabajo de DRC parecería muy reducida. Sin embargo no es así, ya que debe tomarse en cuenta que nuestro oido percibe este plano tiempo-frecuencia en una escala logarítmica de frecuencias.

También se ve claro que por encima de 1 o 2 KHz solo es corregido el sonido directo y la corrección de sala realmente se reduce tan solo a una corrección minimalista de la respuesta de los altavoces.

Durante el desarrollo de DRC el autor ha leído algunas *notas informales en Internet* diciendo que con ventanas de corto tiempo debemos considerar que nuestra percepción del tiempo también es logarítmica (además de la frecuencia). Independientemente de la validez de esta hipótesis, podemos aprovecharla para representar gráficamente las curvas de prefiltering window junto con el límite de Gabor, que se define por la siguiente desigualdad:

$$[?][?]\,f\,*\,[?][?]\,t > 1/2$$

El límite de Gabor define el límte de incertidumbre en el plano tiempo-frecuencia. Esto significa por ejemplo que, viendo la gráfica de abajo, cuando WE < 0.5 la curva de enventanado empieza a violar el límite de Gabor. Entonces DRC hará una evaluación errónea de la función de transferencia de la sala.



Desde la versión 2.6.0 ha sido introducida una curva de prefiltering basada en la transformada bilinear. Esta curva encaja mejor con la resolución del oído típica y también con el comportamiento de la sala típico. La curva difiere de la anterior para valores WE ≠ 1.0. La mejor aproximación al comportamiento del oido se puede ver en la figura 8 donde se muestra que usando una adecuada configuración de los parámetros de windowing es posible alcanzar un emparejamiento casi perfecto con la escala psicoacústica ERB.

4.2.1 Pre-echo truncation

Desde la versión 2.7.0 **esta estapa está implícitamente deshabilitada**. Considerando que la inversión del exceso de fase es llevada a cabo simplemente mediante el inveresión temporal de la componente de exceso de fase, el pre-echo está implícitamente limitado por el enventanado dependendiente de la frecuencia llevado a cabo en el exceso de fase.

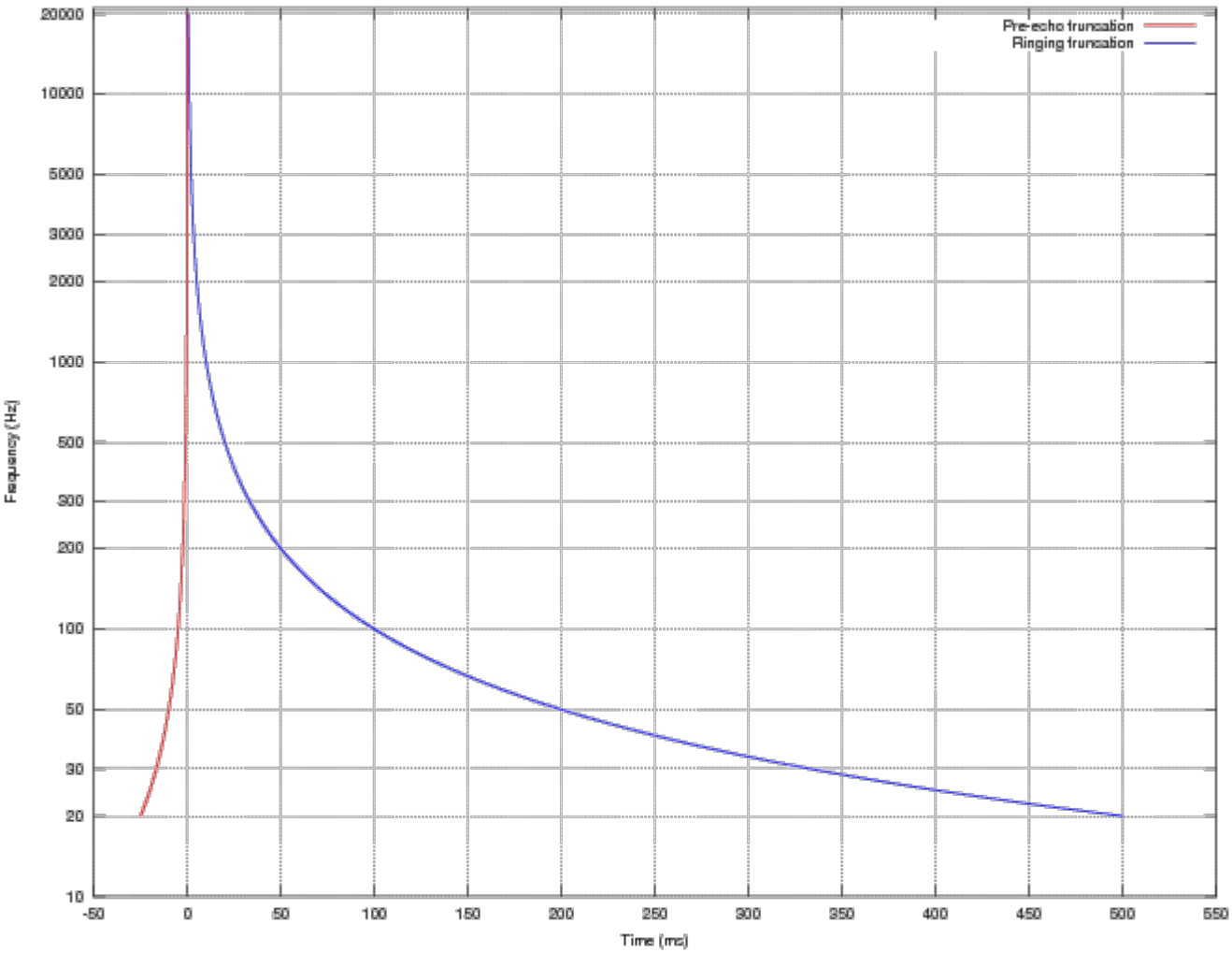
Se puede usar para experimentar. Una reecualización de fase mínima realizada en el truncado del pre-echo es equivalente a una reecualización de exceso de fase en el pre-procesado del exceso de fase, y a la inversa...

Si se habilita, un enventadado dependiente de la fecuencia será usado para truncar el pre-echo causado por la inversión de la parte de exceso de fase de la respuesta a impulso de la sala. El procedimiento de truncado es siempre el mismo pero está implícitamente aplicado en el lado izquierdo del plano tiempo-frecuencia, debido a que la inversión del exceso de fase se corresponde con una inversión temporal. Se usa un enventanado mucho más corto debido a que nuestro oido es MUY sensible al pre-echo.

Enventanando parte de la respuesta impulsiva de la componente de exceso de fase del fitro corrector, por supuesto que deja de ser un filtro pasa-todo, es decir que la parte de la fase exceso ya no tiene una magnitud plana. Para compensar esto, la componente de exceso de fase de la respuesta de magnnitud es reecualizada a plano, usando, después de la inversión, un filtro de fase mínima. Esto por supuesto causa algín post-ringing. Aunque eso solo ocurre en algunas bandas estrechas, puede ser audible, y está limitado por la etapa subsecuente de Ringing Truncation.

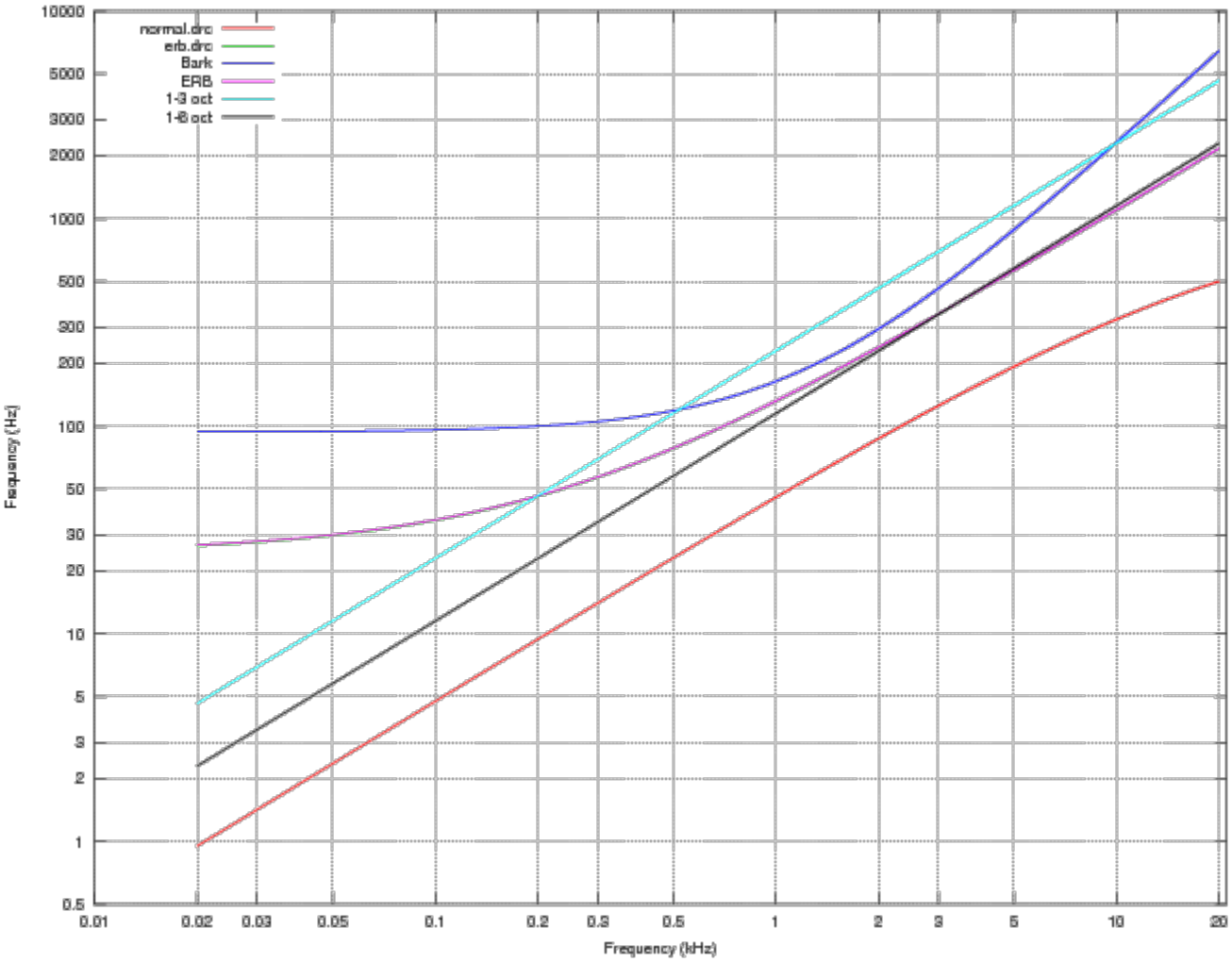
4.2.2 Ringing truncation stage

Desde la versión 2.5.0 se aplica un enventanado (windowing) más dependiente de la frecuencia directamente al filtro después de la etapa de inversión de la respuesta a impulso y de la etapa de limitación de picos. Ello es para eliminar cualquier ringing residual causado por las etapas previas, en especial las etapas de limitación de peaks y dips, incluso si ello supone alguna pérdida en la precisión del filtrado.



Con esto, la respuesta a impulso del filtro queda encerrada en una especia de jaula tiempo-frecuencia definida por el ejuste del windowing del exceso de fase en el lado izquierdo del tiempo 0 y por los ajustes de ringing truncation el el lado derecho del tiempo 0. Por ejemplo la figura de arriba para **normal.drc**.

Considerando que estos límites tienen también algunas implicaciones psicoacústicas, con ello DRC debe ser capaz de truncar automáticamente cualquier parte de la corrección que pudiera causar artifactos audibles. Así podemos considerar que DRC hace una auto optimización psicoacústica, siendo más robusto y menos susceptible a los artifactos.



Aplicando la desigualdad de Gabor a la longitud de la ventana entre las dos curvas de pre-echo y ringing-truncation es fácil obtener una resolución de frecuencia equivalente, como funcion de la frecuencia central, a la del procedimiento de enventanado dependiente de la frecuencia (*frequency dependent windowing*). La figura de arriba muestra el ancho de banda de resolución obtenido con normal.drc y con erb.drc, y es comparado con algún procedimiento estandar de suavizado ampliamente usado, como el **suavizado en fracciones de octava** y como las clásicas escalas psicoacústicas **Bark** y **ERB**.

La figura de arriba muestra que la resolución de corrección usada en DRC supera a cualquiera de los procedimientos estandar de suavizado, al menos con los ajustes **normal.drc** (ver 5.2). Esto significa que la corrección debe proporcionar una percepción de respuesta en frecuencia realmente cercana a la respuesta objetivo (target) configurada.

La gráfica erb.drc muestra la aproximación de la escala ERB que proporciona los ajustes del fichero **erb.drc** (see section 5.2., se asume que $\Delta t * \Delta f = 2$ en lugar de la usual inigualdad de Gabor), esdecir suponiendo el enventanado dependiente de la frecuencia con los ajustes usados tiene una resolución de 4 veces superior al límite

de Gabor. Esta es una estimación aproximada de la verdadera resolución conseguida mediante el procedimiento DRC en esta situación. Esta estimación se ha derivado de considerar el efecto compuesto de varias ventanas superpuestas en diversas etapas del procedimiento de generación de filtro.

4.3 Psychoacoustic target computation (Cómputo psicoacústico de la curva objetivo)

Desde la versión 3.0.0 se ha introducido una etapa opcional usada para calcular un objetivo de respuesta en frecuencia (target) basado en la percepción psicoacústica de la respuesta en frecuencia corregida. Está basado en el cálculo de la envolvente de la respuesta en magnitud de la respuesta del impulso corrector, que una extensión del concepto de envolvente espectral.

Esto se lleva a cabo antes de la aplicación del target usual, de manera que la respuesta del target estandar no es compensada por esta etapa.

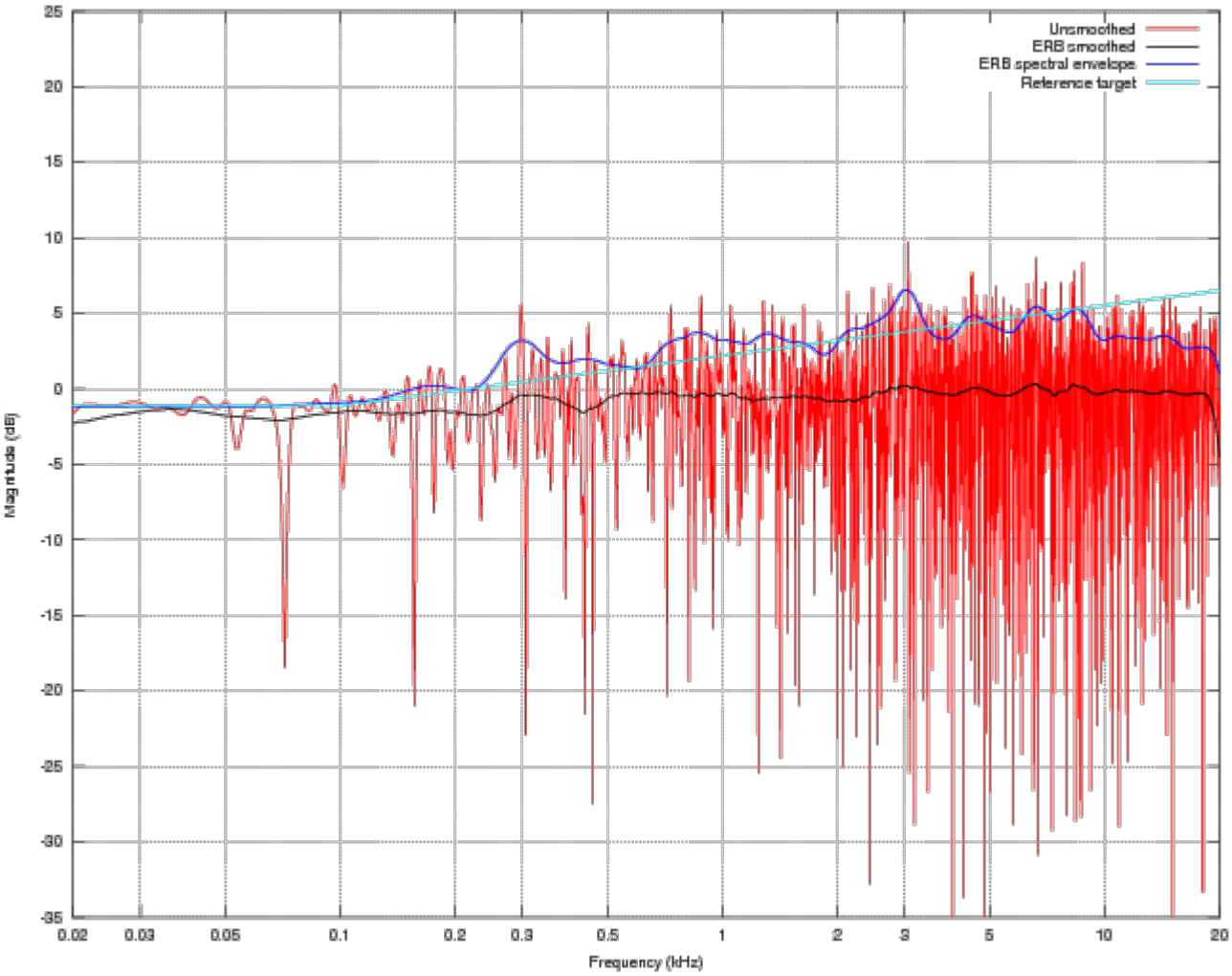
La envolvente espectral es un concepto que ha sido introducido en el campo de la síntesis y análisis de voz, se define simplemente como una suave curva conectando o siguiendo los picos del espectro de la señal. Hay evidencias de que nuestro oido lo utiliza para reconocer los sonidos. Por ejemplo, esto permite al oido reconocer la voz generada por uno mismo, la de un susurro o la generada por otros medios, ya que finalmente tienen envolvente espectral similar.

La envolvente espectral también explica por qué nuestro oido es más sensible a los picos (peaks) que a los valles (dips) de la magnitud de la respuesta de frecuencia.

Una curva basada en los picos de la respuesta de magnitud es por definición poco o nada afectada por dips en la respuesta de frecuencia.

Los procedimientos para el cálculo de la envolvente espectral no adecuados para su aplicación en HiFi. Con DRC ha sido desarrollado un nuevo procedimiento. Este es una variación del habitual procedimiento de suavizado de fracción de octava, usando la media paramétrica Hölder en lugar del habitual promediado simple. Además el suavizado ha sido extendido para proporcionar las escalas de resolución Bark y ERB cuando sea aplicable.

Para afinar los parámetros de cálculo de la envolvente se han tomado en cuenta algunas respuestas de magnitud de salas reales. Se ha optado por que la envolvente proporcione una respuesta lo más cercana posible a la inversa de las respuestas target (objetivo) sugeridas en la literatura. Un ejemplo se reporta en la figura 9.



Los mismos parámetros han sido probados también en algunas salas poco comunes para verificar que todavía proporcionaban los resultados esperados. Por supuesto, considerando que ahora el target básico es proporcionado por la inversa de la envolvente, las respuestas de los target habituales ya no son necesarias, aparte de filtrado subsónico o ultrasónico, y el target ahora debe ser puesto a plano. Por esta razón, la respuesta plana es ahora el default target de todos los ficheros de configuración estandar.

La etapa estandar de target debe usarse solo para ajustar la respuesta al gusto personal, pero, al contrario que en previas versiones, para una respuesta neutral debe usarse na respuesta plana.

Desde un punto de vista subjetivo, un sistema ecualizado con la inversión de la envolvente de la respueta de magnitud, normalmente suena muy neutro.

A pesar de que la mayoría de las veces la envolvente de respuesta de magnitud es diferente entre los diferentes canales, resultando en un obvio desalineamiento entre canales si se evalúa con los procedimientos de suavizado estandar, la imagen estereo normalmente mejora, convirtiéndose en más estable y enfocada. Esto parece confirmar

que la envolvente está muy cerca de la percepción subjetiva de la respuesta de magnitud.

4.4 Medición de la respuesta a impulso de la sala.

Hay dos sencillas utilidades command line: **glsweep** (Generate Log Sweep) and **lsconv** (Log Sweep Convolution). Los pasos son:

1.- Generar el log sweep y el filtro inversor con **glsweep**. Opcionalmente convertir a un formato de archivo adecuado.

2.- Reproducir el log sweep por el altavoz y simultáneamente grabarlo con un micrófono en un archivo.

3.- Con **lsconv** se obtiene lla convolución del log-sweep grabado con el filtro inversor del punto1.

lsconv admite usar un segundo canal como referencia de alineación, usualmente se pone un bucle de la señal enviada a los altavoces. Así se alcanzan medidas de **+-0.1dB**. Además el método log-sweep es **inmune a ruido ambiental**, tiene 90dB S/N incluso en ambientes poco silenciosos, incluso fuerte rechazo a distorsiones no lineales de los altavoces y del amplificador...

4.4.1 The glsweep program

Se recomienda un log-sweep de unos 45 s. La salida del programa es una raw 32bit IEEE float. Uso:

```
$ glsweep rate amplitude hzstart hzend duration silence leadin leadout sweepfile
inversefile
```

Ejemplo de uso:

```
$ glsweep 44100 0.5 10 21000 45 2 0.05 0.005 sweep.pcm inverse.pcm
```

Si queremos convertir a wav de 16bit se usa **sox**:

```
$ sox -t f32 -r 44100 -c 1 sweep.pcm -t wav -c 1 sweep.wav
```

Si queremos tener también el canal de referencia en el wav::

```
$ sox -t f32 -r 44100 -c 1 sweep.pcm -t wav -c 2 sweep.wav
```

El firtro inversor no precisa conversión ya que **lsconv** puede leerlo tal cual.

Si queremos convertir un wav en raw 32bit float:

```
$ sox recorded.wav -t f32 recorded.pcm
```

Si queremos convertir un wav stereo en raw 32bit float individuales:

```
$ sox recorded.wav -t f32 -c 1 recorded.pcm mixer -l
$ sox recorded.wav -t f32 -c 1 reference.pcm mixer -r
```

4.4.2 The lsconv program

```
Usage: LSConv sweepfile inversefile outfile [refsweep mingain [dlstart]]
  sweepfile: sweep file name
  inversefile: inverse sweep file name
  outfile: output impulse response file
  refsweep: reference channel sweep file name
  mingain: min gain for reference channel inversion
  dlstart: dip limiting start for reference channel inversion


Example: $ lsconv sweep.pcm inverse.pcm impulse.pcm refchannel.pcm 0.1 0.8
```

Todos los ficheros raw 32bit float.

Uso con canal de referencia:

```
$ lsconv recorded_sweep.pcm glsweep_inverse.pcm impulse.pcm reference.pcm 0.1
```

Uso sin canal de referencia:

```
$ lsconv recorded_sweep.pcm glsweep_inverse.pcm impulse.pcm
```

`impulse.pcm` es el archivo que procesará el programa DRC.

“0.1” (=20dB) es la ganancia mínima permitida para la inversión del canal de refencia. O sea, no serán corregidos más de 20dB de la respuesta de frecuencia del canal de referencia. Esto se necesita para eludir inestabilidades de cómputo causadas por el fuerte cutt off de los filtros brick wall de DAC y ADC de la tarjeta de sonido.

4.4.3 Sample automated script file

El script **source/contrib/Measure/measure** utiliza glsweep, lsconv, SoX y ALSA (aplay arecord) para automatizar un procedimiento de mediad time-aligned usando un canal de referencia. OjO hay que parar JACK para dejar libre ALSA.

```
Automatic measuring script.
Copyright (C) 2002-2005 Denis Sbragion

Usage:
 measure bits rate startf endf lslen lssil indev outdev impfile [sweepfile]

 bits: measuring bits (16 or 24)
 rate: sample rate
 startf: sweep start frequency in Hz
 endf: sweep end frequency in Hz
 lslen: log sweep length in seconds
 lssil: log sweep silence length in seconds
 indev: ALSA input device
 outdev: ALSA output device
 impfile: impulse response output file
 sweepfile: optional wav file name for the recorded sweep

example: measure 16 44100 5 21000 45 2 plughw plughw impulse.pcm
```

Se asume que el canal de medida es “L” y el de referencia es “R”. 16 bit de profundidad es más que suficiente para medidas acústicas. Se propone una duración de 45 s para Fs=44100 Hz. En ambientes muy ruidosos convendrá aumentar la duración, pero no la profundidad de bits.

El script **source/contrib/Measure/measurejack** está adaptado para usar con JACK en lugar de directamente con ALSA.

4.4.5 How to work around your cheap, resampling, soundcard

Hacer DRC a 48KHz y posteriormente resamplear el filtro obtenido a 44.1KHz con **SoX**.

4.6 Correction tuning

DRC proporciona configuraciones de ejemplo, ver sección 5.2. Pero conviene tener en cuenta algunos asuntos:

4.6.1 Preventing pre-echo artifacts

Aparecen al superar cierto umbral de precisión en la corrección. Son audibles anticipadamente a la ocurrencia de transitorios o ataques afilados en un programa musical. Hay dos opciones:

1) Reducir la corrección en bandas críticas

2) Usar una aproximación de fase mínima para evitar pre-echoes. A costa de tener más ringing tras el pico principal, pero el post-ringing queda enmascarado por nuestro oido y por la reverberación de la sala, resultando casi inaudible.

DRC usa las dos opciones en sus distintas etapas. Deberemos proceder a:

1) Reducir la corrección aplicada al componente de exceso de fase, mediante la reducción del tamaño de la ventana dependiente de la frecuencia que aplicamos. Con los archivos de configuración proporcionados, esto es normalmente todo lo que hay que hacer, ya que el resto de procedimientos ya están tenidos en cuenta.

2) Usar una FFT suficientemente larga cuando esté implicada una conculución circular (basicly homomorphic deconvolution and pre-echo truncation inversion), ya que circular artifacts pueden convertirse fácilmente en pre-echo.

3) Usar el procedimiento de prefilterig de paso bajo de deslizamiento lateral sencillo, debido a pequeños errores numéricos con el enventanado de banda que causan pre-echo en los límites de la banda.

4) Usar la versión de fase mínima en algunos procedimientos, como por ejemplo en la limitación de peaks y dips.

5) Usar la truncación del pre-echo en la deconvolución rápida para el proceso de inversión. Por cierto, si están puestos adecuadamente los parámetros de enventanado del exceso de fase, esto no debería ser necesario.

En situaciones normales podemos usar los archivos de ejemplo de configuración proporcionados, en tanto solo se cambian los parámetros EPLowerWindow, EPWindowExponent, MPLowerWindow and MPWindowExponent para cumplir nuestras necesidades.

4.6.2 Preventing clipping

La operación de convolución puede incurrir en clipping en el DAC.

Todas las etapas de DRC-fir aceptan 4 tipos de normalización:

- (S)uma, también llamada L_1 norm

- (E)uclídea, también llamada L_2 norm

- (M)áx, también llamada L_∞ norm

- (P)eak

La normalización S garantiza no saturar el DAC con valores mayores de 2^(bits_DAC-1) aunque la muestra de entrada multiplicada por el factor de normalización resultara mayor.

De todos modos, con señales musicales típicas y dependiendo de la respuesta de frecuencia del filtro, el uso de S puede incurrir en la pérdida de la resolución disponible en el DAC debido al manejo de niveles muy bajos. Es habitualmente seguro usar un factor de normalizacón S mayor que uno sin que ocurran problemas...

La normalización P proporciona una buena opción. P reescala el filtro tal que el mayor pico en la FR del fltro se lleva a 0dBFS...

Brutefir necesita un filtro pcm normalizado a 1.0 para obtener 0 dB de ganacia entre la entrada y la salida de una etapa de filtrado.

Si usamos Brutefir, podemos poner PSNormFactor=1 y PSNormType=S. Entonces podemos ajustar las ganacias internas de Brutefir y observar si ocurren saturaciones gracias a las facilidades de monitorización de Brutefir (escupe los peaks por la consola). podemos probar el filtro metiendo una fuente de ruido blanco de 0dBFS para tener el escenario de pruebas del peor caso.

Si no queremos hacer la prueba de arriba, mejor seleccionar una normalización tipo P.

Una simple regla de oro es usar el tipo E con un factor de normalización de unos 2 dB por debajo de la máxima ganacia permitida en la fase de peak limiting de DRC-fir. Con los ficheros de configuración estandar, donde la ganacia máx permitida nunca es mayor que 6dB, esto significa usar un factor de normalización de 0.3 ~ 0.4 con convolvers como Brutefir (1.0 = 0 dB gain), o bien usar 10000 ~ 13000 con convolvers que usen 32768 (2^15) = 0dB.

OjO al factor de normalización en relación a la profundidad de bits de nuestro sistema de proceso de audio y a la profundidad de bits de la tarjeta. Especialmente si usamos DAC de 16 bits con dithering o un DAC de 24 bits.

Los ficheros de configuración por defecto usan el tipo E con factor 1, dejando al convolver la tarea de escalado de la ganacia del filtro para evitar clipping. Deberemos ajustar la ganancia del convolvedor a algo por debajo de -6 dB, que es la máxima ganancia permitida en DRC-fir en la etapa de peak limiting.

4.6.3 Some notes about loudspeaker placement

La corrección del filtro DRC puede ser muy dependiente de la posición. Es muy importante igualar las distancias entre cada altavoz y el punto de escucha.

Con DRC suele ser beneficioso experimentar con situaciones poco habituales. Las reflexiones de muros cercanos son más difíciles de corregir cuando están lejos de los altavoces, por tanto es posible obtener mejores resultados colocanso los altavoces en rincones. En ese caso conviene disponer algún material absorbente para minimizar las reflexiones tempranas en agudos, donde DRC solo es capaz de corregir el sonido directo.

El tipo de colocación es opuesto al tradicional sin DRC, en el cual es usual poner los altavoces lejos de las paredes para evitar reflexiones tempranas... Hay que experimentar...

4.6.4 Some notes about channel balance

DRC-fir no hace corrección del balance de canales. Debe hacerse manualmente, preferiblemente con ruido rosa y un sonómetro. De todos modos, debido a que ambos canales tendrán un respuesta plana, es fácil hacerlo a oído, escuchando una voz femenina monofónica y ajustando las ganacias hasta que la voz quede centrada.

También podemos usar los avisos de nivel proporcionados por DRC al final de la ejecución, a condición de que en la medida de los impulsos medidos se haya respetado el balance de canales.

4.6.5 Interchannel time alignment

La versión actual de DRC puede compensar un desalineamiento temporal entre canales de no más de +-8 samples. Además, deben proporcionarse medidas alineadas en el tiempo con precisión, usando bien glsweep+lsconv con ref.channel, o bien otra herramienta que proporcione el mismo grado de precisión.

Para alcanzar este alineamiento temporal, hay que ejecutar dos pasos:

1.- Tomar nota de la posición_X de la muestra del centro del impulso, que se avisa al ejecutar DRC para el primer canal.

2.- Ejecutar DRC para el resto de canales con BCImpulseCenterMode=M y BCImpulseCenter=posicion_X.

Si los canales estuvieran desalineados más de unos pocos samples, los filtros serán incorrectos, tendrán una subida en agudos, y un sonido brillante.

4.6.6 How to tune the filters for your audio system

La mejor manera de afinar DRC para nuestro sistema es chequear los resultados usando los scripts Octave proporcionados con la documentación, ver anexo A. O bien a oido, pero no es fácil.

Es un error usar una corrección excesiva. Algunas pistas musicales delatarán fallos. Para experimentar cómo suena una corrección excesiva, se proporciona una configuración “insane.drc” que producirá artifactos claramente audibles, excepto en salas muy amortiguadas.

Es mejor empezar por una mínima corrección como la de “minimal.drc” o “erb.drc”. Si nuestras medidas de impulsos son del calidad, ya obtendremos grandes resultados, sin ningún artifacto. Si no fuera así, hay que **revisar el proceso de medida**.

Una vez que el resultado de minimal.drc es bueno, podemos ir lentamente cambiando a correcciones más fuertes usando “soft.drc”, “normal.drc”, “strong.drc” and “extreme.drc”. Siempre escuchando los resultados en cada paso, si es posible haciendo conmutación rápida entre filtros. Cuande empiezen a aparecer artifactos, es momento de parar y jugar con algunos parámetros específicos de corrección.

Los primeros parámetros a modificar son aquellos que definen el enventanado de la curva de corrección aplicado a la señal: MPWindowExponent, EPWindowExponent, RTWindowExponent. Redudir lentamente a 0.95, 0.90, 0.85... hasta 0.70, por tanto reduciendo la corrección en las bandas críticas de medios y medios-graves. OjO, eston son parámetros muy sensibles, cambiar alguna centésima puede ser audible especialmente cuando estamos cerca de la frontera de la aparición de artifactos.

Cuando los artifactos de medios desaparecen se puede empezar subiendo levemente (en saltos de 5%) las ventanas aplicadas al rango de graves: MPLowerWindow, EPLowerWindow, RTLowerWindow, hasta que los artifactos empiecen a reaparecer, entonce bajar otra vez los parámetros Window Exponent hasta que los artifactos desaparezcan, y así sucesivamente.

No sobrepasar los valores límite:

Enventanado en medios/medios-graves: xxWindowExponents >= 0.60

Enventanado de la fase mínima y del ringing truncation: MPLowerWindow, RTLowerWindow < 1s

Enventanado del exceso de fase: EPLowerWindow < 100 ms, que vincula a los parámetros de truncado del pre-echo (ISPELowerWindow, ISPEUpperWindow), ver secciones 6.8.4 y 6.8.5.

Se recomienda cambios al vuelo de los filtros en “prueba ciega”...

5 Program compilation and execution

Muchas distros Linux tienen el paquete DRC incluido, y tb está compilado (drc.exe, glsweep.exe, lsconv.exe) para ejecutarlo en Windows...

Tb se puede compilar las fuentes. Durante las pruebas de la versión 3.1.0 Denis Sbragion ha comprobado que Incluso compilado con precisión aritmética sencilla los filtros presentan S/N > 140 dB en el pero caso.

DRC es un programa de consola, sin GUI. Los parámetros de ejecución residen en un fichero de texto plano, por ejempo:

```
$ drc configuracion1.drc
```

También podemos reemplazar los parámetros del archivo del configuración por otros proporcionados en la línea de comando:

```
$ drc --BCInFile=impulso_L.pcm --PSOutFile=filtro_L.pcm --PLMaxGain=3.5 “mi config.drc”
```

Y ver la ayuda:

```
$ drc --help
```

5.2 Sample configuration files

En orden de grado de corrección: **minimal-XX.X.drc, soft-XX.X.drc, normal-XX.X.drc, strong-XX.X.drc, extreme-XX.X.drc**.

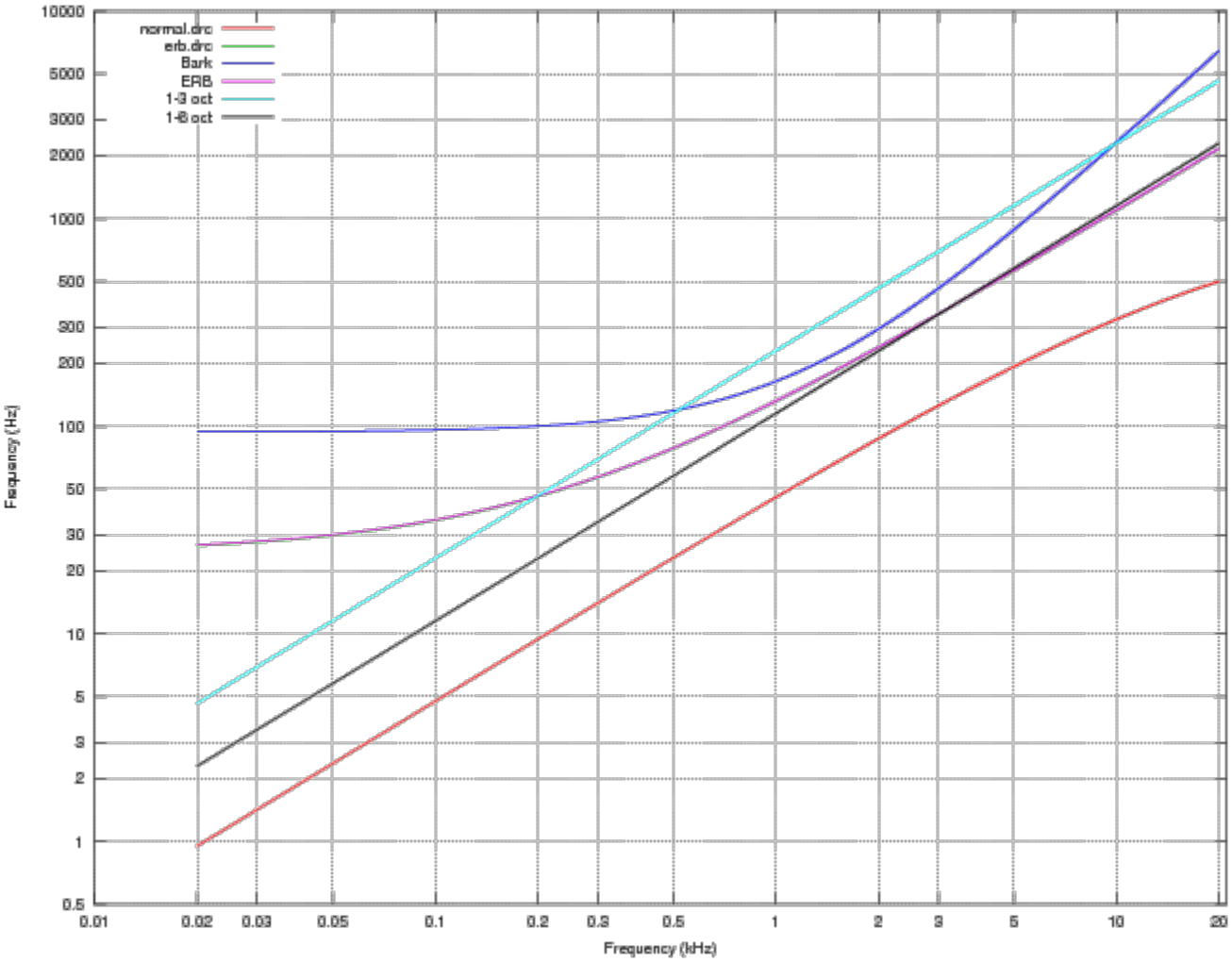
Y como ejemplo que provoca artifactos **insane-XX.X.drc**, muy util para aprender a identifciarlos, ver sección 4.6.6.

XX.X representa la Fs en KHz. La única diferencia entre los ficheros es la Fs.

OjO, strong.drc y extreme.drc producen filtro muy sensibles a la posición de escucha

rs.pcm (32 bit IEEE raw) es un ejemplo de la respuesta a impulso del equipo HiFi del autor

erd.drc proporciona una aproximación precisa de la escala psicoacústica ERB:



Básicamente la corrección **erb.drc** es más fuerte que con minimal.drc, pero siendo ajustada a nuestra resolución psicoacústica, probablemente proporcinará una buena percepción con una mínima dependencia de la posición de escucha, además es adecuada para múltiples situaciones de escucha, como home theater.

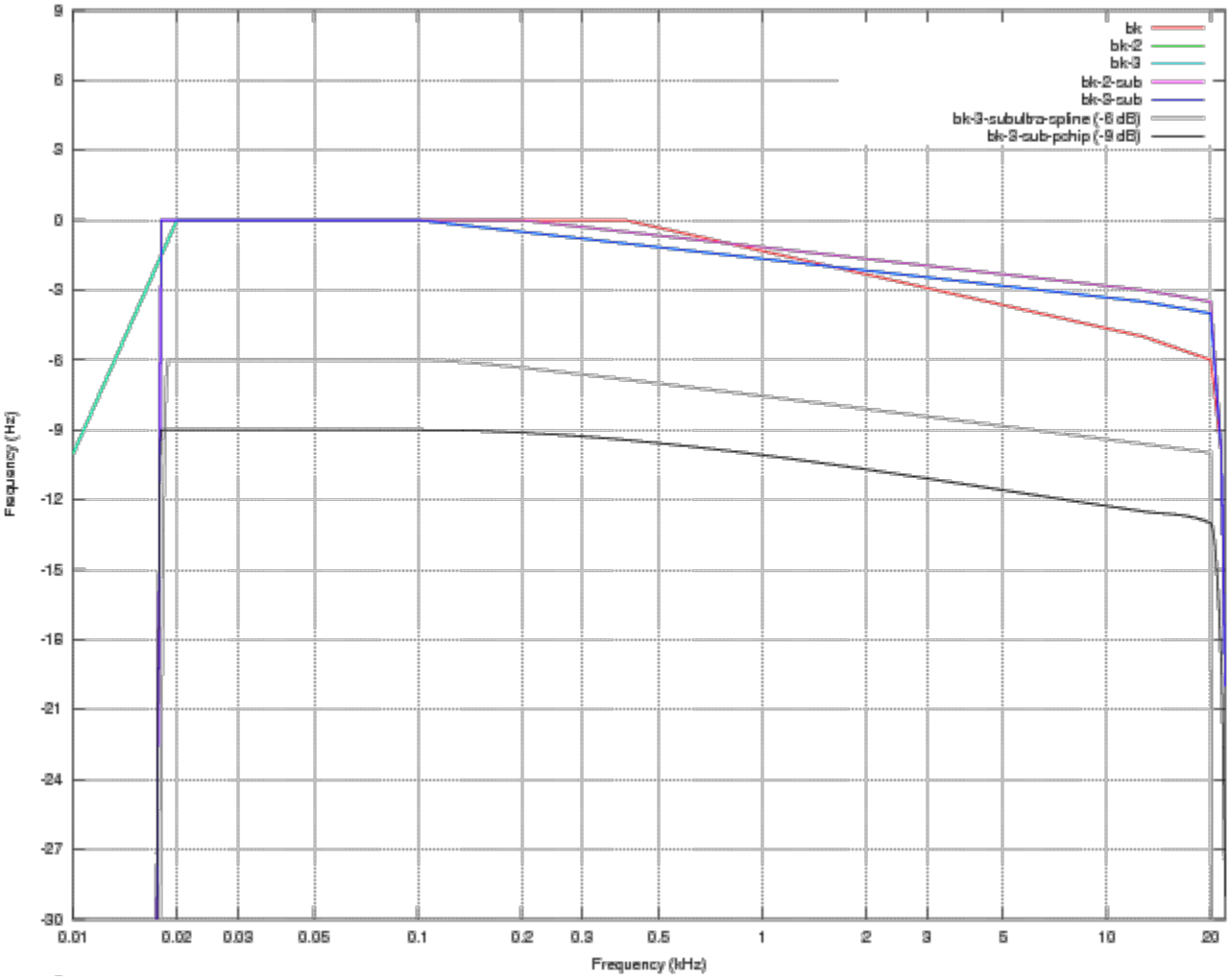
Los ficheros de configuración de ejemplo proporcionan un fichero de filtro (rps.pcm) en 32bit float normalizado a 1.0, adecuado para Brutefir.

5.2.1 Target (objetivo) magnitude response

Desde la versión 3.0.0 se aplica un target automáticamente en base a algunas asunciones psicoacústicas. Ver sección 4.3.

Por tanto, no debe ser necesario definir una curva target específica. En su lugar, debe usarse una curva target plana con alguna limitación subsónica y ultrasónica. Esto se consigue usando el target **pa-XX.X.txt** que es ahora la opción por defecto para todos los ficheros de configuración estándar, es solo una variación del target plano previo, ajustado para trabajar mejor con el procedimiento de interpolación de la función de transferencia “B Spline”.

Los viejos target están disponibles en el directorio source/target para ser usados si fuera requerido. Por supuesto, la etapa de post filtering debe ser usada para ajustar al gusto la magnitud de la respueta, pero para una reproducción neutra el target magnitud de la respueta deje ser dejado plano.



Los más importantes ficheros suministrados para el postfilterirg de la magnitud de respuesta target son **subultra-XX.X.txt, bk-XX.X.txt, bk-2-XX.X.txt, bk-3-XX.X.txt** (ver sección 6.12.7 “PSPointsFile”). Siendo XX.X la Fs.

subultra.txt proporciona solo una simple supresión de la sobre compensación en los extremos del rango de frecuancias y tiene un target lineal. No se ha pintado arriba.

bk.txt sigue las recomendaciones de Bruel & Kjaer para la respuesta de una sala, es decir lineal de 20 Hz a 400 Hz seguido por una lenta disminución de 1 dB/oct hasta 20 KHz

bk-2.txt es similar a bk.txt pero esta es lineal de 20 Hz a 200 Hz seguido por una lenta caída de 0.5 dB/oct hasta 20 KHz

bk-3.txt es algo intermedio entre bk y bk-2, pero lineal de 20 Hz a 100 Hz seguido por una lenta caída de 0.5 dB/oct hasta 20 KHz

Las versiones con el sufijo **sub** son iguales pero añadiendo un filtro subsónico de gran pendiente.

Las versiones con el sufijo **spline** incluyen un conjunto de puntos de control adecuados para el target “B Spline”.

La figura de arriba también muestra un ejemplo de interpolación PCHIP del target bk-3-sub.

En el directorio sample/ también se proporcionan algunos otros ejemplos de ficheros target de postfiltering.

En los ficheros con Fs > 44.1 KHz simplemente se ha movido el último punto de frecuencia hasta la frecuencia de Nyquist. Para la mayoría de ficheros de configuración hay o bien un suave roll-off a frecuencias altas o bien un un filtro supersónico brick-wall por encima de 20 KHz.

6 DRC Configuration file reference

Un fichero de configuración es un simple fichero ASCII con parámetros en la forma:

```
ParamName = value # comentarios opcionales para legibilidad
```

Cada parámetro tiene un prefijo de dos caracteres que define la etapa a la que el parámetro se refiere:

- BC = Base Configuration
- MC = Microphone Compensation stage
- HD = Homomorphic Deconvolution
- MP = Minimum phase Prefiltering stage
- DL = Dip Limiting stage
- EP = Excess phase Prefiltering stage
- PC = Prefiltering Completion stage
- IS = Inversion Stage
- PT = Psychoacoustic Target
- PL = Peak Limiting
- RT = Ringing Truncation stage
- PS = Postfiltering Stage
- MS = Minimum phase filter generation Stage
- TC = Test Convolution stage

DRC hace algunas comprobaciones para asegurar valores consistentes, pero no es a prueba de balas.

Los parámetros importantes están marcados con asterisco (*). A veces también se proporcionan los rangos de valores adecuados (siempre referidos a 44.1 KHz)

Muchos parámetros son potencias de 2 por razones de rendimiento en los cómputos FFT.

Los parámetros suministrados por defecto son para FS = 44.1 KHz.

6.1 BC - Base Configuration

6.1.1 BCBaseDir

This parameter define the base directory that is prepended to all file parameters, like for example BCInFile, HDMPOutFile or PSPointsFile. This parameter allow the implicit definition of a library directory where all DRC support file might be placed.
File parameters supplied on the command line are not affected by this parameter unless the BCBaseDir parameter is also supplied on the command line. File parameters supplied in the configuration file are instead always affected by the BCBaseDir parameter, no matter if it has been supplied in the configuration file or on the command line.

6.1.2 BCInFile
Just the name of the input file with the input room impulse response.

6.1.3 BCInFileType
The type of the input file. D = Double, F = Float, I = Integer.

6.1.4 BCSampleRate
The sample rate of the input file. Usually 44100 or 48000.

6.1.5 BCImpulseCenterMode
The impulse response impulse center may be set manually using the BCImpulseCenter parameter or you may ask DRC to try to find it automatically. If BCImpulseCenterMode is set to A DRC will look for the impulse center within the input file. If BCImpulseCenterMode is set to M DRC uses the impulse center supplied with the BCImpulseCenter parameter.
Be careful when using automatic impulse center recognition. Strong reflections or weird situations may easily fool the simple procedure used by DRC, which simply looks for the sample with the maximum absolute amplitude.

6.1.6 BCImpulseCenter (*)
This is the position, in samples, of the time axis zero of the impulse response read from BCInFile. Usually this is where the peak of the impulse response is, but for complex situations it might not be easy to identify where the zero is. Even few samples displacement in this parameter may cause high frequency overcorrection, causing too bright sound. If BCImpulseCenterMode is set to A this parameter is ignored.

6.1.7 BCInitWindow
Initial portion of the impulse response which is used to perform the correction. It should be long enough to accomodate for any subsequent windowing performed by DRC. The window is symmetrical with respect of the impulse center. If needed, the signal is padded with zeroes. Usual values are between 16384 and 131072, depending on the values of the parameters for the subsequent steps. This initial window may be further limited in subsequent steps, which sets the real window used.

6.1.8 BCPreWindowLen
This is the length of the window used to remove any noise coming before the impulse center. This is usually just few samples, with a typical value of 1024 samples, corresponding to 23.2 ms at a 44.1 kHz sample rate. If this value is 0 this step is skipped

6.1.9 BCPreWindowGap
This is the central flat gap left in the previous windowing operation. Usually it is set to 0.75 * *BCPreWindowLen*, i.e. 768 samples with the standard BCPreWindowLen value.

6.1.10 BCNormFactor

Initial normalization of the input impulse response. 0 means no normalization, which is the default.

6.1.11 BCNormType

Type of normalization applied. M means max normalization, i.e. the input signal is rescaled so that the maximum value of the samples is equal to the normalization factor. E means Euclidean normalization (L2 Norm), i.e. the input signal is rescaled so that the RMS value of the signal is equal to the normalization factor. S means sum normalization (L1 Norm), i.e. the input signal is rescaled so that the sum of the absolute values of the samples is equal to the normalization factor. P means peak normalization i.e. the input signal is rescaled so that the highest peak in the signal magnitude response is equal to the normalization factor.

6.1.12 BCDLType, BCDLMinGain, BCDLStartFreq, BCDLEndFreq, BCDLStart, BCDLMultExponent

These parameters are used to set a mild dip limiting on the input impulse response. For a detailed description of these parameters see the similar procedure described in section 6.5. This stage is used just to prevent overflow or underflow problems in subsequent stages so under standard conditions there is no need at all to change these parameters.

6.2 MC - Microphone Compensation

Within this stage the microphone transfer function is invertend and applied to the input impulse response to compensate for any microphone aberration. If you want a microphone compensated filter you have to enable this stage.
The inversion is carried out by direct inversion of the values supplied in the microphone compensation file. So it is assumed that the microphone response is easily invertible. This is usually true with any decent microphone.

6.2.1 MCFilterType

This is the type of filter used for the microphone compensation stage stage. N means that the mic compensation stage is disabled, L means linear phase filtering, M means minimum phase filtering.

6.2.2 MCInterpolationType

This parameter is the same as the PSInterpolationType parameter (see section 6.12.2) but applied to the mic compensation filter. The default is H.

6.2.3 MCMultExponent

The multiplier exponent used for the homomorphic deconvolution used to compute the minimum phase compensation filter. Usually a value of 2 or 3 is used.

6.2.4 MCFilterLen

Length of the FIR filter used for microphone compensation. Usually between 16384 and 65536.

6.2.5 MCNumPoints

Number of points used for the definitions of the microphone frequency response. If this parameter is 0 DRC automatically counts the number of lines in the microphone frequency response file. See the following parameters for details about the microphone frequency response compensation.

6.2.6 MCPointsFile

This is the name of the file which contains the microphone frequency response to be compensated. The file format is identical to the one defined for the target frequency response (see section 6.12.7). Again any phase specification get wiped out if minimum phase filtering is used. This usually isn't a problem because most microphones suited for measurement tasks are minimum phase systems, so the minimum phase compensation filter already has exactly the phase response needed to compensate for the microphone phase response.
In the sample directory there's a sample compensation file (wm-61a.txt) which is a generic compensation file for the Panasonic WM-61A electrect capsule. This file has been derived from average values available on the Internet, so don't expect to get perfect linear frequency response using it. There could be some difference among different capsules of the same type. In the same directory there's also a compensation file for the Behringer ECM8000 instrumentation microphone. This is the measured frequency response of a single unit, i.e. it isn't even derived from an average over many samples, so it may be even less reliable than the WM-61A compensation file.

6.2.7 MCMagType

This parameter selects how the amplitude of the target frequency response is defined. L means linear amplitude (0.5 means half the level, i.e about −6 dB), D means that the amplitude is expressed in dB.

6.2.8 MCFilterFile

This parameter set the file where the impulse response of the microphone compensation filter will be saved. This might be useful to take a look at the microphone compensation filter or to use it into some other program. By default it is disabled, i.e. commented out.

6.2.9 MCOutWindow

Final window after microphone compensation. Default value set to 0, i.e. no windowing applied.

6.2.10 MCNormFactor

Normalization factor for the microphone compensated impulse response. Usually 0.0, i.e. disabled.

6.2.11 MCNormType

Normalization type for the microphone compensated impulse response. Usually E.

6.2.12 MCOutFile

Output file for the microphone compensated impulse response. Disabled, i.e. commented out, by default. The file generated by enabling this parameter might be used as input for the “createdrcplots” Octave script to generate the uncorrected response graph using a microphone compensated uncorrected response (see section A for more details).

6.2.13 MCOutFileType

Output file type for the microphone compensated impulse response. D = Double, F = Float, I = Integer.

6.3 HD - Homomorphic Deconvolution

6.3.1 HDMultExponent

Exponent of the multiplier of the FFT size used to perform the homomorphic deconvolution. The FFT size used is equal to the first power of two greater than or equal to *BCInitWindow* * (2 *HDMultExponent*). Higher exponents give more accurate deconvolution, providing less circular convolution artifacts.
With older DRC versions achieving low circular artifacts was not so important because they were masked by the higher pre-echo artifacts in other steps. Starting with version 2.0.0 it is possible to achieve really low pre echo artifacts so circular artifacts now are an issue, because when truncated by the pre-echo truncation inversion procedure they may cause errors on the phase correction. In this situation a value of at least 3 is suggested.

6.3.2 HDMPNormFactor

Normalization factor for the minimum phase component. Usually 1.

6.3.3 HDMPNormType

Normalization type for the minimum phase component. Usually E.

6.3.4 HDMPOutFile

Output file for the minimum phase component. Usually not used (commented out).

6.3.5 HDMPOutFileType

Output file type for the minimum phase component. D = Double, F = Float, I = Integer.

6.3.6 HDEPNormFactor

Normalization factor for the excess phase component. Usually 1.

6.3.7 HDEPNormType

Normalization type for the excess phase component. Usually E.

6.3.8 HDEPOutFile

Output file for the excess phase component. Usually not used (commented out).

6.3.9 HDEPOutFileType

Output file type for the excess phase component. D = Double, F = Float, I = Integer.

6.4 MP - Minimum phase Prefiltering

6.4.1 MPPrefilterType

This parameter can be either B for the usual band windowing prefiltering stage or S for the sliding lowpass method. The first method splits the input response into log spaced bands and window them depending on some parameters but basically with a window length which decrease exponentially with the center frequency of the band. The sliding lowpass method instead filters the impulse response with a time varying lowpass filter with a cutoff frequency which decreases exponentially with the sample position with respect to the time axis zero. This last one is a stepless procedure.
Using either a lowercase b or s for the MPPrefilterType parameters enable the single side version of the prefiltering procedures. The procedure is applied starting from the impulse center, leaving the first half of the impulse response unchanged. This gives less pre-echo artifacts, and should be used when the pre-echo truncation inversion procedure is used. Please remember to set the prefiltering parameters to values which are adequate for the procedure used.

6.4.2 MPPrefilterFctn

This parameter sets the type of prefiltering function used, i.e. P for the usual inverse proportional function, or B for the bilinear transform based prefiltering function. For a comparison between the two functions see figure 6. The default is B.

6.4.3 MPWindowGap

This parameter changes a little the window function (Blackman) used for the band windowing prefiltering stage. It sets a small flat unitary gap, whose length is expressed in samples, at the center of the window function, so that even if the impulse center is slightly misaligned with respect to the time axis zero there is no high frequency overcorrection. For band windowing prefiltering procedure usually this overcorrection is in the order of 0.1 − 0.2 dB at 20 kHz for errors of 2 to 3 samples, so it is not important at all in real world situations, but if you want to fix even this small problem this parameter lets you do it.
MPWindowGap should never be more than 2 sample less than MPUpperWindow and it is usually no more than few samples (5 to 10). If in any situation it is bigger than the calculated window DRC automatically reduces the gap to 2 less than the applied window. When MPWindowGap is 0 DRC behaves exactly as in the older versions.
For the sliding lowpass procedure this sets just the window gap used for the initial windowing before the procedure starts.

6.4.4 MPLowerWindow (*)

Length of the window for the minimum phase component prefiltering at the bottom end of the frequency range. Longer windows cause DRC to try to correct a longer part of the impulse response but cause greater sensibility to the listening position. Typical

values are between 16384 and 65536. MPLowerWindow must not be greater than BCInitWindow.

6.4.5 MPUpperWindow (*)

Length of the window for the minimum phase component prefiltering at the upper end of the frequency range. Longer windows cause DRC to try to correct a longer part of the impulse response but cause greater sensibility on the listening position. Typical values are between 22 and 128. MPUpperWindow must be not greater than MPLowerWindow, and usually is much shorter than that.

6.4.6 MPStartFreq

Start frequency for the prefiltering stage. Usually 20 Hz or just something less.

6.4.7 MPEndFreq

End frequency for the prefiltering stage. Usually set at 20 kHz, i.e. 20000. Of course you must be using a sample rate which is greater than 40 kHz to set this above 20 kHz.

6.4.8 MPWindowExponent (*)

This is the exponent used in the frequency dependent window length computation for the band windowing procedure, or in the computation of the time dependent cutoff frequency for the sliding lowpass procedure.
The window length for band windowing is computed with the following expression:

$$W = 1 / [A * (F + Q)^{WE}]$$

Where W is the window length, F is the normalized frequency, WE is the window exponent, A and Q are computed so that W is equal to MPLowerWindow at MPStartFreq and is equal to MPUpperWindow at MPEndFreq. If you set MPLowerWindow equal to the value used for MPInitWindow in DRC 1.2, set MPWindowExponent to the same value of version 1.2 and set MPUpperWindow to the value you got at the upper limit of the frequency range in version 1.2 you should get results much similar to the 1.2 DRC release.

In a similar way the cutoff frequency for the sliding lowpass prefiltering stage is computed with:

$$F = 1 / [A * (W + Q)^{WE}]$$

with identical parameters but reversed perspective, i.e. the cutoff frequency is computed from the window length and not the other way around. In both cases W and F are considered normalized between 0 and 1.

These parametric functions are used when the proportional function is selected using the MPPrefilterFctn (see section 6.4.2) parameter. The parametric functions derived from the bilinear transformation are quite different and more complicated, so they aren't explained here.

Changing the window exponent provides different prefiltering curves, see section 4.2 for a deeper explanation. Increasing the window exponent gives higher correction in the midrange. Typical values are between 0.7 and 1.2.

6.4.9 MPFilterLen

Filter length, in taps, used to perform band splitting or sliding lowpass prefiltering of the input signal. Higher values give better filter resolution but require a longer computation. Typical values for band windowing are between 4096 and 65536. Sometimes may be useful to use short filters (64 - 512 taps) to get a more “fuzzy” correction at lower frequencies.
With the sliding lowpass procedure similar filters should be used. Usually the filter length is in the 512 - 65536 range. Short filters (16 - 64 taps) gives a similar fuzzy correction at the bottom end, but with a different behaviour than band windowing.

6.4.10 MPFSharpness (*)

This parameter applies only to the sliding lowpass prefiltering procedure and control the sharpness of the filtering performed in the filtered region of the time-frequency plane. A value of 1.0 provides the same behaviour of version 2.3.1 of DRC and provides the maximum allowable filtering sharpness without affecting the direct sound, but also creates a substantial amount of spectral spreading in the filter transition region of the time-frequency plane. Values above 1.0 increase the spectral spreading up to a point where it starts affecting also the direct sound, with the introduction of some ripple in the direct sound itself. Values below 1.0 reduce the spectral spreading in the filtered region at the expense of a little reduction in the filter sharpness. Typical values for this parameter are between 0.1 and 0.75, with a default value of 0.25.

6.4.11 MPBandSplit

Fractional octave splitting of band windowing. Band windowing is performed in 1 / *MPBandSplit* of octave bands. Usual values are between 2 and 6. The higher this value the higher should be MPFilterLen. Values greater than 6 usually give no improvements. For the sliding lowpass prefiltering this just gives the rate at which log messages are reported during the prefiltering procedure and has no effect on the prefiltering procedure itself, which is always stepless.

6.4.12 MPHDRecover

After prefiltering the minimum phase component may be no longer minimum phase, with a bit of excess phase component added. Setting this parameter to Y enable a second homomorphic deconvolution on the prefiltered minimum phase component to make it minimum phase again. This is important especially if the pre-echo truncation inversion procedure is used. This procedure assumes that the minimum phase part really is minimum phase, so skipping this step may cause it to fail in avoiding pre-echo artifacts.

6.4.13 MPEPPreserve

Setting this to Y causes the excess phase part of the filtered impulse response to be preserved after the MPHDRecover step. This excess phase part is then convolved with the excess phase part of the filtered impulse response to preserve it and invert it. This

provides a slight improvement in the direct sound phase response. The default value is Y.

6.4.14 MPHDMultExponent

Exponent of the multiplier of the FFT size used to perform the homomorphic deconvolution described above. The FFT size used is equal to the first power of two greater than or equal to *MPPFFinalWindow* * (2*MPHDMultExponent*). Higher exponents give more accurate results, but require a longer computation. Usually a value of 2 or 3 is used. If this parameter is less than 0 no multiplier will be used. Be careful because if the FFT size isn't a power of two the procedure can take a long time to complete.

6.4.15 MPPFFinalWindow

Final window of the prefiltering stage. Usually the same as MPLowerWindow or just something more. If set to 0 no windowing is applied.

6.4.16 MPPFNormFactor

Normalization factor for the minimum phase component after prefiltering. Usually 0.

6.4.17 MPPFNormType

Normalization type for the minimum phase component after windowing. Usually E.

6.4.18 MPPFOutFile

Output file for the minimum phase component after band windowing. Usually not used (commented out).

6.4.19 MPPFOutFileType

Output file type for the minimum phase component after windowing. D = Double, F = Float, I = Integer.

6.5 DL - Dip Limiting

6.5.1 DLType

To prevent numerical instabilities during the inversion stage, deep dips in the frequency response must be limited or truncated. This parameter sets the type of dip limiting performed. L means linear phase, i.e. it applies a linear phase filter that removes dips below a given threshold, M means minimum phase, i.e. it uses a minimum phase filter to achieve the same result. Both procedures are also available with a logarithmic frequency weighting of the magnitude response, so that, for example, the 20 Hz - 200 Hz range weigths the same as the 2000 Hz - 20000 Hz range. Setting DLType to P activates the log weighted verions of the linear phase dip limiting and setting DLType to W activates the log weighted verions of the minimum phase dip limiting.
Starting with version 2.0.0 DRC performs this step only on the prefiltered minimum phase part, just before performing the second homomorphic deconvolution, if enabled. So if the MPHDRecover parameter is set to Y and the MPEPPreserve parameter is set to N there is almost no difference between the two procedures, because the subsequent homomorphic deconvolution stage wipes out any phase difference giving

just a minimum phase signal. Any difference would be caused just by numerical errors.

6.5.2 DLMinGain

This is the minimum gain allowed in the frequency response of the prefiltered signal. Values lower than this will be truncated. Typical values are between 0.1 and 0.5. These are absolute values with respect to the RMS value, i.e. 0.1 is about −20 dB, 0.5 is about −6 dB.

6.5.3 DLStartFreq

Start frequency where the reference RMS level used for dip limiting is computed.

6.5.4 DLEndFreq

End frequency where the reference RMS level used for dip limiting is computed.

6.5.5 DLStart

Setting this parameter to a value between 0.0 and 1.0 enables the “soft clipping” dip limiting procedure. Everything below *DLStart* * *DLMinGain* , with respect to the RMS value, get rescaled so that it ends up between *DLStart* * *DLMinGain* and DLMinGain. Values for this parameter usually are between 0.5 and 0.95, with a typical value of 0.75. Setting this parameter to a value equal to or greater then 1.0 cause DRC to switch to hard clipping of the frequency response.

6.5.6 DLMultExponent

Exponent of the multiplier of the FFT size used to perform the dip limiting stage. The FFT size used is equal to the first power of two greater than or equal to (*MPBWFinalWindow* + *EPBWFinalWindow* − 1) * (2*DLMultExponent*) . Higher exponents give more accurate dip limiting, but requires a longer computation. Usually a value of 2 or 3 is used. If this parameter is less than 0 no multiplier will be used. Be careful because if the FFT size isn’t a power of two the procedure might take a long time to complete.

6.6 EP - Excess phase Prefiltering

The excess phase prefiltering is performed pretty much the same way as the minimum phase prefiltering, so the parameters are almost identical, even though with different values.

6.6.1 EPPrefilterType

Same as MPPrefilterType but for the excess phase component.

6.6.2 EPPrefilterFctn

Same as MPPrefilterFctn but for the excess phase component.

6.6.3 EPWindowGap

Same as MPWindowGap but for the excess phase component.

6.6.4 EPLowerWindow (*)

Same as MPLowerWindow but for the excess phase component. Typical values are between 1024 and 4096. As a rule of thumb you can take:

EPLowerWindow = MPLowerWindow / A

with *A* going from 16 to 32 and a typical value of 24. EPLowerWindow must be not greater than BCInitWindow.

6.6.5 EPUpperWindow (*)

Same as MPUpperWindow but for the excess phase component. Typical values are between 22 and 128. As a rule of thumb you can take:

EPUpperWindow = MPUpperWindow

6.6.6 EPStartFreq

Start frequency for the prefiltering stage. Usually 20 Hz or just something less.

6.6.7 EPEndFreq

End frequency for the prefiltering stage. Usually set at 20 kHz, i.e. 20000. Of course you must be using a sample rate which is greater than 40 kHz to set this above 20 kHz.

6.6.8 EPWindowExponent (*)

Same as MPWindowExponent but for the excess phase component. See discussion on MPWindowExponent. Usual values for this parameter are between 0.5 and 1.2, depending on the value of EPInitWindow. As a rule of thumb you can take:

EPWindowExponent = MPWindowExponent

6.6.9 EPFilterLen

Filter length, in taps, used to perform band splitting of the input signal or sliding lowpass prefiltering. Higher values gives better filter resolution but require a longer computation. Typical values for band windowing are between 4096 and 65536. Sometimes may be useful to use short filters (64 - 512 taps) to get a more “fuzzy” correction at lower frequencies.
With the sliding lowpass procedure similar filters should be used. Usually the filter length is in the 512 - 65536 range. Short filters (16 - 64 taps) gives a similar fuzzier correction at the bottom end, but with a different behaviour than band windowing.

This value is usually equal to MPFilterLen.

6.6.10 EPFSharpness (*)

Same as MPFSharpness but applied to the excess phase part.

6.6.11 EPBandSplit

Fractional octave splitting of band windowing. Band windowing is performed in 1 / *MPBandSplit* of octave bands. Usual values are between 2 and 6. The higher this value the higher should be MPFilterLen. Values greater than 6 usually give no improvements. For the sliding lowpass prefiltering this just gives the rate at which log messages are reported and has no effect on the prefiltering procedure, which is always stepless.

This value is usually equal to MPBandSplit.

6.6.12 EPPFFinalWindow

Final window of the prefiltering stage. Usually the same as EPInitWindow or just something more. If set to 0 no windowing is applied.

6.6.13 EPPFFlatGain

After band windowing the excess phase component usually need reequalization to get the flat frequency response it must have. This is the gain applied with respect to the RMS level of the signal to get this flat frequency response. Usually 1, a value of 0 disables this step. Skipping this step, i.e. setting this parameter to 0, usually gives bad results.

6.6.14 EPPFOGainFactor

This parameter controls how the excess phase flattening set by the previous parameter is performed. Setting this to 0 tries to get a perfectly flat excess phase component, as in version 1.3.0 of DRC. This parameter has been introduced to balance between the need of a flat excess phase response and a perfect control of the direct sound, usually achieved without any flattening. Unfortunately so far the supposed balance always proved really difficult to find in any real world situation, so this parameter is always set to 0 in the standard configuration file. The procedure has been left just for experimental purposes if some unusual situation need to be handled.
Furthermore this parameter applies only to the linear phase and minimum phase excess phase flattening, it isn't available for the D type of excess phase flattening.

6.6.15 EPPFFlatType

This is the type of procedure adopted for the excess phase component renormalization. L means applying linear phase renormalization, M means applying minimum phase renormalization, D means applying another homomorphic deconvolution stage to extract just the excess phase component of the prefiltered excess phase component. L applies a linear phase filter that equalizes the excess phase amplitude response to flat, M means minimum phase, i.e. it uses a minimum phase filter to achieve the same result. The D procedure provides the same effect of the M procedure when EPPFOGainFactor is equal to 0. Any difference is just caused by numerical errors.

6.6.16 EPPFFGMultExponent

Exponent of the multiplier of the FFT size used to perform the frequency response flattening. The FFT size used is equal to the first power of two greater than or equal to *EPBWFinalWindow* * (2 *EPPFFGMultExponent*) . Higher exponents give more accurate results, but require a longer computation. This parameter should be set using the same criteria described in HDMultExponent. If this parameter is less than 0 no multiplier will

be used. Be careful because if the FFT size isn't a power of two the procedure might take a long time to complete.

6.6.17 EPPFNormFactor

Normalization factor for the excess phase component after band windowing. Usually 0, i.e. disabled.

6.6.18 EPPFNormType

Normalization type for the excess phase component after windowing. Usually E.

6.6.19 EPPFOutFile

Output file for the excess phase component after windowing. Usually not used (commented out).

6.6.20 EPPFOutFileType

Output file type for the excess phase component after windowing. D = Double, F = Float, I = Integer.

6.7 PC - Prefilter Completion

The prefilter completion stage combines the prefiltered minimum phase and excess phase parts together again. The impulse response recovered after prefilter completion defines the impulse response of the system as seen by the correction applied by DRC.

6.7.1 PCOutWindow

Final window after prefiltering completion stage and before impulse inversion. This is usually between 8192 and 65536. Values greater than 65536 make no sense, giving a filter resolution lower than 1 Hz at a 44.1 kHz sample rate. Furthermore inversion of signals longer than 65536 samples may require a lot of time. Starting with version 2.0.0 this step is no longer needed with the pre-echo truncation fast deconvolution inversion method, which works directly on the minimum and excess phase components from the prefiltering stages. So if PCOutFile is not defined and ISType is set to T this step is completely skipped.

6.7.2 PCNormFactor

Normalization factor for the prefiltered signal. Usually 0, i.e. disabled.

6.7.3 PCNormType

Normalization type for the prefiltered signal. Usually E.

6.7.4 PCOutFile

Output file for the prefiltered signal. Usually not used (commented out).

6.7.5 PCOutFileType

Output file type for the prefiltered signal. D = Double, F = Float, I = Integer.

6.8 IS - Inversion Stage

6.8.1 ISType

Type of inversion stage. L uses the usual Toeplitz least square inversion, T activates the pre-echo truncation fast deconvolution.

6.8.2 ISPETType

This sets the type of pre echo truncation applied when ISType is T. f means a fixed pre-echo truncation, s means a time dependent pre-echo truncation applied using the usual single side sliding low-pass procedure, but with reversed behaviour, i.e. only what comes before the impulse center is processed. Starting with version 2.7.0 this is set to f and pre-echo truncation is basicly disabled because it is already carried out by the excess phase prefiltering procedure.

6.8.3 ISPrefilterFctn

Same as MPPrefilterFctn but for the pre-echo truncation windowing. It is used only when ISPETType is set to s.

6.8.4 ISPELowerWindow

When ISPETType is f this is the number of samples before the impulse center where the inverted impulse response is considered pre-echo. Starting with version 2.7.0 this is is usually set to half the value of EPLowerWindow so that the pre-echo truncation procedure provides just a mild windowing. When ISPETType is s this is the number of samples considered pre-echo at the ISPEStartFreq frequency, with a typical value equal to EPLowerWindow.

6.8.5 ISPEUpperWindow

When ISPETType is f this is the number of samples before the impulse center where the pre-echo region, defined by the previous parameter, ends, and the full impulse response of the inverted filter should start. Starting with version 2.7.0 this is usually set to about 0.75 * *ISPELowerWindow* so that the pre-echo truncation procedure is limited to a mild windowing used only to avoid small steps in the impulse response attack caused by small numerical errors. When ISPETType is s this is the number of sample considered pre-echo at the ISPEEndFreq frequency, with a typical value equal to EPUpperWindow.

6.8.6 ISPEStartFreq

Start frequency for the sliding low pass pre-echo truncation procedure. Usually 20 Hz. Used only when ISPETType is s.

6.8.7 ISPEEndFreq

End frequency for the sliding low pass pre-echo truncation procedure. Usually 20000 Hz. Used only when ISPETType is s.

6.8.8 ISPEFilterLen

Length of the filter used for the pre-echo truncation sliding lowpass procedure. Usually 8192. Used only when ISPETType is s.

6.8.9 ISPEFSharpness

Same as MPFSharpness, but applied to the inversion stage pre-echo truncation. Here slightly bigger values usually provide better results because of the shorter windowing. Used only when ISPETType is s. The default value is 0.5.

6.8.10 ISPEBandSplit

For the sliding lowpass prefiltering this just gives the rate at which log messages are reported and has no effect on the prefiltering procedure, which is always stepless. Used only when ISPETType is s.

6.8.11 ISPEWindowExponent

Window exponent applied to the pre-echo truncation sliding lowpass procedure. Usual values goes from 0.5 to 1.5, with a typical value of 1.0. Used only when ISPETType is s.

6.8.12 ISPEOGainFactor

This parameter has the same effect of the EPPFOGainFactor (see section 6.6.14) but applied to the renormalization of the excess phase part of the inverse filter after pre-echo truncation. Used in conjunction with the EPPFOGainFactor parameter, this parameter can be used to balance the amount of correction applied to the direct sound compared to the amount of correction applied to the reverberant field. A negative value disables the renormalization. Default is 0.0.

6.8.13 ISSMPMultExponent

This is the exponent of the multiplier for the S inversion stage, using the longest of the input and output signals as a basis. This parameter should be set using the same criterion used for the MPHDMultExponent parameters and a values of at least 3 is suggested.

6.8.14 ISOutWindow

Final window after inversion stage. Usually 0, i.e. disabled, with the L type inversion stage. With the S type this is the output filter size and can be any length but usually is between 8192 and 65536. If it is 0 than a length equal to *MPPFFinalWindow* + *EPPFFinalWindow* − 1 , i.e. the length of the convolution of the two components together, is assumed and no windowing is applied to the output filter.

6.8.15 ISNormFactor

Normalization factor for the inverted signal. Usually 0, i.e. disabled.

6.8.16 ISNormType

Normalization type for the inverted signal. Usually E.

6.8.17 ISOutFile

Output file for the inverted signal. Usually not used (commented out).

6.8.18 ISOutFileType

Output file type for the inverted signal. D = Double, F = Float, I = Integer.

6.9 PT - Psychoacoustic Target

This stage computes a psychoacoustic target response based on the magnitude response envelope.

6.9.1 PTType

Defines the type of psychoacoustic target filter to use. N means no filter, thus skipping the psychoacoustic target stage completely. M means that a minimum phase filter is used and L means that a linear phase filter is used. The default is M.

6.9.2 PTReferenceWindow (*)

This parameter define the size used to window the corrected impulse response. The windowed response is then used to compute the magnitude response envelope that the target response is based upon. Usually a portion of the impulse response going from 150 ms to 500 ms is used. The default value is 26460, corresponding to a symmetric window, 300 ms long on each side, at 44100 Hz sample rate.

6.9.3 PTDLType, PTDLMinGain, PTDLStartFreq, PTDLEndFreq, PTDLStart, PTDLMultExponent

These parameters are used to set a small dip limiting on the corrected impulse response in order to avoid numerical problems in the inversion of the magnitude response envelope. For a detailed description of these parameters see the similar procedure described in section 6.5. This stage is used just to prevent overflow or underflow problems so under standard conditions there is no need at all to change these parameters.

6.9.4 PTBandWidth (*)

This parameter define the resolution used for the computation of the magnitude response envelope. It is defined as fraction of octaves, so a value of 0.25 means a resolution of 1/4 of octave. Values below 0 down to -1 causes the adoption of the Bark scale, values below -1 causes the adoption of the ERB scale. The default value is -2, which means that the computation is performed on the ERB scale.

6.9.5 PTPeakDetectionStrength (*)

This parameter define how close the magnitude response envelope will be to to the peaks in the unsmoothed spectrum. Higher values provide a closer match. Typical values are between 5 and 30, with the default value, based on documented psychoacoustic assumptions, set to 15. Values above 50 are probably going to cause numerical problems and should be avoided.

6.9.6 PTMultExponent

Multiplier exponent for the computation of the magnitude response envelope. Default is 0.

6.9.7 PTFilterLen

Length of the psychoacoustic target filter. Default set to 65536.

6.9.8 PTFilterFile

Output file for the psychoacoustic target filter. Usually not used (commented out).

6.9.9 PTFilterFileType

Output file type for the psychoacoustic target filter. D = Double, F = Float, I = Integer.

6.9.10 PTNormFactor

Normalization factor for the inverted signal after convolution with the psychoacoustic target filter. Usually 0, i.e. disabled.

6.9.11 PTNormType

Normalization type for the inverted signal after convolution with the psychoacoustic target filter. Usually E.

6.9.12 PTOutFile

Output file for the inverted signal after convolution with the psychoacoustic target filter. Usually not used (commented out).

6.9.13 PTOutFileType

Output file type for the inverted signal after convolution with the psychoacoustic target filter. D = Double, F = Float, I = Integer.

6.9.14 PTOutWindow

Normalization factor for the inverted signal after convolution with the psychoacoustic target filter. Usually 0, i.e. disabled.

6.10 PL - Peak Limiting

The peak limiting stage limits the maximum allowed gain of the filter to prevent amplification and speaker overload.

6.10.1 PLType

Type of peak limiting applied. L means linear phase, M means minimum phase, P means log weighted linear phase, W means log weighted minimum phase. For an explanation of the log weighted versions of the procedures see section 6.5.1. If PSFilterType is set to T PLType should be set to M or W to ensure that the initial zero valued part is preserved. Since version 3.1.2 the default and suggested value for this parameter is W.

6.10.2 PLMaxGain

Maximum gain allowed in the correction filter. Peaks in the correction filter amplitude response greater than this value will be compressed to PLMaxGain. Typical values are between 1.2 and 4. These are absolute values with respect to the RMS value, i.e. 1.2 is about 1.6 dB and 4 is about 12 dB. This peak limiting stage is used to prevent speaker or amplifier overloading, resulting in dynamic range limitations which are subjectively

worse than some narrow dip in the frequency response. A typical value is 2.0, i.e. 6 dB.

6.10.3 PLStart

Setting this parameter to a value between 0.0 and 1.0 enables the “soft clipping” peak limiting procedure. Everything above *PLStart* * *PLMaxGain* , with respect to the RMS value, get rescaled so that it ends up between *PLStart* * *PLMaxGain* and about PLMaxGain. Values for this parameter usually are between 0.5 and 0.95, with a typical value of 0.75. Setting this parameter to a value equal to or greater than 1.0 switch to hard clipping of the magnitude response.

6.10.4 PLStartFreq

Start frequency where the reference RMS level used for peak limiting is computed. Default value set to 20 Hz.

6.10.5 PLEndFreq

End frequency where the reference RMS level used for peak limiting is computed. Default value set to 20000 Hz.

6.10.6 PLMultExponent

Exponent of the multiplier of the FFT size used to perform the peak limiting stage. The FFT size used is equal to the first power of two greater than or equal to *PSOutWindow* * (2 *PLMultExponent*) . Higher exponents give more accurate peak limiting, but requires a longer computation. Usually a value of 2 or 3 is used. If this parameter is less than 0 no multiplier will be used. Be careful because if the FFT size isn’t a power of two the procedure can take a long time to complete.

6.10.7 PLOutWindow

Final window after peak limiting. Usually 0, i.e. disabled.

6.10.8 PLNormFactor

Normalization factor for the final filter. Usually 0, i.e. disabled.

6.10.9 PLNormType

Normalization type for the peak limited filter, usually E.

6.10.10 PLOutFile

Output file for the peak limited filter. Usually disabled (commented out).

6.10.11 PLOutFileType

Output file type for the final filter. D = Double, F = Float, I = Integer.

6.11 RT - Ringing Truncation

The ringing truncation stage applies a further frequency dependent windowing to the correction filter. The truncation parameters are pretty similar to those of the prefiltering stage and usually have also much similar values.

6.11.1 RTType

This parameter can be either B or b for the band windowing method, S or s for the sliding lowpass method or N to disable the ringing truncation stage. See section 4.2 and 6.4.1 for further details.

6.11.2 RTPrefilterFctn

Same as MPPrefilterFctn but for the ringing truncation windowing.

6.11.3 RTWindowGap

This parameter changes a little the window function (Blackman) used for the band windowing or the sliding lowpass windowing. See section 6.4.3 for further details.

6.11.4 RTLowerWindow (*)

Length of the window at the bottom end of the frequency range. Usually set to the same value of MPLowerWindow.

6.11.5 RTUpperWindow (*)

Length of the window at the upper end of the frequency range. Usually set to the same value of MPUpperWindow.

6.11.6 RTStartFreq

Start frequency for the windowing. Usually 20 Hz or just something less.

6.11.7 RTEndFreq

End frequency for the windowing. Usually set to 20000.

6.11.8 RTWindowExponent (*)

This is the exponent used in the frequency dependent window length computation for the band windowing procedure, or in the computation of the time dependent cutoff frequency for the sliding lowpass procedure. See section 6.4.8 for further details. Usually set to the same value of MPWindowExponent.

6.11.9 RTFilterLen

Filter length, in taps, used to perform band splitting or sliding lowpass prefiltering of the input signal. Usually the same as the one used in the prefiltering stage.

6.11.10 RTFSharpness (*)

This parameter applies only to the sliding lowpass prefiltering procedure and control the sharpness of the filtering performed in the filtered region of the time-frequency plane. See section 6.4.10 for further details.

6.11.11 RTBandSplit

Fractional octave splitting of band windowing. See section 6.4.11 for further details.

6.11.12 RTOutWindow

Final window of the stage. Usually set to 0, i.e. disabled.

6.11.13 RTNormFactor
Normalization factor for the minimum phase component after windowing. Usually 0.

6.11.14 RTNormType
Normalization type for the minimum phase component after windowing. Usually E.

6.11.15 RTOutFile
Output file for the filter after windowing. Usually not used (commented out).

6.11.16 RTOutFileType
Output file type for the filter after windowing. D = Double, F = Float, I = Integer.

6.12 PS - Postfiltering Stage
During the postfiltering stage the final target transfer function is applied to the filter and the filter is normalized to suitable values for the convolver used.

6.12.1 PSFilterType
This is the type of filter used for the postfiltering stage. L means the usual linear phase filtering, M means minimum phase filtering, T means minimum phase filtering with initial zero truncation. If the pre-echo truncation inversion is used and the final post filtering stage is minimum phase all the filter taps before ISPELowerWindow are zero (there could be some roundoff errors that make them different from zero, but considering them zero makes no difference for our needs). So this initial all zero part can be windowed out without changing the filter behaviour. This way the filter becomes almost zero delay, providing a delay of just ISPELowerWindow samples. This sometimes may be low enough to make it usable even with home theater systems where audio delay is a major issue. Of course to ensure that the initial all zero part is preserved the minimum phase peak limiting should also be used.

6.12.2 PSInterpolationType
This parameter defines the type of interpolation used between the points of the target transfer function. L means the usual linear interpolation, G means logarithmic interpolation, i.e. interpolation performed on a bilogarithmic scale, R means interpolation using Uniform Cubic B Splines, S means interpolation using Uniform Cubic B Splines on a bilogarithmic scale, P means interpolation on a linear scale using a monotone Piecewise Cubic Hermite Interpolating Polynomial (PCHIP), H means interpolation on a logarithmic scale using PCHIP. The logarithmic interpolation makes the definition of the target transfer function easier, without the need to define intermediate points to get the desired behaviour on a bilogarithmic scale. The default is S.
The B Splines interpolation options allow for the definition of smooth target transfer functions which provides less ringing. Be careful when using this option because defining the right control points to get the desired target transfer function might be tricky. B Splines don’t interpolate the supplied points but are instead tangent to the lines connecting the control point. If you want sharp corners in the transfer function just

place few close control points near to the desired corner. Remember that B Splines of the type used are unaffected by control points which are more than two control points away from any given point on the curve. Take a look at the supplied examples for some simple transfer function definition.

The PCHIP procedure provides a monotonic interpolation procedure. The resulting target is less smooth than the one supplied by B Splines, but being a true interpolation PCHIP makes the definition of the control points much easier.

The use of the B Spline or PCHIP interpolation procedures is often useful also for the definition of the mic compensation transfer function, especially when only few points are available.

6.12.3 PSMultExponent

The multiplier exponent used for the homomorphic deconvolution used to compute the minimum phase post filter. Usually a value of 2 or 3 is used.

6.12.4 PSFilterLen

Length of the FIR filter used during the postfiltering stage. Usually between 16384 and 65536.

6.12.5 PSNumPoints

Number of points used for the definition of the post filter frequency response. If this parameter is 0 DRC automatically counts the number of lines in the post filter definition file. See the following parameters for further details about the post filter frequency response.

6.12.6 PSMagType

This parameter selects how the amplitude of the target frequency response is defined. L means linear amplitude (0.5 means half the level, i.e about −6 dB), D means that the amplitude is expressed in dB.

6.12.7 PSPointsFile (*)

File containing the post filter frequency response definition. This file should contain PSNumPoints lines, each line in the form “Frequency Gain”, with the gain expressed as a linear gain or in dB depending on the PSMagType parameter value. The following examples are in dB. The first line must have a frequency equal to 0, the last line must have a frequency equal to *BCSampleRate* / 2 . A post filter definition file must have the following format:

```
0 -40
18 -20
20 0
20000 0
21000 -40
22050 -100
```

This is for a 44.1 kHz sample rate.

The post filter stage is usually used to prevent overcompensation in the subsonic or ultrasonic range, but may be used also to change the target frequency response from linear to a more euphonic one.

Starting from version 2.0.0 DRC lets you specify the phase for the target post filter stage. Phase specification should be placed after the amplitude specification and should be expressed in degrees. Following the example above:

```
0 -40 0
18 -20 45
20 0 90
20000 0 180
21000 -40 90
22050 -100 0
```

If not specified a value of 0 is assumed. Setting a phase different than 0, i.e. flat, is useless within normal HiFi systems in almost all circumstances. Furthermore the phase specification is used only if the PSFilterType is L, else any phase specification is wiped out by the minimum phase filter extraction.

6.12.8 PSOutWindow

Final window after post filtering. This is also the length of the generated correction filter. Usual values are between 8192 and 65536. Filter with 65536 taps gives about 0.5 Hz resolution at 44.1 kHz sample rate, 16384 is usually enough for most situation and 8192 gives somewhat good results with much less computing needs during real time convolution.

6.12.9 PSNormFactor

Normalization factor for the correction filter. Usually 1.0. See section 4.6.2 for some instructions on how to set this parameter.

6.12.10 PSNormType

Normalization type for the correction filter. Usually E. See section 4.6.2 for some instructions on how to set this parameter.

6.12.11 PSOutFile

Output file for the correction filter. This file contains the filter to be used with the convolution engine.

6.12.12 PSOutFileType

Output file type for the correction filter. D = Double, F = Float, I = Integer.

6.13 MS - Minimum phase filter extraction Stage

The minimum phase extraction stage creates a minimum phase filter from the correction filter. A minimum phase filter corrects just the magnitude response and the minimum phase part of the phase response, but it is usually almost artifacts free and as basically zero latency. If microphone compensation is enabled the filter includes microphone

compensation.

6.13.1 MSMultExponent

Exponent of the multiplier for the homomorphic deconvolution used to extract a zero delay minimum phase version of the correction filter. A value of 2 or 3 is usually enough.

6.13.2 MSOutWindow

Output window size for the minimum phase filter. Typical values are about half of PLOutWindow.

6.13.3 MSFilterDelay

This parameter add an initial delay to the filter, making it possible to align it with other filters. Usually it is set to the same value assigned to EPLowerWindow, so that the filter has the same latency of the mixed phase filter when PSFilterType is set to T. If you want a zero delay filter set this parameter to 0.

6.13.4 MSNormFactor

Normalization factor for the minimum phase filter. The same considerations of section 4.6.2 should be applied.

6.13.5 MSNormType

Normalization type for the minimum phase filter. Usually E. See section 4.6.2 for some instructions on how to set this parameter.

6.13.6 MSOutFile

Output file name for the minimum phase filter.

6.13.7 MSOutFileType

Output file type for the minimum phase filter.

6.14 TC - Test Convolution

6.14.1 TCNormFactor

Normalization factor for the output of the final convolution stage. Usually 0.0.

6.14.2 TCNormType

Normalization type for the output of the final convolution stage. Usually E.

6.14.3 TCOutFile

Output file for the final test convolution. If this is not supplied the test convolution stage is skipped.

6.14.4 TCOutFileType

Output type for the file above. D = Double, F = Float, I = Integer.

8 Similar software

There are other software packages providing functionalities similar or comparable to those of DRC. Here are some examples:

- Acourate: http://www.acourate.com/
- Room Eq Wizard: http://www.hometheatershack.com/roomeq/
- Audiolense: http://www.juicehifi.com/

9 Commercial products

A complete commercial package, based on DRC for the filter generation procedure, is available from the small Italian company AVA Italy. I have no involvement in the development of the product so those interested in this package should contact AVA Italy directly. Contact informations are available on the AVA Italy web site:

http://www.avaitaly.it/

A Resultados de muestra (ver en la página web o en la doc del paquete)