DRC Digital Room Correction

Imprimer au format PDF

From Wikipedia :

> Digital room correction (or DRC) is a process in the field of acoustics where digital filters designed to ameliorate unfavorable effects of a room's acoustics are applied to the input of a sound reproduction system. Modern room correction systems produce substantial improvements in the time domain and frequency domain response of the sound reproduction system.

DRC by Denis Sbragion

Denis Sbragion proposes a set of tools to measure the Impulse Response of a system and to make a digital correction.

In a first time, it is the measurement tools I used.

The full documentation is available here

The tools :

- glsweep = sine log sweep and inverse function generator (see IR page)
- lsconv = impulse response extraction by convolution
- drc = digital correction

Installation

Under Debian, drc is available from repository.

Under Manjaro, download the .tar file

- Extract the archive
- In the "sources" folder, open a terminal and enter "make" (glsweep, lsconv, drc are compiled); enter "sudo make install" to install them (they are copied in /usr/bin)

glsweep

Usage: glsweep rate amplitude hzstart hzend duration silence leadin leadout sweepfile inversefile

Parameters:

rate: reference sample rate amplitude: sweep amplitude hzstart: sweep start frequency hzend: sweep end frequency duration: sweep duration in seconds silence: leading and trailing silence duration in seconds leadin: leading window length as a fraction of duration leadout: trailing window length as a fraction of duration sweepfile: sweep file name inversefile: inverse sweep file name

Example:

- `glsweep 44100 0.5 10 21000 45 2 0.05 0.005 sweep.pcm inverse.pcm`
- `glsweep 44100 0.8 20 20000 10 2 0.05 0.005 sweep.pcm inverse.pcm`
- `glsweep 44100 0.8 20 18000 5 1 0.05 0.05 sweep2018k5s.pcm inverse2018k5s.pcm`
- `glsweep 44100 0.8 20 20000 0.3 0.1 0.05 0.05 sweep2020k03s.pcm inverse2020k03s.pcm`
- `glsweep 44100 0.8 20 20000 1.6 0.15 0.05 0.05 sweep2020k1d6s.pcm inverse2020k1d6s.pcm`

Convert .pcm in .wav with SOX

SOX documentation

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

Remark : the "inverse file contains very low amplitudes. Use -n option and gain to normelize

```
sox -t f32 -r 44100 -c 1 inverse.pcm -t wav -c 1 inverse.wav gain -n -1
```

Convertir de wav en pcm avec sox

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

or to extract from stereo to mono:

- `sox ZOOM0003.WAV -t f32 ramb0003_200210_micleftt.pcm remix 1` => for left

- `sox ZOOM0003.WAV -t f32 ramb0003_200210_micright.pcm remix 2` => for right

lsconv

Usage: LSConv sweepfile inversefile outfile [refsweep mingain [dlstart]]

Parameters:

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

To get the impulse response without the use of a reference channel just use something like:

```
lsconv recorded.pcm inverse.pcm impulse.pcm
```

drc

When an IR is available why not testing a digital room correction.

Due to the number of parameters and the length of documentation for DRC, an example of use is welcome : see in this forum

In short:

> DRC-FIR needs the impulse response in a “raw” format as 32-bit floating point numbers. I use sox to convert from WAV to the raw format:

```
$ sox impulse.wav -t f32 impulse-44100.pcm rate -v -s 44100
```

> Here I’ve converted the WAV file to raw bytes and downsampled it to 44100. DRC-FIR has a ton of adjustable parameters that affect how the correction is computed, but it also provides a set of standard configuration files with preset parameters with different levels of correction. These are named: minimal, erb, soft, normal, strong, extreme, and insane. These config files are provided for 44.1, 48, 88.2 and 96 kHz sample rates. So finally, to turn the measured impulse response into a correction filter, the drc command is

```
$ drc --BCInFile=impulse-44100.pcm --PSOutFile=filter-l-44100.pcm --MCPointsFile=7032857.txt "/usr/share
/drc/config/44.1 kHz/normal-44.1.drc"
```

> Breaking it down: BCInFile is our input impulse response measurement, PSOutFIle is where the correction filter will be saved, MCPointsFile is the microphone frequency response correction file (You can just leave this empty if you don’t have a correction file) and the final file name on the line is the DRC-FIR config file with the parameters that will be used. By default, drc uses a “psychoacoustic” target file, but several target files, such as the popular B&K target, are also provided, or you could make your own target file (it’s just frequency/amplitude pairs).

My first use : `drc --BCInFile=201028_IR_L.pcm --PSOutFile=201028_filter-L-44100.pcm "./44.1 kHz/normal-44.1.drc"`

The filter is then also generated for the right channel.

Both are converted from .pcm to .wav and then put together in a stereo .wav file.

The file is downloaded in the convolver of PulseEffect.

PORC Python Open Room Correction

PORC repo in Github

DRC (DRC or PORC), A rhythmbox plugin for digital room correction.

DRC rhythmbox plugin repo in Github

The repo details the installation procedure which is quiet simple :

- download the repo in a folder
- open a terminal
- `sudo make install`
- open Rhythmbox
- in settings, plugin check the DRC box

By ChristianV