Parte I - Manipulação de ficheiros de áudio WAV

1 - *wav_hist*

```
Usage: ./wav_hist <input file> <option>
       List of available options:
       -c <channel> [k]  for desired channel and optional constant k for bins with 2^k values
       -m           for MID
       -s           for SIDE
```

O programa ***wav_hist*** permite apresentar um histograma das amplitudes de um ficheiro de áudio WAV à partir de uma lista de opções de histogramas disponíveis:

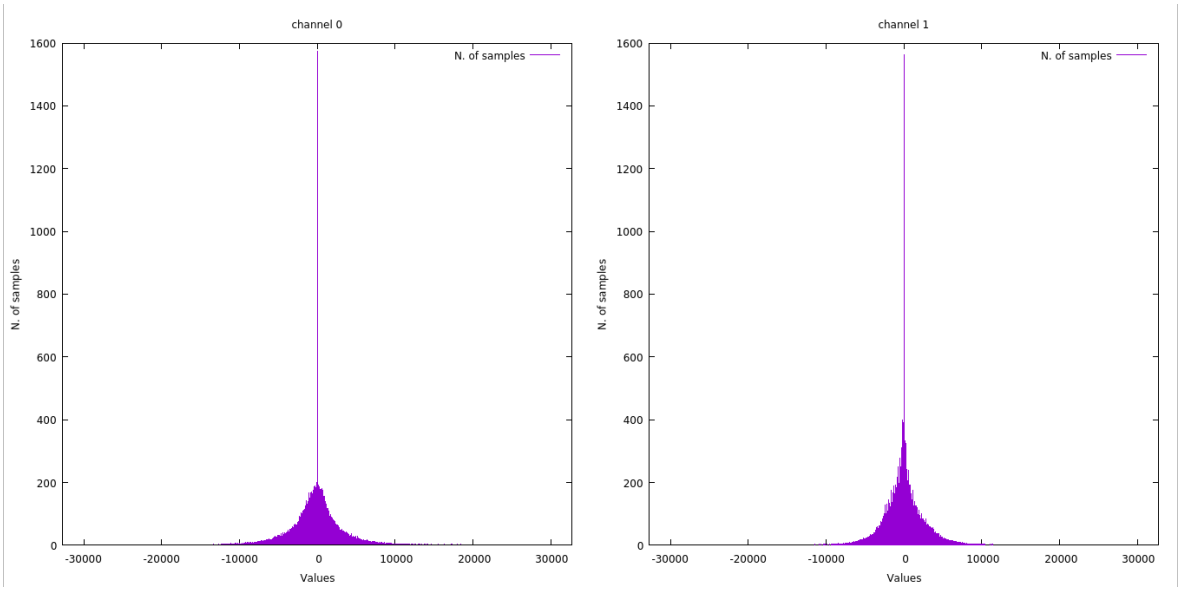
- **-c (channel)**: permite selecionar o canal desejado para o histograma e ainda definir uma constante k opcional para especificar os ***coarser bins*** (agrupar as *samples* em ***bins*** de 2^k valores)

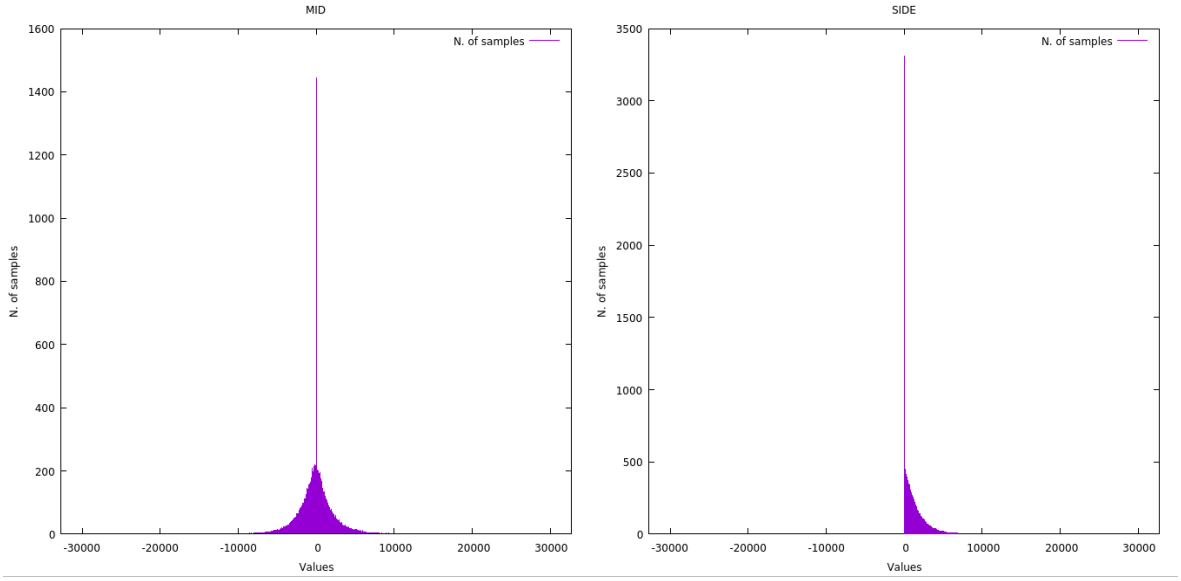
- **-m (MID channel)**: permite selecionar o canal MID para o histograma (média dos canais)

- **-s (SIDE channel)**: permite selecionar o canal SIDE para o histograma (diferença dos canais)

Posteriormente, através de uma aplicação externa como o ***gnuplot***, por exemplo, podemos visualizar graficamente os histogramas gerados.

Histograma dos canais do ficheiro *“sample02.wav”*





2 - *wav_cmp*

```
Usage: ./wav_cmp <edited file> <orignal file>
```

O programa ***wav_cmp*** permite comparar dois ficheiros de áudio (ficheiro de áudio editado com o ficheiro original) para cada canal e para a média dos canais utilizando 3 parâmetros diferentes:

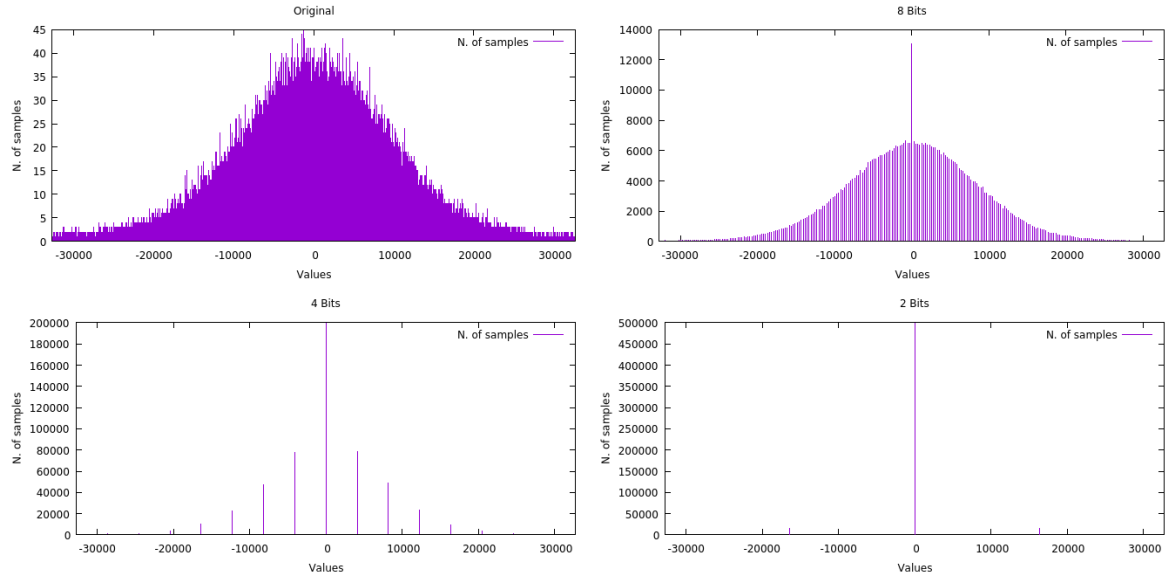
- **Norma L^2**: média dos desvios quadrados entre as amostras editadas e originais
- **Norma L^∞**: desvio máximo entre as amostras editadas e originais
- **SNR (*signal-to-noise ratio*)**: razão entre o quadrado das amostras originais e a média dos desvios quadrados entre as amostras editadas e originais

3 - *wav_quant*

```
Usage: ./wav_quant <input file> <sampleSize> <output file> <newSampleSize>
```

O programa ***wav_quant*** permite reduzir o número de bits utilizado para representar as amostras de um ficheiro de áudio através de uma quantização escalar uniforme.

Efeito da quantização nos histogramas do ficheiro "sample.wav"



4 - *wav_effects*

```
Usage: ./wav_effects <input file> <effect> <params> <output file>

        List of available effects:
        -d <delay>               for the DELAY  effect
        -e <delay> <decay>       for the ECHO   effect
        -r <maxDelay> <maxDecay> for the REVERB effect

        List of available configuration params:
           <delay>: delay in seconds
           <decay>: decay factor in the interval ]0,1[
           <maxDelay>: maximum delay in seconds
           <maxDecay>: maximum decay factor in the interval ]0,1[
```

O programa ***wav_effects*** permite aplicar diferentes efeitos a um ficheiro de áudio. É possível escolher entre 3 efeitos de variação temporal e definir parâmetros de configuração para cada um:

- **-d (*Delay*)**: permite aplicar o efeito *delay* e configurar o seu tempo de atraso em segundos

- **-e (*Echo*)**: permite aplicar o efeito *echo* e configurar o seu tempo de atraso em segundos e um fator de decaimento

- **-r (*Reverb*)**: permite aplicar o efeito *reverb* e configurar o seu tempo de atraso máximo e seu fator de decaimento máximo

A implementação dos algoritmos para a aplicação dos efeitos é definida pela interface ***Wav_Effects*** com o método ***apply()***.
Cada efeito define uma implementação diferente dessa interface e contém atributos de configuração específicos. Todas elas utilizam de um vetor circular para aplicação sucessiva do efeito que é realizada por *frames* e de um parâmetro de ***wet/dry mix*** interno para balancear as intensidades do áudio original e do efeito no resultado final.

Parte II - Manipulação de ficheiros a nível binário

5 - BitStream

A classe BitStream contém, principalmente, funções para leitura e escrita de bits de ficheiros. Em tais funções, foi escolhido passar como parâmetro a referência para o objeto 'fstream' a ser utilizado, pois isso dá maior liberdade aos programas que requisitam as funcionalidades em termos de endereçar os ficheiros, além de facilitar sua implementação por não ter que lidar com a criação/destruição dos objetos.

Também optou-se por criar funções de conversão de vetores booleanos em vetores de caracteres e vice-versa, inicialmente visando auxiliar na implementação do Ex 6, mas não houve conveniência em sua utilização.

6 - encoder-decoder

O programa encoder-decoder é relativamente simples, usado para converter um ficheiro txt de caracteres '1' e '0' em sua respectiva representação binária - interpretando '1' e '0' como os valores 0b1 e 0b0 - e vice-versa. Para isso foram criadas as funções encode e decode.

```
Usage: ../sndfile-example-bin/encoder-decoder <inputFile> <-d|-e> <outputFile>
```

Em termos de implementação:

- **encode:**
  Foi utilizada a função ***BitStream::readNBytes*** para ler os caracteres do ficheiro txt em sucessão. Para cada 8 bytes, cria-se um byte representando os caracteres encontrados, que é em seguida escrito para o ficheiro binário.

- **decode:**
  De modo similar à função encode, usou-se a função ***readNBits*** para analisar bit a bit a informação do ficheiro binário de

entrada. Para cada bit, foi escrito no ficheiro de saída o caractere equivalente.

Parte III - codec para ficheiros de áudio

7 - codec

A funcionalidade principal desta etapa foi criada no ficheiro **codec.h**, que contém as funções necessárias para a transformação **DCT** e quantização e suas respectivas operações inversas. As operações de compressão e descompressão são feitas à custa de tais transformações - DCT seguida de quantização para a compressão, e quantização reversa seguida de IDCT para a descompressão.

Assim, o programa principal executa uma das operações, com a especificação do número de bits do ficheiro comprimido (nBits) para ambos os casos. A chamada para executá-lo é a seguinte:

```
Usage: ../sndfile-example-bin/codec <-c|-d> <nBits> <inputFile> <outputFile>
```

Vale notar que, na operação de compressão, o codec aceita apenas ficheiros .wav (com 16 bits por amostra) para normalizar as funções implementadas e de modo manter coerência com a Parte I. Da mesma forma, os ficheiros de áudio produzidos pela descompressão têm o mesmo formato.