

SAE 1.01-02 - BUT INFORMATIQUE - GROUPE 7

Comparative Study of Two Low-Pass Filters for Sound Data Playback: Analysis and Performance

Auteur:

Yahia KHERZA(A1)

Olivier MARAVAL(A1)

Client:

Gilles PERROT

Référent:

Olivier MARAVAL

January 10, 2024

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 6 |
| 2 | Methodology | 7 |
| 2.1 | DosSend and DosRead usage | 7 |
| 2.1.1 | Using DosSend to create a wav file | 7 |
| 2.1.1.1 | Compiling the Program | 7 |
| 2.1.1.2 | Run the Program | 7 |
| 2.1.1.3 | Output | 7 |
| 2.1.2 | Using DosRead | 8 |
| 2.1.2.1 | Compiling the Program | 8 |
| 2.1.2.2 | Run the Program | 8 |
| 2.1.2.3 | Output | 8 |
| 2.2 | Explanation of the Low-Pass Filters | 9 |
| 2.2.1 | LPFilter1 : Moving Average Filter | 9 |
| 2.2.2 | LPFilter2 : Exponential Moving Average Filter | 9 |
| 2.3 | Method to monitor the speed and accuracy of the filters | 10 |
| 2.3.1 | Monitoring the speed of the filters | 10 |
| 2.3.2 | Monitoring the accuracy of the filters | 10 |
| 3 | Results | 11 |
| 3.1 | Speed | 11 |
| 3.1.1 | LPFilter1's Speed | 11 |
| 3.1.1.1 | Short File | 11 |
| 3.1.1.2 | Medium File | 11 |
| 3.1.1.3 | Long File | 11 |
| 3.1.2 | LPFilter2's Speed | 11 |
| 3.1.2.1 | Short File | 11 |
| 3.1.2.2 | Medium File | 11 |
| 3.1.2.3 | Long File | 11 |
| 3.2 | Accuracy | 11 |
| 3.2.1 | LPFilter1's Accuracy | 11 |
| 3.2.1.1 | Low n | 11 |
| 3.2.1.2 | High n | 11 |
| 3.2.2 | LPFilter2's Accuracy | 11 |
| 3.2.2.1 | Low | 11 |
| 3.2.2.2 | high | 11 |
| 4 | Discussion | 13 |
| 5 | Conclusion | 14 |

Chapter 1

Introduction

The objective is to examine and compare two low-pass filter designs with the end goal of enhancing the clarity of text data transmitted via audio signals. This comparison is a component of a larger project, the DosOok challenge, which involves the development of the two programs DosSend and DosRead for sending and receiving text data through sound.

In this context, a low-pass filter is crucial as it allows the desired signal to pass while attenuating frequencies that are not needed. The necessity for this comparison comes from the fact that we need to find the best speed to noise reduction ratio.

For this study, two distinct low-pass filtering techniques have been selected. The criteria for their evaluation are the speed of processing and their smoothing capability. The filter will get a pass if the initial text isn't altered after the signal has been processed.

This report will detail the theoretical concepts behind each filter, the process of their implementation in Java, and the outcomes of their performance tests. The aim is to determine which filter provides a superior solution for the problem at hand.

Chapter 2

Methodology

The sound containing the data in a text file is created with the DosSend program. It will be read back with DosRead to see how efficiently it can process the signal without altering the data it contains.

2.1 DosSend and DosRead usage

2.1.1 Using DosSend to create a wav file

2.1.1.1 Compiling the Program

Compile the Java code using a Java compiler

```
1 javac DosSend.java StdDraw.java
```

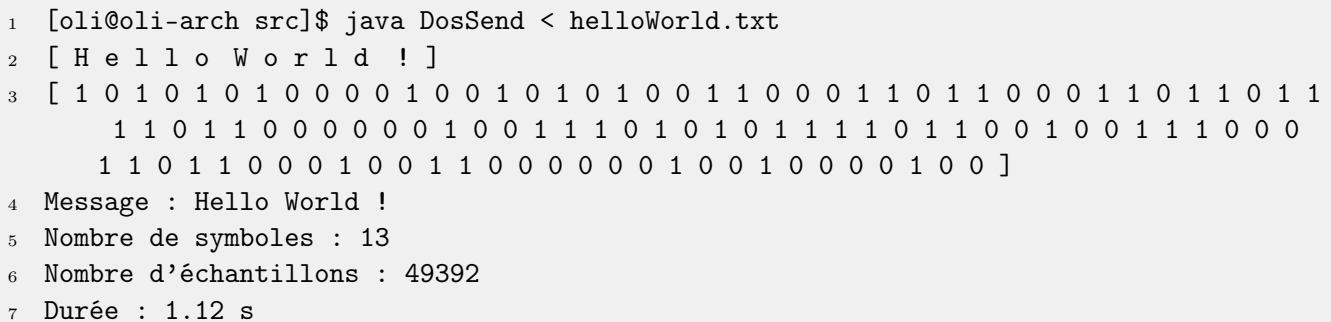
2.1.1.2 Run the Program

Run DosSend with Java and send it your data written in a .txt file via standard input.

```
1 java DosSend < textfile.txt
```

2.1.1.3 Output

The program will convert the input text into an audio signal and save it as `DosOok_message.wav`. It will also print characteristics of the signal to the console and display a graphical representation of the signal waveform. It is a sinusoidal wave modulated using OOK modulation.



```
1 [oli@oli-arch src]$ java DosSend < helloWorld.txt
2 [ H e l l o W o r l d ! ]
3 [ 1 0 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 1 1 0 1 1 0 0 0 1 1 0 1 1 0 1 1
   1 1 0 1 1 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 1 0 1 1 1 1 0 1 1 0 0 1 0 0 1 1 1 0 0 0
   1 1 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 ]
4 Message : Hello World !
5 Nombre de symboles : 13
6 Nombre d'échantillons : 49392
7 Durée : 1.12 s
```

Figure 2.1: Console output for the text "Hello World !"

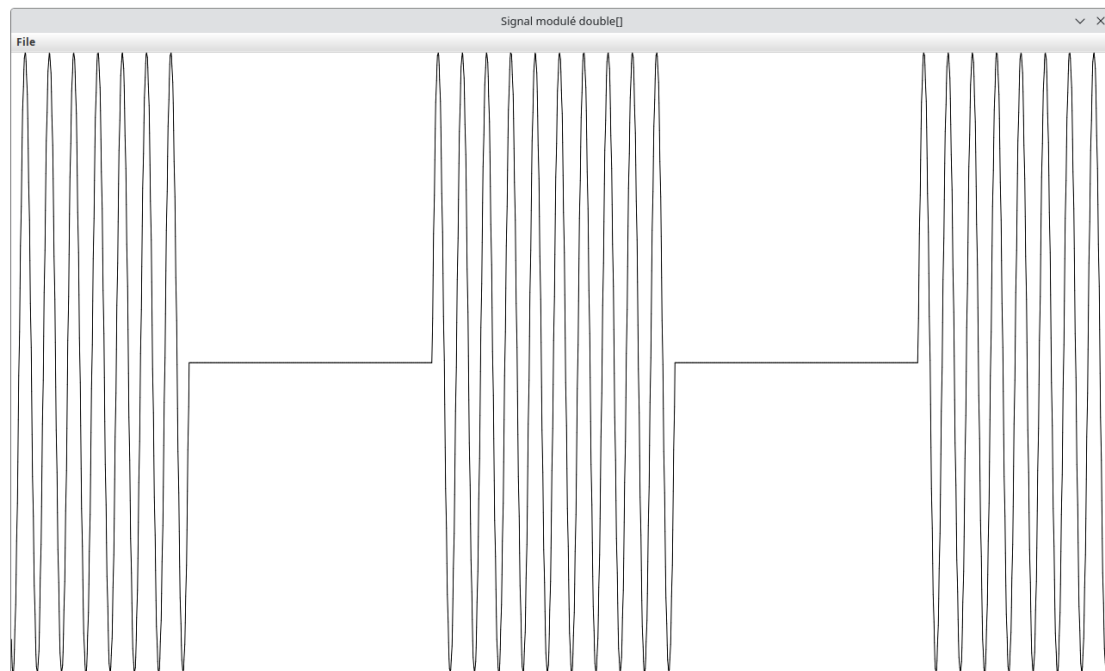


Figure 2.2: Signal output for the text "Hello World !" with start:300 stop:1000 mode:line

2.1.2 Using DosRead

2.1.2.1 Compiling the Program

Compile the Java code using a Java compiler

```
1 javac DosSend.java StdDraw.java
```

2.1.2.2 Run the Program

Run DosRead with Java and add your sound file as a parameter.

```
1 java DosRead sound.wav
```

2.1.2.3 Output

The program will read, process (through a low pass filter), and analyze audio data from the WAV file. It will then output information about the file and the text corresponding to the binary sequence to a terminal. It will show a graphical representations of the original signal and it's filtered counterpart.

```

1 [oli@oli-arch src]$ java DosRead DosOok_message_sample.wav
2 Fichier audio: DosOok_message_sample.wav
3 Sample Rate: 44100 Hz
4 Bits per Sample: 16 bits
5 Data Size: 97660 bytes
6 Message décodé : H e l l o W o r l d !

```

Figure 2.3: Console output for a wav file containing the text "Hello World !"

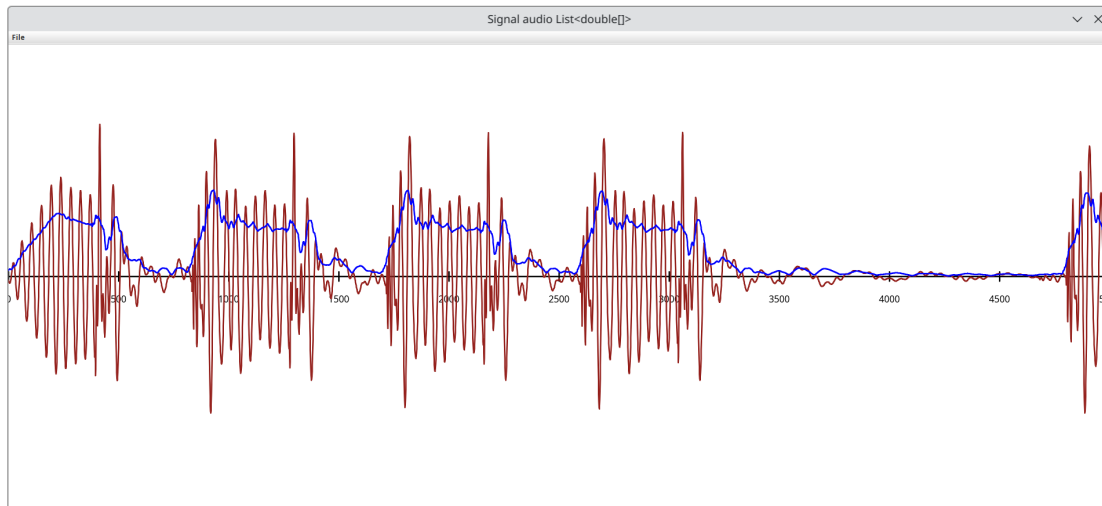


Figure 2.4: Signal output for the text "Hello World !". In red the signal before processing, in blue the same signal after processing

2.2 Explanation of the Low-Pass Filters

2.2.1 LPFilter1 : Moving Average Filter

The moving average filter is a basic type of Low Pass FIR filter that is widely employed to smooth out a sequence of data points or a signal. It operates by averaging a set number, M , of input samples at once, and then outputs a single averaged data point. While this method is effective for signals that are relatively free from noise, its performance significantly deteriorates when dealing with noisy signals.

2.2.2 LPFilter2 : Exponential Moving Average Filter

The exponential moving average (EMA) filter is a type of discrete, low-pass, infinite-impulse response (IIR) filter. It prioritizes recent data by giving it more weight and discounting older data in an exponential manner. This behavior is similar to the discrete first-order low-pass RC filter. In contrast to a simple moving average (SMA) this ensures that the trend is maintained by still considering a significant portion of the reactive nature of recent data points. Compared to the SMA, this filter effectively suppresses noisy components, making it optimal for denoising signals in the time domain. It is a 1st order infinite impulse response (IIR) filter.

2.3 Method to monitor the speed and accuracy of the filters

2.3.1 Monitoring the speed of the filters

Each filter will process 3 files : A short one (two words), A medium one (one paragraph, 97 words), A long one (10 paragraphs, 939 words). Each of the wav files will have been created with DosSend.java and processed within DosRead.java. This will create ideal conditions for the filters as the noise will be minimal. For each file 3 settings with increased filtering power will be used and each of them must not alter DosRead decoding capability.

2.3.2 Monitoring the accuracy of the filters

Each filter will process 3 files with noise added to it : A clean one, A noisy one and A noisier one.

Chapter 3

Results

3.1 Speed

3.1.1 Values

- : $1000 = 0.0 = 1$

3.1.2 LPFilter1's Speed

3.1.2.1 Short File

3.1.2.2 Medium File

3.1.2.3 Long File

3.1.3 LPFilter2's Speed

3.1.3.1 Short File

3.1.3.2 Medium File

3.1.3.3 Long File

3.2 Accuracy

3.2.1 LPFilter1's Accuracy

3.2.1.1 Low n

3.2.1.2 High n

3.2.2 LPFilter2's Accuracy

3.2.2.1 Low

3.2.2.2 high

On the first plot, we have the input that is going into the moving average filter. The input is noisy and our objective is to reduce the noise. The next figure is the output response of a 3-point Moving Average filter. It can be deduced from the figure that the 3-point Moving Average filter has not done much in filtering out the noise. We increase the filter taps to 51-points and we can see that the noise in the output has reduced a lot, which is depicted in next figure. Frequency Response of Moving Average Filters of various lengths We increase the taps further to 101 and

501 and we can observe that even-though the noise is almost zero, the transitions are blunted out drastically (observe the slope on the either side of the signal and compare them with the ideal brick wall transition in our input). Frequency Response: From the frequency response it can be asserted that the roll-off is very slow and the stop band attenuation is not good. Given this stop band attenuation, clearly, the moving average filter cannot separate one band of frequencies from another. As we know that a good performance in the time domain results in poor performance in the frequency domain, and vice versa. In short, the moving average is an exceptionally good smoothing filter (the action in the time domain), but an exceptionally bad low-pass filter (the action in the frequency domain)

Chapter 4

Discussion

Chapter 5

Conclusion