signal with a duration of N, possibly weighted by a window, then the cepstral coefficients have the expression:

$$c(k) = \frac{1}{L}\sum_{\ell=0}^{L-1} S(\ell)e^{2j\pi n\ell/L} \text{ with } S(\ell) = \log\left|\sum_{n=0}^{N-1} x(n)e^{-2j\pi n\ell/L}\right|$$

It can be proven that the first cepstral coefficient represents the energy of the signal segment and that the following d coefficients, where d corresponds to a duration of a few milliseconds, characterize the shape of the vocal tract. In speech recognition, the relevant information is essentially characterized by the shape of the vocal tract. This is why most recognition methods are based on the use of the first cepstral coefficients. Most of the time, the cepstral coefficients are calculated on a logarithmic frequency scale; this is called *MFCC*, for *Mel Frequency Cepstral Coefficients*. Exercise 12.6 only uses a linear frequency scale, leading to a sequence of length d vectors that will undergo the recognition processes.

Exercise 12.6 (DTW word recognition)

1. Write a function that extract the short-term cepstrum from a speech signal sampled at 8,000 Hz using the operations:
   - multiplication of the time window by a Hamming window: choose a window duration corresponding to 20 ms with a temporal overlap of 50%;
   - computation of the logarithm of the modulus of the Fourier transform over $L = 256$ points;
   - computation of the inverse Fourier transform;
   - extraction of the 12 values useful to recognition.
2. Write a program that uses the DTW to compute, based on the sequences of cesptral coefficients, the "distance" between two recordings of the same pronounced word, and between two recordings of two different pronounced words.

12.5 Modifying the duration of an audio signal

Modifying the temporal scale of a sound has applications in many fields, such as solutions for the hearing impaired, speech design and recognition, movies, TV and radio advertisement, etc. A simple way of performing this modification is to reconstruct the signal with a sampling frequency different from the one used for the acquisition. Unfortunately, this method causes frequency distortion, because, as we know, if $X(f)$ represents the Fourier transform of $x(t)$, then the Fourier transform of $x(\gamma t)$, with $\gamma > 0$, is given by $\gamma^{-1}X(f/\gamma)$. Therefore,

the frequency axis is dilated or contracted, depending on whether γ is greater or smaller than 1. You can observe the effects on a speech signal with the use of the MATLAB® function `sound` and by trying different reconstruction frequencies: when the frequency is greater than the acquisition frequency, the pitch seems higher. To prevent this type of distortion, several techniques have been suggested. The most popular one is probably the technique called *PSOLA*, for *Pitch Synchronous Overlapping Addition*, which works in the time domain [67]. Another one is referred to as the "phase vocoder", which works in the frequency domain [36]. One of the drawbacks of PSOLA is to add unwanted noises. As for the phase vocoder, it causes a reverberation effect.

12.5.1 PSOLA

To reduce the total duration of the signal, while preserving the frequency scale, we can simply eliminate small segments of the signal throughout the recording. Conversely, to increase the total duration of the signal, we can duplicate some of its segments. However, these segments have to be long enough to prevent spectrum aliasing, but short enough compared with the duration of the basic acoustic units: with PSOLA [67], the durations of the segments have to be chosen equal to the "instantaneous" pitch period. PSOLA is essentially comprised of two steps:

- **Analysis**: the signal $s(t)$ is time-"marked" according to a sequence of analysis times $t_a(i)$ such that $t_a(i) = t_a(i-1) + P_a$, where P_a refers to the "instantaneous" fundamental period (the pitch) estimated with a window long enough, starting at the time $t_a(i-1)$.

- **Synthesis**: the modified signal $\tilde{s}(t)$ is constructed by an overlap-add operation on the basic segments $s_i(t)$ of the original signal *relocated* at synthesis times $t_s(j)$ according to the expression:

$$\tilde{s}(t) = \sum_n \tilde{s}_i(t - t_s(n))$$

  where $\tilde{s}_i(t - t_s(n)) = h(t - t_a(i(n)))s(t)$ is a signal segment centered in $t_a(i(n))$. The synthesis times are such that $t_s(n) = t_s(n-1) + P_a(i(n))$. If γ refers to the speed modification rate, the index $i(n)$ is the closest integer to the value $n\gamma$.

Exercise 12.7 (PSOLA)

1. **Analysis**: use the `f0cor.m` function, which estimates a signal's pitch, to construct the sequence of analysis times $t_a(n)$ in the following way:

- The initial values are set as $t_a(1) = 1$ and $P_a = L_{10}$ where L_{10} is the number of samples corresponding to 10 ms.
- The window starting at the time $t_a(n)$ and lasting at least two pitch periods P_a is extracted. In practice, the duration has to be chosen equal to twice the smallest period expected in the signal.
- Pitch detection is performed on the window. If the signal is voiced, we get the pitch period P_a and the analysis time $t_a(n+1) = t_a(n) + P_a$. If the signal is unvoiced, the value $t_a(n+1) = t_a(n) + L_{10}$ is chosen.

At the end of this first process, we have a sequence of N_a analysis times (positions in the file) $t_a(n)$ that are *synchronous* with the pitch period.

2. **Synthesis**: we wish to modify the prosodic speed by a factor γ. This is done by generating the sequence of synthesis times t_s as follows:

   - Initially, $t_s = 1$.
   - The following synthesis time is calculated by executing:

     ```
     te=te+gamma;
     ie=ceil(te);
     ```

     The index is used to point out the analysis time after which the segment of the signal used for creating the desired signal is extracted. Hence, if $\gamma < 1$, which corresponds to a slower utterance, the value of i_e after n iterations will be smaller than n. Hence, some segments of the signal will be repeated and the signal created will last longer.
   - The sequence of synthesis times is generated by executing:

     ```
     ts=ts+(ta(ie+1)-ta(ie));
     ```

     This is equivalent to generating a window with a length equal to the pitch (hence the phrase Pitch Synchronous), because (using obvious notations), we can write:

$$t_s(n+1) = t_s(n) + \underbrace{(t_a(i_e(n)+1) - t_a(i_e(n)))}_{P_a}$$

Array 2 shows some of the values for the sequences that were found for $\gamma = 0.8$. As you can see, the portion centered in 7,668 in the original signal is in position 9,610 in the synthesis signal. Notice also that, because the signal is slowed down, as we wanted it to be, we have to duplicate the portion centered un 7,629, once in position 9,532, and once more in the following position 9,571. The signal synthesis is performed as follows:

- the segment centered in $t_a(i_e)$ and with a length of $2(t_a(i_e+1) - t_a(i_e))$ is extracted from the original signal;

n	1	...	199	200	201	202	...
$t_e(n)$	1	...	159.4	160.2	161.0	161.8	...
$i_e(n)$	1	...	160	161	161	162	...
$t_s(n)$	1	...	9,493	9,532	9,571	9,610	...
$t_a(i_e(n))$	1	...	7,590	7,629	7,629	7,668	...

Table 12.2 – Sequences of synthesis and analysis times corresponding to $\gamma = 0.8$, that is a slower utterance

– the extracted segment is multiplied by a Hann window then added to the previous portion with a 50 % overlap.

Write a function that consecutively performs the two steps of the process.

12.5.2 Phase vocoder

The basic idea behind the *phase vocoder* [36] is to perform a short-term Fourier analysis. If the spectrum is comprised of narrow band spectral components, which means that the sound is closer to a voiced sound than it is to an unvoiced sound, and that the analysis window is much longer than the pitch period, then the values of every pitch harmonic will be clearly identified. In this case, if the spectral characteristics are maintained, for a duration slightly lower or slightly greater than the original one, then the value of the pitch is preserved.

The only difficulty is that the phases associated with each frequency component in the modified signal's spectrum have to be calculated in such a way as to ensure the proper alignment of the consecutive phases during the overlap-add operation.

Exercise 12.8 (Hann window)
Consider the sequence (called a *Hann window*):

$$h(n) = \begin{cases} 0.5(\cos(2\pi n/L) + 1) = \sin^2(\pi n/L) \text{ for } n \in \{0, \ldots, L-1\} \\ 0 \text{ otherwise} \end{cases}$$

Let $\alpha \in (0, 1)$, and [2] $n_0 = \lfloor \alpha L \rfloor$. Write a program that plots against n the sequence:

$$x(n) = \sum_k h^2(n - kn_0) \tag{12.16}$$

Notice that the sequence $h(n)$ has a finite length. Therefore, to construct $x(n)$, all you need to do is to calculate the sum of a finite number of segments distant from each other by $h^2(n)$.

[2] $\lfloor \alpha L \rfloor$ refers to the integer part of αL.

What happens when L varies, when α varies? Try other powers of $h(n)$ in expression 12.16.

Exercise 12.9 (Phase vocoder)
Write a program that performs the following operation:

- Calculation of the DFTs of the signal segments, each segment weighted by a length L Hann window $h(n)$. The consecutive windows are distant from each other by a number of points $n_0 = \lfloor \alpha L \rfloor$ where $\alpha \in (0, 1)$. With such an overlap, we have:

$$\sum_k g(n - kn_0) = C$$

  where $g(n) = h^2(n)$ and where C is constant depending on α that you will determine. According to what we saw in exercise 12.8, if L is a power of 2, the distance between the windows has to be in the form $n_0 = L/2^m$.

- Generating the sequence of synthesis times with a factor γ compared with the original sequence according to the method suggested in exercise 12.7.

- Calculation of the modified DFTs by performing an amplitude interpolation, proportional to the original DFTs on the interval containing the synthesis time. For the phase calculations, sum the phase increases of the original DFTs.

- Calculation of the inverse DFTs of the modified DFTs, each DFT being once more weighted by a length L Hann window. For $\gamma = 1$, everything happens as if the window $g(n) = h^2(n)$ were applied and, therefore, the correct amplitudes can theoretically be found by dividing the obtained signal by the constant C.

12.6 Quantization noise shaping

Shannon [88] showed that, for a given level of distortion, there is an encoding that minimizes the *rate* measured in bits per second (bps). The first problem is to define a measurement of distortion. We saw one possible definition in the example on the signal-to-quantization noise ratio. A second difficulty resides in the fact that, in practice, the relation between distortion and minimal rate is usually impossible to determine, because it would require an analytical model for the signal statistics, which most of the time we don't have. Last but not least, the theorem does not tell us how to reach the fundamental limit.

Acquisition

Audio frequency signals are usually real, B band signals. The signal is sampled at a frequency $F_s \geq 2B$, then its samples are quantized, by linear quantization using N bits. The rate is then equal to $F_s \times N$ bps. To increase the SNR, we can therefore increase N during the acquisition; remember that if $F_s = 2B$, the SNR increases by about 6 dB per quantization bit. On the other hand, we can also increase F_s beyond the Nyquist frequency $2B$. One solution (Figure 12.16) to take advantage of this oversampling consists of rejecting part of the quantization noise's power outside the useful band $(-B, +B)$. This noise shaping operation was first introduced by Cutler in 1954.

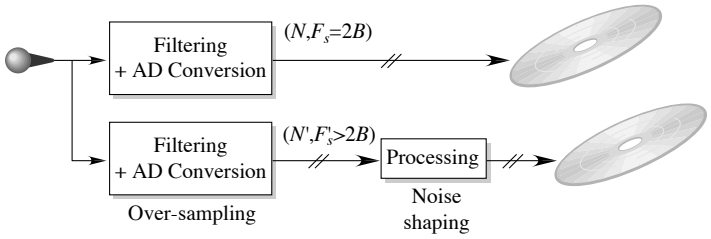

Figure 12.16 – *Acquisition for an audio CD with and without oversampling*

Generally speaking, in order to decrease the rate for a given level of distortion, the signal statistics have to be taken into account, but also the way the signal is perceived. This is what has been implemented in certain compression techniques where the samples are replaced by quantities, such as the prediction coefficients in the case of speech, or by the DCT coefficients in the case of music.

Concerning the sampling of $B = 22$ kHz band audio signals, the compromise between sound quality and technical feasibility has led to several formats, the most common of which are:

- the audio CD, which uses a sampling frequency of $F_s = 44.1$ kHz and an $N = 16$ bits quantization. The resulting rate is roughly 706 kbps, with an SNR of about 96 dB;

- the audio DVD, which uses a sampling frequency up to $F_s = 192$ kHz and a quantization up to $N = 24$ bits. This corresponds to a rate of 4.6 Mbps and an SNR of about 144 dB;

- the DSD (Direct Stream Digital) which performs a *one* bit quantization at a sampling frequency $F_s = 128 \times B$, hence a rate of 2.8224 Mbps. The resulting SNR is approximately 120 dB. Notice that the SNR is the same as the one resulting from a 20 bit format sampled at 44.1 kHz,

which corresponds a third of the rate. The DSD is usually chosen for technological reasons.

Reconstruction

The noise shaping operations mentioned previously can also be used for the reconstruction (Figure 12.17). The first operation consists of "increasing" the sampling frequency: this actually an *interpolation* which is sometimes inaccurately called oversampling. This interpolation is followed by quantization noise shaping. Here are a few implementation examples:

- Probably one of the oldest systems was the one imagined by Philips in audio CD players to recover 16 bit quality at 44.1 kHz while using a 14 bits analog-to-digital converter, but set to four times the initial frequency, hence $4 \times 44.1 = 176.4$ kHz.
- The more recent *MASH* digital-to-analog converter (for *Multi-stAge noise-SHaping*), introduced by the Nippon Telephone and Telegraph, operates on 4bits at a frequency of about 3 MHz. It is used in certain audio DVD players.

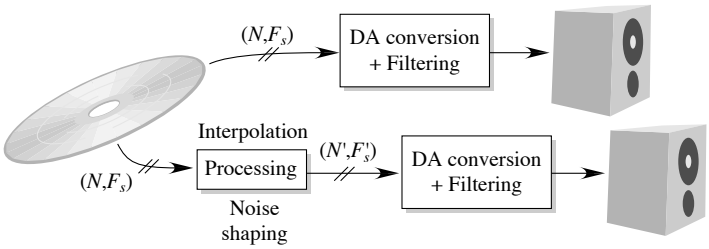

Figure 12.17 – *Reconstruction of audio CD signal, with and without processing*

- Finally, we will mention the One-Bit Stream technique introduced by Philips for audio CDs: the digital-to-analog converter only has a *one* bit resolution but operates at an interpolation frequency between 128 and 256 times the Nyquist frequency. The audio CD's initial quality can be restored by quantization noise shaping and a post-filtering.

Notice that in the case of the DSD format, it is possible to go back to the 20 bit format at the frequency 44.1 kHz with the initial 120 dB quality. However, this requires very long filters to suppress the frequencies outside the useful band. The DSD can be seen as some kind of generic format, based on which an entire range of other formats can be defined with varying levels of quality.

The following exercise is meant to illustrate these interpolation and quantization noise shaping techniques.

Exercise 12.10 (Spectral quantization noise shaping)
In this exercise, the signal $x(t)$ is the sum of three sines with the respective frequencies 437, 504 and 1,367 Hz. The following program generates samples of $x(t)$ for the sampling frequencies 44,100 Hz (`SE1`) and $4 \times 44{,}100$ Hz (`SE2`):

```
%===== MISFQ.M:
% Signal generation
clear; surech=4;            % Oversampling factor
AA=[1.2 3.2 2.7];           % 3 amplitudes
freq=[437 504 1367]'/44100; % 3 frequencies
T=200;
SE1=AA*cos(2*pi*freq*(0:T-1));
SE2=AA*cos(2*pi*freq*(0:surech*T-1)/surech);
```

1. In order for the errors introduced by the quantization to be seen, they have to be much greater than the errors introduced by the interpolation operations. The object of this first question is to justify the choices that will be made concerning the number of bits used to encode in order to perform a simulation that shows the properties of the quantization.

   Using the the interpolation function[3] obtained in exercise 4.14 on page 152, calculate the sequence `SE2int` corresponding to the frequency $f_{e2} = 4 \times f_{e1}$. Create a program that compares `SE2int` with the correct sequence `SE2`. In order to do this, calculate the ratio of the signal's root mean square value to the difference `(SE2-SE2int)` using the MATLAB® function `std`. Perform this computation on a reduced range so as to avoid the side effects caused by the interpolation function.

2. The samples taken at the frequency $f_{e1} = 44{,}100$ Hz are $N_1 = 8$ bit quantized. Let `SE1Q1` be the resulting sequence. Write a program that evaluates the signal-to-quantization noise ratio for the signal. Formula 7.34 indicates that you should have roughly 48 dB.

   Same question with $N_2 = 6$ bits, where `SE1Q2` is the resulting sequence. Check that there is a loss of about 12 dB.

3. The sequence `SE1Q1` is interpolated at the frequency $f_{e2} = 4 \times f_{e1}$. This leads to the sequence `SE2Q2`. It is important that you notice that the sequence `SE2Q2` contains quantization noise outside the frequency band of the signal $x(t)$, and therefore that `SE2Q2` has to be filtered before comparing the result with `SE2`. This is done by using the `rif` function obtained in exercise 4.8. Based on the expression of the quantization noise's power, show that there is a gain of about 6 dB. Check this result with a simulation.

[3] Those who have access to the Signal Processing Toolbox can also use the `interp` function, which at the cost of an algorithm which is a bit more complicated leads to a slightly better interpolation.

4. Consider the process represented by the diagram in Figure 12.18 where Q is a system that goes from 8 bits to 6 bits by eliminating the two least significant bits.

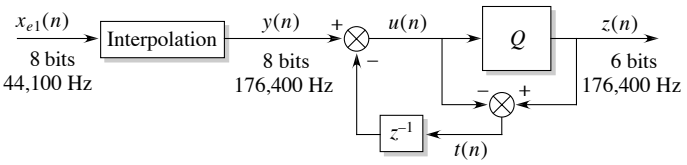

Figure 12.18 – *Noise shaping*

Therefore, it is equivalent, based on the diagram in Figure 7.8 on page 271, to adding a white noise $e(n)$ with the power spectral density equal to $q^2/12$ in the band.

Use this result to show that $z(n) = y(n) + e(n) - e(n-1)$. By noticing that going from $e(n)$ to $\varepsilon(n) = e(n) - e(n-1)$ is a linear filtering, infer that the signal-to-noise ratio is enhanced by about 6 dB compared with a system that would directly apply the quantization to $y(n)$.

This is an interesting result, from a practical point of view, since it allows you to find basically the same SNR as in question 1. Notice that the suggested process cannot be done without the interpolation operation conducted on the quantized signal with many more bits than what is necessary, in this case 8 at the beginning compared with the 6 at the end.

12.7 Elimination of the background noise in audio

For practically every signal processing system, we are faced with the problem of extracting a useful signal corrupted by noise. In the case of audio frequencies, there are several different causes for the noise. We will mention for example the noise due to the defects in the sensors and, in other recording devices, the noise related to the environment in which the recording was done (conversation noises in a public place, doors opening and closing, the tinkling of silverware in a restaurant, etc.) or also the decay of the recording medium. Among all the types of noises encountered in practice the following two types occur in almost all audio systems:

- The *background noise* which can be represented by a stationary random process. It often originates from the electronic circuits of the amplification and reconstruction devices, hence the name.
- The *impulse noises* which can be represented as a sequence of very brief impulses, with random amplitudes and locations. Listening to a scratched record is a perfect example.

To reduce these noises and restore the recording as best as possible, we are going to consider them and deal with them separately.

Eliminating background noise is a particularly difficult problem, not to say impossible to solve, since we have to find what characterizes the difference between the useful part of the signal and the unwanted part. Take the example of a musical recording where we can hear the sound of a violin amidst background noise. The program assigned with the separation would have to transform into some kind of a music expert to distinguish the sound of the violin from the background noise. As a consequence, the solution can sometimes be worse than the problem when reducing background noise in musical recordings.

If we only concern ourselves with speech signals, we can mention [32, 33, 11] among the different denoising methods in a stationary background noise. These methods give satisfactory results, such as for example in cellular technology, or for restoring "old" recordings, but only if the settings of the algorithm are very well chosen. They usually require for the spectral properties of the noise to be known. They can do this by using a portion of the signal that contains nothing but noise. This portion is selected either by a direct "acoustic" observation of the signal, or by automatically detecting the active areas of the useful signal. In [11], the author suggests a very simple method leading to an acceptable result that we are going to implement. If we assume that the noise is white with the power σ^2, known or estimated from a portion of the signal containing nothing but noise, the short-term amplitude spectrum of the denoised signal σ^2 is estimated by the expression:

$$\widehat{X}(k) = G(k)X(k) \tag{12.17}$$

where:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-2j\pi nk/N}$$

refers to the DFT of a length N window of the noisy signal $x(n)$ and where:

$$G(k) = \begin{cases} \left(1 - \lambda \dfrac{\sigma\sqrt{N}}{|X(k)|}\right) & \text{if } \lambda \dfrac{\sigma\sqrt{N}}{|X(k)|} < 1 \\ \mu & \text{otherwise} \end{cases}$$

where the parameters λ and μ are adjusted experimentally so as to obtain the best sound result. This method is sometimes called spectral subtraction. According to 12.17, everything happens as if we were applying a gain $G(k)$ adapted to each component of the DFT. The results show that the background noise is rather well eliminated, but the method introduces whistling noises, that get louder as μ decreases. In particular, the value $\mu = 0$ leads to the plain and simple elimination of the spectral components below the threshold,

causing certain components of the noise to be isolated and to behave as isolated "peaks". This "whistling" noise, referred to as *musical noise*, can be reduced by decreasing λ and increasing μ, but at the cost of a lower noise reduction.

Exercise 12.11 (Denoising a speech signal)
First, record a speech signal, sampled at 8,000 Hz, which will serve as the reference signal. Construct a noisy version of it, by adding noise so as to ensure a signal-to-noise ratio of 10 dB.

Implement the method described by equation 12.17 to denoise the noisy file by considering that σ is known. Try the program for different values of N, of λ and of μ.

12.8 Eliminating the impulse noise

We are now going to look into the restoration of recordings containing errors with a relatively large amplitude and with a very brief duration (less than a millisecond), referred to as *clicks*, and assumed to be in small numbers. Thus, for the signal represented in Figure 12.19, originating from record, shows a scratch spread over a few samples. Cracks are not always as "visible" as this one, and the recording often has to be listened to in order to detect them.

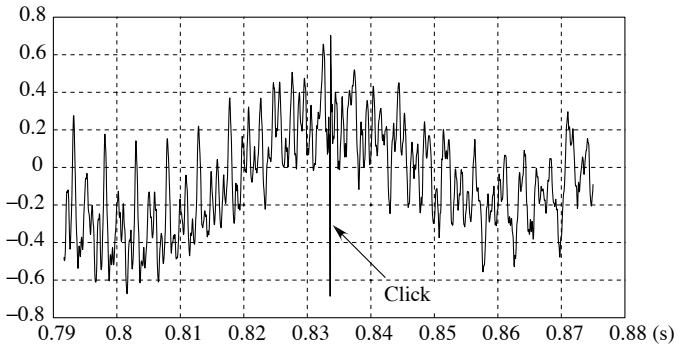

Figure 12.19 – *Signal originating from a record and containing a click*

The considered restoration method is comprised of two separate steps:

1. the detection of clicks in the signal;
2. the restoration of corrupted samples.

12.8.1 The signal model

When observing the signal $s(n)$ shown in Figure 12.19, what makes us decide that there is a click in position n_0 is that the value in n_0 is noticeably different