1

Generating Elementary Signals

1.1. General points

A signal is a *function* of one or more variables, which has the specificity of being generated by a physical phenomenon and of carrying information rather than energy. Simple examples of signals include speech signals, acoustic signals and the electrical voltages provided by sensors. We may distinguish between different types of signals:

– Continuous time signals, which have a (measurable) value at any time.

– Discrete time signals, which only have (measurable) values at certain moments.

– Sampled signals, which are time-discrete signals for which the time gap between two consecutive measurements is constant.

– Deterministic signals, for which the value may be perfectly predicted using a mathematical model to calculate exact values at any time.

– Random signals, which cannot be precisely predicted, either because it is impossible or because it is not necessary.

– Causal signals, which are null prior to the instant selected as time 0. These signals show the repercussions of a particular event occurring at time 0.

– Stationary signals, which carry the same information throughout the observation period. All periodic signals are stationary, but not all stationary signals are periodic: for example, the deterministic signal defined by the expression $x(t) = 10\sin(2\pi 50t) + 5\sin(2\pi 50\sqrt{2}t)$ is not periodic, as it is the sum of two sinusoids with frequencies in an irrational relationship.

1.2. Generation of elementary wave forms

To test the processing operations and analyses shown later in this book, it is useful to generate synthetic test signals (which are not the result of the observation of a real physical phenomenon). To do this, we use the `synth` command. The command file below (`Generation1.bat`) includes comment lines (which begin with the instruction `rem`) and six calls to the SoX program:

```
rem generation de signaux elementaires avec SoX
rem F. Auger, IUT Saint-Nazaire, dep. MP, dec. 2009

sox -n s1.mp3 synth 3.5 sine 440
sox -n s2.wav synth 90000s sine 660:1000
sox -n s3.mp3 synth 1:20 triangle 440
sox -n s4.mp3 synth 1:20 trapezium 440
sox -V4 -n s5.mp3 synth 6 square 440 0 0 40
sox -n s6.mp3 synth 5 noise

for %%i in (200,300,400) do ^
sox -n s7_%%i.mp3 synth 15 sine %%i

rem ecoute du resultat : le fichier de sortie est le canal
rem de sortie par defaut (-d), c'est a dire la carte son

sox s1.mp3 -d

pause
```

– The first command generates a file `s1.mp3` containing a recording of 3.5 s of a sinusoid with a frequency of 440 Hz. The `-n` (null) command is used to indicate the absence of an input signal.

– The second command generates a file `s2.wav` containing a recording of $90,000$ samples of a sinusoidal signal with a frequency that varies in a linear manner from 660 to $1,000$ Hz.

– The third command generates a file `s3.wav` containing a recording of 1 min and 20 s of a triangular signal.

– The fourth command generates a file `s4.wav` containing a recording of 1 min and 20 s of a trapezoidal signal.

– The fifth command generates a file `s5.wav` containing a recording of 6 s of a square signal with a null offset (no continuous component), a null phase and a duty cycle of 50%. The command `-V4` is used to choose the amount of information shown by SoX: its verbosity ranges from 0 (no information sent back to the user) to 4 (maximum information displayed).

– The sixth command generates a file `s6.wav` containing a recording of 5 s of a white noise with a value ranging from 1 to 1 with an average value of 0.

– The seventh command uses a repetition structure to generate three files containing a recording of 15 s of a sinusoid. The variable `%%i` is used both for the file name and for the frequency of the sinusoid. The symbol `^` indicates that the instruction continues onto the following line.

– The final command sends the contents of the file `s1.mp3` to the default peripheral `-d`, i.e. the sound card, allowing us to listen to the signal.

EXERCISE 1.1.– Write a command file to generate 10 files containing recordings of 2.5 s of sinusoids, for which the frequencies should be linearly spaced from 100 to $1,000$ Hz. What is the analytical expression of these signals?

1.3. Elementary signal processing operations

A number of elementary operations may be applied to existing signals. The command file below (called `Generation2.bat`) shows some examples:

```
rem transformations elementaires de signaux avec SoX
rem F. Auger, IUT Saint-Nazaire, dep. MP, jan. 2010

sox s1.mp3 s1_faible1.mp3 vol -6 dB
sox s1.mp3 s1_faible2.mp3 vol -0.4 amplitude

sox s1.mp3 s2.mp3 DeuxSons1.mp3
sox s2.mp3 -v 0.6 s1_faible1.mp3 DeuxSons2.mp3

sox s1.mp3 s1_avec_silence1.mp3 pad 1
sox s1.mp3 s1_avec_silence2.mp3 pad 1 0.5
sox s1.mp3 s1_avec_silence3.mp3 pad 1 5000s@3 0.5

sox -m s3.mp3 s4.mp3 SommeSons.mp3

sox s1.mp3 s1_avec_offset.mp3 dcshift 0.05

sox s1.mp3 s1_a_l_envers.mp3 reverse

sox s3.mp3 s3_morceau.mp3 trim 1.5 2

sox s3.mp3 s3_faded.mp3 fade t 10 1:00 20

rem pause
```

– The first two commands apply gains to the signal contained in file `s1.mp3`; 6 dB in the first case (corresponding to a factor of 0.5) and 0.4 in the second case (corresponding to a reduction in amplitude of 60% and a phase shift of 180^o).

– The next two commands place two signals in sequence. In the second case, the amplitude of the first signal is reduced by a factor of 0.6.

– The following three commands add null values (i.e. silence) to the signal contained in the file `s1.mp3`. The first command adds 1 s of silence at the beginning, the second

adds 1 s at the beginning and 0.5 s at the end and the third additionally inserts 5,000 samples from the third second of the signal.

– The following command mixes the two signals `s3.mp3` and `s4.mp3`.

– The next command adds a DC component equal to 0.05.

– The following command reverses a signal (the end becomes the beginning, and vice versa).

– The next command extracts a section of the signal contained in the file `s3.mp3`, beginning at 1.5 s with a duration of 2 s. This command is used to easily extract sections of signals for analysis or modification.

– The final command is used to modify the amplitude of a signal. In this way, it is possible to vary the amplitude of a signal progressively, at the beginning and at the end. Parameter `-t` corresponds to a linear modulation. The second parameter indicates that the amplitude of the signal moves from 0 to its maximum value in the course of the first 10 s; the next parameter indicates that a decrease begins after 1 min, and the final parameter indicates that the linear decrease continues for more than 20 s. These last two parameters are optional. Other types of modulation are possible, as shown in the software user manual.

EXERCISE 1.2.– Write a command file that generates a file we will call `succession.mp3`, containing the 10 signals generated in Exercise 1.1, placed in sequence. Next, write another command file to generate a second file, `Sommesons.mp3`, containing the sum of the 10 files generated in Exercise 1.1. What is the analytical expression of this second signal? What mathematical property does it satisfy?

EXERCISE 1.3.– Write a command file to generate a recording of 1 min of signal equal to the sum of two sinusoids

with frequencies of 440 and 441 Hz and with the same amplitude. What does this signal look like? Justify its appearance using one of the following two relationships:

$$\begin{aligned}\cos(\theta_1)+\cos(\theta_2) &= 2\cos\left(\frac{\theta_1+\theta_2}{2}\right)\cos\left(\frac{\theta_1-\theta_2}{2}\right)\\ \sin(\theta_1)+\sin(\theta_2) &= 2\sin\left(\frac{\theta_1+\theta_2}{2}\right)\cos\left(\frac{\theta_1-\theta_2}{2}\right)\end{aligned}$$

Finally, write a command file to extract 3 s from the file

```
AriaCantilenaVillaLobosPetibon.mp3
```

beginning at 5 min 24 s, then listen to and visualize this extract using `Audacity`. How is this connected to the previous question?

EXERCISE 1.4.– First generate a file `Sinus.mp3` containing a sinusoid of frequency 440 Hz, then a file `Noise.mp3` containing random noise, both with a length of 7 s. Then, use a command such as

```
sox -m Sinus.mp3 -v 0.01 Noise.mp3 SinusNoised.mp3
```

to add the sinusoid to the noise, with the noise being attenuated by a factor of 0.01. Beyond what value of the attenuation factor is the noise no longer perceptible to the human ear?

Finally, use a command such as

```
sox -m Noise.mp3 -v 0.01 Sinus.mp3 SinusNoised.mp3
```

to add the noise to the sinusoid, with the sinusoid being attenuated by a factor of 0.01. Beyond what value of the attenuation factor is the sinusoid no longer perceptible to the human ear?

EXERCISE 1.5.– The aim of this exercise is to show that it is possible to generate relatively complex files, which may then be used for tests or monitoring.

Write a command file to generate an `mp3` file containing a recording of 2 s of the sum of six sinusoids of frequency f_0, $2\,f_0$, $3\,f_0$, $4\,f_0$, $5\,f_0$ and $6\,f_0$, where $f_0 = 523$ Hz. Next, generate two other files obtained using values of 659 and 830 Hz, respectively, for f_0. Finally, generate a file containing the sequence of these three sounds, and then a last file that repeats the previous file three times[1].

For this exercise, a spreadsheet may be used with file `GenerationCode.xls`, which automatically generates a set of SoX instructions; these instructions may then be copied into a command file.

EXERCISE 1.6.– Write a command file that:

1) generates two sinusoidal signals with a duration of 10 s and frequencies of $12{,}000$ and $12{,}440$ Hz. These two signals should be stored in files with extension `.wav`.

2) calculates the sum of these signals and stores them in a third file with extension `.wav`.

1 In 1968, Jean-Claude Risset used this principle to create a perpetually ascending sound, which gives the impression of constantly becoming higher and higher.

Listen to the obtained signal (preferably using headphones). What do you hear? Justify this result[2].

1.4. Using other file formats

In the previous section, signals were stored in `mp3` format files. However, SoX is able to handle a wide variety of file formats, both in terms of reading and writing. These formats, described in detail in the user manual, include `.wav`, used by the Windows operating system, and `.dat`, which corresponds to text files (coded in ASCII), in which the first line corresponds to the sampling frequency and the following lines contain two numbers for time and the value of the signal. The command file below (`Generation3.bat`)

```
rem generation d'un fichier texte au format .dat
rem F. Auger, IUT Saint-Nazaire, dep. MP, jan. 2010

sox s1.mp3 s1.dat silence 0 trim 0.1 10s
```

generates a file `s1.dat` containing 10 successive values (`10s`) of file `s1.mp3` taken 0.1 s after the start of the signal (having removed the null values from the beginning of the file). The contents of file `s1.dat` are as follows:

```
; Sample Rate 48000
; Channels 1
               0      -0.68915808
  2.0833333e-005      -0.65037082
  4.1666667e-005      -0.60942747
       6.25e-005      -0.56646369
  8.3333333e-005      -0.52161656
   0.00010416667      -0.47505816
        0.000125      -0.42689275
   0.00014583333      -0.37733861
```

2 This technique is used by directional speakers to generate acoustic waves using ultrasonic transducers. See [MUI 11].

```
   0.00016666667      -0.32652545
       0.0001875      -0.27462651
```

By creating files of this type, we are able to manipulate measurement signals of non-acoustic origin using SoX. We must simply normalize these signals so that they always fall between -1 and 1. By choosing a sampling frequency suited to human audition, we are able to listen to non-acoustic phenomena. The program below (called `GenerationFichierDat2010.ch`), written in C for the Ch environment, reads numbers from a text file, extracts the maximum and the minimum, and then generates a `.dat` file, applying cross-multiplication to the values of the text file to convert values between $x_{\min}$ and $x_{\max}$ into values between -1 and 1. The result may then be converted into an `mp3` file, and may be listened to using the command

```
sox -V4 Mesures.dat Mesures.mp3
```

```
//
Programme permettant de transformer un fichier de mesures au format texte
//  en fichier de mesures au format .dat
//  F. Auger, janvier 2010

#include <stdio.h>

int main()
{
 char   NomFichierTexte[] = "Mesures.txt" ;
 char   NomFichierDat[] = "Mesures.dat" ;
 FILE   *PointeurFichierTexte, *PointeurFichierDat ;
 double x, xmax, xmin, xnorm, Somme, Difference, Fe=16000.0, Te=1.0/Fe, t;

 // on ouvre une premiere fois le fichier pour trouver le min et le max
 PointeurFichierTexte = fopen(NomFichierTexte, "r") ;
 if ( PointeurFichierTexte == NULL )
  {printf("Impossible d'ouvrir le fichier %s en lecture \n",
          NomFichierTexte);}
 else
  {
   // premiere lecture dans le fichier pour initialiser xmin et xmax
   if (fscanf(PointeurFichierTexte,"%lf", &x) != EOF)
    {xmin=x ; xmax=x;}
```

```
   while (fscanf(PointeurFichierTexte,"%lf", &x) != EOF)
    {
     if (x>xmax) {xmax=x;}
     if (x<xmin) {xmin=x;}
    }
   fclose(PointeurFichierTexte);

   Somme=xmax+xmin; Difference = xmax-xmin; t=0;

   // generation du fichier .dat par une relecture du fichier texte
   PointeurFichierTexte=fopen(NomFichierTexte,"r");// ouverture en lecture
   PointeurFichierDat  =fopen(NomFichierDat,
"w");// ouverture en ecriture

   if ( PointeurFichierDat == NULL )
    {printf("Impossible d'ouvrir le fichier %s en ecriture\n",
            NomFichierDat);}
   else
    {
     // on precise la frequence d'echantillonnage du signal de sortie, qui
     // n'est pas forcement la meme que celle du signal d'entree
     fprintf(PointeurFichierDat,"; Sample Rate %8.2f\n", Fe);

     // un seul signal = une seule voie = un seul canal
     fprintf(PointeurFichierDat,"; Channels 1\n");

     while (fscanf(PointeurFichierTexte,"%lf", &x) != EOF)
      {
       xnorm=(2*x-Somme)/Difference;
       fprintf(PointeurFichierDat, "%f %f \n", t, xnorm);
       t=t+Te;
      }

     fclose(PointeurFichierTexte);// c'est fini, on ferme les fichiers
     fclose(PointeurFichierDat);
    }
  }
 return 0;
}
```

EXERCISE 1.7.– If x is between $x_{\min}$ and $x_{\max}$, which intervals must contain

$$\frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad \text{and} \quad 2\,\frac{x - x_{\min}}{x_{\max} - x_{\min}} - 1 \quad ?$$

Provide a justification of the expression of `xnorm` used in the program. Next, modify the previous program to convert the file `ecg.txt` into `.dat` format. This file contains an electrocardiogram sampled at 720 Hz and acquired using a 12 bit analog–digital converter[3]. This file may then be converted into `.wav` format using SoX, then visualized using `Audacity`.

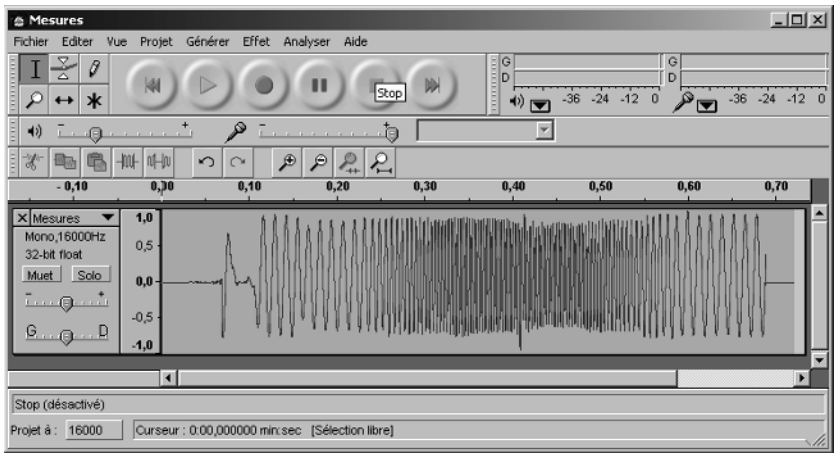

Figure 1.1. *Result of the creation of an `mp3` file based on electrical currents measured at the terminals of an asynchronous machine*

This format may also be used to process signals generated by SoX in other programs, for example in a spreadsheet program such as `LibreOffice Calc`[4].

3 Source: http://www.physionet.org/physiobank/database/aami-ec13/.

4 See http://fr.libreoffice.org/.