Digital Signal Processing Labs

Hugues Garnier

December 2022

Contents

Introduction

MATLAB, for MATrix LABoratory is a powerful software that can deal with complex problems in various scientific fields, exploit and analyse data in an interactive environment.

It is particularly widely used by scientists and engineers to perform various signal processing tasks.

The *Signal Processing* toolbox includes functions and graphic interfaces (Apps) for generating, viewing, processing and filtering signals. Algorithms are available to:

- approach the signal spectrum by discrete Fourier transform;
- calculate the power spectra of random signals;
- design and analyse filters;
- estimate parametric models of random signals;
- perform time-series forecasting.

This toolbox is used to analyse and compare the signals in the time, frequency and time-frequency domains, identify patterns and trends, infer characteristics, and develop and validate customized algorithms in order to extract relevant information on data coming from very diverse fields.

You will apply digital signal processing on audio data. Bring your headphones in order to hear the effects of your digital signal processing as well as your smartphone to record audio sounds.

Lab 1

Fourier analysis of digital signals

You are not asked to submit a report for this 4h00 lab which will not therefore be marked. You are, however, encouraged to write a personal report that will be useful for your preparation to the final exam which will include some questions related to this lab.

Downloading of the data needed for the lab

1. Download the zipped file **Lab1_DSP.zip** from the course website and save it in your Matlab working directory.
2. Start Matlab.
3. By clicking on the browse for folder icon  , **change the current folder of Matlab so that it becomes your Lab1 folder** that contains the files needed for this lab.

Recording of sound files

You will do the recordings of sounds with your own hardware, for example, with the microphone of your smartphone.

Layout of the Lab

The lab is structured into two main parts:

1. A tutorial introduction of the graphical interface `signalAnalyzer` to determine the fundamental frequency of a tuning fork.
2. Several assignments where you have to analyze your data recordings and other real-life data with the methods and theory that you learn on Fourier analysis.

1.1 Tutorial introduction of the SignalAnalyser App - Determination of the fundamental frequency of a tuning fork

The Signal Analyzer App in the Signal Processing toolbox lets you visualize, analyze, and compare signals in the time, frequency and time-frequency domains without needing to write any code.

You can explore signals and identify and extract key features. You can also easily apply various preprocessing techniques to the signals in the app. The app provides a way to work with many signals at the same time and in the same view. To watch a short introduction on how to use the Signal Analyzer App, watch the following 4 mn video:
`fr.mathworks.com/support/search.html/videos/signal-analysis-made-easy-with-the-signal-analyzer-app-1604485876749.html`

A tuning fork is a small steel tool consisting of a U-shaped rod as shown in Figure 1.1 which can be used to tune certain musical instruments.

Figure 1.1: Tuning fork

Load the tuning fork data into MATLAB by typing the following command:
`load tuning_fork;`
This file contains a `y` column vector with the sound samples produced by the tuning fork and a `fs` scalar for the sampling frequency.
Listen to the sound made by the tuning fork by entering the following command:
`sound(y,fs);`

Convert the signal to a MATLAB timetable object. This will facilitate the automatic labelling of the time and frequency axis in the SignalAnalyzer tool.
`tuning_fork = timetable(seconds((0:length(y)-1)'/fs),y);`

Matlab now has a `signalAnalyzer` graphical interface used to display the features of a signal in the time and frequency domains.
At any time, you can consult the technical documentation of this graphical tool by entering, in the command window of MATLAB :
`doc signalAnalyzer`

The graphical interface can be open by clicking on its icon in the Apps menu or by typing in the command window:
`signalAnalyzer;`

A graphical window should then open. Select and drag the timetable to a main display windows on the right frame. The time-domain plot of the tuning fork data should be displayed as shown in Figure 1.2.

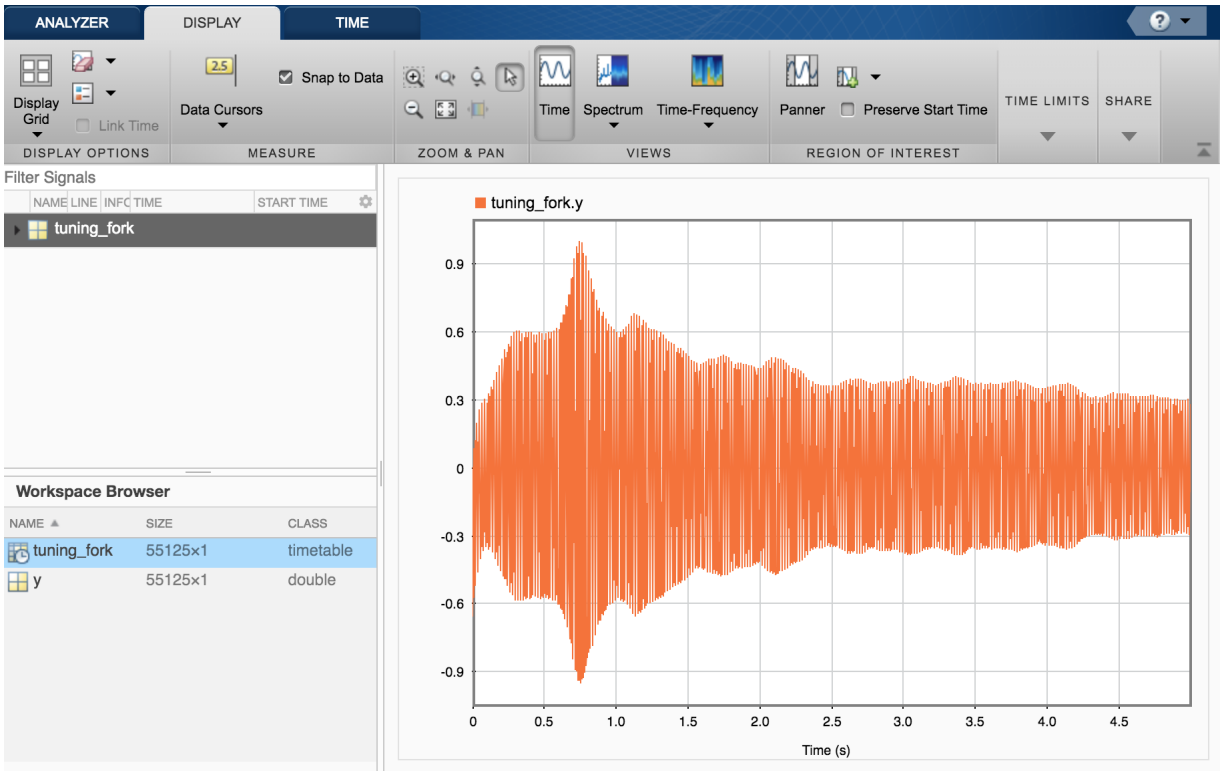

Figure 1.2: Time-domain plot of the tuning fork data in the SignalAnalyser App

Check that the signal is plotted over a time range of about 5s.

From the time-domain plot, it can be noticed that the amplitude of the signal decreases much slowly after 2s. We will concentrate on this time range for the signal analysis.

Click on the Spectrum tab to display the spectrum of the signal.
Click then on the paner tab and select the time window corresponding to 2 to 2.5 s as shown in Figure 1.3.

From the power spectrum displayed, we can clearly observe two main peaks. Click on the Data Cursors tab and select two, so as to make appear two vertical lines on the spectrum that you can move to measure the frequency and amplitude of the two main peaks.

The first peak having the largest amplitude for $f = 440$ Hz represents the fundamental frequency of the tuning fork. The second peak appears at $f = 880$ Hz and is therefore a harmonic of the periodic signal with much lower amplitude (-54 dB) than the fundamental (-10.8 dB), which can be therefore neglected here.

It is recalled that the frequency band audible by the human ear ranges from 20 Hz to 20 kHz. When the frequency ranges from 20 to 200 Hz, the sound is low-pitched (*son grave*). The sound is said medium for a frequency between 200 and 1000 Hz and we get a high-pitched sound (*son aigu*) when the frequency is between 1000 and 15000 Hz. The tone produces by the tuning fork is therefore medium.

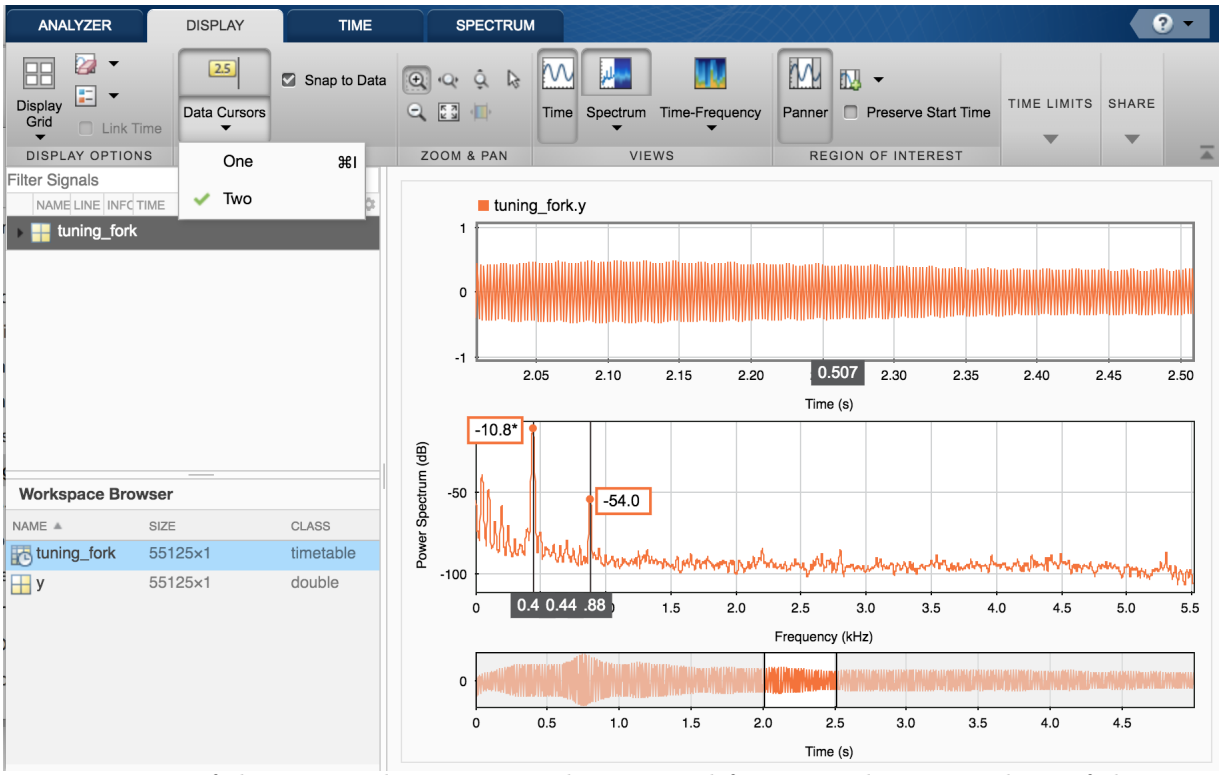

Figure 1.3: Use of the paner tab to improve the time and frequency-domain analysis of the tuning fork data in the SignalAnalyser App

Reduce further the time window of the paner around 2.5s for example to make appear the sine wave form of the tuning fork sound as shown in Figure 1.4.

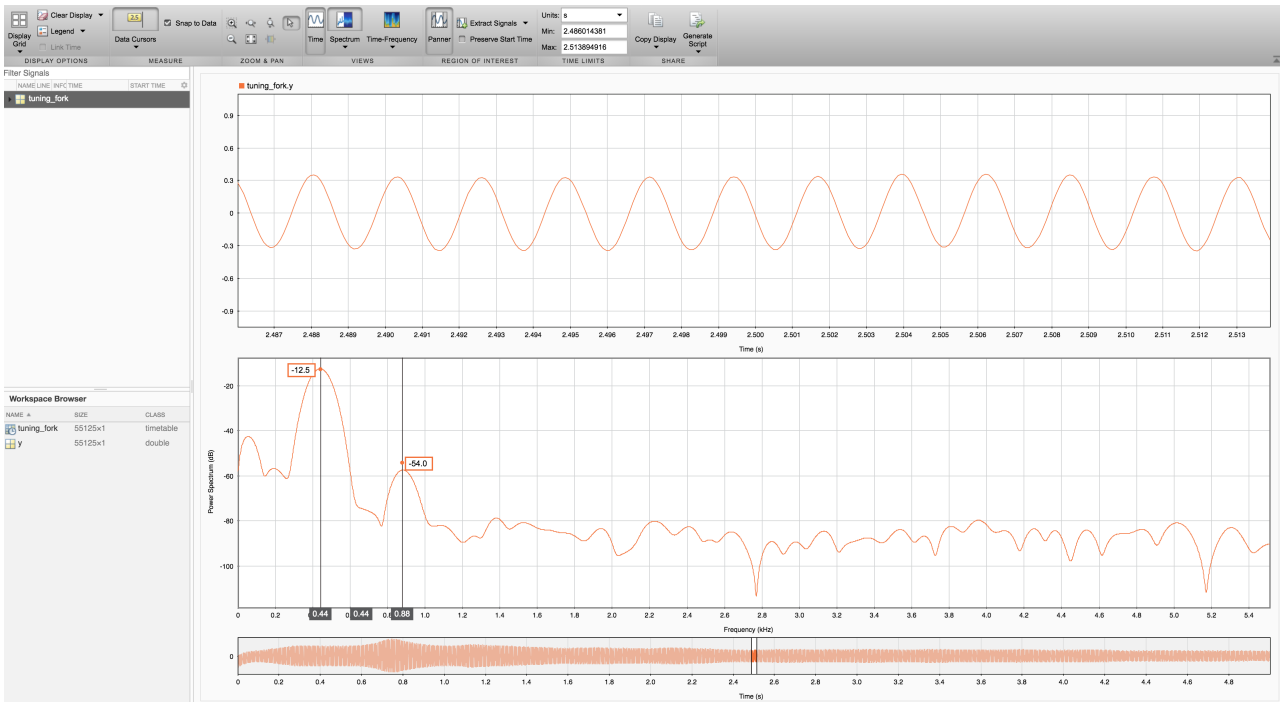

Figure 1.4: Use of the zoom tab to improve the time and frequency-domain analysis of the tuning fork data in the SignalAnalyser App

1.2 Assignments - Application of Fourier analysis in different fields

The assignments aim at analyzing real-life signals by using Fourier analysis coming from different fields :

1. Music instruments: how to detect a spurious signal in trumpet sounds?
2. Telecom: how to decode a phone number from dual-tone multi-frequency data?
3. Human whistle: how to determine the main frequency of your whistle?
4. Animal sounds: what is the type of sound produced by a blue whale vocalization ?
5. Solar physics: how to determine the period in years of the sunspot activity?
6. Image compression with the FFT: can you see the difference ?

1.2.1 Spectral analysis of trumpet sounds

This assignment aims at determining the fundamental frequency of actual trumpet sounds by Fourier analysis.

The first trumpet sound playing note B was sampled at the standard CD sampling frequency stored in the variable `fs=44.1` kHz.
Load the trumpet data into MATLAB and listen to the sound made by the trumpet by typing the following command:

```
load trumpet;
sound(y,fs);
```

Convert the signal to a MATLAB timetable object.

```
t=(0:length(y)-1)'/fs;
trumpet = timetable(seconds(t),y);
```

By using the signal analyzer App, determine:

1. the fundamental frequency of this trumpet sound
2. the number of non-negligible harmonics and their frequencies. A -40 dB attenuation of a harmonic in comparison with the fundamental or with the harmonic having the largest power will be considered here as the threshold of audibility. Indeed, the sounds below this value are of little interest because they are often masked by louder sounds, and therefore can be considered as uninformative sounds.

Repeat the analysis above to the data stored in the file named `spurious_trumpet.mat`. It contains audio data recorded from an actual trumpet still playing note B but where an interfering or spurious signal is added to it.

Listen to the sound made by the spurious trumpet.

By using the signal analyzer App:

1. determine the fundamental frequency of the spurious sound.
2. suggest a way to eliminate the spurious sound from the recorded data. The signal analyzer App can quite easily apply different type of filters on the recorded signals. You can investigate this option but this is not required for this assignment. *This will be done during the second lab.*

1.2.2 Decoding of a dual-tone multi-frequency (DTMF) signal

This assignment aims at decoding a phone number from dual-tone multi-frequency (DTMF) signaling tone data by Fourier analysis.

A dual-tone multi-frequency (DTMF) or VF (Voice Frequency) is a telecommunication signaling system using the voice-frequency band over telephone lines between telephone equipment and other communication devices and switching centers. These codes are emitted when pressing a key on the telephone keypad, and are used for dialing phone.
Technically, each key on a telephone corresponds to a pair of two audible frequencies that are transmitted simultaneously. Here we look at the DTMF code used in the United States which uses seven distinct frequencies that can encode the twelve telephone keypad shown in Figure 1.5. These seven frequencies are shown at the left edge and at the bottom of Figure 1.5.
They are also listed in the table below.

697 Hz	770 Hz	852 Hz	941 Hz	1209 Hz	1336 Hz	1477 Hz

The sound generated by pressing a key result in the sum of two sine waves with associated frequencies. Pressing button 1 will therefore produce the following signal:

$$y_1(t) = 2 \times \big(\sin(2\pi \times 697 \times t) + \sin(2\pi \times 1209 \times t)\big);$$

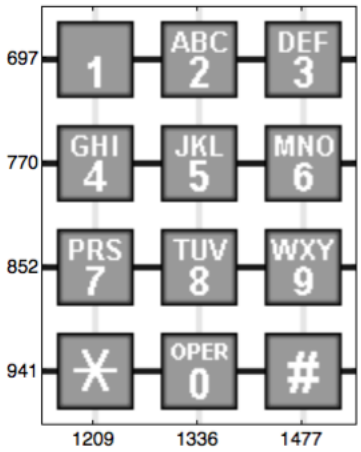

Figure 1.5: telephone keypad

Figure 1.6 represents the signal and its amplitude spectrum when pressing the "1" key of the dial pad. The two frequencies can be clearly identified from the spectrum.

A file named `hidden_number.mat` containing an American telephone number is available in the downloaded folder.
This file contains the column vectors `y` et `t` containing the telephone number samples and

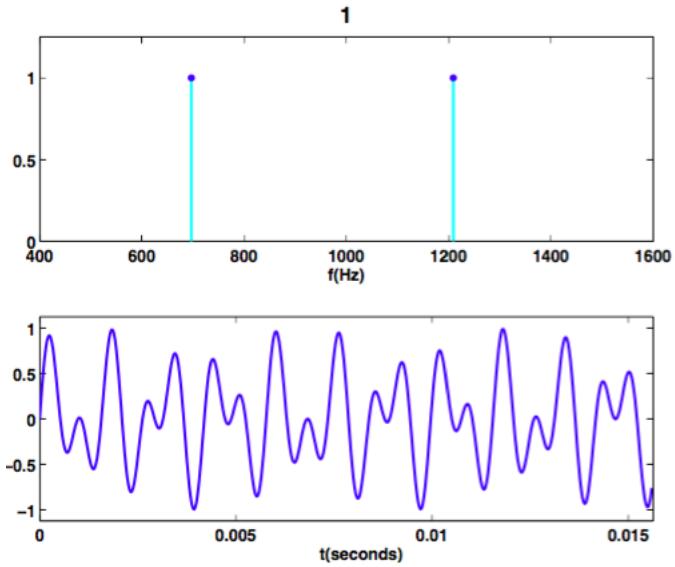

Figure 1.6: Time evolution and spectrum of the signal resulting from pressing the "1" key on the dial pad

recording time-instants as well as sampling frequency `fs`.

Load the DTFM data in MATLAB by typing the command:
`load hidden_number;`

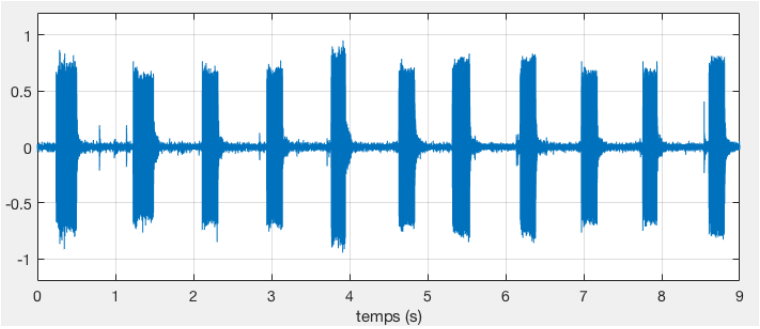

Figure 1.7: Time evolution and spectrum of the DTFM signal

Plot the time evolution of the telephone number over the 9 seconds of recording using the following command :
`plot(t,y),grid,shg`
Make sure you get a similar plot to the one shown in Figure 1.7.
Address the following questions:

1. How many digits are in the hidden phone number?
2. Is it possible to determine the phone number from the time evolution of the composite signal?

Listen to the recorded number by typing the following command:
`sound(y,fs);`
Convert the signal to a MATLAB timetable object.

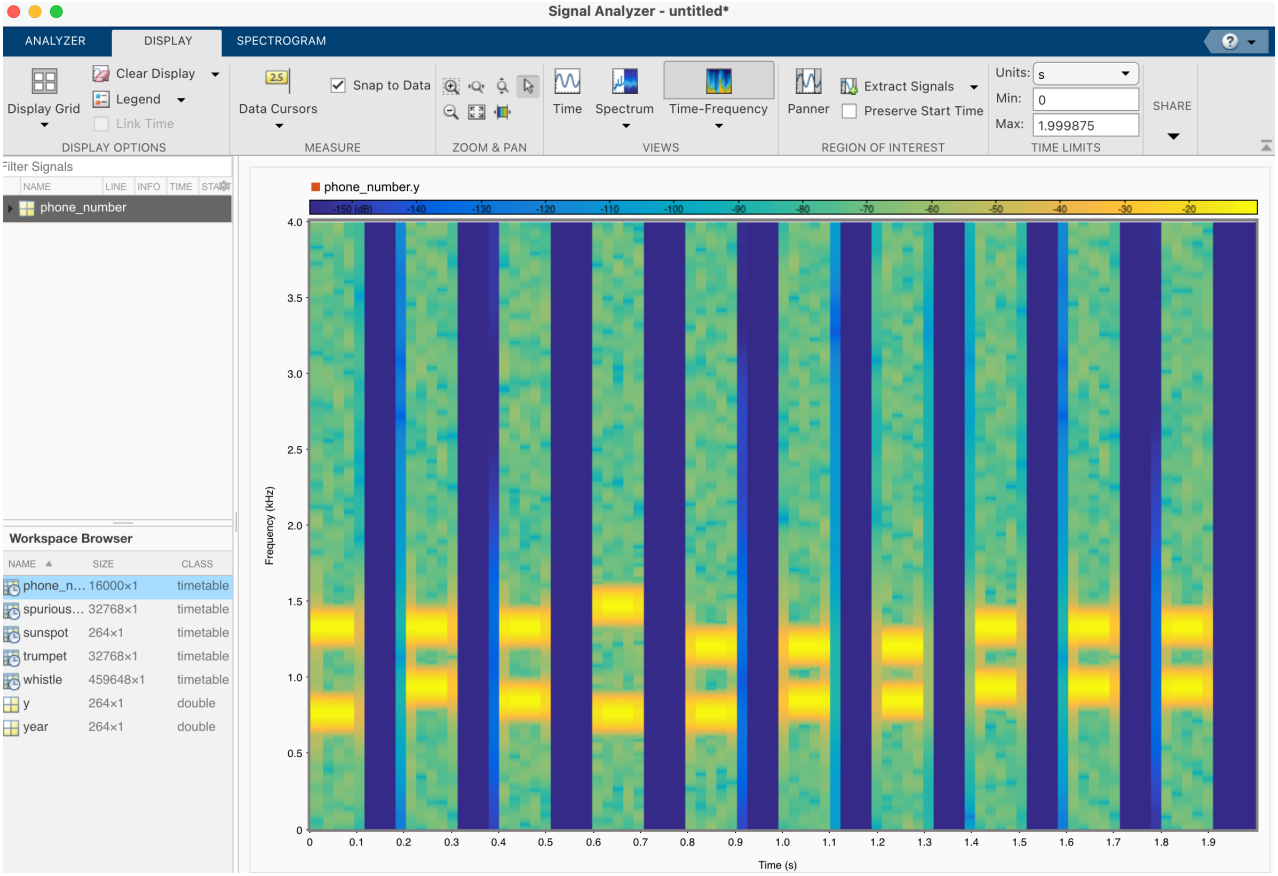

Figure 1.8: Spectrogram of the DTFM signal

```
t=(0:length(y)-1)'/fs;
phone_number = timetable(seconds(t),y);
```

Use the `SignalAnalyzer` App to address the following questions:

1. Display the time-domain evolution and the spectrum of the recorded best two second signal.

2. Plot the spectrogram view (time-frequency representation) of the hidden number by clicking the Time-Frequency tab. You should get the spectrogram as shown in Figure 1.8. The y-axis scale representing the frequency can be used to identify each frequency over every tenth second. From the spectrogram view, it can now be observed that the first tone has its frequency content concentrated around 770 Hz and 1336 Hz, corresponding to the digit '5' in the DTMF standard. Continue the analysis for determining the rest of the phone number.

3. Another way to determine the phone number is to use the panner and to define time-windows of about 0.1s. The spectrum obtained for the time-domain windowed signal by using the Short-Term Discrete Fourier Transform (STDFT) should allow you to determine the phone number.

4. Using the Internet, find the name of the company which owns the number.

5. Find the address in the United States and use Google Maps to view the main entrance of the US company. A clue is provided in Figure 1.9.

Figure 1.9: Clue to determine the owner of the hidden number to be identified

In practice, the detection of DTMF tones is performed by using digital signal processing techniques such as the Goertzel algorithm. To learn more, browse:
`https://en.wikipedia.org/wiki/Goertzel_algorithm`

1.2.3 Determination of the type of sounds produced by your whistle

This assignment will assess your whistling talents and particularly determine if you are able to whistle the purest way possible. The technique of tightening the lips is certainly the most common form of melodic whistle. The pitch or frequency and musical intonation depend on the performer, and its length is limited by the breath of the whistler. It is recalled that the frequency range audible to the human ear ranges from 20 Hz to 20 kHz. When the frequency is low, the sound is low-pitched (20 to 200 Hz). A medium frequency is between 200 and 1000 Hz and a high-pitched sound is when the frequency is between 1000 and 15000 Hz.
Use your mobile phone to record you whistling for about 10s. Try to whistle in the medium band if possible. Do not blow straight into the microphone. Hold the microphone from the side, so the sound is recorded from the side. This will reduce the incidence of disturbing blowing sounds.

Send the recorded file by e-mail to your address and save it to your working directory. A whistle data recorded from an iPhone 7 is available in the downloaded file for those who cannot record their whistle by using their own smartphone.

Open the signal analyzer App and select the best two seconds.

Address the following questions/tasks:

1. Display the spectrum of the recorded best two second signal.
2. What is the sampling frequency by default of the mobile phone ?
3. Does your mobile phone include an anti-aliasing filter applied to the analogue signal before digitizing it? If so, estimate the low-pass filter cut-off frequency and check its consistency with the sampling frequency.
4. Determine the dominating frequency of your whistle from the spectrum.
5. Can you observe any significant harmonics from the spectrum ?

6. Deduce the type of sound (low-pitched, medium or high-pitched) produced by your whistle.

7. Use the different options available in the `Zoom and Pan tab` to zoom in the low-frequency part of the spectrum. Observe the nominal frequency of the electrical supply voltage.

1.2.4 Blue whale vocalization

This assignment aims at analyzing a blue whale vocalization in the time and frequency domains.
A Pacific blue whale recording is available in Matlab. The file is from the library of animal vocalizations maintained by the Cornell University Bioacoustics Research Program. Load the blue whale vocalization data:

```
[y,fs] = audioread('bluewhale.au');
```

Convert the signal to a timetable object.

```
t=(0:length(y)-1)'/fs;
whale = timetable(seconds(t),y);
```

The time scale in the data has been compressed by a factor of 10 to raise the pitch and make the sound more audible. To hear the sound made by the blue whale, type:

```
soundsc(y,fs);
```

Open the Signal Analyzer App and drag the timetable to the main frame on the right. Four features stand out from the noise. The first is known as a trill, and the other three are known as moans.

By using the signal analyzer App, analyze the time and frequency contents of :

1. the trill;

2. each of the three moans;

3. plot the spectrogram

4. deduce the type of sound (low-pitched, medium or high-pitched) produced by a blue whale vocalization.

To learn more about how to extract regions of interest from whale song, browse:
`https://fr.mathworks.com/help/signal/ug/extract-regions-of-interest-from-whale-song.html`

1.2.5 Prediction of the Sun's activity over the coming decade

This assignment aims at analyzing sunspot data which is available from NASA for the years 1749-2012.

Sunspots are temporary phenomena on the Sun's photosphere that appear as spots darker than the surrounding areas on the side of the Sun visible to an Earth observer, as shown in Figure 1.10.

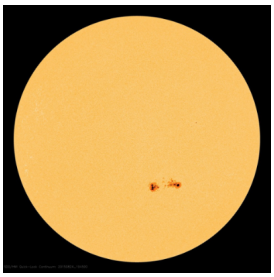

Figure 1.10: Sunspots observed on the Sun - August 24, 2015. NASA.

The Sun is the source of the solar wind: a flow of gases from the Sun that streams past the Earth at speeds of more than 500 km/s. Disturbances in the solar wind shake the Earth's magnetic field and pump energy into the radiation belts. Regions on the surface of the Sun often flare and give off ultraviolet light and x-rays that heat up the Earth's upper atmosphere.

This **"Space Weather"** can change the orbits of satellites and shorten mission lifetimes. The excess radiation can physically damage satellites and pose a threat to astronauts. Shaking the Earth's magnetic field can also cause voltage surges in power lines that destroy equipment and knock out power over large areas on the planet. As we become more dependent upon satellites in space and electricity power on Earth, we will increasingly feel the effects of space weather and therefore we need to better understand and predict it.
To learn more about space weather, browse:
`https://www.futura-sciences.com/sciences/actualites/...`
`soleil-nouveau-cycle-solaire-pourrait-etre-plus-intense-jamais-observe-84601`
`https://en.wikipedia.org/wiki/Space_weather`

The nature and causes of the sunspot cycle constitute one of the great mysteries of solar astronomy. While we now know many details about the sunspot cycle, we are still unable to produce a model that will allow us to reliably predict future sunspot numbers using basic physical principles. This problem is a little like trying to predict the severity of next year's winter or summer weather.
For almost 300 years, astronomers have tabulated the number and size of sunspots every year. Load the sunspot data available in the folder that you have downloaded :

```
load sunspot;
```

Convert the signal to a timetable.

```
sunspot = timetable(years(year),y);
```

Use the signal analyzer App to address the following questions/tasks:

1. Analyze the time and frequency contents of the sunspots since 1749.
2. Determine the main cycle/period in years of the sunspot activity.
3. Predict in which year we should reach a new maximum sunspot number.

1.2.6 More practical examples

To get more practical examples of frequency-domain signal analysis, execute the command below and run every section included in the Matlab file:
`openExample('signal/FrequencyAnalysisExample');`

1.2.7 Image compression with the FFT

The two-dimensional FFT is effective for image compression, as many of the Fourier coefficients are small and may be neglected without loss in image quality. Thus, only a few large Fourier coefficients must be stored and transmitted.
Watch the video from Steven Brunton that shows how to compress images with the FFT:
`https://www.youtube.com/watch?v=KGiV_2i713It=620s`

Open and run the Matlab file `ImageCompression_by_FFT.m`.

Modify the threshold to keep 5%, 1%, and 0.2% of the largest Fourier coefficients.

Lab 2

Applications of digital filtering

The work done during this 4h00 lab will be noted in a report and marked. You are asked to follow the instructions given below to write your report.

Instructions for writing your lab report

A lab report is a scientific document. It should be self-contained: that is, someone who has never seen the lab instructions should be able to understand the problems that you are solving and how you are solving them. All the choices you have made should be clearly motivated. Comment and explain all your plots and your Matlab-code, so as to make it easy to follow your way of thinking.

Your lab report can be written in French or in English but do not mix both languages. It should be organized as follows:

- a general introduction specifying the objectives of the lab
- For each assignment or exercise:
  - ▷ a brief presentation of the expected outcomes
  - ▷ a description of the obtained results in graphical and or numerical form
  - ▷ a critical analysis of the results
  - ▷ a short conclusion
- a general conclusion explaining what has been understood during the lab and any difficulties encountered.
- an appendix containing the Matlab programs you have developed.
- It is highly recommended to use the Matlab `Live Editor` to generate your report. In the Live Editor, you can create live scripts that show output together with the code that produced it. Moreover, from the .mlx file, you can create a pdf file very easily to generate your report. If necessary, you can watch the tutorial introduction to the Matlab Live Editor available at:
  `www.mathworks.com/videos/using-the-live-editor-117940.html?s_tid=vid_pers_recs`

Sending your report to the tutor

The report should be written in groups of two students and sent by email to the tutor in the form of a single pdf format file attached to the message by the deadline decided with your instructor. Please indicate the following "subject" for your email to send your report to the tutor : **Lab2_DSP_Name_Group"**

Downloading of the data needed for the lab

1. Download the zipped file **Lab2_DSP.zip** from the course website and save it in your Matlab working directory.
2. Start Matlab.
3. By clicking on the browse for folder icon  , **change the current folder of Matlab so that it becomes your Lab2 folder** that contains the files needed for this lab.

Recording of sound files

You will do some recordings of sounds with your own hardware, for example, with the microphone of your smartphone.

Layout of the Lab

The lab is structured into two main parts:

1. A tutorial introduction of the interactive `Filter Designer` app to design digital FIR and IIR filters and to implement them by using the command line routine `filter`.
2. Several assignments where you have to analyze your data recordings and other real-life data with the methods and theory that you learned during the Lectures.
   The assignments aim at designing digital filtering for diverse applications in different fields :
   - Whistles: how to separate two whistles simultaneously recorded?
   - Temperature anomalies: how to smooth monthly temperature anomaly data to better observe the trend?
   - Sunspots: how to smooth daily sunspot data?
   - Underwater robotic vehicles: how to preprocess recorded noisy data ?
   - Noisy artefact: how to filter out an artefact noisy harmonic in a musical signal?
   - Spikes: how to cancel out unwanted spikes on an electrical signal?

2.1 Tutorial introduction of the Filter Designer App

We will create a signal to be used in this tutorial. The signal is a 100 Hz sine wave in additive white Gaussian noise of standard deviation 0.5.

```
Fs = 1000;
N = 10000;
t = linspace(0,10,N)';
x0 =cos(2*pi*100*t);
x = x0+0.5*randn(size(t));
noisy_sine_v0=timetable(seconds(t),x);
noisy_sine=noisy_sine_v0;
```

- Start the App by entering `filterDesigner` at the command line or by selecting it from the App Menu. The main window of the Filter Designer App should then open.
- Set the Response Type to Lowpass.
- Set the Design Method to FIR and select the Window method.
- Under Filter Order, select Specify order. Set the order to 20.
- Under Frequency Specifications, set Units to Hz, Fs to 1000, and Fc to 110.
- Click Design Filter.

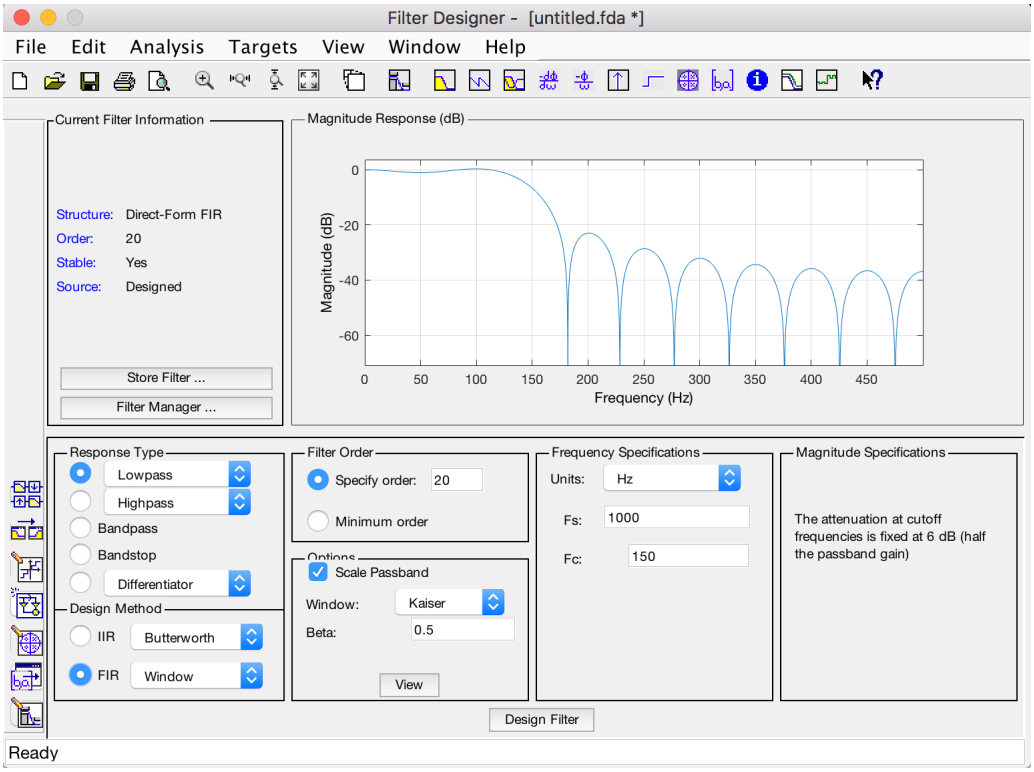

Figure 2.1: Main window of the Filter Designer App

- Select File > Export... to export your FIR filter to the Matlab workspace as coefficients or a filter object. In this example, export the filter as coefficients. Specify the variable name as `Num`

- Click Export.

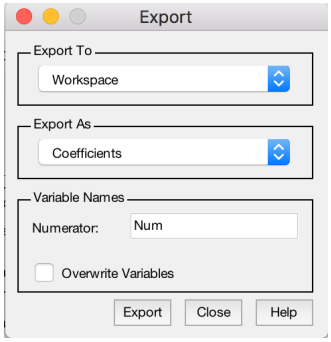

Figure 2.2: Window to export the designed filter as coefficients

- Go to the command window and implement the filtering of the noisy sine signal with the `filter` function by using the exported filter. Plot the result for the five periods of the 100 Hz sinusoid.

```
xf = filter(Num,1,x);
plot(t,x,t,xf)
xlim([0.5 0.55])
xlabel('Time (s)')
ylabel('Amplitude')
legend('Noisy signal','Filtered signal')
```

- You can implement zero-phase filtering with the `filtfilt` routine:

```
xzp = filtfilt(Num,1,x);
plot(t,x0,t,x,t,xf,t,xzp)
xlim([0.5 0.55])
xlabel('Time (s)')
ylabel('Amplitude')
legend('Noise-free','Noisy signal','Filtered signal','Zero-phase filtered
signal')
grid
```

Note that the filter transfer function (via its numerator coefficients here stored in the variable `Num` can also be directly obtained by using the Matlab filter designer routine `fir1` for FIR filters with the following code:

```
fc=110;
N=20;
Num=fir1(N,fc/(Fs/2),boxcar(N+1));
```

You can then implement the filtering with the `filter` or `filtfilt` routines.

Another alternative that can be very useful in the following assignments is to exploit the `signalAnalyzer` graphical interface used in Lab 1. It is then easy to display the features of the original signal in the time and frequency domains but also to apply some filtering operations and see the obtained results in the time and frequency domains. Note however that if you choose to apply the filtering in the `signalAnalyzer` tool, you will have however less options for the type of filters (FIR or IIR) to be applied but it might be sufficient for the proposed analysis.

2.2 Assignments - Applications of digital filtering in different fields

2.2.1 Separation of two whistles

Use your mobile phone to record you and your partner whistling simultaneously for about 10s. One of you should whistle in the medium band and the other in the high-pitched band.
Address the following questions/tasks:

1. Display the spectrum of the recorded best two second signal.
2. Determine the dominating frequency of each of the two whistles.
3. Design a digital processing approach to separate the two whistles.
4. Implement your approach and check its efficiency by listening to the two filtered signals.

2.2.2 Smoothing of temperature anomalies

You are going to investigate the monthly temperature anomalies since 1880 that has used to illustrate the data smoothing concept during the last lecture.

Trends in annual global temperatures are an important indicator of the magnitude of climate change and its possible impacts. Global temperature has been rising steadily since the end of the 19th century. The rate of increase has been particularly high since the 1970s.
The goal is to obtain a smoother version of the temperature anomalies so that the trend for the upcoming decade can be better foreseen.
Address the following questions/tasks:

1. Open and run the `temp_anomalies_plot.m` file.
2. Observe the temperature anomaly increase over time.
3. Apply a moving average (MA) filter to the temperature anomalies described by the following difference equation:
$$y(k) = \frac{1}{2}e(k) + \frac{1}{2}e(k-1)$$
It is recalled that MA filters are special types of the more general family of FIR filters. The smoothing MA filter above is therefore a simple low-pass FIR filter of order 1. Plot the frequency response of the MA filter of order 1.
4. Set the order of the MA filter to 9 and apply it to the temperature anomalies.
5. Can you better observe the trend over the years ?
6. Plot the frequency response magnitude of the MA filter of order 9 and compare it with the frequency response magnitude of the first-order MA filter.
7. Apply an IIR filter to the temperature anomalies described by the following difference equation:
$$y(k) = \frac{4}{5}y(k-1) + \frac{1}{5}e(k)$$

8. Plot the frequency response magnitude of the IIR filter and compare it with the best FIR filter frequency response magnitude.

2.2.3 Daily sunspot data

The file `daily_sunspot_data.mat` contains the daily sunspot data stored in `sunspot` since 1800. The days of measurement are stored in `day`.
Address the following questions/tasks:

1. Plot the daily sunspot data.

2. Develop and implement a processing method to smooth the sunspot data.

3. Determine the main cycle/period in years of the sunspot activity from the daily data.

4. The file `yearly_sunspot_data.mat` contains the daily sunspot data stored in `sunspot` since 1700. The years of measurement are stored in `years`. Superpose the processed daily data and the yearly daily data.

5. If the two signals are different, figure out why. Suggest how the yearly sunspot data have been computed from the daily measurement. Implement your suggestion to validate your conjecture.

2.2.4 Removing means and filtering out noise in thruster data

Having a model is often a key step you to design and test a controller. Data-driven system identification is an interactive and systematic way of modeling dynamical system behavior by building mathematical models on the basis of experimental input/output data. System identification includes several stages such as importing, analyzing and preprocessing the data, trying out different model structures, and evaluating the resulting models. Two datasets of input and output data from the actuator (a thruster) used in underwater robotic vehicles are used to demonstrate the data preprocessing stage.
The T200 Thruster from Blue Robotics, shown in Figure 2.3, is one of the world's most popular underwater thruster for ROVs, AUVs or surface vessels. Its flooded motor design makes it compact but efficient and powerful.

Figure 2.3: The T200 Thruster from Blue Robotics.

The T200 thruster is in use on thousands of marine robotic vehicles including the BlueROV2 as shown in Figure 2.4.

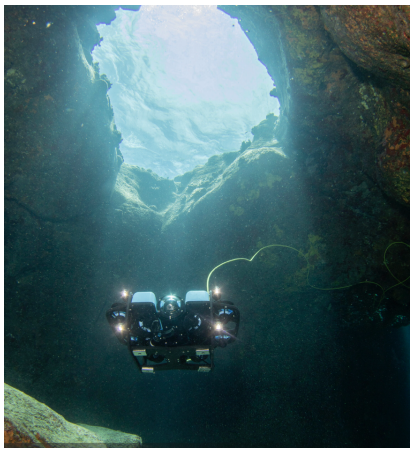

Figure 2.4: The marine robotic vehicle BlueROV2 - `www.bluerobotics.com`.

1. Open and run the file `import_thruster_data.mlx` in Matlab that loads dataset[1] from two different inputs (sine chirp and square chirp signals) applied to a Blue Robotics T200 thruster. You can watch a short video showing one of the two experimental tests at: `www.youtube.com/watch?v=RU_LOtrE0Bo` (go to the 17mn25).

We will concentrate on the analysis and pre-processing of the square chirp signal and the response of the thruster.
A chirp is a signal in which the frequency increases or decreases with time. In some application areas, the term chirp is used interchangeably with sweep signal. Chirp signals are commonly applied to sonar, radar or laser systems, and to other applications.
Address the following questions/tasks:

1. By using the signalAnalyzer App, plot the sine chirp input and its spectrum.
2. Plot its spectrogram and observe that the chirp input frequency increases with time.
3. Plot the thruster response and its spectrum.
4. Develop a pre-processing to the thruster response that removes its mean and low-pass filter out the unwanted high-frequency noisy and nonlinear components. All the pre-processing can be done in the SignalAnalyser App.
5. Implement your approach and check its efficiency by superposing the spectrum of the original (you would need first to duplicate the original signal in Matlab) and filtered signal.
6. Repeat the analysis for the square chirp input.

2.2.5 Special filtering for removing a noisy artefact in a piece of music

The file `music_track.mat` contains a musical signal stored in `y` and sampled at `fs` that is contaminated by a noisy artefact.
Address the following questions/tasks:

[1] MathWorks Student Competitions Team (2022). MATLAB and Simulink Robotics Arena : From Data to Model (www.mathworks.com/matlabcentral/fileexchange/65919-matlab-and-simulink-robotics-arena-from-data-to-model), MATLAB Central File Exchange

1. Listen to the noisy signal.
2. By using the signalAnalyzer App, determine the fundamental frequency of the spurious tone.
3. The goal now is to design a notch filter to cancel out the spurious harmonic. Refer to the course slides to recall the transfer function $H(z)$ of a so-called notch filter.
4. Matlab has no function dedicated to this type of notch filter. It is possible to directly define the coefficients of the transfer function $H(z)$ based on the following solution:

```
f0=...; %interference frequency to be deleted \Omega0=2*pi*f0/fs;
a=...; %coefficient influencing the selectiveness of the notch filter
num=[...]; %coefficients of the numerator of the notch filter in decreasing power of z
den=[...];  %coefficients of the denominator of the notch filter in decreasing power of z
figure
freqz(num,den,1024,fs); %Calculation of the Bode diagram for 1024 points
yf=filter(num,den,y); % filtering of the trumpet signal by the designed notch filter
soundsc(yf,fs);
```

5. The effectiveness of the filter may also be observed using the signalAnalyzer App by superimposing the spectrum of the original and filtered signals. Optimise the settings of the notch filter to most effectively remove the disruptive harmonic and recover the original trumpet signal with the less distorsion possible.

2.2.6 Special filtering for spiky electrical voltage

The file `electrical_voltage.mat` contains a voltage signal stored in `y` and sampled at `fs` that is contaminated by unwanted spikes.
Address the following questions/tasks:

1. Plot the electrical signal and observe the spikes.
2. Develop and implement a median filter to cancel out the unwanted spikes on the electrical signal (read the lecture slides on median filter).
3. Determine the fundamental frequency of the cleaned electrical signal.
4. Was the signal recorded in France. Explain your answer ?

2.2.7 Summary and reflections

Summarize and reflect on what you have done and learned during this lab.

English to French glossary

dial pad	:	clavier numérique
high-pitched	:	aigu
low-pitched	:	grave
spurious	:	parasite
sunspots	:	tâches solaires
voltage surges	:	surchages électriques
FIR	:	Finite Impulse Response
IIR	:	Infinite Impulse Response