CINTAL - Centro de Investigação Tecnológica do Algarve

Universidade do Algarve

Acoustic pressure and particle motion power spectrum estimation with Matlab®

S.M.Jesus

Rep 03/19 - SiPLAB
Date: 31/October/2019
Revised: 28/December/2020

University of Algarve
Campus de Gambelas
8005-139, Faro
Portugal

tel: +351-289800131
fax: +351-289864258
cintal@ualg.pt
www.ualg.pt/cintal

Work requested by	University of Algarve (under project JONAS)
Laboratory performing the work	SiPLAB - Signal Processing Laboratory FCT, Campus de Gambelas, Universidade do Algarve, 8005-139 Faro, Portugal tel: +351-289800949 info@siplab.uceh.ualg.pt, www.siplab.fct.ualg.pt
Projects	JONAS (EAPA-52/2018) and BIOCOM
Title	Acoustic pressure and particle motion power spectrum estimation with Matlab®
Authors	S.M.Jesus
Date	October 31, 2019 (revised: December 28, 2020)
Reference	03/19 - SiPLAB
Number of pages	35 (thirty five)
Summary	This report recalls definitions and makes a brief review on power spectrum estimation. Various implementations are compared using Matlab® widely used routines, discrete functions and matched with standard texts. This is then applied to underwater acoustics for computing received levels at discrete frequencies from measured or modelled data, both for sound pressure and particle motion.
Clearance level	UNCLASSIFIED
Distribution list	SiPLAB, CINTAL, UALG, UCC, QO, CEFAS, SHOM, UPC, MS, IH and PLOCAN
Attached	None
Total number of recipients	11 (eleven)

Approved for publication

A.B. Ruano

President Administration Board

Foreword and Acknowledgment

This work was performed during the processing of the data and the writing of the BIOCOM'19 experiment data report [1], while finding difficulties to find clear and practical well accepted definitions for estimating the power spectral density of underwater acoustic pressure and particle motion. Thanks are due to several colleagues with which I discussed this subject, namely Fábio Xavier (from IEAPM, Brasil), Cristiano Soares (from Marsensing) and Florent Le Courtois (from SHOM, France), which comments are greatly appreciated and resulted in several improvements.

This version has been revised two times: one (February 2020) to include a justification for the usage of a slightly different definition for the DFT, where an appendix was added for that purpose, and also an additional test case with a real data record; the other (December 2020) was a correction on equations (2.7) to (2.9).

intentionally blank

Contents

List of Figures

Abstract

This document briefly describes the background on power spectrum estimation using well known reference books and texts. Reference definitions are then implemented into Matlab® using discrete functions and matched to widely used Matlab® routines. Basic simulations and respective codes are given to illustrate the various cases.

In a second chapter this experience is applied in the underwater acoustics case for producing received level estimates at discrete frequencies from measured and modeled data both for acoustic pressure and for particle motion, the latter being in most case not referred to in current standards.

This report will be useful for those scientists involved in comparing measured noise fields from place to place or over time and in integrating that knowledge on modeled sound maps, or in whichever task spectral power amplitude is most important.

intentionally blank

Chapter 1

Introduction

In many situations power spectrum estimation is used to determine the frequency content of a given data record. For decades the scope of a large body of work has been to estimate frequencies with enough precision and resolution to discriminate closely spaced spectral lines. This led to the so-called high resolution methods which, most of them, are not strictly speaking true spectral estimators since the focus was on spectral resolution and not on amplitude estimation. The analogy between spectral lines and plane waves, and between frequency and direction of arrival (DoA), makes this body of work directly applicable to array signal processing. See classical literature on the subject in the review paper of Kay and Marple [2], in the book of Marple [3] and for DOA applications on Krim *et al.* [4].

The class of true spectral amplitude estimators is restricted to two types: the classical methods and the minimum-variance based estimators. The classical estimators encompass the periodogram and the correlogram. The former is based on the principle of conservation of energy (or Parseval theorem) and the latter is based on the Wiener-Khinchin theorem. The minimum-variance based estimators belong to the so-called high resolution methods and use the assumption that the data record is formed by one sinusoid embedded in white noise. The derived estimator (one of which is the Minimum Variance Distortion Less - MVDR estimator) attempts to minimize all frequencies (respectively directions, in the DOA case) but that of the sinusoid (plane wave). The assumption behind this class of methods makes it well suited for estimating sinusoids but not extended background sources, so we will concentrate in the classical estimators and in particular in the periodogram. The reason for this is mostly computational since the periodogram (and its variants) are based on the Fast Fourier Transform (FFT) which makes it much faster than the correlogram.

This is also the approach followed in Matlab where widely used routines such as `spectrogram` and `pwelch` implement variants of the periodogram algorithm. One of the issues we address here is to go back to the classical definitions and compare discrete implementations with those obtained by these "ready-to-use" routines.

In the second chapter of this work we apply the Power Spectral Density (PSD) and Power Spectrum (PS) estimators' definitions in the underwater acoustics context, namely for determining sound pressure level, particle acceleration and pressure equivalent particle velocity PSD and PS.

Chapter 2

Power Spectrum Estimators and Matlab®

2.1 Power and energy

For deterministic signals, the conservation of energy (also called the Parseval theorem) implies that the total energy is conserved from the time to the frequency domain, *i.e.*,

$$\int |h(t)|^2 dt = \int |H(f)|^2 df, \tag{2.1}$$

where $H(f)$ is the Fourier Transform (FT) of signal $h(t)$. Since the term on the right hand side of (2.1) is the total energy integrated over the frequency domain, we may say that

$$ESD(f) = |H(f)|^2, \tag{2.2}$$

is the energy per unit of frequency, or the *Energy Spectral Density* (ESD), which unit should be [J/Hz]. Since energy is defined as power multiplied by time, we may define the *Power Spectral Density* (PSD), by the ratio of ESD divided by time, *i.e.*,

$$PSD(f) = \frac{ESD(f)}{T} = \frac{|H(f)|^2}{T}, \tag{2.3}$$

now in [W/Hz]. Of course the total mean power in a given band $B = [f_1, f_2]$ is the integral of the PSD over that band, that is

$$P_B = \int_{-f_2}^{-f_1} PSD(f)df + \int_{f_1}^{f_2} PSD(f)df, \tag{2.4}$$

where the negative frequency side of the PSD was included for completeness. The *Power Spectrum* (PS) is given as the incremental value of the PSD for a frequency window Δf [Hz]

$$PS(f) = PSD(f)\Delta f, \tag{2.5}$$

simply in watts [W].

For random signals the PSD is defined as the FT of the signal autocorrelation function via the Wiener-Khintchine theorem. It is shown that in order to define the PSD directly

from the signal FT the following relation holds (see *e.g.* Marple [3])

$$PSD(f) = \lim_{T\to\infty} \frac{1}{T} E\left[\left|\int_0^T x(t)e^{-j2\pi ft}dt\right|^2\right], \tag{2.6}$$

where mathematical expectation $E[\,]$ assumes that the mean power of $x(t)$ in the interval $[0, T]$ is finite, its variance is zero and its mean value is equal to the PSD, when T tends to ∞. The expectation operator in (2.6) is the most often forgotten / ignored aspect in classical spectral estimation.

For discrete signals recorded in limited time intervals, the energy theorem (2.1) is usually written as

$$\sum_{n=0}^{N-1} |x(n)|^2 = \sum_{k=0}^{N-1} |X(k)|^2. \tag{2.7}$$

Marple [3] claims that a better approximation for the discrete summation over time and frequency is to consider incremental steps weighted by the sampling intervals both, in the time and frequency domain. This results in a scaling of the DFT which implications are shown in appendix A. Among others, this claim also holds for the energy theorem since it means that the energy incremental continuous step $|x(t)|^2 dt$ is better approximated in the discrete case by $|x(n)|^2\Delta T$, where ΔT is the sampling interval. Thus, the energy theorem becomes, in the discrete case

$$\sum_{n=0}^{N-1} |x(n)|^2\Delta T = \sum_{k=0}^{N-1} |X(k)|^2\Delta f, \tag{2.8}$$

where Δf is the frequency sampling interval. Since $\Delta f = f_s/N$ and $f_s = 1/\Delta T$, where f_s is the sampling frequency, we may rewrite (2.8) as

$$N\Delta T \sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{\Delta T} \sum_{k=0}^{N-1} |X(k)|^2, \tag{2.9}$$

which will be useful later.

2.2 Classical power estimators

In practice, sound recording, like any other experimental activity, involves random quantities or noise, that may be of acoustic and/or electronic origin, generated in the recorder itself. The recorded data is then assimilated to a stochastic process and therefore only estimates of moments (mostly first and second order) are meaningful. This is true both in time and frequency, which clearly means that the PSD given by (2.3) or the PS by (2.5) will not be determined exactly and only their estimates will be obtained from finite data records. As mentioned above the so-called classical spectral PSD estimators are the periodogram and the correlogram, described in every signal processing text book (see *e.g.* Marple [3]).

2.2.1 Power Spectral Density

To make the long story short, the periodogram is basically a time average of the module squared of the Fourier Transform (FT), or DFT in the discrete case, of a stretch of

recorded signal. As mentioned above time average to implement the expectation of (2.6) is essential to obtain a stable PSD estimate. Adopting a common notation the *sample PSD* can be written as (see eq. (5.31) of [3])

$$\begin{aligned} PSD(f) &= \frac{1}{N\Delta T}\Big|\Delta T \sum_{n=0}^{N-1} x(n)e^{-j2\pi fn\Delta T}\Big|^2, \\ &= \frac{\Delta T}{N}\Big|\sum_{n=0}^{N-1} x(n)e^{-j2\pi fn\Delta T}\Big|^2, \end{aligned} \tag{2.10}$$

where the Discrete Time Fourier Transform (DTFT) was used, $\Delta T = 1/f_s$ is the sampling interval and N is the number of samples in the data record. The scaling of the DFT by the sampling interval ΔT allows to obtain a correctly scaled PSD (see appendix A and discussion in [3] pp.41-43). So, $N\Delta T$ is the total record time and (2.10) represents a PSD, according to definition (2.3), as the total energy divided by record duration, which unit is therefore [W/Hz]. The reason why this is a *sample PSD* is because it is based on a single N-sample data record and, therefore, yields unstable estimates for random processes, simply because the expectation operator (see (5.30) in [3]) was ignored. It is a common error to take (2.10) as a PSD estimator, which it is not. For finite ergodic (and stationnary) data records expectation is replaced by time averaging, for which strategies abound in the literature but the most used is probably that proposed by Welch: the Welch periodogram. As in all periodogram-style PSD estimators, the Welch periodogram divides the data record in, say P, data segments. Then a sample PSD such as (2.10) is calculated on each p-th segment and the result averaged over the segments. Compared to other periodogram estimates (*e.g.* Daniel's or Bartllet) the Welch periodogram has the following features: i) by proposing an overlap between data segments within the finite data record it allows for an increased stability and therefore a reduction of the variance of the estimator while keeping a constant resolution and ii) it proposes time windowing in general, and the Hann window in particular, as a mean for attenuating sidelobes at an expense of a slight loss in resolution. In other words, Welch proposes a well balanced compromise between resolution and stability, which is known to be the plague of classical PSD estimators. Moreover, the implementation of the Welch periodogram using the FFT algorithm greatly contributed to its success and widespread.

So, the Welch periodogram PSD estimate $PSD_W(f)$ is given as the average of the P sample PSD estimated over each data segment as

$$PSD_W(f) = \frac{1}{P}\sum_{p=0}^{P-1} PSD^{(p)}(f), \tag{2.11}$$

where, using (2.10), the (p)-th segment sample spectrum is given by

$$PSD^{(p)}(f) = \frac{\Delta T}{UD}\Big|\sum_{n=0}^{D-1} w(n)x(n+pS)e^{-j2\pi fn\Delta T}\Big|^2, \quad -f_s/2 < f \leq f_s/2, \tag{2.12}$$

where D plays the role of N of (2.10) and is the number of samples in a data segment, $w(n)$ is the considered time window function, and U is the time window power, given as the total energy divided by total time duration, as

$$\begin{aligned} U &= \frac{1}{D\Delta T}\Big[\Delta T \sum_{n=0}^{D-1} w^2(n)\Big], \\ &= \frac{1}{D}\sum_{n=0}^{D-1} w^2(n), \end{aligned} \tag{2.13}$$

used here for power normalization, and is given by the discrete integration over the $D\Delta T$ interval of the window function. Note that a rectangular window $\{w(i) = 1, i = 1, \ldots, D\}$ leads to a correct power normalization of $U = 1$. The data record of length N is therefore segmented into P segments of length D with S samples of overlap between segments. The number of segments is given as the integer part of $(N-D)/S+1$. The Welch periodogram uses a Hann function time window and an overlap of $S = D/2$, *i.e.*, a 50%, which has been shown to held optimal results. It is clear from this formulation that the number of samples taken for the FT has decreased from N to D, which is the price to pay for increased stability.

In the discrete case, we use a DFT of length $N_f \geq D$, normally chosen as a power of 2 for FFT efficiency, that gives

$$PSD^{(p)}(k) = \frac{\Delta T}{UD}\Big| \sum_{n=0}^{N_f-1} w(n)x(n+pS)e^{-j2\pi kn/N_f}\Big|^2, \tag{2.14}$$

and therefore

$$PSD_W(k) = \frac{1}{P}\sum_{p=0}^{P-1} PSD^{(p)}(k), \tag{2.15}$$

where now the frequency discrete index k runs as $-N_f/2 + 1 \leq k \leq N_f/2$, which is equivalent to $-f_s/2 < f \leq f_s/2$. In many cases, for computation efficiency, the FFT block size N_f is chosen equal to the data window size D if D is a power of 2 and, if not, N_f is the next power of 2 larger than D and $N_f - D$ zeroes are appended to the data record, which does not change window or signal energy, and has the effect of performing frequency interpolation.

When the PSD of real signals is represented as one-sided for $f \in [0, f_s/2]$, it requires scaling by a factor of 2 in order to account for the negative half of the spectrum. A remark should be made for discrete time, in which case the spectrum is periodic and the spectrum samples $P(0)$ and $P(N_f/2)$ are not repeated and therefore should not be doubled. So, the final PSD representation should be

$$PSD_W^{\text{real}}(k) = \begin{cases} PSD_W(k) & k = 0 \\ 2PSD_W(k) & k \in [1, N_f/2 - 1] \\ PSD_W(k) & k = N_f/2 \end{cases} \tag{2.16}$$

where $PSD_W(k)$ is given by (2.15).

2.2.2 Power Spectrum

The Power Spectrum (PS) is just a scaled version of the PSD so the power estimates at each discrete frequency represent the true power estimate at that frequency, while for the PSD that same power is distributed along the frequency bin width centered at the given discrete frequency. Therefore, in order for the value of the spectrum to reflect the true power spectrum estimate, as given in (2.5), one needs to integrate the power over the frequency bin window. In the discrete spectrum case, integration over the frequency bin width becomes simply as multiplying the mean PSD value in the bin by the bin width, which is $\Delta f = f_s/N_f$ which, used in (2.14) gives

$$PS^{(p)}(k) = \frac{1}{UDN_f}\Big| \sum_{n=0}^{N_f-1} w(n)x(n+pS)e^{-j2\pi kn/N_f}\Big|^2, \tag{2.17}$$

since $\Delta T \times \Delta f = 1/N_f$, and of course

$$PS(k) = \frac{1}{P}\sum_{p=0}^{P-1} PS^{(p)}(k). \tag{2.18}$$

2.3 Implementation with Matlab®

For Matlab® implementation we will focus on two widely used routines: `spectrogram` and `pwelch`, both available in release 2017a. `Spectrogram` is normally used to analyze the spectral content evolution through time of a given data record: the data is divided into segments and a sample power spectral density (equivalent to (2.14)) is calculated and displayed. The basic command line of the `spectrogram` routine is

```
[S,T,F,P] = spectrogram(x,hann(window),noverlap,nfft,fs)
```

with `window` an integer $\leq$ `nfft`, allows for calculating two time-frequency matrices `S` and `P` with the complex sample spectrum (also known as Short Time Fourier Transform) and the sample PSD of data record `x`, respectively, using data segments of length `window`, Hann windowed, and `nfft` samples for FFT calculation, with `noverlap` = `nfft`/2 denoting a 50% overlap between data segments. There is no averaging done in the computation of matrices `S` or `P` so, they can not represent power spectral estimates. If signal `x` is real, both `S` and `P` are one sided and (2.16) applies to `P`. `T` and `F` are arrays with the time and frequency center bin values, obtained from the number of samples `nfft` and the frequency `fs` (in Hz). In this implementation, the parallel to the definitions of the previous section can be made by allowing `window` = D, `nfft`=N_f and `fs`=f_s. It is unclear why the online Matlab manual of `spectrogram` says that

P = *Power spectral density (PSD) or power spectrum, returned as a matrix. If x is real, then P contains the one-sided modified periodogram estimate of the PSD or power spectrum of each segment*

when `P` is not a peridogram estimate of the PSD or PS since it has the same dimension than `S` and therefore no averaging is performed, whatsoever.

As an alternative, `pwelch` efectively calculates the Welch periodogram PSD estimate as

```
[Pw,Tw,Fw] = pwelch(x,hann(window),noverlap,nfft,fs)
```

where all variables have the same meaning as for the `spectrogram` case. `Pw` is the Welch periodogram PSD estimate obtained by averaging over the entire record duration `x`. If `x` is real, `Pw` is one sided. In our quest to understand what is exactly done in `spectrogram` and `pwelch`, we compare them with the plain FFT derived periodogram using (2.14) - (2.16). So, four alternative algorithms are tested:

1. **spectrogram P**: the mean PSD matrix `P` directly calculated by routine `spectrogram`;
2. **spectrogram S**: the PSD matrix `S` calculated by `spectrogram` is used in (2.14) and then plugged in (2.15) as

$$PSD_W(k) = \frac{1}{UPDf_s}\sum_{p=0}^{P-1} |S^{(p)}(k)|^2, \tag{2.19}$$

and then the weighting of (2.16) is used;

3. **P Welch**: Welch periodogram directly calculated by the `pwelch` Matlab routine initialized with the same parameters as `spectrogram`;

4. **FFT**: the reference PSD estimate with a direct FFT using (2.14) - (2.16) above.

Matlab routines `spectrogram` and `pwelch` also have a switch named `'power'` which, according to the documentation, allows for computing the true power at each frequency. The test is re-run for PS estimation, activating the `'power'` switch for options 1 and 3 above. According to (2.17), a PS estimate may be obtained from the PSD by multiplying by the frequency interval Δf such that the weighting in (2.19) becomes

$$\Delta f \times \frac{1}{UPDf_s} = \frac{1}{UPDN_f} \tag{2.20}$$

the same occurring for the FFT based reference PS estimate.

This test is carried out for two examples: one is a simulated signal of known amplitude so that absolute power estimates may be evaluated and compared; the other is an experimental recording containing a mixture of ocean noise and man made signals.

2.3.1 Example 1: two sinusoids

A simulated $N = 10000$ sample deterministic signal composed of two sinusoids at frequencies $f_1 = k_1\Delta f$ and $f_2 = k_2\Delta f$ with $\Delta f = f_s/N_f$ and k_1, k_2 the closest integers to $f_s/5\Delta f$ and $2f_s/5\Delta f$[1]. The two sinusoids have peak amplitudes $\sqrt{2}$ and $10\sqrt{2}$ volts, respectively. The factor $\sqrt{2}$ allows for the two sinusoids to have exact root-mean-square (rms) amplitudes of 1 and 10 volts, respectively. The sampling rate is $fs = 10547$ Hz, `window`=`nfft`=1024, *i.e.*, the whole data record.

Fig. 2.1 shows the PSD estimate results for the four algorithms. In all cases the results are shown in a non-normalized dB scale as $10\log_{10}[P(k)]$ relative to 1W/Hz. It is remarkable to notice that the four PSD curves perfectly superimpose for this case. This result is deemed as correct since the total energy is maintained from time to frequency as stated in (2.1). In effect the following output is obtained:

```
...from PSD
Energy in time domain : 50.146213 dB
Energy in freq domain by spectrogram power : 50.146213 dB
Energy in freq domain by STFT from spectrogram : 50.146213 dB
Energy in freq domain by Welch periodogram : 50.146213 dB
Energy in freq domain by discrete FFT : 50.146213 dB
```

The energy in time domain is calculated as the sum of the module squared of the time samples in one window. In the frequency domain the energy is the sum of the PSD

[1]this is not strictly necessary but is done here in order to avoid "scalop loss", *i.e.*, the power split between the true signal frequency and the sampled frequency in the frequency domain. Of course, the "scalop loss" effect decreases with increasing `nfft`.

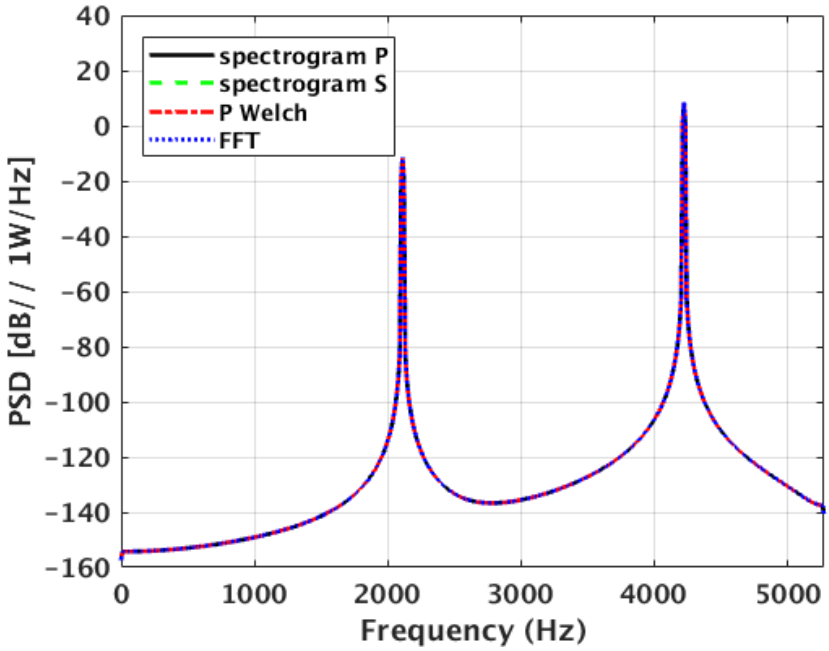

Figure 2.1: *power spectral density (PSD), in dB // 1W/Hz, obtained in the two sinusoid deterministic signal case for:* $N = 10000$, $f_s = 10547$ *Hz,* `window=nfft=1024`, *Hann windowed and a segment overlap of 50%.*

samples over the frequency band multiplied by the total bandwidth, which is the sampling frequency f_s since, according to (2.9),

$$\begin{aligned}\sum_{n=0}^{n-1}|x(n)|^2 &= \frac{1}{\Delta T}\frac{1}{N\Delta T}\sum_{k=0}^{N-1}|X(k)|^2,\\ &= f_s\sum_{k=0}^{N-1}\frac{1}{N\Delta T}|X(k)|^2,\\ &= f_s\sum_{k=0}^{N-1}PSD(k). \end{aligned} \tag{2.21}$$

However, the amplitudes of the two estimated sinusoids at f_1 and f_2 do not coincide with the correct amplitudes which should be 0 dB (1 Vrms) and 20 dB (10 Vrms), respectively. This is so because the PSD is a "power density" estimator and not a strict power estimator.

Using the `'power'` switch and the definition (2.17) allows to obtain the result of Fig. 2.2. In a first approach, the four algorithms gave very similar results and in coincidence with the actual amplitudes of the sinusoids of 1 and 10 Vrms, respectively, 0 and 20 dB. However, if we look closely by making a zoom as shown in Fig. 2.3, we see that the Matlab routines `spectrogram` and `pwelch` with the "power" switch (red and black lines) gave the correct power amplitude estimates while the other two methods for the discrete S matrix calculated through `spectrogram S` and the discrete FFT, gave power estimates short by approximately 1.35 dB at the sinusoid frequency peaks. On the other hand while making an estimate of the total energy in time and frequency (see output below) it becomes clear that the `spectrogram S` and `FFT` methods mantained the correct

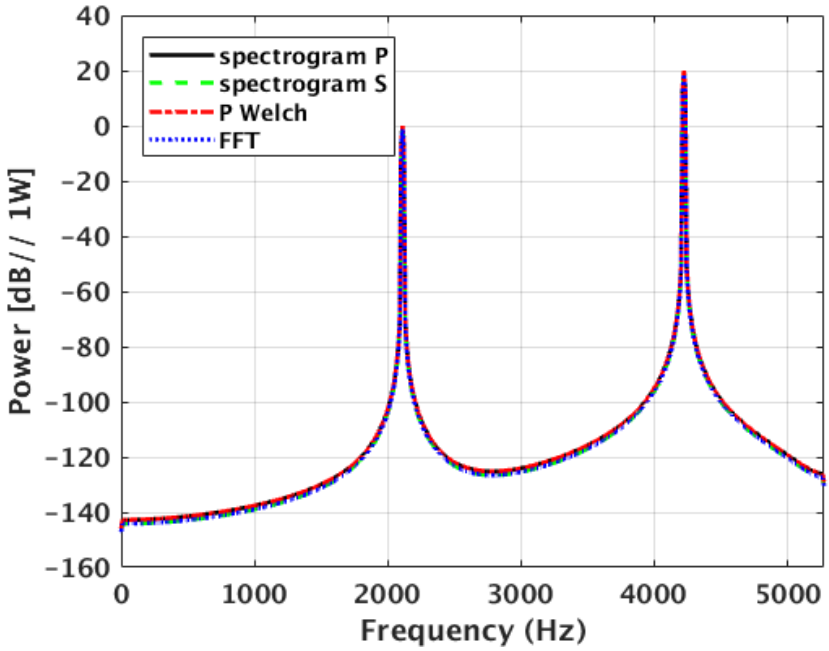

Figure 2.2: *spectrum power estimate, in dB // 1W, obtained in the two sinusoid deterministic signal case for:* $N = 10000$, $f_s = 10547$ *Hz,* `window=nfft=`*1024, Hann windowed and a segment overlap of 50%.*

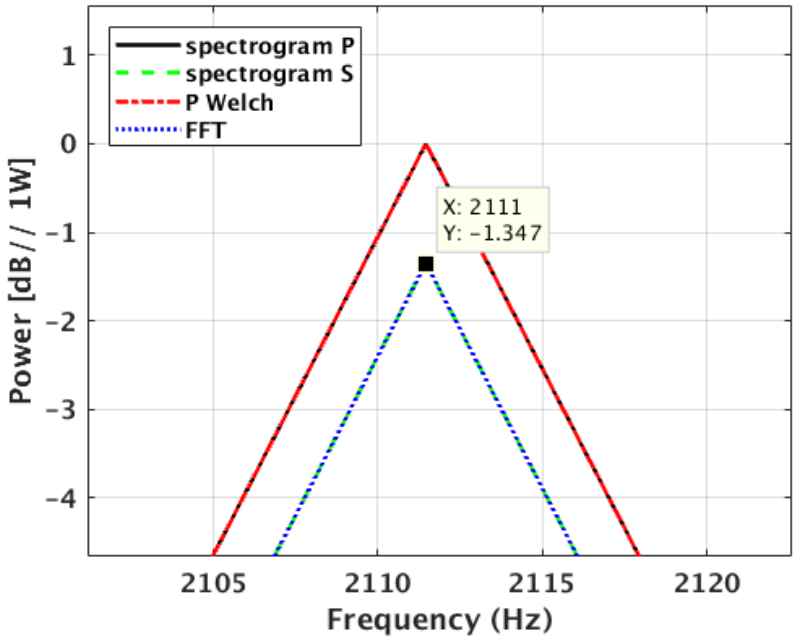

Figure 2.3: *zoom of the spectrum power estimate of plot 2.2 close to the peak frequency.*

value while the `P-spectrogram` and `P-welch` algorithms overrated the correct energy by approximately 1.35 dB. At this stage it is not clear where this difference comes from.

```
...from Power Spectrum
Energy in time domain : 50.146213 dB
Energy in freq domain by spectrogram power : 51.493671 dB
Energy in freq domain by STFT from spectrogram : 50.146213 dB
Energy in freq domain by Welch periodogram : 51.493671 dB
Energy in freq domain by discrete FFT : 50.146213 dB
```

As it can be see from the code in appendix B, the normalization of the frequency energy is obtained by multiplying the power sum over frequency bins by `nfft`, the block size, or the number of frequency bins, since from (2.21)

$$\begin{aligned}\sum_{n=0}^{n-1}|x(n)|^2 &= \frac{1}{\Delta T}\frac{1}{N\Delta T}\sum_{k=0}^{N-1}|X(k)|^2,\\ &= N\sum_{k=0}^{N-1}\frac{\Delta f}{N\Delta T}|X(k)|^2,\\ &= N\sum_{k=0}^{N-1}PS(k). \qquad (2.22)\end{aligned}$$

A further comparison was attempted with the software PAMGuide provided in attach of [5]. The two sinusoid data set was written on a WAV file (see code in appendix B) and then read within PAMGuide and processed to obtain the plot in Fig. 2.4. In order to

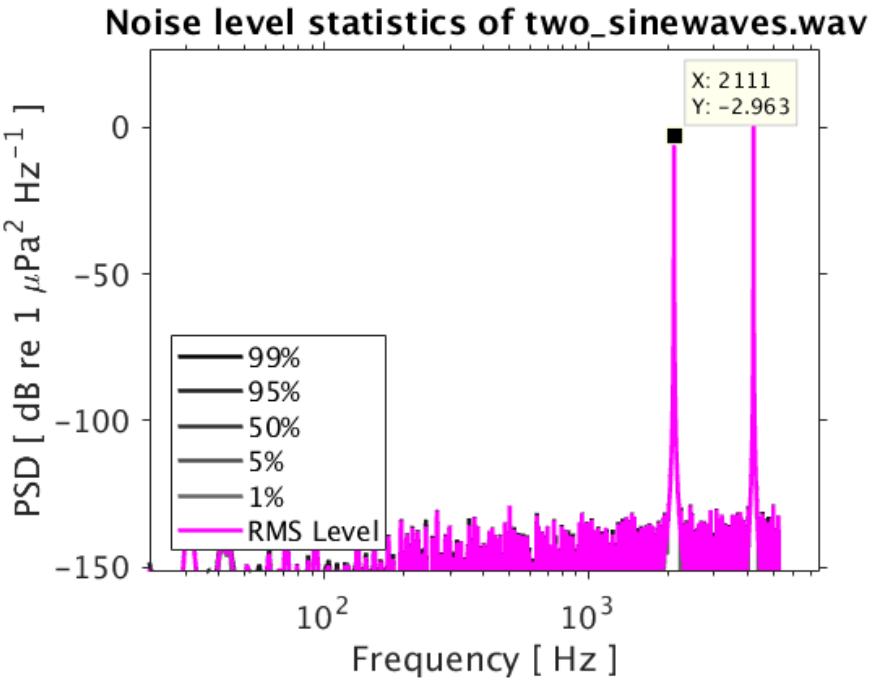

Figure 2.4: *zoom of the power spectrum estimate of the two sinusoid example using the PAMGuide software package.*

make the results compatible with those of the previous figures, a 0 dB // 1V/μPa, and a gain of -40 dB were used, since the signal was divided by 100 before saving into file in order to stay within the 1v limit required for the WAV format. The power level of the two sinusoids approximately corresponds to the expected but falls short by nearly -3 dB.

2.3.2 Example 2: real data record

The real data record used in this test was recorded in Ria Formosa in November 2018 during a field experiment of the course on Bioacoustics under the Master of Marine Biology (MBM). The record contains 120 seconds of data at fs = 52734 Hz. The following parameters were used `nfft`=`window`=4096 and `noverlap`=2048 (50%). The results are shown for PSD and PS estimates in figure 2.5(a) and (b), respectively. It can be noticed

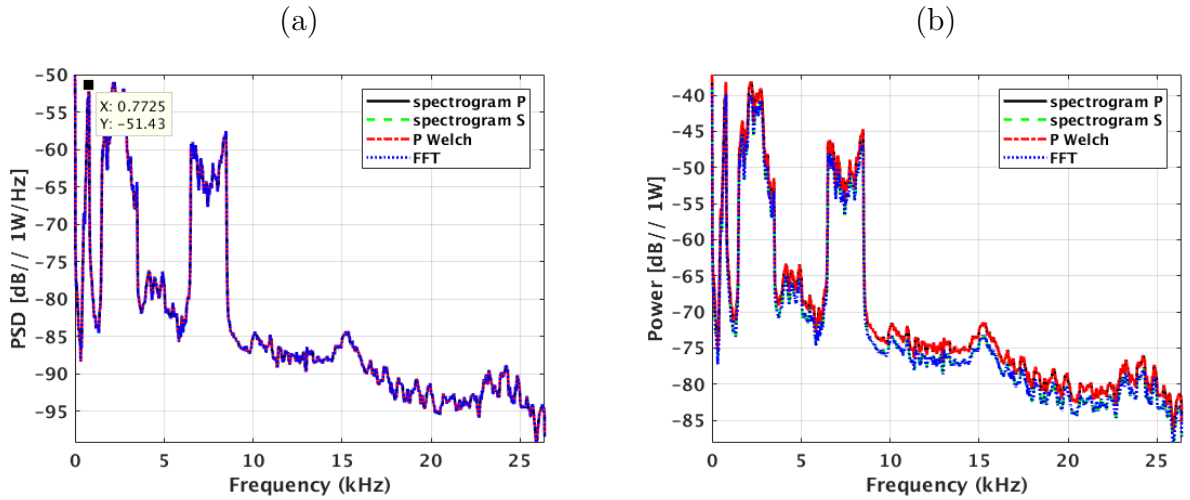

Figure 2.5: *real data record spectrum estimates for PSD (a) and PS (b) with* `nfft`=`window`*=4096,* `fs`*=52734, Hann windowed and 50% segment overlap.*

that also here the four estimators perfectly coincide for the PSD but there is a slight amplitude disagreement for the PS estimates (plot b). The zoom of figure 2.6 shows that the difference equates to approximately -1.7 dB for the `spectrogram S` and `FFT` algorithms when compared to the `spectrogram P` and `P Welch`.

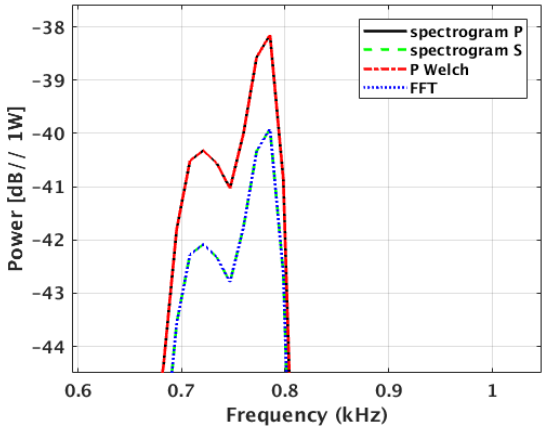

Figure 2.6: *zoom of the spectrum power estimate of plot 2.5(b) close to the peak frequency.*

Chapter 3

Ocean acoustic measurements: pressure and particle motion

This section deals with the application of the definitions and tested implementation routines of the previous chapter to the estimation of ocean acoustic noise and measurements. Very often, spectrograms of recorded acoustic data are used to analyze time and frequency variability, as well as relative levels. When absolute values are necessary, then the exact definition of Power Spectrum (PS) or Power Spectral Density (PSD) is required in order to be able to compare levels between different times and locations, or obtained by different users / instruments. This problem of portability of definitions becomes even more stringent when applied to particle motion, much less used than acoustic pressure, and therefore where standard methods and procedures are not yet well established.

3.1 Pressure and particle motion

It is well known from ocean physics that an acoustic wave (or any other mechanical wave) propagates through the motion of tiny mass particles. Two types of motion may be observed: one that is formed by mass particle "in place" oscillation transmitting this oscillation to the nearby particle, then to the next, etc, and the actual back and forth movement of particles through a short distance. That short distance is approximately one wavelength, but may vary close to the sound source or close to hard boundaries, that in the ocean are rocks, the ocean bottom or the sea surface.

So, there are two effects resulting from acoustic wave propagation: one is due to "in place" particle oscillation and the other is the actual "back-and-forth" particle motion. The former gives rise to the acoustic pressure that can be measured at long distance from the source, and the latter is called particle motion and can only be observed within short distances from the sound wave origin. Beyond that short distance the two effects merge into acoustic pressure only. For many years particle motion was disregarded mainly for two reasons: one was that sensitivity range was too short at practical frequencies and the other was that there were no easy to use sensors for measuring particle motion.

In order to introduce the topic let us start by giving some useful definitions.

3.1.1 Wave equation for sound pressure

The wave equation in linear regime starts from three equations [6]. The equation for conservation of mass

$$\frac{\partial \rho'}{\partial t} = -\nabla \cdot (\rho_0 \mathbf{v}), \tag{3.1}$$

Euler's equation

$$\frac{\partial \mathbf{v}}{\partial t} = -\frac{1}{\rho_0} p'(\rho), \tag{3.2}$$

and the equation of state

$$\frac{\partial p'}{\partial t} = c^2 \left(\frac{\partial \rho'}{\partial t} + \mathbf{v} \cdot \nabla \rho_0 \right), \tag{3.3}$$

where c is the speed of sound in an ideal fluid, ρ is the media density, $\mathbf{v}$ is the particle velocity vector which module is assumed to be much smaller than c, and where p is the pressure. The time independent quantities are identified with a subscript $_0$, while the small perturbations assumption was used for both pressure $p = p_0 + p'$ and for density $\rho = \rho_0 + \rho'$.

It can be shown that the manipulation of (3.1)-(3.3) leads to the typical wave equation

$$\nabla^2 \nu - \frac{1}{c^2} \frac{\partial^2 \nu}{\partial t^2} = 0, \tag{3.4}$$

where ν is a generic variable that may be replaced by the pressure perturbations p' or density perturbations ρ'.

3.1.2 Wave equation for particle velocity

Alternatively, taking the divergence of (3.1) and the time derivative of (3.2) and combine the two using (3.3) leads to the wave equation for the particle velocity $\mathbf{v}$

$$\frac{1}{\rho} \nabla (\rho c^2 \nabla \cdot \mathbf{v}) - \frac{\partial^2 \mathbf{v}}{\partial t^2} = \mathbf{0}, \tag{3.5}$$

which is a vectorial equation with three components. It is common to define the velocity potential field ϕ as

$$\mathbf{v} = \nabla \phi, \tag{3.6}$$

so that another wave-equation similar to (3.4) on ϕ may be defined.

3.1.3 Relation between pressure and particle velocity

The acoustic field supports plane-wave solutions for the wave equation and therefore, for frequencies above the cut-off frequency

$$f_{\text{cut-off}} = \frac{c(\pi - \rho_{\text{sed}}/\rho_{\text{water}})}{2\pi H \sin(\arccos(c/c_{\text{sed}}))}, \tag{3.7}$$

where ρ_{sed} and ρ_{water} are the densities in the sediment and in the water, respectively, c and c_{sed} are the sound speed in the water and in the sediment, respectively and H is the water

depth, one may assume that the velocity potential along, say, the x-axis, $\phi = f(x - ct)$, is some travelling wave, function of x and t. According to the Euler's equation (3.2), and using definition (3.6), we have that

$$\frac{\partial p}{\partial x} = -\rho \frac{\partial v_x}{\partial t} = -\rho \frac{\partial^2 \phi}{\partial x \partial t} \tag{3.8}$$

so that integrating for p along x, gives

$$p = -\rho \frac{\partial \phi}{\partial t} = \rho c f'(x - ct) \tag{3.9}$$

and therefore using the fact that according to (3.6), $v_x = f'(x - ct)$ we can write an essential relation between sound pressure and particle velocity for plane waves

$$p = \rho c v \tag{3.10}$$

where velocity v is a scalar, $v = |\mathbf{v}|$. At short range r from the sound source the following approximate relation is used

$$p = \rho c v \left(1 + \frac{\lambda}{2\pi r}\right)^{-1/2} \tag{3.11}$$

where $\lambda = c/f$ is the wavelength. The acoustic pressure obtained by either (3.10) or (3.11) are sometime called the *pressure equivalent particle velocity* and noted p_v in order to avoid confusion with the plain pressure field.

Another way to see this equivalence between pressure and particle velocity is to remark that (3.8) may be written for a narrow band signal, in the frequency domain as

$$\frac{\partial P(\omega)}{\partial x} = -\rho j \omega V_x(\omega) \tag{3.12}$$

which simply states that the particle velocity at a given frequency ω and along a given axis is proportional to the pressure gradient (pressure difference) along the same axis. This is why one practical technique for measuring particle velocity is by doing the difference of the output of closely spaced pressure sensors.

Including the particle velocity equivalent $V_{px} = \rho c V_x$, in (3.12) allows to write the relation between plain pressure and pressure equivalent particle velocity as

$$V_{px}(\omega) = \frac{1}{jk} \frac{\partial P_x(\omega)}{\partial x} \tag{3.13}$$

where $k = \omega/c$ is the wavenumber. It is easy to note that the unit of pressure equivalent particle velocity is conveniently [Kg][m^{-1}][s^{-2}] which is also equivalent to [N][m^{-2}] or Pascal [Pa].

3.1.4 Particle acceleration, velocity and displacement

Since particle displacement, velocity and acceleration are related by successive time derivatives, we may write in the frequency domain, and for each spatial component $i = (x, y, z)$, that

$$A_i(\omega) = j\omega V_i(\omega) \qquad V_i(\omega) = j\omega \Xi_i(\omega) \tag{3.14}$$

where A_i, V_i and Ξ_i are the i-th axis acceleration, velocity and displacement, as frequency domain representations of a_i, v_i and ξ_i, respectively. These three quantities are measured in ms^{-2}, ms^{-1} and simply m, respectively.

3.2 Definitions for measurements

3.2.1 Sound pressure

The commonly accepted definition of Sound Pressure Level (SPL), usually in dB, is that given by Urick [7]

$$\mathrm{SPL_{dB}} = 10\log_{10}\frac{I}{I_{\mathrm{ref}}}, \tag{3.15}$$

where I is not the true "intensity" but the sound field intensity of a plane wave defined by

$$I = \frac{p_{\mathrm{rms}}^2}{\rho c}, \tag{3.16}$$

where p_{rms} is the root mean square pressure field of the plane wave taken over a time interval T, *i.e.*,

$$p_{\mathrm{rms}} = \sqrt{\frac{1}{T}\int_0^T p^2(t)dt}, \tag{3.17}$$

where $p(t)$ is the instantaneous acoustic pressure at time t. Most often T is taken equal to 1s. In (3.15) I_{ref} is the reference sound field intensity for calculating the SPL in dB. According to (3.16), I_{ref} should depend on a rms reference sound pressure p_{ref} and on the media impedance ρc that would cancel out of (3.15) if constant in time and space. Otherwise, the intensity ratio in (3.15) depends on the impedence ratio Z_{ref}/Z, where Z is the actual media impedance calculated at the time and location where the measurement is taking place. This requires the definition of Z_{ref}. Strangely enough, as p_{ref} has been clearly defined to be 1μPa in the water both by ANSI and IEC since 1969, the reference value for the impedance is not mentioned in any standard posterior to 1960 [8]. So, while before 1960 the standard for impedance in the US was set to $Z_{\mathrm{ref}} = 1.53507\ 10^6$ Pa s/m due to a reference sound speed of $c = 1500$ m/s and a water density ρ=1.02338 g/cm^3, after that date Z_{ref} was left undefined. Despite this lack of definition, the community continued to use the reference pressure of 1μPa and has dismissed the conversion to intensity, implicitely admitting that impedance was cancelling out in (3.15). Nowadays it is common usage to refer simply to SPL in units of dB // 1μPa which, justifies for the name of "sound pressure level" and not "sound intensity level" as it should have been: it is indeed a ratio of squared pressures. Assuming a space-time invariant impedance we can write (3.15) as

$$\mathrm{SPL_{dB}} = 20\log_{10}\frac{p}{p_{\mathrm{ref}}}, \tag{3.18}$$

in [dB // 1μPa]. In the frequency domain definitions (2.14) - (2.16) for PSD and (2.17) - (2.18) for PS, apply with the signal equal to the pressure field, *i.e.*, $x(n) = p(n)$. The units will become [dB // μPa^2/Hz] for the PSD and [dB // μPa^2] for the power spectrum. Some authors define SPL in spectral amplitude, in which case the units become [dB // μPa/$\sqrt{\mathrm{Hz}}$] for the PSD amplitude and [dB // μPa] for the PS amplitude.

3.2.2 Received level

Very often, when attempting to produce SPL maps using modeled data, one is bound to determine the received level (RL) as a combination of contributions of the various sound sources on a given field at a given time. Assuming in a first approximation - often made

- that the field is composed of I discrete sources and that the media behaves as a linear system, *i.e.*, that the system output is the sum of the contributions of each input taken individually, one may write

$$\mathrm{RL}_m = 10\log_{10}\sum_{i=1}^{I} 10^{(\mathrm{SL}_i - \mathrm{TL}_{im})/10}, \tag{3.19}$$

where RL_m is the received level at location m, SL_i is the level of discrete source i and TL_{im} is the transmission loss between location of source i to location of receiver m. All quantities are expressed in dB. Note, that the summation in (3.19) is performed over individual source received power in a linear scale, not in dB. This is due to the assumed linear system property referred to above, and is essential for the calculation of the estimated received level RL.

3.2.3 Particle motion

The focus of interest on particle motion is the now widely spread understanding that many marine species are sensitive to the particle motion field. The most common quantities under use have been *pressure equivalent particle velocity* and *particle acceleration*. The pressure equivalent notation is normally used when particle velocity is to be combined or compared with acoustic pressure, while particle acceleration is meaningful when determining acoustic noise effects on aquatic life.

Since, most common particle motion sensors nowadays use accelerometers, the measured field is $a_i(t)$, particle acceleration in units [m/s^2], where $i = (x, y, z)$-axis. The next step is to equate the sample FT $A_i(f)$, as

$$A_i(f) = \sum_{n=0}^{N-1} a_i(n) e^{-j2\pi n f \Delta T}, \tag{3.20}$$

to get the PSD or power spectrum estimates using similar expressions to those in section 2 using, for example, for the sample spectrum

$$\tilde{P}_{Ai}(f) = \frac{\Delta T}{N}|A_i(f)|^2, \tag{3.21}$$

and the subsequent relations for the Welch periodogram PSD and PS estimates, conveniently expressed in units of [dB // (m/s^2)^2/Hz] or [dB // (m/s^2)^2], respectively.

Alternatively, using (3.13) and (3.14), one may get the particle velocity pressure equivalent along axis i,

$$V_{pi}(f) = \frac{\rho c}{j2\pi f} A_i(f). \tag{3.22}$$

In this case and since V_{pi} is pressure equivalent, unit [Pa] applies, and therefore equivalent PSD and power spectrum definitions as for the pressure field are used.

Chapter 4

Conclusions

It is common for researchers to present results of power spectrum estimates at conferences, workshops and other events. Often, spectral power levels are compared among various experiments taken with a variety of instruments, many of which have embedded software for automatically determining those levels. Power spectrum estimation is a well established discipline that has lead to implementations in commonly used software packages such as Matlab®. However, there are underlying assumptions and associated definitions that are not always taken into account and sometimes misused.

It is therefore important to draw a reference procedure by comparing the fundamental definitions with common used software so that every researcher will be able to tune their own processing in order to be able to compare results up to, say, the level of 1 dB.

So, the comparison of the results obtained with reference methods and ready-to-use routines in a simple example allowed to draw the following conclusions:

1. the PSD calculated by Matlab's `spectrogram` and `pwelch` routines exactly coincided;
2. the PSD obtained by the STFT matrix `S` exactly coincided with that of the plain FFT and are consistent with definitions; in order to make those two definitions fit among them, an appropriate and sensible scaling using the Welch periodogram definitions in [3], namely by dividing by the time window energy, the number of samples in the data segment and the sampling frequency, allowed to obtain the same results as those of Matlab;
3. the rms amplitudes of the sinusoids could be obtained through Matlab `spectrogram` and `pwelch` by using the `'power'` switch, but could not be replicated exactly by integrating the discrete PSD implementations using `S` from spectrogram and direct FFT, over the frequency bin interval;
4. conservation of energy between time and frequency domains could only be achieved by the discrete `S`-spectrogram and FFT implementations;
5. definitions of particle motion are given for measuring and displaying PSD and power spectrum of pressure equivalent particle velocity and particle acceleration quantities, as an attempt to provide a consistent set for representing noise fields.

This work is a contribution to the implementation of standard definitions for comparative purposes of sound pressure level, power spectral densities and power spectra, within task

4.3 of project JONAS (EAPA 52/2018) and for the analysis of the particle motion data set acquired during experiment BIOCOM'19 (see report [1], under project BIOCOM funded by program "Science without Borders" (Ciência sem Fronteiras) 2015 - 2019, Brasil.

Bibliography

[1] S.M. Jesus, L. Maia, F. Xavier, R. Vio, and E. Vale. Biocom'19 experiment data report: particle motion measurements. Report 02/19 - SiPLAB, CINTAL, University of Algarve, 8005-139 Faro (Portugal), August 2019.

[2] S.M. Kay and S.L. Marple Jr. Spectrum analysis - a modern perspective. *Proceedings IEEE*, 69(1):1380–1419, 1981.

[3] S.L. Marple. *Digital Spectral Analysis with Applications*. Prentice-Hall, New Jersey, USA, 1987.

[4] H. Krim and M. Viberg. Two decades of array signal processing research. *IEEE Signal Processing Magazine*, 96:67–94, July 1996.

[5] N.D. Merchant, K.M Fristrup, M.P. Johnson, P.L. Tyack, M.J. Witt, P. Blondel, and S.E. Parks. Measuring acoustic habitats. *Methods in Ecology and Evolution*, 6:257–265, 2015.

[6] F. Jensen, W. Kuperman, M. Porter, and H. Schmidt. *Computational Ocean Acoustics*. AIP Series in Modern Acoustics and Signal Processing, New York, 1994.

[7] R.J. Urick. *Principles of Underwater Sound*. McGraw-Hill, New York, 1983.

[8] M.A. Ainslie. A century of sonar: planetary oceanography, underwater noise monitoring, and the terminology of underwater sound. *Acoustics Today*, 11(1):12–19, Winter 2015.

[9] M. Frigo and S.G. Johnson. Fftw: An adaptive software architecture for the fft. In *Proc. of Int. Conf. on Acoustics, Speech and Signal Proc.*, volume 3, pages 1381–1384, 1998.

Appendix A

DFT scaling issues

Let us recall that the usual definition DFT pair of a discrete time signal $x(n)$ of N samples, is

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}, \tag{A.1}$$

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} X(k) e^{j2\pi nk/N}, \tag{A.2}$$

The Matlab manual refers that the definition of [9] is adopted, that states in `www.fftw.org` that the DFT pair is given by

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}, \tag{A.3}$$

$$x(n) = \sum_{n=0}^{N-1} X(k) e^{j2\pi nk/N}, \tag{A.4}$$

the only difference being an amplitude factor of $1/N$. Instead in the reference text of Marple [3] (pp.37) it is stated that the following definition is adopted

$$X(k) = \Delta T \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}, \tag{A.5}$$

$$x(n) = \frac{1}{N\Delta T} \sum_{n=0}^{N-1} X(k) e^{j2\pi nk/N}, \tag{A.6}$$

where now the sampling interval ΔT multiplies the direct transform and divides in the inverse transform. The question is why those differences and what is their impact on the final PSD estimate ?

Estimating PSD or PS through the periodogram mainly deals with the direct DFT so we will concentrate on that one. We need to show whether (A.5) provides a more consistent and better calibrated energy (or power) expression than that provided by (A.1) (or (A.3)).

The first remark is that the sampling interval factor ΔT of (A.5) and (A.6) has no influence in the forward-backward transform pair.

The second remark holds on the justification given in [3] (eq. 2.57, pp.42), that the factor ΔT gives a better discrete approximation of the FT integral, that is

$$\sum_{n=0}^{N-1} x(n)e^{-j2\pi nf\Delta T}\Delta T \approx \int_0^{N\Delta T} x(t)e^{-j2\pi ft}dt, \tag{A.7}$$

simply because the discrete version of the incremental factor $x(t)dt$ is given by $x(n)\Delta T$.

This change has, of course, impact on the expression of the energy theorem in the discrete case. So, we now prove (2.9) with the help of the elaborated explanation given in [3] pp. 40-43.

Let us calculate the signal power in the frequency domain using (A.5),

$$\begin{aligned}
\sum_{k=0}^{N-1} |X(k)|^2 &= \sum_{k=0}^{N-1} \Delta T^2 \left| \sum_{n=0}^{N-1} x(n)e^{-j2\pi nk/N} \right|^2, \\
&= \sum_{k=0}^{N-1} \Delta T^2 \left(\sum_{n=0}^{N-1} |x(n)|^2 + \sum_{n=0}^{N-1} \sum_{\substack{m=0 \\ m\neq n}}^{N-1} x(n)x^*(m)e^{-j2\pi k(n-m)/N} \right), \\
&= N\Delta T^2 \sum_{n=0}^{N-1} |x(n)|^2 + \Delta T^2 \sum_{n=0}^{N-1} \sum_{\substack{m=0 \\ m\neq n}}^{N-1} x(n)x^*(m) \sum_{k=0}^{N-1} e^{-j2\pi k(n-m)/N}
\end{aligned} \tag{A.8}$$

The latter summation on k in the second term of (A.8) appears as the DFT, of a signal equal to 1 in the interval $[0, N-1]$. Therefore it results on a Dirac $\delta(n-m)$. Since this Dirac will only be different from zero at $n = m$, the double summation in the second term of (A.8) will always be zero. Therefore,

$$\sum_{k=0}^{N-1} |X(k)|^2 = N\Delta T^2 \sum_{n=0}^{N-1} |x(n)|^2, \tag{A.9}$$

or in a more stricking equivalent form

$$\frac{1}{\Delta T} \sum_{k=0}^{N-1} |X(k)|^2 = N\Delta T \sum_{n=0}^{N-1} |x(n)|^2, \tag{A.10}$$

that simply states the energy theorem for a time discrete finite signal, where in each domain the power summation should be multiplied by the extent of time and frequency intervals, $N\Delta T$ and $1/\Delta T = f_s$, respectively. If the ΔT scaling factor was not used in (A.5), the right scaling could only be obtained for $\Delta T = 1$, which somehow is assumed in the current DFT definitions.

Appendix B

PSD and power spectrum simulation code

```
% test_spectrogram.m: test spectrogram and pwelch for PSD and Power
% 19jul2019
%
% last update:  08ago19
%               21ago19 - including Cristiano's remarks
%               30ago19 - with energy calculation
%               02feb20 - test with N > Nfft
%               04feb20 - change to Hann window (default in Welch)
%=================================================================

clear
close all

% window    = actual # time samples taken from the signal
% nfft      = FFT length, if nfft > window zeros are appended to signal
% overlap   = segment overlap in %
% noverlap  = # of samples of overlap
% fs        = sampling frequency [Hz]
% dt        = sampling interval in time [s]
% df        = sampling interval in frequency [Hz]
window  = 1024;
nfft    = 2^nextpow2(window);
overlap = 50;       % percentage of segment overlap
noverlap= floor(overlap * window /100);
fs      = 10547;
dt      = 1/fs;
df      = fs/nfft;

% test signal: sum of two sinusoids at f0 with amplitude 1Vrms and 2*f0
% with amplitude 10 Vrms; f0 is approx fs/5, where approx means the nearest
% integer frequency bin in order to avoid "scalop loss"
N = 100000;
n = 0:N-1;
% make sinusoide frequency coincide exactly with a discrete frequency bin
f0 = round((fs/5)/df)*df;
```

```
w0 = 2*pi*f0;
x = sqrt(2)*sin(w0*n*dt) + sqrt(2)*10*sin(2*w0*n*dt);

% pick hann window and calculate energy U
% note that energy should equate to 1 for rectangular window
w = hann(window);
U = (1/window)*sum(w.^2);

%
% PSD estimates
%
%=============================================
figure(1)
% mean PSD calculated by spectrogram
[S,F,T,P] = spectrogram(x,hann(window),noverlap,nfft,fs);
Pp = mean(P,2);
plot(F/1000,10*log10(Pp),'k-','Linewidth',2)
hold on

% mean PSD calculated with S given by spectrogram
Ps = (mean(abs(S).^2,2))/(U*window*fs);
Ps(2:end-1) = 2*Ps(2:end-1);
plot(F/1000,10*log10(Ps),'g--','Linewidth',2)

% Welch periodogram PSD
[Pw,Fw,Tw] = pwelch(x,hann(window),noverlap,nfft,fs);
plot(Fw/1000,10*log10(Pw),'r-.','Linewidth',2);

% emulate Pw, Pp, Ps with FFT estimated periodogram
Nseg = floor((N-window)/(noverlap)) + 1;      % number of segments
k2 = 0;
Pxmean = zeros(1,nfft/2+1);
y = zeros(1,nfft);
for pseg = 1:Nseg,
    k1 = k2 - (window-noverlap);
    if k1 < 0, k1 = 0; end
    k2 = k1 + window;
    y(1:window) = x(k1+1:k2).*w';
    X = fft(y,nfft);
    Px = (abs(X(1:nfft/2+1)).^2)/(U*window*fs);
    Px(2:end-1) = 2*Px(2:end-1);
    Pxmean = ((pseg-1)/pseg)*Pxmean + Px/pseg;
end
Fx = linspace(0,fs/2,nfft/2+1);
plot(Fx/1000,10*log10(Pxmean),'b:','Linewidth',2)
axis tight
grid on
legend('spectrogram P','spectrogram S','P Welch','FFT','Location','NorthWest')
xlabel('Frequency (kHz)','Fontsize',15);
ylabel('PSD [dB// 1W/Hz]','Fontsize',15)
set(gca,'Fontweight','bold','Fontsize',12);

% calculate time and frequency energy
```

```
Et = 10*log10(sum(abs(x(1:window)).^2));     % time
Ep = 10*log10(sum(fs*Pp));                  % PSD calculated by spectrogram
Es = 10*log10(sum(fs*Ps));                  % PSD calculated with S from spectrogram
Ew = 10*log10(sum(fs*Pw));                  % PSD calculated by pwelch
Ex = 10*log10(sum(fs*Pxmean));              % PSD calculated from discrete fft
disp ('...from PSD ');
fprintf('Energy in time domain : %f dB\n', Et);
fprintf('Energy in freq domain by spectrogram power : %f dB\n', Ep);
fprintf('Energy in freq domain by STFT from spectrogram : %f dB\n', Es);
fprintf('Energy in freq domain by Welch periodogram : %f dB\n', Ew);
fprintf('Energy in freq domain by discrete FFT : %f dB\n', Ex);

%
% Power spectrum
%
%=============================================
clear Pp Ps Pw Fw Tw Px S F T P Pxmean
figure(2)
% power per bin calculated by spectrogram
% power key allows for calculating the noise/signal present on each
% frequency bin
[S,F,T,P] = spectrogram(x,hann(window),noverlap,nfft,fs,'power');
Pp = mean(P,2);
plot(F/1000,10*log10(Pp),'k-','Linewidth',2)
hold on

% power per bin calculated with S by spectrogram
Ps = (mean(abs(S).^2,2))/(U*window*nfft);
Ps(2:end-1) = 2*Ps(2:end-1);
plot(F/1000,10*log10(Ps),'g--','Linewidth',2)

% Welch periodogram PSD
[Pw,Fw,Tw] = pwelch(x,hann(window),noverlap,nfft,fs,'power');
plot(Fw/1000,10*log10(Pw),'r-.','Linewidth',2);

% emulate Pw, Pp, Ps with FFT estimated periodogram
Nseg = floor((N-window)/(noverlap)) + 1;      % number of segments
k2 = 0;
Pxmean = zeros(1,nfft/2+1);
y = zeros(1,nfft);
for pseg = 1:Nseg,
    k1 = k2 - (window-noverlap);
    if k1 < 0, k1 = 0; end
    k2 = k1 + window;
    y(1:window) = x(k1+1:k2).*w';
    X = fft(y,nfft);
    Px = (abs(X(1:nfft/2+1)).^2)/(U*window*nfft);
    Px(2:end-1) = 2*Px(2:end-1);
    Pxmean = ((pseg-1)/pseg)*Pxmean + Px/pseg;
end
Fx = linspace(0,fs/2,nfft/2+1);
plot(Fx/1000,10*log10(Pxmean),'b:','Linewidth',2)
axis tight
```

```
grid on
legend('spectrogram P','spectrogram S','P Welch','FFT','Location','NorthWest')
xlabel('Frequency (kHz)','Fontsize',15);
ylabel('Power [dB// 1W]','Fontsize',15)
set(gca,'Fontweight','bold','Fontsize',12);

% calculate time and frequency energy
disp ('...from Power Spectrum ');
Et = 10*log10(sum(abs(x(1:window)).^2));       % time
Ep = 10*log10(window*sum(Pp));                 % PSD calculated by spectrogram
Es = 10*log10(window*sum(Ps));                 % PSD calculated with S from spectrogra
Ew = 10*log10(window*sum(Pw));                 % PSD calculated by pwelch
Ex = 10*log10(window*sum(Pxmean));             % PSD calculated from discrete fft
fprintf('Energy in time domain : %f dB\n', Et);
fprintf('Energy in freq domain by spectrogram power : %f dB\n', Ep);
fprintf('Energy in freq domain by STFT from spectrogram : %f dB\n', Es);
fprintf('Energy in freq domain by Welch periodogram : %f dB\n', Ew);
fprintf('Energy in freq domain by discrete FFT : %f dB\n', Ex);

% save 10 seconds of data into wav file
T = 10;
Nsamp = T * fs;
nt = 0:Nsamp-1;
X = sqrt(2)*sin(w0*nt*dt) + sqrt(2)*10*sin(2*w0*nt*dt);
% apply a gain so, as if the signal was recorded with a gain = 10 (20dB)
X = X/10;
% apply a 2.5 V peak max excursion at ADC input and 24 bit coding
Xint = int32(X * 2^23/2.5);
audiowrite('two_sinewaves.wav',Xint,fs,'BitsPerSample',24);
clear X nt Nsamp T

%==============================
%      real data example
%==============================
% reading in WAV file
clear
[Y,fs] = audioread('DATA_TP1_0218_333110007.WAV');
x = double(Y(:,1));
clear Y

% set parameters
window      = 4096;
nfft        = 2^nextpow2(window);
overlap     = 50;       % percentage of segment overlap
noverlap    = floor(overlap * window /100);
dt          = 1/fs;
df          = fs/nfft;
N           = floor(length(x));

% pick hann window and calculate energy U
% note that energy should equate to 1 for rectangular window
w = hann(window);
U = (1/window)*sum(w.^2);
```

```
%
% PSD estimates
%
%================================================
figure(3)
% mean PSD calculated by spectrogram
[S,F,T,P] = spectrogram(x(1:N),hann(window),noverlap,nfft,fs);
Pp = mean(P,2);
plot(F/1000,10*log10(Pp),'k-','Linewidth',2)
disp(' PSD P spectrogram - done');
hold on

% mean PSD calculated with S given by spectrogram
Ps = (mean(abs(S).^2,2))/(U*window*fs);
Ps(2:end-1) = 2*Ps(2:end-1);
plot(F/1000,10*log10(Ps),'g--','Linewidth',2)
disp(' PSD S spectrogram - done');

% Welch periodogram PSD
[Pw,Fw,Tw] = pwelch(x(1:N),hann(window),noverlap,nfft,fs);
plot(Fw/1000,10*log10(Pw),'r-.','Linewidth',2);
disp(' PSD Welch - done');

% emulate Pw, Pp, Ps with FFT estimated periodogram
% missing the case where nfft > window
Nseg = floor((N-window)/(noverlap)) + 1;      % number of segments
k2 = 0;
Pxmean = zeros(1,nfft/2+1);
for pseg = 1:Nseg,
    k1 = k2 - (window-noverlap);
    if k1 < 0, k1 = 0; end
    k2 = k1 + window;
    X = fft(x(k1+1:k2).*w,nfft);
    Px = (abs(X(1:nfft/2+1)).^2)/(U*window*fs);
    Px(2:end-1) = 2*Px(2:end-1);
    Pxmean = ((pseg-1)/pseg)*Pxmean + Px/pseg;
end
Fx = linspace(0,fs/2,nfft/2+1);
plot(Fx/1000,10*log10(Pxmean),'b:','Linewidth',2)
disp(' PSD FFT - done');
axis tight
grid on
legend('spectrogram P','spectrogram S','P Welch','FFT','Location','NorthEast')
xlabel('Frequency (kHz)','Fontsize',15);
ylabel('PSD [dB// 1W/Hz]','Fontsize',15)
set(gca,'Fontweight','bold','Fontsize',12);

%
% Power spectrum
%
%================================================
clear Pp Ps Pw Fw Tw Px S F T P Pxmean
```

```
figure(4)
% power per bin calculated by spectrogram
% power key allows for calculating the noise/signal present on each
% frequency bin
[S,F,T,P] = spectrogram(x,hann(window),noverlap,nfft,fs,'power');
Pp = mean(P,2);
plot(F/1000,10*log10(Pp),'k-','Linewidth',2)
disp(' PS P spectrogram - done');
hold on

% power per bin calculated with S by spectrogram
Ps = (mean(abs(S).^2,2))/(U*window*nfft);
Ps(2:end-1) = 2*Ps(2:end-1);
plot(F/1000,10*log10(Ps),'g--','Linewidth',2)
disp(' PS S spectrogram - done');

% Welch periodogram PSD
[Pw,Fw,Tw] = pwelch(x,hann(window),noverlap,nfft,fs,'power');
plot(Fw/1000,10*log10(Pw),'r-.','Linewidth',2);
disp(' PS Welch - done');

% emulate Pw, Pp, Ps with FFT estimated periodogram
% missing the case where nfft > window
Nseg = floor((N-window)/(noverlap)) + 1;      % number of segments
k2 = 0;
Pxmean = zeros(1,nfft/2+1);
for pseg = 1:Nseg,
    k1 = k2 - (window-noverlap);
    if k1 < 0, k1 = 0; end
    k2 = k1 + window;
    X = fft(x(k1+1:k2).*w,nfft);
    Px = (abs(X(1:nfft/2+1)).^2)/(U*window*nfft);
    Px(2:end-1) = 2*Px(2:end-1);
    Pxmean = ((pseg-1)/pseg)*Pxmean + Px/pseg;
end
Fx = linspace(0,fs/2,nfft/2+1);
plot(Fx/1000,10*log10(Pxmean),'b:','Linewidth',2)
disp(' PS FFT - done');
axis tight
grid on
legend('spectrogram P','spectrogram S','P Welch','FFT','Location','NorthEast')
xlabel('Frequency (kHz)','Fontsize',15);
ylabel('Power [dB// 1W]','Fontsize',15)
set(gca,'Fontweight','bold','Fontsize',12);
```