amplitude spectrum
phase spectrum

Normalization of Spectral Analyses

Andrew J. Barbour

January 28, 2013

Abstract

Units matter, and having correct ones is crucial to interpretation of spectral analyses. Here we outline the normalization used by `rlpSpec`, namely the power spectral density, and compare it to other quantities commonly encountered in spectral analysis.

Contents

1 Background

There can often be confusion about the different quantities used in spectral analysis [1], partly due to myriad nomenclature within the incredibly vast literature on the subject. Regarding nomenclature: Phrases including "amplitude spectrum", "energy spectral density", "power", "power spectra", and even "spectra" all mean *something*, but are rarely equivalent.

Let us, for the sake of brevity, assume we are in the time domain, and we are considering a discrete stationary signal f of length N, having Fourier transform $\mathfrak{F}\{f\}$[2] which is complex, and represented by$F$. The **amplitude spectrum** of this transform pair is simply the amplitude of F, or mod$\{F\}$; we will denote this as $S^{(\mathrm{A})}$. The pair's corresponding **phase spectrum** is the phase angle of F, or arg$\{F\}$, denoted by $S^{(\phi)}$.

How do we interpret the quantities

Table with equivalent expressions

The latter assumes that the spectrum has the normalization used in power – which is that the Nyquist is assumed to be 1,0, always (power doesn't ask for a sample interval). But

[1] This post to `R-help` illustrates the confusion common in spectral analyses:
`http://r.789695.n4.nabble.com/Re-How-do-I-normalize-a-PSD-td792902.html`

[2] A proper Discrete Fourier Transform (DFT) must be normalized by the length of the series; however, most DFT programs (including `stats::fft`) eschew this normalization for efficiency's sake.

Table 1: Representors of various spectral quantities.

Expression	Representing	Equivalent expression
f	stationary timeseries	
F	Fourier transform of f	$\mathfrak{F}\{f\}$
$S^{(\mathrm{A})}$	**amplitude spectrum** of F	$\lvert F\rvert$ or $\mathrm{mod}\{F\}$
$S^{(\phi)}$	**phase spectrum** of F	$\arg\{F\}$

Table 2: A comparison of functions comparable to `rlpSpec`, excluding raw periodogram estimators.

Function	Namespace	Sine m.t.?	Adaptive?	Reference
`mtapspec`	`RSEIS`	Yes	No	Lees and Park (1995)
`spectrum`	`stats`	No	No	R Core Team (2012)
`spec.mtm`	`multitaper`	Yes	Yes	Rahim and Burr (2012)
`SDF`	`sapa`	Yes	No	Percival and Walden (1993)

psd assumes that the interval is 1 (unless you specify otherwise) so that the Nyquist is 0.5. So say you have a white noise with variance 1: power will return a flat spectrum with level 0 db (=1) but psd will return a level of 2 (=3 db), both so that the level times the Nyquist will be 1. But logsmoo, told that the interval is 1, will multiply the spectrum by 2 to convert from lc**2/Nyquist to lc**2/Hz: giving 3 dB for the spectrum from power, but 6 dB (=4) for the spectrum from psd.

A straightforward way to understand normalization in spectral analysis is to analyze a real, stationary series which is normally distributed with known variance, $x = \mathcal{N}(\mu, \sigma^2)$. A fundamental result found in many texts on spectral analysis is

$$\mathrm{var}\{x\} \equiv \sigma_x^2 = \int_{-1/2}^{1/2} S(f) df \tag{1}$$

which says if we integrate the power spectral density over all frequencies we can obtain the process variance. If we have a $\mathcal{N}(0, 1)$ process, and assume the sampling interval is once per second, we should expect a flat spectrum of 2 units^2/Nyquist across all frequencies $[0, 0.5]$ so that the area under the spectrum is equal to one.

```
> set.seed(1234)
> N <- 256
> x <- rnorm(N, mean = 0, sd = 1)
> xv <- var(x)
> X <- fft(x)
> class(X)

[1] "complex"

> length(X)
```

```
[1] 256
```

Let us first find the amplitude and phase response:

```
> Sa <- Mod(X) # Amplitude spectrum
> Sp <- Arg(X)  # Phase spectrum
```

followed by the energy spectral densities, noting the equivalence between the complex conjugate based estimates. between

```
> XC <- Conj(X)
> all.equal(Se <- Sa**2, Se_2 <- Mod(XC * X), Se_2R <- Mod(X * XC))
```

```
[1] TRUE
```

So now we are able to find single-sided power spectral density estimates
We test the integrated spectrum result in Equation X:

```
> dB <- function(y) 10*log10(y)
> # Approximations of process variance:
> xv_1 <- mSn * fNyq
> xv_2 <- sum(S)/Nf * fNyq
> all.equal(xv_1, xv_2)
```

```
[1] TRUE
```

```
> xv_1/xv
```

```
[1] 0.9933334
```

We test the integrated spectrum result in Equation X:

```
> # crude representation of the integrated spectrum, which
> # should equal the variance of the original series (xv)
> S_sum <- function(S_ave, fr, fl=0) S_ave*(fr - fl)
> S_sum(mSn, fNyq)/xv             #1
```

```
[1] 0.9933334
```

```
> S_sum(mSn, fNyq, -fNyq)/xv      #2
```

```
[1] 1.986667
```

```
> sum(Se)/N/N                     #1
```

```
[1] 1.020075
```

```
> ## Single sided PSD
> Nf <- N/2
> S <- 2*Se[1:Nf]/N
> fsamp <- 1  # sampling freq, Hz
> fNyq <- fsamp/2   # nyquist or shannon(?) freq
> nyfreq <- seq.int(from=0, to=fNyq, length.out=Nf) # frequencies
> plot(nyfreq, S, type="h", xlab="Nyquist frequency", ylab="units**2 / Nyq.-freq.", ma
> #lines(nyfreq, S, type="s")
> mSn <- mean(S)
> #abline(h=xv/fNyq, lwd=2) # expected PSD for N(0,1)
> polygon(c(0,fNyq,fNyq,0,0), c(0,0,mSn,mSn,0), lwd=2, lty=3, border="red")
```

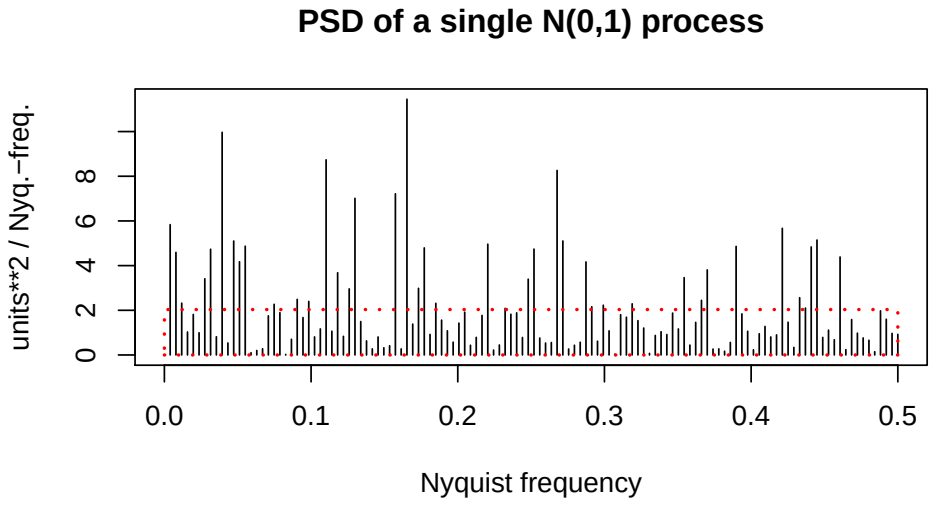

Figure 1:

```
> mSe <-round(mean(Se))
> S_sum(mSe, fNyq)/xv              #64

[1] 127.4321

> S_sum(mSe, fNyq)/Nf/xv           #1

[1] 0.9955633

> freq <- fsamp*nyfreq # Nyq-freq to Hz
> S0 <- dB(2)
```

1.1 stats::spectrum

Included in the core distribution of R is `stats::spectrum`, which accesses `stats::spec.ar` or `stats::spec.pgram` for either parametric and non-parametric estimation, respectively. The user can optionally apply a single cosine taper, and/or a smoothing kernel. Our method is non-parametric; hence, we will compare to the latter.

```
> spec.pgram(X, pad=1, taper=0.2, detrend=FALSE, demean=FALSE, plot=FALSE)
```

However, the logical arguments `detrend` and `demean` to `psdcore` are passed to `spec.pgram`; they are, by default, both TRUE.

As a matter of bookkeeping, we must deal with the working environment accessed by `rlpSpec` functions. Specifically, we should ensure `psdcore` does not access any inappropriate information by setting `refresh=TRUE`. We can then re-calculate the multitaper PSD and the raw periodogram with `plotpsd=TRUE`. The results are shown in Figure **??**.

1.2 multitaper::spec.mtm

1.3 SDF::sapa

References

Lees, J. M. and Park, J. (1995). Multiple-taper spectral analysis: A stand-alone C-subroutine. *Computers & Geosciences*, 21(2):199–236.

Percival, D. and Walden, A. (1993). *Spectral analysis for physical applications*. Cambridge University Press.

R Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Rahim, K. and Burr, W. (2012). *multitaper: Multitaper Spectral Analysis*. R package version 1.0-2.

Index