amplitude
spectrum
phase spectrum

Normalization of Spectral Analyses

Andrew J. Barbour

January 26, 2013

Abstract

Normalizations come in many forms, depending on the quantity in consideration. 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 stationary signal f having Fourier Transform $\mathfrak{F}\{f\}$ represented by F. The of this transform pair is simply the amplitude of complex F, or mod$\{F\}$; we will denote this as $^{(\mathtt{A})}S$. The corresponding is thus $\arg F$, denoted $^{(\phi)}S$.

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 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 covert 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 generate a stationary series, normally distributed with variance 1, and expectation 0.

```
> no <- ne <- 128
> x <- rnorm(no, mean = 0, sd = 1)
> X <- fft(x)
> class(X)

[1] "complex"
```

[1] This post to `R-help` very eloquently describes the problem, and provides some guidance: `http://r.789695.n4.nabble.com/Re-How-do-I-normalize-a-PSD-td792902.html`

```
> length(X)
```

```
[1] 128
```

Let us first find the amplitude and phase response:

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

And the energy spectral density function, noting the equivalence to Equation XX and. We find:

```
> Se <- Sa**2
> all.equal(Se_2 <- Mod(Conj(X) * X), Se_2r <- Mod(X * Conj(X)), Se)
```

```
[1] TRUE
```

```
> dB <- function(y) 10*log10(y)
> 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`

Index