Subject: Re: cross-spectral analysis
Posted by Craig Markwardt on Fri, 09 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

t.osborn@uea.ac.uk (T.Osborn) writes:

>
>
> I need to do some cross-spectral analysis (preferably in IDL).
> In particular, I need to compute the "squared coherence"
> between two time series. Is there a built-in routine to do
> this in IDL, or does someone have their own? I'm not very
> au fait with spectral analysis, so I hope it's not too
> difficult!

Yes, this is definitely do-able, I have a specialized routine to do it
for light curves in X-ray astronomy.

There are unfortunately many definitions for the squared coherence in
the engineering literature. Sometimes it can be difficult to
understand which one to use.

Generally speaking, if F1 and F2 are the complex fourier coefficients
of two time series at a given frequency, then the squared coherence is

```
  gamma^2 = |<F1* F2>|^2 / ( <|F1|^2> <|F2|^2> )
```

where |x| is the absolute value of x, <x> is the ensemble average of
x, and x* is the complex conjugate of X. It is very important to
realize that the coherence cannot be computed from a single
measurement. You need to measure the fourier coefficients many times,
and average over that entire sample. In X-ray astronomy, this often
means breaking the observation up into small segments and doing lots
of FFTs. The classic reference for this kind of thing is Bendat and
Piersol, "Random Data: Analyses and Measurement Procedures."

How to do it in IDL? Well, it's probably not appropriate to
regurgitate all of my code. Here is a snippet of the important bits.

```
for i = 0, nmeasurements - 1 do begin

  ;; You must construct the time series here
  ts1 = [Time series 1, measurement i]
  ts2 = [Time series 2, measurement i]
  ;; Apply your window function as needed.

  ;; Compute the FFT of each time series
  f1 = fft(ts1, +1)
  f2 = fft(ts2, +1)
  ;; f1 and f2 are complex vectors

  ;; Extract only the useful elements of FFT output
  nfbins = n_elements(ts1) / 2
  f1 = f1(1:nfbins)    ;; Ignores DC value, but keeps Nyquist value
  f2 = f2(1:nfbins)

  ;; Compute complex cross amplitude
  c12 = CONJ(f1) * f2  ;; Cross amplitude 1 vs. 2
  p1  = abs(f1)^2      ;; Power spectrum series 1
  p2  = abs(f2)^2      ;; Power spectrum series 2

  ;; Accumulate many samples
  if i EQ 0 then begin
    sum_c12 = c12
    sum_p1  = p1
    sum_p2  = p2
    nsamples= 1L
  endif
    sum_c12 = sum_c12 + c12
    sum_p1  = sum_p1  + p1
    sum_p2  = sum_p2  + p2
    nsamples= nsamples+ 1
  endelse
endfor

;; Compute squared coherence value
gam2  = abs(sum_c12)^2 / (sum_p1 * sum_p2)
;; Uncertainty in the mean of squared coherence value
egam2 = sqrt( (2D * gam2*(1D - gam2)^2) / nsamples )
```

The result, gam2, is the squared coherence value at each frequency, and egam2 is the standard error of the mean of gam2. Of course, there are tweaks. Several corrections can apply if your signal is noise-dominated. Computing the lag between two signals is also straightforward:

```
;; Lag at each freq., computed in radians
lag  = atan(imaginary(sum_c12), double(sum_c12))
;; Uncertainty in the lag
elag = sqrt( (1D - gam2)/ (2D * gam2) / nsamples )
```

The resulting value, lag, is positive when the first series leads the second.

Good luck!

Craig

```
--
 ------------------------------------------------------------ --------------
Craig B. Markwardt, Ph.D.         EMAIL: craigmnet@astrog.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 ------------------------------------------------------------ --------------
```