Bringing Least-Squares Spectral Analysis to Octave

Ben Lewis

McGill University

20 July 2012

Outline

1. Introduction
   - Spectral analysis
   - Mathias' implementation
2. Currently completed work
3. Further work

1 Introduction
- Spectral analysis
- Mathias' implementation

2 Currently completed work

3 Further work

Types of Least-Squares implementations

Lomb-Scargle

A periodogram implementation, this method will find the best fit for any given frequency, thus it is not an invertible method, however it is also highly capable of testing a wide range of possible values.

Types of Least-Squares implementations

Korenberg

Determines a sparse set of coefficients from an overdetermined set of values, also called “Fast Orthogonal Search”; uses a form of Cholesky decomposition.

Types of Least-Squares implementations

Chen & Donoho

Also determines a sparse set of coefficients, but minimises the Manhattan distance (L_1 norm, Taxicab norm . . .) and handles the problem as a linear programming problem (called "basis pursuit".)

Mathias' implementation

Lomb-Scargle periodogram

$$\tau = \left[\arctan\left(\frac{\sum_j \sin 2\omega t_j}{\sum_j \cos 2\omega t_j}\right)\right](2\omega)^{-1}$$

This provides a time-delay τ for the specific frequency ω being examined, which is then used to determine the result:

$$c = \frac{1}{2}\left(\frac{\left(\sum_j x_j \cos(\omega t_j - \tau)\right)^2}{\sum_j \cos^2(\omega t_j - \tau)} + \frac{\left(\sum_j x_j \sin(\omega t_j - \tau)\right)^2}{\sum_j \sin^2(\omega t_j - \tau)}\right)$$

Mathias' implementation

Limitations of implementation

Because this method considers each frequency independently of all others, the values produced cannot precisely represent the data set when summed, however they accurately represent how well each frequency compares to the data involved.
Overcome slightly by wavelet functions, which provide both frequency coefficient and location in the data set, however building a function is still not really feasible.

1 Introduction
- Spectral analysis
- Mathias' implementation

2 Currently completed work

3 Further work

cubicwgt, a cubic weighted function.

$$w(t) = \begin{cases} 1 - 3t^2 + 2t^3, & |t| \leq 1 \\ 0, & |t| > 1 \end{cases}$$

Compares favourably to a single cosine peak, and does not involve trigonometric functions; this is used in the unaccelerated wavelet transform and associated functions.

lscomplex, lsreal

`lscomplex` and `lsreal` are each implementations of the Lomb-Scargle periodogram; they each take the input of a time series of complex or real values (depending on the function) and will compute a non-invertible set of coefficients, for each frequency over a given number of octaves. There are also fast versions of these functions in the works, but they are currently not working.

lswaveletcoeff

Determines the coefficient of the nonuniform wavelet transform of the provided series at one frequency, one time, and for a specified windowing function and radius, which default to `cubicwgt` and 1 respectively.

lscorrcoeff

Tests the correlation coefficient of the wavelet transform of two time series at the same time and frequency; it's possible to specify a windowing function, which defaults to `cubicwgt`, and a window radius, which defaults to 1.

1 Introduction
- Spectral analysis
- Mathias' implementation

2 Currently completed work

3 Further work

lsrealwavelet

This function will implement the lsreal transformation over a given window, provided by the cubicwgt function combined with some scaling.

fastlscomplex, fastlsreal

These functions are divide-and-conquer based versions of the lscomplex and lsreal functions; the current version of fastlscomplex in the SVN is correct to 10^{-6} or thereabouts for the first octave of results, however in the process of merging the computation ranges (as part of the D-a-C scheme), enough error is introduced to bring later error up to 10^{-2} or higher. This is, clearly, unacceptable; I'm actively searching for a way to improve that result, and also working on getting fastlsreal functional to the same degree (at least.)

fastlsrealwavelet

A divide-and-conquer method similar to the regular methods, but implements the wavelet transform.
In the interest of fulfilling my GSoC mandate, however, I am more interested in providing a “slow” variant of the lsrealwavelet function before getting the fast code working to the same degree, simply so that there is a working function available.

An invertible method

I am happy with this code, but if I have the ability to do so within the constraints of GSoC, I intend to code a basis-chasing or other invertible Least Squares method, and (if possible) a fast variant.