Timing Analysis of X-ray Lightcurves

Michael Nowak, mnowak@space.mit.edu

X-ray Astronomy School; Aug. 1-5, 2011

Introduction

This exercise is designed to give a brief introduction to one aspect of timing analysis: Fourier transforms (executed via a Fast Fourier Transform, FFT, of discrete data) and their associated Power Spectral Densities (PSD). Parts of this exercise have been cribbed from Tomaso Belloni's tutorial that was presented at the European "2$^{\rm nd}$ School on Multiwavelength Astronomy" (June 28–July 9, 2010), which can be found at:

`http://www.black-hole.eu/index.php/schools-workshops-and-conferences/2nd-school-on-multiwavelength-astronomy/program`

There you can also find a very good presentation by Michiel van der Klis describing in much more detail the fundamentals of Fourier techniques in X-ray timing.

It is important to note that Fourier techniques are by no means the only tools employed in timing analysis. However, they are very commonly used, and often are the starting point, or at least the comparison point, for other analyses. Here we shall introduce the Power Spectral Density (PSD; sometimes also referred to as Power Density Spectra, PDS).

A Fourier Transform is a decomposition of a signal into a sum of complex exponential (i.e., sinusoidal) components, with each component having a complex amplitude and a complex phase. When dealing with discretely sampled, uniformly binned lightcurves, one uses the discrete Fourier transform. Specifically, if x_k are the lightcurve amplitudes measured at times t_k (subdivided into N uniform steps of time bins with width $1/T$), then the discrete Fourier transform is given by

$$a_j = \sum_k x_k e^{i\omega_j t_k} \quad . \tag{1}$$

Here i is the imaginary unit, and ω_j are the discrete Fourier frequencies given by

$$\omega_j \equiv 2\pi\nu_j = \frac{2\pi j}{T} \quad , \tag{2}$$

with j being the *integers*

$$j = -\frac{N}{2}+1, \ldots, \frac{N}{2} \quad . \tag{3}$$

The frequency $\nu_{\rm Ny} \equiv \nu_{N/2} = N/2T$ is referred to as the Nyquist frequency, and is the highest frequency one can explore in a discretely sampled lightcurve. (Please refer to the above cited lecture by Michiel van der Klis for *much* more detail on this issue.)

The Fourier transform can be inverted:

$$x_k \;=\; \frac{1}{N}\sum_j a_j e^{-i\omega_j t_k} \;=\; \frac{1}{N}\sum_{j=-N/2+1}^{N/2} a_j e^{-i2\pi jk} \quad . \tag{4}$$

If the signal x_k is real (i.e., like all the signals we deal with in X-ray timing analysis!), then for the Fourier transform of equation (1), the imaginary terms at $-j$ and $+j$ cancel out in the sum, leaving us with terms of the form:

$$2|a_j|\cos(\omega_j t_k - \phi_j) \ . \tag{5}$$

Since real signals are what we deal with in X-ray astronomy, one usually only deals with the positive Fourier frequencies, ω_j, and refers to $2|a_j|$ as the Fourier amplitude, and ϕ_j as the Fourier phase.

In this exercise, we will only be dealing with the squared Fourier amplitudes, $\propto a_j^* a_j = |a_j|^2$. The run of these amplitudes vs. frequency is what is referred to as the PSD. It is a measure of the "power" of a signal over a given range of frequencies. Specifically, the PSD is a measure of the fractional variability over a given range of frequencies. Often, you will see the PSD in units of $(RMS)^2$/Hz, i.e., (root mean square variability)^2/unit frequency. In this formulation,

$$\left(\int df\ PSD(f)\right)^{1/2} \ , \tag{6}$$

is the fractional variability of a signal over that frequency range.

This leads to a final point: PSD normalization conventions. In the above formulas, the negative sign in front of the imaginary unit could have been associated with either the "forward" (time to frequency) or "reverse" (frequency to time) transform. Likewise, the term involving N can be associated with one or the other transform, or split between both. And sometimes factors of 2 or $\sqrt{2}$ are introduced. That is, different software packages will define the transform pairs differently (but they will always come in pairs that are self-consistent, in that applying the reverse transform after the forward transform will get you back to where you started, modulo numerical roundoff errors). Always be careful to know the conventions of the software package that you are using!

In X-ray astronomy you will usually see one of two conventions employed, which can be distinguished by the properties of the PSD of Poisson noise. Poisson noise is referred to as "white noise" since the expectation for its PSD is that it is, on average, flat (i.e., constant) as a function of Fourier frequency. Astrophysical PSD are normalized such that the Poisson noise level is expected to be 2 (often referred to as the "Leahy normalization"), or such that the Poisson noise is expected to be 2/count rate (referred to as "RMS" or "Belloni-Hasinger" or "Miyamoto" normalization). For the former, the amplitude of an astrophysical signal scales with count rate (but the noise level is constant), whereas for the latter, the astrophysical signal is independent of count rate (but the noise level increases with decreasing count rate). In the exercise that follows, we will use the Leahy normalization.

Practice Creating Your Own Signals

The following presumes you are using `ISIS`, and have downloaded the `SITAR` package from:

```
http://space.mit.edu/cxc/analysis/SITAR/
```

`SITAR` will be used for two purposes: create a PSD from a lightcurve, and then register this PSD as a fittable dataset. Any system wherein one can: read a FITS file, perform an FFT, and then register the resulting PSD as a fittable dataset would suffice. The "value added" in `SITAR` is that it properly takes care of the normalization issues and gets the statistics correct, as we explain below. Reproducing the functionality that we will use here is not terribly difficult in a number of other systems. We leave it as an exercise for the ambitious student!

First, load `SITAR` from the same directory in which you are running `ISIS`.

```
isis> require(''sitar'');
```

Now create a linear grid of $1024 * 16$ time points ranging from 0–16 sec.

```
isis> (tlo,thi) = linear_grid(0,16,1024*16);
isis> tavg = (tlo+thi)/2;
```

Create a constant+sinusoidal signal with a period of 1/100 sec.

```
isis> period=0.01;
isis> omega=2*PI/period;
isis> lc=10+sin(omega*tavg);
```

Plot these data over a subsection of the lightcurve.

```
isis> xrange(0,0.5); xlabel(''Time''); ylabel(''Amplitude'');
isis> plot(tavg,lc);
```

A real signal will have noise associated with it, so let's add some to the lightcurve. Specifically, we will add Gaussian noise with zero mean and unit variance.

```
isis> lc += grand(1024*16);
isis> plot(tavg,lc);
```

Now use `SITAR` to create an *average* PSD, `psd`, at frequencies `f`, where the length of the individual PSD making up the average is 1024*16 (i.e., the whole lightcurve), and where the uniform time bin size is 1/1024. Plot the results!

```
isis> (f,psd,navg,cts) = sitar_avg_psd(lc,1024*16,1./1024,tavg);
isis> xrange();          % Reset the xrange of the data to autoscale
isis> xlog; ylog;                                % Use logarithmic axes
isis> xlabel(''Frequency (Hz)'');
isis> ylabel(''PSD'');
isis> plot(f,psd);
```

Here `navg` is the number of lightcurve segments used in the average PSD (here, 1, since we created a PSD of the same length as the input lightcurve), and `cts` is the average counts per segment (here, total counts). Note that her we have not used proper Poisson statistics, so we shouldn't worry too much about the overall normalizations at this point.

Why does one usually average the PSD from multiple lightcurve segments? One often assumes that real astrophysical signals represent *stochastic processes* (i.e., the lightcurve is not deterministic, but has well-defined statistical properties). In this case, we expect the lightcurve to have a well-defined average PSD, but a standard deviation equal to the average. We improve our estimate of the mean PSD by averaging many lightcurve segments and/or binning over adjacent frequency bins. Although averaging gives us a better estimate of the mean PSD, we sacrifice knowledge of the low frequency behavior (shorter segments means a higher minimum frequency for any given segment), and have to be careful not to average over so many frequency bins as to hide any interesting behavior. (One often uses a logarithmic binning approach where the number of frequency bins averaged over increases as one goes to higher frequency. Remember, the frequency bins are uniformly spaced with resolution of $1/T$.)

The lightcurves created above *do not* represent a stochastic process; therefore, we didn't bother averaging over multiple lightcurve segments. Averaging, however, will improve the estimates of the noise level. Try that now by using shorter data segments, and plot the results:

```
isis> (f,psd,navg,cts) = sitar_avg_psd(lc,1024,1./1024,tavg);
isis> plot(f,psd);
```

Repeat the exercise, but increase the amplitude of the gaussian noise component. How does the signal change? Try two sinusoids at different frequencies. Add phase shifts to the sinusoids, and compare the power spectra to what you obtained previously.

Power Spectra of Astrophysical Sources

I have placed on a web site a set of lightcurves from an RXTE observation of a neutron star source. The gzipped tar file can be found at:

```
http://space.mit.edu/home/mnowak/data/events.tar.gz
```

The file unpacks to files named `events_18_39_*.lc`, each of which contain counts vs. time. Start with the file `events_18_39_a.lc`. Read the time and counts from this file.

```
isis> (t,c) = fits_read_counts(''events_18_39_a.lc'',''time'',''counts'');
```

What is the width of the time bins in this lightcurve? What is its total length? What is the range of frequencies that you can explore with the PSD? Try plotting this lightcurve, and look at various short segments.

Create a PSD from this lightcurve. (Suggestion: FFTs tend to run fastest for lightcurves whose lengths are a power of 2. In fact, the RXTE clock was specifically designed to have its fundamental clock ticks be in units of a power of 2 times a second, e.g., 2^{-10} sec.) Where is the noise level? Are there any other interesting features present?

Let's assume that you have assigned the outputs of `sitar_avg_psd` to variables `f`, `psd`, `navg`, and `cts` as before. We can further bin the PSD by averaging over frequency bins.

```
isis> (aflo,afhi,apsd,nf) = sitar_lbin_psd(f,psd,0.01);
```

will average over bin widths $\Delta f/f = 0.01$. Each new frequency bin is averaged from `nf` frequency bins, while the original PSD was created from `navg` lightcurve segments. If the PSD error goes as the PSD value divided by the square root of the number of averages, what is the PSD error for a given frequency bin?

Fit a model to this PSD in the frequency range 500–1500 Hz. To do this, we must first "register" the PSD data: frequency bin values, PSD values, and PSD error bars. `SITAR` contains a function to do this in `ISIS`, which we apply as follows.

```
isis> Minimum_Stat_Err = 1.e-30;
isis> id = sitar_define_psd(aflo,afhi,apsd,apsd/sqrt(navg*nf));
```

The first command above resets the `ISIS` minimum error from the usual value of 1 (often appropriate when using Poisson statistics and counts/bin as the data) to a much lower value. Here the error on the PSD value can be as low as 10^{-3}. The second command assigns the frequencies, PSD, and PSD errors as a dataset, with the frequency units of Hz being treated like energy units of keV. Although programs like `ISIS` and `Sherpa` are in fact agnostic about the units of fittable datasets, many of their built in models (inherited from legacy `XSPEC` models) presume bin units of keV. If one wants to use these legacy models, it is convenient to make the implicit transformation of 1 Hz (Fourier frequency, *not* frequency of light!) = 1 keV.

Restrict the "energy" range to 500–1500 Hz (keV), and create a fit function that is a constant plus a gaussian, and then fit the data. Plot your results.

```
isis> xnotice_en(id,500,1500);            % Restrict the energy range
isis> fit_fun(''constant+gaussian'');
isis> list_par;                 % Look at the model default parameters
isis> set_par(1,2,0,1,3);     % Set parameters to reasonable starting
isis>                   % values, with sensible limits on the ranges.
isis>                      % The 0 means free (not frozen) parameter.
isis> set_par(''const*'',2,0,1,3);       % Alternative to the above -
isis>                        % use the parameter's name; wild cards OK
isis> % Next choose sensible gaussian parameters (not shown) ...
```

```
isis> fit_counts;
isis> xlin; ylin; xrange(500,1500); yrange(1.9,2.2);
isis> rplot_counts(id);
```

You'll notice that in the above plot, the x-axis says "Energy [kev]" and that the y-axis says "Counts/bin". Again, we are making the association of 1 Hz=1 keV in order to be able to use the `XSPEC` legacy models. (Custom plotting routines are available that will produce nice plots, and replace the axis labels with more reasonable defaults.)

What is the frequency of the gaussian feature? What is its width in comparison to this frequency? The frequency divided by the width is called the Q-value, with large Q-values indicating combinations of long persistence times, and stable frequency and phases, of an oscillating feature. Is this a high Q-value feature? A more realistic profile to fit to such a feature would be a Lorentzian function that was properly averaged over the width of the individual frequency bins. Such a function is not hired to write (i.e., a simple scripted fit function would suffice). We leave it as an exercise for the reader to write and fit such a function!

You can determine error bars for the parameters using the `conf` command. E.g.,

```
isis> conf(1);
```

will give the 90% confidence limit for the `constant` component. Is this constant statistically different than the expected value of 2? Note that deadtime in the detector *can* reduce the expected PSD of Poisson noise below 2. Is there evidence for detector deadtime here? Find the error bars for the gaussian feature as well.

Look at the other data files. Are the amplitude and frequency of the feature persistent? Is it present in all of the lightcurves? What changes can you note? Is there any evidence for a harmonic of this feature (e.g., another feature at twice the frequency). Feel free to combine all of the power spectra from all of the lightcurves. How would you go about doing that?