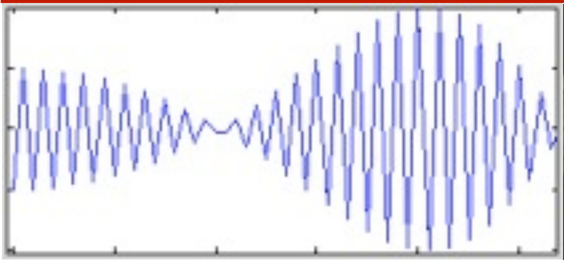
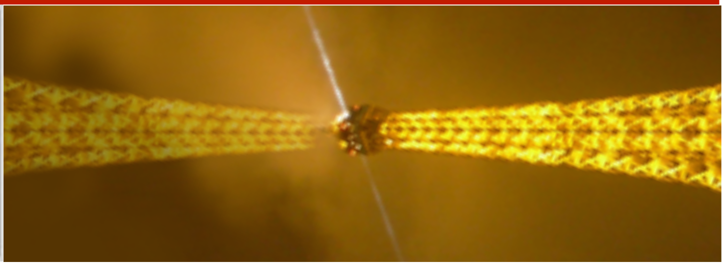
Paris 2009

Hands-on session on timing analysis





Introduction

During this session, we'll approach some basic tasks in timing analysis of x-ray time series, with particular emphasis on the typical signals found in the data from Black-Hole Binaries. This document is simply meant to provide rather general guidelines and is not intended to be exhaustive.

Some warm-up exercise

Before jumping to the data, let us start with some simple simulation. First we should take care of some initialization. In order to start the ISIS/SITAR session, you should load the modules

```
()=evalfile("sitar.sl");
()=evalfile("isis_fancy_plots.sl");
()=evalfile("isis_utility_functions.sl");
()=evalfile("isis_utility_functions_prerelease_1.5.sl");
```

Let us simulate a periodic sinusoidal curve; the commands should be self-explaining

```
variable ti, th, co, omega, period;
(ti,th)=linear_grid(0,16,1024*16);  % 1024*16 time points from 0 to 16s
period = 0.01;                      % our period is 1/100 s
omega=2.*3.14159/period;
co=10+sin(omega*ti);                % trivial sinusoid plus a constant
plot(ti,co);                        % have a look
xrange(0,1); plot(ti,co);           % have a meaningful look
```

This is not a serious signal: let us add some Gaussian noise, zero average, unit variance

```
 co=co+grand(1024*16);
```

We now produce a power spectrum of our signal `co`, using intervals of `1024*16` length (i.e. only one interval), `1/1024` is the time resolution and `ti` is the time array

```
(f,pa,na,ctsa) = sitar_avg_psd(co,1024*16,1./1024,ti);
```

Outputs are frequency array `f`, power array `pa`, number of intervals averaged `na` (in this case 1), and average intensity `ctsa` of the `co` signal. Without looking, you should be able to guess the first and last values of `f` and its number of elements. In other words: what is the minimum frequency, what the maximum frequency and how many frequencies do you have? You can examine the nth element of an array with `print(f[n]).` Arrays start from 0.

Plotting the power spectrum should show a peak at 100 Hz

```
xlabel("Frequency (Hz)\n");
ylabel("PSD\n");
xrange; xlog; ylog;     % xrange resets the x range to automatic
```

```
plot(f,pa);
```

The rest is noise. We are not dealing with counting noise here (we simply added some additional noisy signal), so we should not be concerned with the normalization. If you go on a linear frequency axis (`xlin`) and zoom on the peak at 100 Hz with `xrange`, you can see that the peak has a width: you should know how broad it is even without looking. Let us try to reduce the length of the time series by splitting it into shorter intervals (16 intervals of 1 second each) and averaging the resulting power spectra

```
(f,pa,na,ctsa) = sitar_avg_psd(co,1024,1./1024,ti);
```

What are now the minimum and maximum frequency? How broad is the peak now?

The signal we played with until now is too good. Try increasing the amount of noise (for instance, its variance) and see whether you can still detect the signal. It is instructive to have a visual look at the light curves as well: Fourier analysis is a very powerful technique and can pick up signals that are completely invisible by eye.

Before moving to the real data, play with signals for a bit and get a feeling of what you can do with just a simple power spectrum commend such as `sitar_avg_psd`.

- ☑ Try two sinusoids at different periods, harmonically related or not
- ☑ Add a phase shift to the sinusoid(s) and compare power spectra
- ☑ Try some weird combination of signal and noise

The available data

We have selected four RossiXTE data sets to cover typical examples. In order to allow the students to concentrate upon the timing analysis aspect of the work, we have made available already extracted binned light curves for the four cases:

- **bln.lc** - An observation of Cygnus X-1 in its Low-Hard State. The short-term time variability is dominated by a number of strong band-limited noise components.

- **typec.lc** - An observation of XTE J550-564 in its Hard Intermediate State. Here, a strong nd narrow Quasi-Periodic Oscillation, with typical frequencies between 1 and 10 Hz is present, together with band-limited noise. The QPO has additional harmonic peaks.

- **typeb.lc** - An observation of GX 339-4 in its Soft Intermediate State. The power density spectrum is dominated by the presence of a strong 4-8 Hz QPO, accompanied by a power-law noise. The QPO has additional harmonic peaks and jitters on a time scale of ~10 seconds.

- **hfqpo.lc** - An observation of GRS 1915+105 where a high-frequency QPO is detected. These oscillations are weak and elusive. This is the best case ever, so that detection should not prove to be a problem.

Some initialization

In order to start the ISIS/SITAR session, remember to load the modules

```
()=evalfile("sitar.sl");

()=evalfile("isis_fancy_plots.sl");

()=evalfile("isis_utility_functions.sl");
```

```
()=evalfile("isis_utility_functions_prerelease_1.5.sl");
```

and declare some convenient variables. This is needed for any of the four datasets

```
variable ta,ca,pa,na,ctsa,f,cts,aflo,afhi,apsd,nf;
```

Low-Hard State

Before moving to the frequency domain with a power spectrum, let us have a look at the raw data and see what the light curve looks like. The time resolution of this curve is 1.953125 ms, corresponding to 512 bins per second. The total length is 3634 seconds, just over one hour. We can load the data and convert the number of counts to integer (to speed up the FFT and save space) with

```
(ta,ca) = fits_read_col("events_18_39_a.lc","time","counts");

ca = typecast(ca,Integer_Type);
```

Time to plot your light curve

```
xlabel("Time  (sec)");

ylabel("Counts");

plot_bin_integral;

xlin; ylin;

xrange; yrange;

plot(ta-ta[0],ca);
```

Difficult to see much at this high time resolution. Zooming on the first 10 seconds

```
xrange(0,10);
```

reveals some structure, but there are so few counts in each bin that one can see the jumps between integers. At any rate, the data are quite variable, as expected.

Let move to the frequency domain and produce a PDS: we'll calculate a PDS for each 16s interval, then average the PDS together

```
(f,pa,na,ctsa) = sitar_avg_psd(ca,8192,1./2^9,ta);
```

here `ca` is the time series, `8192` is the number of points per data interval (i.e. 16 seconds at 512 pps), `1./2^9` is the time resolution and `ta` are the times for the bins, so that gaps can be taken care of. There should not be any gap here. Plotting the PDS is not particularly enlightening

```
xlabel("Frequency (Hz)\n");

ylabel("PSD\n");

plot(f,pa);
```

Plotting in loglog seems to be a much better idea

```
xlog; ylog;

plot(f,pa);
```

Some rebinning is in order. A logarithmic rebinning with a factor of 1%, meaning that each frequency bin is 1% broader than the previous one, leads to a much more satisfying result

```
(aflo,afhi,apsd,nf) = sitar_lbin_psd(f,pa,0.01);
hplot(aflo,afhi,apsd);
```

The flattening at high frequencies is due to the Poissonian noise, which should be at an average level of 2 but is modified (lowered) by the effects of detector dead time. The two possible courses to follow in order to remove it from the data are to estimate it through the available models or to fit it with a constant value. Then subtract (see below). Just to have a look, let us pretend it is exactly at 2 and subtract it

```
dumpsd = apsd - 2.0;
hplot(aflo,afhi,dumpsd);
```

The full shape of the PDS is now clearly visible.

We must fit this PDS with a suitable model in order to extract useful information. First we must assign the PDS as a fittable set, with errors (power/sqrt(number of average)

```
variable id = sitar_define_psd(aflo,afhi,apsd,apsd/sqrt(na*nf));
```

If needed, we can restrict the fittable part of the PDS to a range of frequencies, say 0.1 to 100 Hz

```
xnotice_en(id,0.1,100);
```

Here it is not really needed. Now we define a suitable model for the fit as a constant (for the Poisson noise) and two zero-centered Lorentzians for the source intrinsic noise

```
fit_fun("constant(1)+zfc(1)+zfc(2)");
set_par("constant(1).factor",1.99);
```

notice the value lower than 2

```
set_par("zfc(1).norm",100);
set_par("zfc(1).f",8);
set_par("zfc(2).norm",50);
set_par("zfc(2).f",0.8);
```

Now we can rem=normalize to the integral, to help the fit, and try a fit

```
() = renorm_counts;
() = fit_counts;
```

Looking at the parameters gives some impression, but the reduced chi square is still high

```
list_par;
```

We need to plot the fit to see how we are faring. First setting (laboriously) some variables

```
d_width=3; r_width=3; m_width=3; de_width=1; re_width=1;
set_frame_line_width(3);
charsize(1.12);
Plot_Unit("psd_leahy");
popt.dsym=0;
popt.dcol=4;
popt.decol=5;
popt.rsym=0;
popt.rcol=4;
popt.recol=5;
popt.xrng=[0.05,256];          X range: not necessary, but you can see how we do it
popt.res=1;                    Plot also residuals
```

and then plot the resulting fit

```
plot_counts(id,popt);
```

In case you wanted to save it as a ps file, you can do this

```
variable pid = open_print("my_first_pds.ps/vcps");
resize(21,0.85);                         your first PDS must look good
plot_counts(id,popt);
close_print(pid,"gv");
```

The fit was not very good, but we will deal with that later. For now, let us be content with the result and estimate some error on the parameters

```
(,) = conf_loop([1:5];save,prefix="first_bln");
() = system("more first_bln.save");
```

In case you want to start from scratch, you can delete all data and restart

```
delete_data(all_data);
```

Now you can try to do more. For instance:

☑ Add another zero-centered Lorentzian component to the fit and see if it gets any better.

☑ Difficult to see how the model is doing even with the residuals. You can restrict the fit to the high frequency part, say above 100 Hz, where Poissonian noise dominates, fit with a constant, then subtract it from your data and go on without Poissonian noise.

☑ Produce power spectra extending to lower frequencies to see how the noise flattens. Can you also extend it to higher frequencies?

☑ Divide the data in more than one segment and see whether there are differences between the segments.

☑ Can you calculate the integrated fractional rms of the different components? What do you need to do that?

Type-C Quasi-Periodic Oscillation

Here you encounter a low-frequency QPO in addition to band-limited noise.

☑ Extract the light curve and have a look. Does it look any different from the previous one?

☑ Extract the power spectrum and see the QPO. How many noise components and how many QPO peaks do you think you will need? Always better to start with an underestimate and add as you go. The model to use for a QPO is, not surprisingly, qpo. Its parameters are norm, Q and f. They correspond to normalization, quality factor (centroid divided by FWHM of the peak) and centroid frequency.

☑ Do you see any differences in PDS taken at different times?

☑ Can you calculate the integrated fractional rms of the different components? What do you need to do that

Type-B Quasi-Periodic Oscillation

Yet another low-frequency QPO on top of some power-law noise

☑ Extract the light curve, as usual. Do you see some section which is different from the others?

☑ Produce the power spectrum and have a look. Does the peak look like a Lorentzian?

☑ Try fitting. You can use a powerlaw (model powerlaw) for the continuum. See whether this is sufficient. Does a Lorentzian model work for the QPO? You might want to try a gaussian (gaussian).

☑ Extract a power spectrum from the first 120 seconds of observation. What does the QPO look like here? Compare it with any other 120 s interval.

☑ Try to follow the QPO on higher time scales, a few seconds.

☑ Again, fractional rms of the components.

High-frequency Quasi-Periodic Oscillation

Here we move to higher energies. The source, GRS 1915+105, is also pretty weird.

☑ A light curve will produce some interesting plots.

☑ Produce a power spectrum and concentrate on frequencies above 10 keV, away from the messy part. That's where the important signal lies.

☑ Fit the HFQPO and get its parameters. Is it significant? At what confidence level?

☑ How strong (in fractional rms) is the oscillation?