X-ray Spectra Part II: Fitting Data and Determining Errors

Michael Nowak, mnowak@space.mit.edu

July 5, 2010

Setting Up the Data

This exercise presumes that you've downloaded and installed the `.isisrc` files located at:

`http://space.mit.edu/home/mnowak/isis_vs_xspec/download.html`

If these files are placed in your home directory, and the path variable in the main `.isisrc` file is edited to point to your home directory, then these will automatically be loaded when you start ISIS. Further you need to have downloaded the data from the web page:

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

These files need to be placed in whatever directory where you will be running ISIS.

As in Part I of the exercise, we are first going to start with the PCA data. This time we will explicitly set systematic error bars, then group the data, then restrict the noticed energy range.

1. Load the data, and then add 0.5% systematic errors to all PCA energy channels with the `set_sys_err_frac` function. There are 129 channels (you can see this with the `list_data` function). Normally `set_sys_err_frac` adds errors in wavelength order, but since we are adding uniform errors, we can ignore that distinction.

```
variable pca = load_data("pca.pha");
set_sys_err_frac(pca,Double_Type[129]+0.005);
```

Again, the `variable` declaration is only required in scripts, not on the command line, so we drop its use from now on in this exercise. Use the `list_data` and `list_rmf` commands to verify what you've loaded.

2. Group the data to a minimum signal to noise of 5 above a lower bound of 3 keV. (Note: the signal-to-noise calculation ignores systematic errors.) Then restrict the data to the 3–22 keV range. As always, plot this to see what it looks like.

```
group(pca;min_sn=5,bounds=3,unit="kev");
notice_values(pca,3,22;unit="kev");

xlog; ylog;
plot_data(pca);
```

Defining the Fit Function and Fitting

3. Start off with a simple fit function, an absorbed powerlaw. Set the parameters either via the `edit_par` command, or on the command line via the `set_par` command. Since PCA doesn't have terribly good low energy coverage, fix the absorption to a reasonable interstellar value. Choose a "canonical" value of the powerlaw index.

```
fit_fun("phabs(1)*powerlaw(1)");
list_par;                        % See the parameters you need to set
set_par("phabs(1).nH",0.6,1);    % Freeze the column
set_par("p*Index",1.7,0,1,3);    % Limit the index to a sensible range
```

It's good general advice to choose broad but "sensible" minimum and maximum ranges for the parameter values. Error bar searches will be faster and more accurate if the ranges aren't unreasonably wide.

4. The closer your parameters are to a good fit to begin with, the better the odds are that the fit procedure will work. So, evaluate and plot the model, and change parameters by hand if you think you can get a better start. When ready to start in earnest, renormalize first, then fit!

```
() = eval_counts;
plot_data(pca;res=2);
() = renorm_counts;
() = fit_counts;
```

Plot your results, and look at the residuals, first with χ^2, and then ratio residuals.

```
plot_data(pca;res=2);
plot_data(pca;res=3);
```

5. Looking at the fit statistic and the residuals, you'll see that things can be improved. Clearly there's a line around 6.4 keV, i.e., the fluorescent Fe line. Add one into the fit model. Don't let its energy wander too far off, nor let its width get too wide. (Otherwise, it might start subsuming continuum components.) Follow the above procedure: set parameters, plot, reevaluate the set parameters, renormalize, then fit.

```
fit_fun("phabs(1)*(powerlaw(1)+gaussian(1))");
set_par("*LineE",6.4,0,6,7);   % Keep the line between 6 & 7 keV
set_par("*Sigma",0.1,0,0,1);   % Don't let the line get wider than 1 keV

()=eval_counts;
plot_data(pca;res=5);  % The line is too strong.  Shrink, then renorm, & fit

set_par("g*norm",1.e-4,0,0,1);
()=renorm_counts;
()=fit_counts;
```

As always, plot the results of your efforts!

```
plot_data(pca;res=2);
plot_data(pca;res=3);
```

6. There looks like there is a break in the powerlaw, so let's change the power law model to a broken powerlaw model. Choose sensible parameters, renormalize, and then fit again. Try changing fit methods. "subplex" is a very slow, but very robust fit method worth trying at times.

```
fit_fun("phabs(1)*(bknpower(1)+gaussian(1))");
set_par(2,1.5,0,0.1,10);  % bknpower has same normalization as powerlaw
set_par(3,1.6,0,1,3);     % slope wasn't too far off
```

```
set_par(4,10,0,8,13);     % break looks to be around 9 or 10 keV
set_par(5,1.4,0,1,3);     % second power law looks "harder"

()=renorm_counts;
()=fit_counts;

subplex;  % Script defined shorthand for set_fit_method("subplex");
() = fit_counts;
```

Searching for Error Bars

7. Perform an error bar search. As `subplex` is a fairly slow method, it's best to revert back to the `lmdif` method. ISIS has several functions for searching for error bars. You can search on individual parameter error bars using `conf` or `vconf` [=verbose form of `conf`]). It has been customary in X-ray astronomy to search for "90% confidence level" error bars (level `1` in the ISIS error bar functions), but 68% (level `0`) and 99% (level `2`) are also available as standard choices. (Arbitrary levels are possible using the `fconf` or `vfconf` commands.) Rather than searching for error bars on one parameter, we can search a whole set of parameters, and have ISIS refit the data any time a new, lower χ^2 value is found. (Error bar searching is in fact a great way to tweak your fits, and make sure that you really have found a global minimum for the fit statistic.)

```
lmdif; % Script defined shorthand for set_fit_method("lmdif");

% The blank means "do all parameters", 1 means 90% confidence level, and
% the 0.1 means only refit if the statistic changes by more than 0.1

(,) = conf_loop(,1,0.1;save,prefix="bknpower.");

% To see your results from the command line (!= do system commands):
!more bknpower.save
% Or to do the same from a script:
() = system("more bknpower.save");
```

As always, plot your results, and take a look at them! Here, however, let's also take a look at the flux corrected spectra in physical units. The default behavior of these plotting routines is to also show the predicted model counts "flux corrected" in exactly the same manner as the data counts. That is, the smearing of the model due to the response matrix is included.

```
plot_data(pca;res=5);
plot_data(pca;res=6);

fancy_plot_unit("kev","Ergs");
plot_unfold(pca;power=3);
plot_unfold(pca;power=3,con_mod=0); % To see the model without smearing
```

Adding a Second Dataset

8. Add the HEXTE data into the mix. Load the data, group it to a minimum signal-to-noise of 8 per channel, and restrict the noticed energy range to 18–200 keV. Then evaluate the current model *without any refitting*, and plot it to see how close you've come when just using the PCA data alone.

```
variable hxt = load_data("hxt.pha");
group(hxt;min_sn=8,bounds=18,unit="kev");  % Bin more than for PCA
notice_values(hxt,18.,200.;unit="kev");

() = eval_counts;
plot_data(hxt;res=2);
plot_data({pca,hxt};res=2);
```

9. Although the PCA only fit isn't too far off. Fit the data, *without changing the model*, to see how much closer we get to a good description of the joint data. Again, plot your data after the fits. Here, however, you might find that plotting the flux corrected data gives you some clues as to how to proceed further.

```
() = fit_counts;
plot_data({pca,hxt};res=2);
plot_unfold({pca,hxt};res=2);
plot_unfold({pca,hxt};res=3);
```

You should notice two things in the above plots. First, the normalization of the PCA and HEXTE instruments don't quite agree. This is *very common* in X-ray detectors. Obtaining an absolute flux value for any given space-based detector is very, very difficult. One should expect 10% disagreements to be common. The other thing that you should notice is that the HEXTE data "rollover" around 100 keV. Such a spectral cutoff is common in the hard state of black hole candidate systems.

10. Add a normalization constant to your model that you define to be `1` for the PCA, but let be a free parameter for the HEXTE data. Also add an exponential rollover to the HEXTE data using the `highecut` model. When you have redefined the model, fit the joint data.

```
fit_fun(
"phabs(1)*highecut(1)*constant(Isis_Active_Dataset)*(bknpower(1)+gaussian(1))"
);
list_free;

set_par("constant(1).factor",1,1);           % Freeze the constant
                                             % for PCA to 1
set_par("constant(2).factor",0.9,0,0.5,1.5); % But let the HEXTE
                                             % constant be free
set_par("high*cutoffE",20,0,8,200);          % Don't let the cutoff
                                             % start by the Fe line,
                                             % nor above the HEXTE
                                             % upper bound
set_par("*foldE",100,0,0.01,500);            % Reduce the upper bound
                                             % for the folding energy
```

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

Never forget to plot your results to see that they make sense!

```
plot_unfold({pca,hxt};res=2);
plot_unfold({pca,hxt};res=3);
```

Saving and Loading Parameters

11. In the `conf_loop` call used up above, since the `save` qualifier was used parameter files for the fits were saved along the way. To explicitly save the parameters, use the `save_par` command. Then, to reload those parameters use the `load_par` command.

```
save_par("bknpower_joint.par");
load_par("bknpower_joint.par");
```

Also do the full error bar search as above, and save the results along the way. The `bknpower_joint.save` file has the lower and upper bounds of the 90% confidence limits placed in the min/max columns. (This needs to be remembered if you want to use the `save` file to start a fit to other data. More likely you would want to use a parameter file with the default or input min/max columns.)

```
(,) = conf_loop(,1,0.01;save,prefix="bknpower_joint.");
!more bknpower_joint.save
```

12. Note that in the above we got very low χ^2 for the PCA data, so perhaps the systematic errors in this particular case were overkill. Let's turn off the systematic errors, and see how the results change:

```
set_sys_err_frac(pca,NULL);
() = renorm_counts;
() = fit_counts;
save_par("bknpower_joint_nosyst.par");
(,) = conf_loop(,1,0.01;save,prefix="bknpower_joint_nosyst.");
() = system("more bknpower_joint_nosyst.save");
```

Error Contours

It's also very common for parameters to be correlated with one another. To explore this possibility, it's useful to perform *joint error bar analysis*. The `conf_loop` function and their ilk look at one parameter at a time: that parameter is held frozen at a value, the fit is redone, and the change in χ^2 from its minimum value is calculated. *Error contours* work exactly the same way, but for two parameters. (In principle, one can do this for as many parameters as you like, but the lack of N-dimensional plotting routines, as well as long computational times, typically limits one to two parameters at a time.)

13. Use the ISIS `conf_map_counts` function to calculate the error bars for the two photon indices. You first have to define the grid over which you want to calculate the contours (using the `conf_grid` function). You don't want to make this grid too large, otherwise the calculation might take a very long time. Use the 90% confidence limits to gauge reasonable ranges over which to calculate the grid. You can plot your results using the `plot_conf` function, and save the contours to a FITS file using `save_conf`.

```
variable x = conf_grid(6,1.675,1.695,20);
variable y = conf_grid(8,1.45,1.51,60);
variable cntr = conf_map_counts(x,y);
xlin; ylin;
xrange; yrange;                          % Make sure ranges autoscale
xlabel("Photon Index, \\gG\\d1");
ylabel("Photon Index, \\gG\\d2");
plot_conf(cntr);
() = save_conf(cntr,"gamma1_vs_gamma2_contour.fits");
```

Line and Model Fluxes

Now that we've got a good fit to the joint data, we can use the model fits to calculate the model-dependent strengths of the line and continuum flux. The former is usually referred to in terms of an equivalent width. If staying within the well-calibrated bounds of the detector, the flux will be about the same for any "good" fit. The equivalent width, however, is likely to be more heavily model-dependent.

14. Use the script functions to calculate the line equivalent width, and the model fluxes in several energy bands. Calculate the flux in the same energy band for both PCA and HEXTE separately.

```
eqw(pca,"gaussian(1).norm";print);

model_flux(1,2.,10.;unit="kev",print);
model_flux(1,10.,20.;unit="kev",print);
model_flux(2,20.,200.;unit="kev",print);
```

Where does most of the energy come out in this spectrum? Does this agree with the flux corrected spectra plots?

More Complex Models - Extending the Grid

The broken powerlaw model is a very simple phenomenological model. We can try something a little bit more sophisticated: reflection. The reflection model, however, relies on the model being calculated well-outside of the normal bounds of the PCA and HEXTE energy ranges. Calculation of the model at 1000 keV affects the calculated spectrum at 10 keV. Thus in order to accurately calculate the reflection model, we have to use a custom grid.

15. ISIS allows one to create custom grids for any data set. The `.isisrc` scripts have some functions to automate this process somewhat for common situations as outlined above. Specifically, the `usr_grid` function creates a logarithmic energy grid. Use this function to create a grid that runs from 100 eV to 1 MeV.

```
usr_grid([1,2],-1,3,0.005); % Create a grid from 0.1 keV to 1 MeV, in
                            % 800 logarithmic steps
() = eval_counts;
```

Note that the χ^2 will have changed *a litle*. This gives you some idea of the numerical accuracy in these models.

16. Define the fit function to have a reflected powerlaw (`pexrav`) instead of the broken powerlaw. The normalization and slope parameters for the `pexrav` model should be the same as for the broken power (use the soft energy slope of the broken powerlaw). Renormalize the model, then fit, plot your results, and save your parameters.

```
fit_fun("phabs(1)*constant(Isis_Active_Dataset)*(pexrav(1)+gaussian(1))");

list_free;

set_par("pex*norm",0.27,0,0,1);
set_par("pex*Index",1.68,0,1.5,2.);
set_par("pex*foldE",300,0,50,100000);
set_par("pex*refl",0.5,0,0,2);

() = renorm_counts;
() = fit_counts;

xlog; ylog;
plot_data({pca,hxt};res=2);
plot_data({pca,hxt};res=3);

save_par("reflect_joint.par");
```

17. Now find the error bars for the reflection model fit. Since the reflection model is a little bit slow, this can take a little bit of time. (That's why we save it for last!)

```
(,) = conf_loop(,1,0.1;save,prefix="reflect_joint.");
() = system("more reflect_joint.save");
```