X-ray Spectra Part III: High Resolution Data

Michael Nowak, mnowak@space.mit.edu

May 12, 2016

Starting up ISIS

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. You also need to have downloaded the data from the location:

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

These files need to be placed in whatever directory where you will be running ISIS. They represent Chandra high energy transmission gratings observations of GRO J1655−40 (Miller et al., 2008, ApJ, 680, 1359), specifically ObsID 5461. These data files were downloaded from *TGCat*, the *Transmission Gratings Catalog*, which can be accessed at:

`http://tgcat.mit.edu`

There you can browse, plot, and download spectral products for all publicly available Chandra gratings observations.

Loading Gratings Data

There are two sets of Chandra transmission gratings: the High Energy Grating (HEG) and the Medium Energy Grating (MEG), each of which disperses in two directions away from the aimpoint (the negative and positive dispersion orders). Furthermore, at any location along the dispersed spectra, one finds multiple dispersion orders corresponding to wavelengths λ, $\lambda/2$, $\lambda/3$, ... These orders are separated from one another using the energy resolution of the CCD. The standard spectral extraction routines typically create spectra for the first three orders of each set of gratings in each direction. That is, one extracts twelve spectra: HEG -3,-2,-1, 1, 2, 3, and MEG -3, -2, -1, 1, 2, 3. Rather than create twelve separate spectral files, all twelve spectra are stored in a *single* FITS file, referred to as a "Type 2 PHA" file. For the case of the HETG spectra, the 12 spectra are stored in the order listed above. The advantage is that there is one file with all the associated spectra. The disadvantage is that there aren't standard protocols for storing the information about the names of the associated arf and rmf files for each spectrum.

Reading such a PHA2 file is not a problem for either ISIS or XSPEC. One just has to make sure to also read the proper arf and rmf files, and then associate them with the correct spectra. Here we will work with just the first order spectra (both positive and negative dispersion orders), since they contain the vast majority of counts, and are also the best calibrated of the spectra.

1. Read the HEG ± 1 and MEG ± 1 spectra from the PHA2 file (the third, fourth, ninth, and tenth spectra in the file) with the `load_data` function. Also read the arf (`load_arf`) and rmf (`load_rmf`) files, and then

associate them with the data using the `assign_arf` and `assign_rmf` functions. (You can get fancy by writing a simple loop to do the response read and assignment, or you can do it by hand.)

```
() = load_data("pha2.gz",[3,4,9,10]); % First order spectra-> Data 1-4

variable order=["heg_-1","heg_1","meg_-1","meg_1"], i;
_for i (0,3,1)
{
   () = load_arf(order[i]+".arf.gz");
   () = load_rmf(order[i]+".rmf.gz");
   assign_arf(i+1,i+1);               % Assigns ARF # -> Data #'s
   assign_rmf(i+1,i+1);               % Assigns RMF # -> Data #'s
}
```

2. Plot the data. Since the gratings disperse *linearly* in wavelength, and the spectra have constant width wavelength bins, let's first plot the spectra in Å. The useful range of the Chandra HETG is $\approx$ 1.5–30 Å. We'll be doing a lot of plotting, so in this case it might be best to define a structure variable with the plot options set.

```
fancy_plot_unit("a");
popt.dsym={0,0,0,0};
popt.dcol={1,4,2,8};
popt.decol={15,5,9,7};
popt.rsym=@popt.dsym;
popt.rcol=@popt.dcol;
popt.recol=@popt.decol;
xlog; ylog;
plot_counts({1,2,3,4},popt;xrange={1,28},yrange={1,8000});
```

Combining Data

To combine or not combine data? In principle, if one uses the proper statistical tests, there isn't any real advantage to combining data. However, combining data might allow one to raise the counts/bin sufficiently to use χ^2 statistics, it might serve the purposes of "averaging over" systematic deviations from one observation to another (or in this case, among the four different dispersed spectra), and it reduces the computational time. (The model is evaluated once, by default, rather than four times.) Combined data might also be easier to plot and visualize; however, in ISIS one can combine the data in a plot without having to combine the data for a fit.

Since combining data is something that you sometime might want to do, for purposes of this exercise, we will add together all of the gratings spectra. Before we add the spectra, however, we must place them on the same spectral grid. The HEG data have twice the spectral resolution of the MEG (i.e., over a given wavelength interval there are two HEG bins for ever one MEG bin); therefore, it's best if we match the HEG data to the MEG data. We do this using the ISIS `match_dataset_grids` function. The first dataset in the list defines the grid which we will match, and all subsequent data sets list will be regridded. (Note that the MEG -1 and +1 data are already are on the same grid.) This procedure works well for Chandra HETG grids since there is a factor of two difference between them; however, for other grids you might notice artifacts due to linear interpolation where the grids haven't nicely lined up. We then tell ISIS to add the data by using the `combine_datasets` command.

3. Place all the data on a common grid (the MEG grid), combine the data, group it to a minimum signal-to-noise of 5 and a minimum of two channels per bin (≈ half width half maximum resolution of the MEG, and full width half maximum for the regridded HEG data).

```
match_dataset_grids([3,1,2]);
variable combo_id = combine_datasets([1:4]);

group([1:4];min_sn=5,min_chan=2,unit="a",bounds=1.7);
notice_values([1:4],1.7,28;unit="a");
```

The response matrices for the HETG are *almost* diagonal, so using "flux corrected" data is *a little* less dangerous for this case. (Again, revert back to "detector space" plots to check your data and your fits!) The plotting routines from the `.isisrc` scripts will combine any data set indices that are input together in an array, denoted by []. *This will only work if they share a common grid, and will happen regardless of whether or not they have been combined for fitting purposes.* (If they don't share a common grid, an error will occur; even if they have been combined for fitting, they will be plotted separately if they are not in an array together.)

4. Plot the flux corrected spectra, and then zoom in on 12–16 Å region. Note the features that you see here. We'll be fitting some of these. You can get a good idea of their wavelength location by using the `cursor` command. This will give you cross hairs on the plot that allow you click and obtain coordinate locations.

```
plot_unfold({[1,2,3,4]},popt;xrange={1.5,28});
xlin;
plot_unfold({[1,2,3,4]},popt;xrange={10,20});
plot_unfold({[1,2,3,4]},popt;xrange={12,16});

cursor;
```

Fitting an Edge and Lines

You should notice an absorption edge in these data, as well as several prominent absorption lines. We're going to do a *local* fit to describe these features. That is, we are not going to attempt to describe the global spectrum, rather we are going to try to describe the location and depth of the edge, as well as the location of the absorption lines.

5. Restrict the range of the noticed data to 13.5–15 Å. Feel free now to switch into keV units (and maybe flux units for the y-axis). Start with a really simple local continuum model – a powerlaw – fit this, and look at the ratio residuals.

```
notice_values([1:4],13.5,15;unit="a");
plot_unfold({[1,2,3,4]},popt;xrange={13.5,15});

fancy_plot_unit("kev","ergs");
plot_unfold({[1,2,3,4]},popt;xrange={NULL,NULL});

fit_fun("powerlaw");
() = renorm_counts;
() = fit_counts;
plot_unfold({[1,2,3,4]},popt;res=6,xrange={NULL,NULL});
```

Note that the qualifier choices `res=4--6` indicate that the residuals for the data sets are to be combined. (Choosing `res=1--3` leaves the residuals uncombined, regardless of whether or not the data is combined. This allows you to see the individual contributions to the fits.)

The ratio residuals should give you an idea as to the depth of the absorption edge. (The fractional residual at the edge will be close to the optical depth.) You can use the `cursor` command to get a good idea of the location of the edge.

6. Add an edge to the model and fit the data. In general, when attempting to fit high resolution features in such data, it's best to restrict the locations and widths of the model components, to prevent them from wandering off, or becoming broad and fitting continuum features instead.

```
fit_fun("edge*powerlaw");
set_par("*Tau",0.2);
set_par("*edgeE",0.87,0,0.86,0.88);
() = fit_counts;
```

The presence of narrow features embedded in a broader, noisy continuum makes fitting these data a good candidate for the `subplex` method, even if it is slower. So long as we don't have too many bins, and aren't attempting to fit many, many lines, it won't slow us down too much in this case, and it might help us better find a global minimum.

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

If you look at the fit parameters, you'll notice that the powerlaw slope is pegged against the lower bound. Let's set it to the model "hard limit" and freeze it there. (Again, we're mostly concerned with the narrow features, not the broader continuum. An exercise for the reader, however, is to determine the systematic changes in the fitted narrow band parameters with different assumed continuum models.)

```
set_par("*Index",-3,1,-3,0);
() = fit_counts;
plot_unfold({[1,2,3,4]},popt;res=5);
```

7. We've improved the fit and have obtained a first estimate of the edge parameters; however, there are clearly absorption lines present in the data. Use the `cursor` function to constrain the location of the most prominent one. Incorporate it into the model by subtracting a `gaussian` function. Fit the data and plot your results. Again, constrain the `gaussian` parameters to help the fit from becoming "lost", and to keep the `gaussian` from becoming broad and fitting continuum features instead.

```
cursor;
fit_fun("edge*(powerlaw-gaussian)");
set_par("g*LineE",0.848,0,0.845,0.852);
set_par(5,1.e-4,0,0,1);
set_par(7,1.e-3,0,0,1.e-2);

() = eval_counts;                    % Initial parameters look OK?
plot_unfold({[1,2,3,4]},popt;res=5);

() = fit_counts;                     % If yes, procede to fitting
plot_unfold({[1,2,3,4]},popt;res=5);
```

The fit has improved; however, additional absorption lines remain, including a possible line very close to the edge. Let's add three more gaussians to the fits. Use the cursor function to get an idea of their location, incorporate them into the model, and fit.

```
cursor;
fit_fun("edge*(powerlaw-gaussian(1)-gaussian(2)-gaussian(3)-gaussian(4))");
set_par("g*Sigma",5.e-4,0,0,0.002);
set_par("g*norm",8.e-4,0,0,0.01);
set_par("g*(2).LineE",0.838,0,0.835,0.841);
set_par("g*(3).LineE",0.855,0,0.85,0.858);
set_par("g*(4).LineE",0.867,0,0.865,0.87);
() = renorm_counts;
() = fit_counts;
plot_unfold({[1,2,3,4]},popt;res=5);
```

8. Now run an error bar search on all the parameters, and save the final fit results to a file. For use later in the exercise, also save the fit statistic information.

```
mofit;          % Switch back to mpfit as the faster method
(,) = conf_loop(,1,0.01;save,prefix="edge.");
() = system("more edge.save");
save_par("edge.par");

plot_unfold({[1,2,3,4]},popt;res=5,con_mod=0);
plot_unfold({[1,2,3,4]},popt;res=5,oplt=1);

variable info_strt;
() = eval_counts(&info_strt);
```

Note that in the above we have plotted the model both with and without smearing by the detector response. The reason for this is that a gaussian line is only an *approximate model* for real absorption. When subtracting a gaussian, it's completely possible for the summed model to *become negative*, which then goes unnoticed after the model is smeared by the detector response. (The forward folding in ISIS doesn't care that the model has gone negative - it's just a vector of numbers related to a function that ISIS is trying to minimize.) *There have been spectral analyses published in the literature where this has occurred.* So, be careful, and double check your work, and make sure you are in a regime where a gaussian line is an acceptable approximation. Use a more sophisticated model, such as a Voigt profile, if warranted and required by your data!

The expected location of the Neon edge is 14.295 ± 0.003 Å. The Neon II 1s$\rightarrow$2p line is expected at 14.608 ± 0.002, and the Neon III 1s$\rightarrow$2p is expected at 14.508 ± 0.002. (Note that for many X-ray lines of ionized species, Chandra HETG observations have provided better determinations of their positions than either theoretical calculation or laboratory measurements!) How close do your values come to the above? Do your results argue for the edge and line being intrinsic to the black hole system, or due to absorption by the interstellar medium?

Monte Carlo Simulations

9. The next most prominent residual occurs at ≈ 859 eV. Is this another significant absorption line? We can add one in, fit the data, and run the error bars. Plot and save your results, and save the fit statistic for use in the next step.

```
cursor;

fit_fun("
edge*(powerlaw-gaussian(1)-gaussian(2)-gaussian(3)-gaussian(4)-gaussian(5))"
);
set_par("g*(5).LineE",0.859,0,0.857,0.860);
set_par("g*(5).Sigma",1.e-4,0,0,1.e-3);
set_par("g*(5).norm",1.e-4,0,0,1.e-2);

subplex;          % Let's do the initial fit with subplex ...
() = fit_counts;
mpfit;            % ... and then the error bars with mpfit
(,) = conf_loop(,1,0.01;save,prefix="edgeII.");
() = system("more edgeII.save");

variable info_strt_II;
() = eval_counts(&info_strt_II);

save_par("edgeII.par");
plot_unfold({[1,2,3,4]},popt;res=5);
```

The results of the above error bar search suggest that this fifth line is indeed significant – it's 90% confidence value lower limit for the line flux is well above zero. But should we believe that? At what point do we start worrying that we have just fit a random noise fluctuation with a narrow gaussian? (Narrow gaussians will probably describe well any noise fluctuation that's only a few bins wide.) Here is where simulations can be very useful.

The idea is that we take the model parameters from our fit with only four lines, simulate data of the same exposure as our real data, use the same grouping/noticing criteria, then fit this data with the five line model. We then store the difference in χ^2 values, and repeat many, many times. We then histogram our results, and see how many times the simulated data (which we *know* has only four lines) yields an improvement in χ^2 as large as the one we found with the real data. (Those with a good statistics background will already note some objections to even this scenario. We discuss some of these further below.)

To obtain the most meaningful results for such simulations *we need to replicate our analysis procedures as closely as possible* (and ideally our analysis procedure should be one that is well-defined and quantitative). In this case that would mean that we fit, and then run the error bar search to guarantee that we have found the best fit. That's going to be very time consuming. As a compromise, we will run the fits with `subplex` (which already will be slow enough). This is a good "first cut", designed to see if the fifth line has a chance of remaining significant. Before publishing the results, we would likely increase the fidelity of the simulations.

10. Run a script to evaluate these Monte Carlo simulations. You first have to delete the real data. Then assign the HETG response matrices to "blank" data sets. This tells ISIS that these data IDs will be used for fake datasets. As before, match the dataset grids, load the four line model parameters, then create fake data with the ISIS `fakeit` command. Group and notice these fake data exactly as we did for the real data, and fit the four line model. Store the χ^2 value. Add another line parameter as before, and fit the data. Store this χ^2 value. Repeat many times. (More than 1000 might be prohibitively long depending upon the speed of your computer. Those with slower computers might want to start with 300.) Histogram the results. How many simulations reach or exceed the χ^2 value that we found with the real data?

```
% Create variable to hold the results:
variable info_I, info_II, ntrial=1000, delta_chi=Double_Type[ntrial];

delete_data(all_data);     % Get rid of the real data

_for i (1,4,1)
{
   assign_arf(i,i);        % Assign the response matrices to
   assign_rmf(i,i);        % "blank" data sets
}
match_dataset_grids(3,1,2);

subplex;                   % Go back to the subplex fitting method

Fit_Verbose=-1;  % Keep ISIS a little quieter during the fits

_for i (0,ntrial-1,1)
{
   load_par("edge.par");   % Base the fake data on these parameters
   fakeit;

   % Group and notice the data as you did above
   () = combine_datasets([1:4]);
   group([1:4];min_sn=5,min_chan=2,unit="a",bounds=1.7);
   notice_values([1:4],13.5,15;unit="a");
   () = fit_counts(&info_I);

   % Define the new fit function as above
   fit_fun("
edge*(powerlaw-gaussian(1)-gaussian(2)-gaussian(3)-gaussian(4)-gaussian(5))
   ");
   set_par("g*(5).LineE",0.859,0,0.857,0.860);
   set_par("g*(5).Sigma",1.e-4,0,0,1.e-3);
   set_par("g*(5).norm",1.e-4,0,0,1.e-2);
   () = fit_counts(&info_II);

   % Find the chi^2 difference
   delta_chi[i]=info_I.statistic-info_II.statistic;
}

% Histogram the results with the ISIS histogram function

variable lo,hi;
(lo,hi) = linear_grid(min(delta_chi),max(delta_chi),50);
variable ndelta = histogram(delta_chi,lo,hi);
hplot(lo,hi,ndelta);
```

Here's the big, obvious objection to the above. When adding the fifth line, we added it to *exactly* the same region as we did for the real data. However, we *chose* that location based upon the fact that it was the largest remaining residual in the spectrum. If it had been *some other* location with that large of a residual, we would have chosen that instead. Therefore, we really should modify the above script to repeat that procedure. First, find the largest remaining residual, then look for a line in a limited band pass around that residual. That procedure undoubtedly would increase the number of simulations with as large χ^2 changes.

In fact, one might argue that we should just run through all possible independent wavelength regions, and try adding a line. We might expect that we have ≈ 40 such regions given the energy range we allowed for the fifth line. Given these considerations, how would you expect our Monte Carlo-derived significances to change? What changes would you make to the above simulation script? How many trials would you run? (We leave these as an exercise for the reader!)