X-RAY SPECTROSCOPY: PART II - FITTING DATA

MICHAEL A. NOWAK

(MIT-CHANDRA X-RAY SCIENCE CENTER)

-WITH HELP OVER THE YEARS FROM-

JÖRN WILMS, JOHN DAVIS, JOHN HOUCK,

DAVE HEUNEMOERDER, MIKE NOBLE

OUTLINE (PART II):

- [ ] THE DEFINITION OF THE FIT EQUATION
  - [ ] THE DANGERS OF INVERTING THE EQUATION
- [ ] THE FORWARD FOLDING PARADIGM
  - [ ] DEFINING A MODEL & SETTING PARAMETERS
- [ ] STRATEGIES FOR FITTING - A WORKED EXAMPLE
- [ ] SAVING YOUR FITS
- [ ] ADVANCED STUFF

SOLVING THE FIT EQUATION:

$$C(h) \;=\; \int_0^\infty \sum_i R_i(h,E) A_i(E)\; S_i(E)\; dE\; dT \;+\; B(h)$$

REALLY JUST A MATRIX EQUATION-

$$C_h \;=\; T \sum_i \sum_E R^i_{hE}\; A^i_E\; S^i_E\; dE \;+\; B_h$$

CAN'T WE INVERT IT?

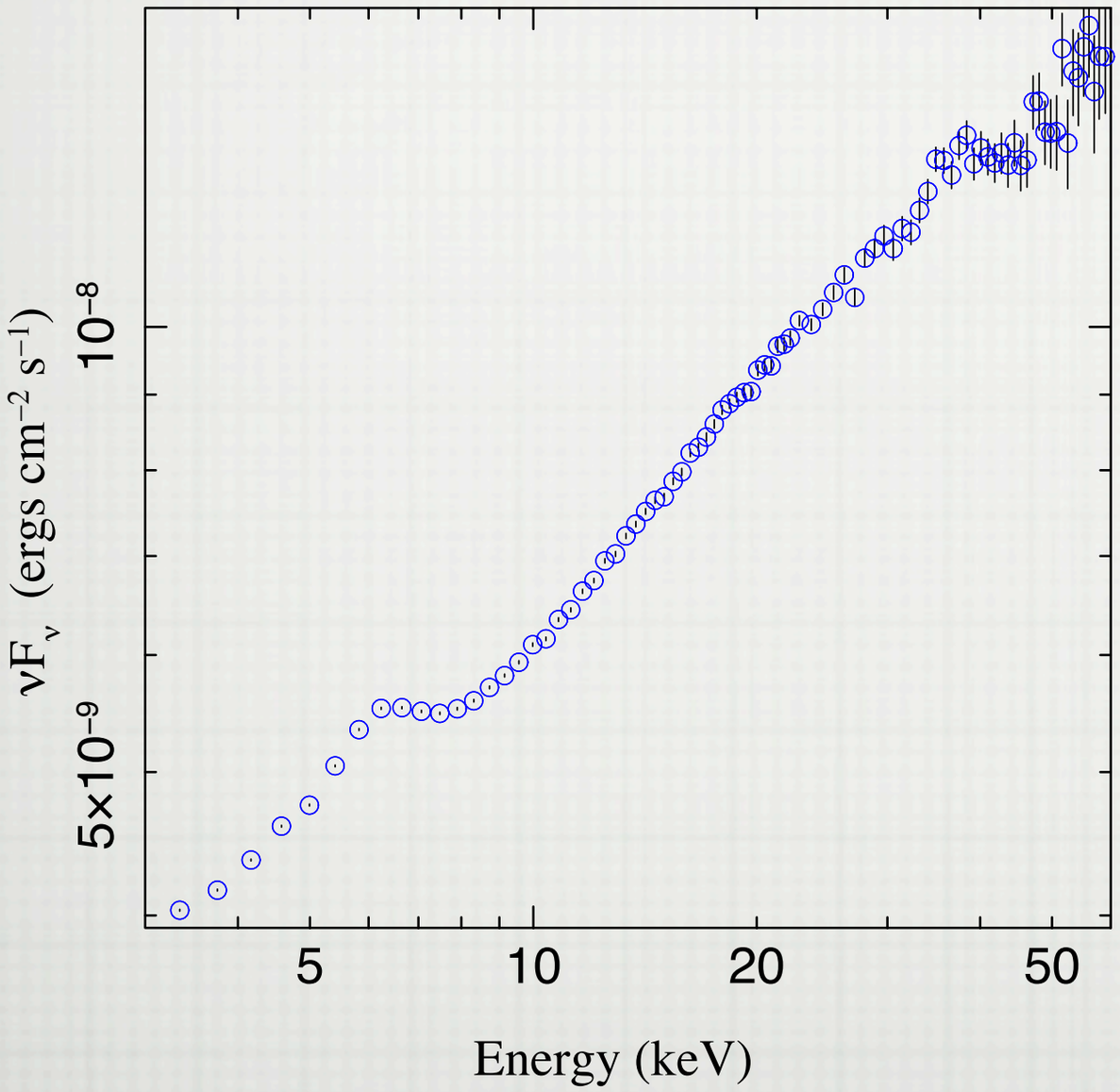
NO!
HORRIBLY
UNSTABLE!

$$R^{-1}_{hE} \frac{(C_h - B_h)}{T A_E} \;=\; S_E dE$$

NOISE

UNCERTAINTIES

WHAT ABOUT "UNFOLDING" THE SPECTRUM? DANGEROUS!!!

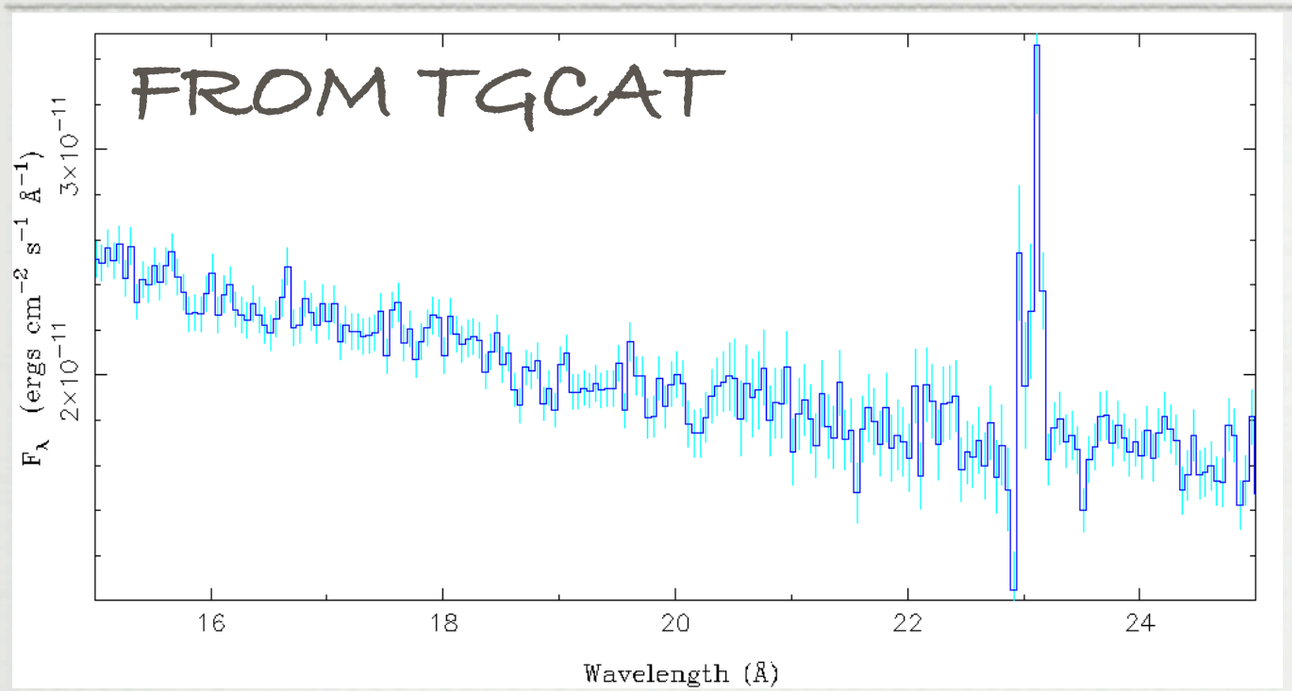


$$\mathcal{F}(h) = \frac{C(h) - B(h)}{T \sum_E R_{hE} A_E}$$

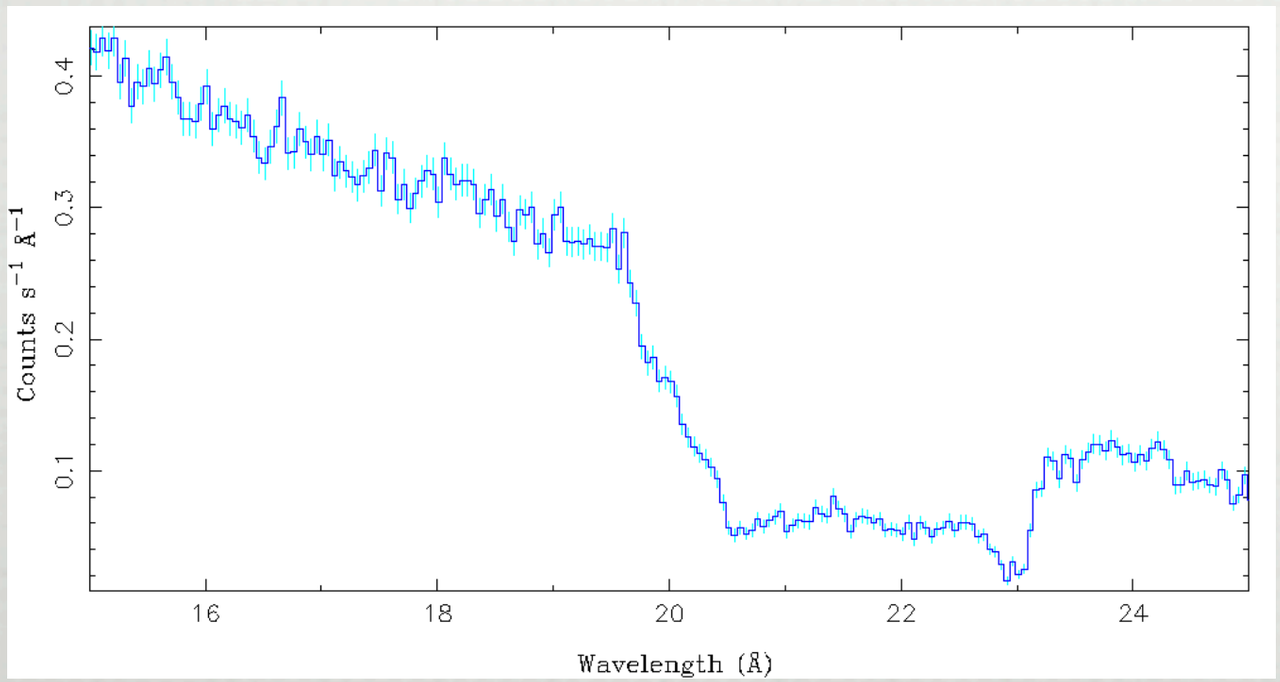
MOST "UNFOLDED" SPECTRA IN THE LITERATURE HAVE *NOT* BEEN DONE THIS WAY!
(THEY'RE EVEN MORE DANGEROUS!)

```
isis> fancy_plot_unit("kev","ergs");
isis> plot_unfold(pca;dsym=4,dcol=4,xrng={3,60});
```

UNFOLDED SPECTRA CAN BE VERY MISLEADING



THIS LOVELY "EMISSION LINE" IN XTE J1118+480



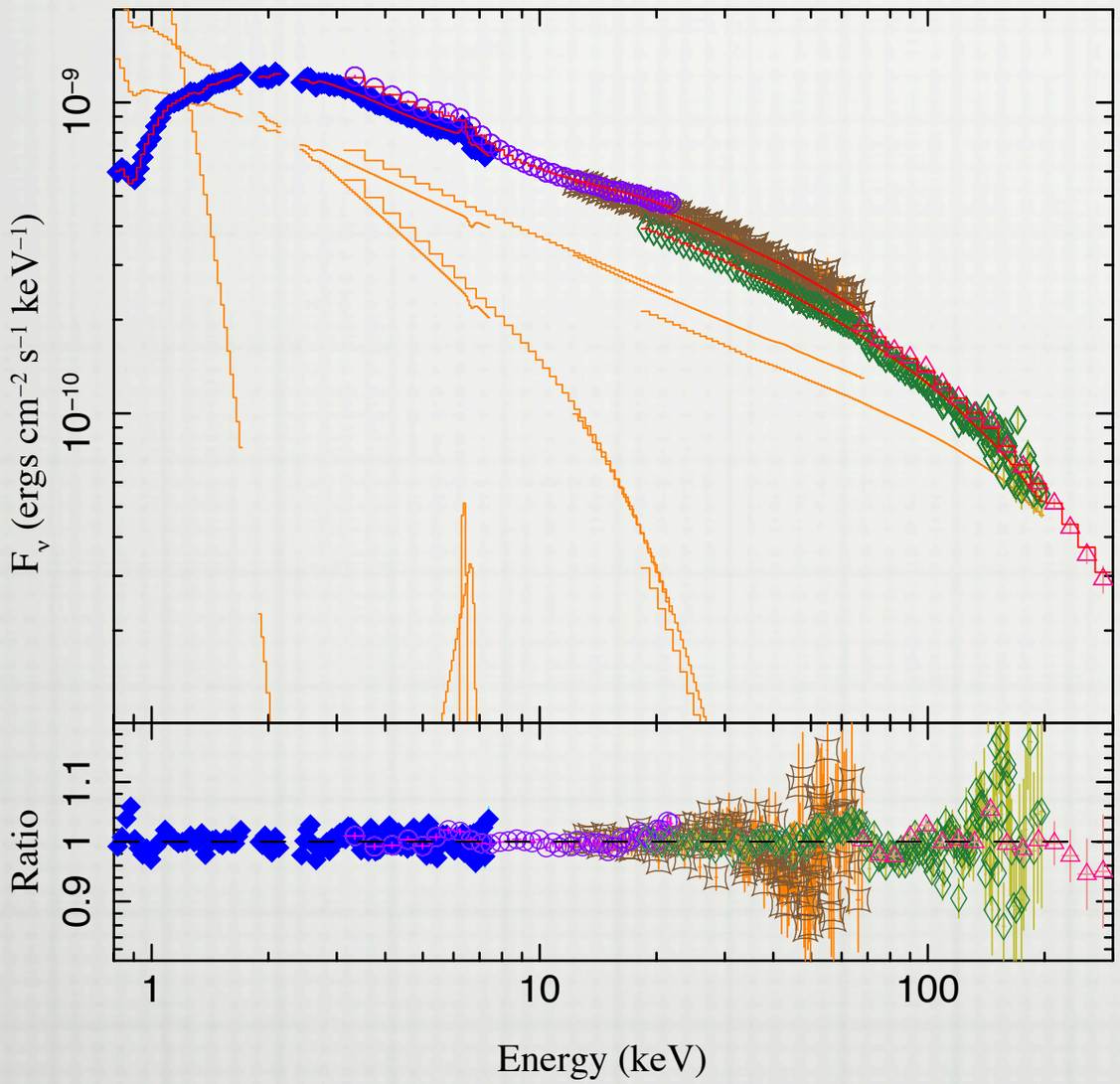
IS REALLY A DETECTOR FEATURE

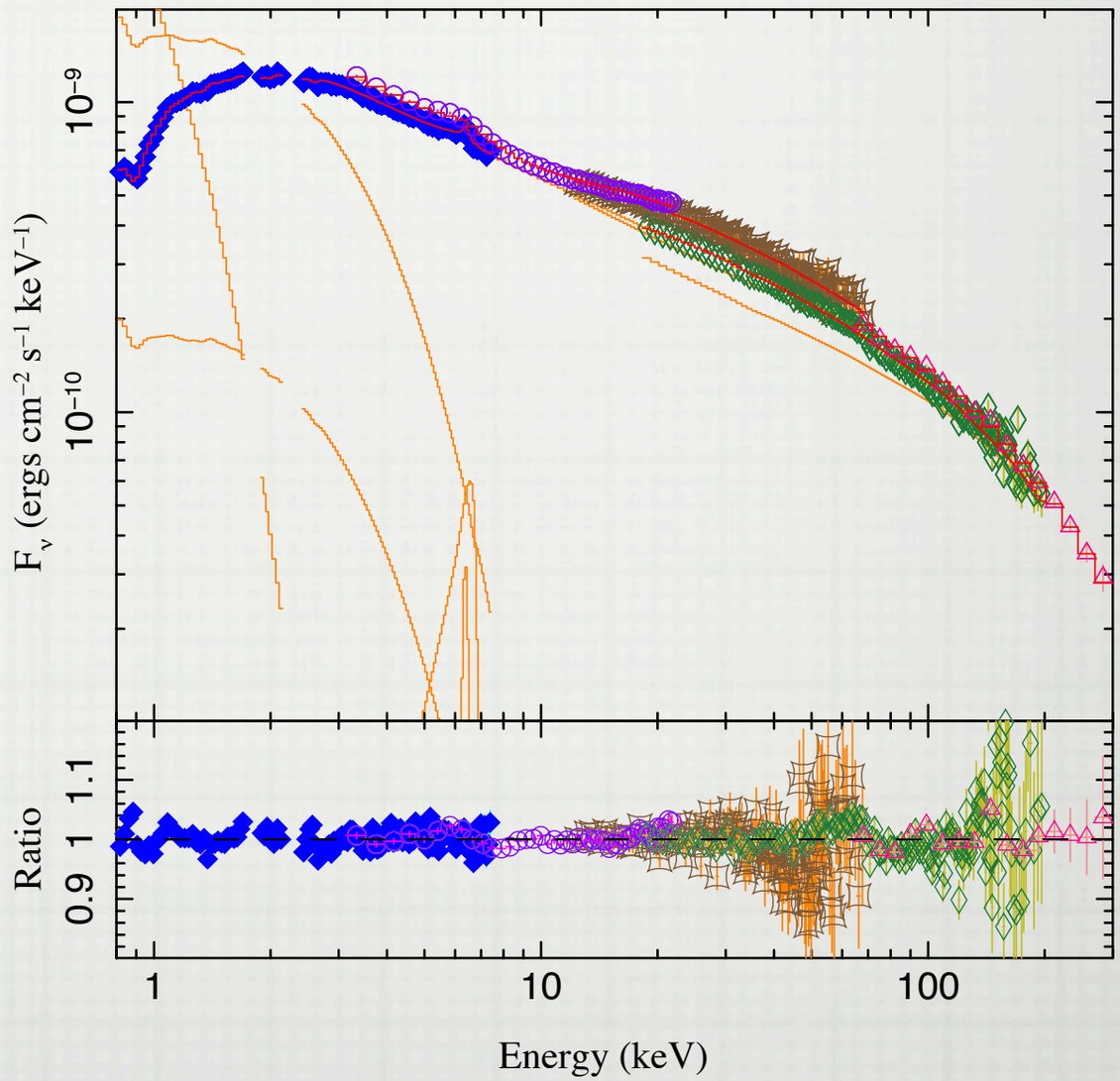
SOLVING THE FIT EQUATION:

$$\sum_h \frac{[C_h \; - \; B_h \; - \; T \; \sum_i \sum_E \; R^i_{hE} \; A^i_E \; \mathcal{F}_E(M_p)]^2}{\sigma_h^2}$$

- "FORWARD FOLDING" - VARY A SET OF MODEL PARAMETERS, CALCULATING COUNTS/BIN, AND MINIMIZE A STATISTIC - TYPICALLY CHI^2
- THIS IS ALSO NOT "UNIQUE"! DIFFERENT MODELS CAN YIELD NEARLY IDENTICAL FIT STATISTICS.

IDENTICAL DATA, DIFFERENT FITS, SIMILAR STATISTICS





STEPS FOR PERFORMING A SUCCESSFUL X-RAY ANALYSIS

- ☐ LOAD THE DATA - CHECK THAT IT IS WHAT YOU WANT!
- ☐ PLOT THE DATA - *SEE* THAT IT IS WHAT YOU WANT!
- ☐ GROUP & NOTICE THE DATA OF INTEREST. PLOT!
- ☐ DEFINE A MODEL. PUT IN "REASONABLE" STARTING PARAMETERS. *EVALUATE THE MODEL*, AND PLOT!
  - ☐ THINK! REASONABLE? IF NOT, ITERATE!

STEPS FOR PERFORMING A SUCCESSFUL X-RAY ANALYSIS

- ☐ FIT THE DATA. PLOT THE RESULTS. LOOK AT PARAMETERS.
  - ☐ THINK! REASONABLE? IF NOT, ITERATE!
- ☐ SEARCH FOR ERROR BARS. LOOK AT RESULTS.
  - ☐ THINK! REASONABLE? IF NOT, ITERATE!
- ☐ TRY OTHER MODELS. DO THEY WORK AS WELL? DO THEY ALSO MAKE PHYSICAL SENSE?

GROUPING & NOTICING

- [ ] WE SET SYSTEMATIC ERRORS, & GROUP THE DATA

```
isis> set_sys_err_frac(pca,Double_Type[129]+0.005);
isis> group(pca;min_sn=5,bounds=3,unit="kev");
isis> notice_values(pca,3,20;unit="kev");
```

- [ ] GROUP CAN GET FANCY FOR DATA ON THE SAME GRID:

```
isis> group([1,2];min_sn={5,8},min_chan={1,4},
                  bounds={3,15},unit="kev",sn_data=1);
isis> notice_values([1,2],3,5,8,20;unit="kev");
```

- [ ] EXCLUDE/INCLUDE DATA WITHOUT CHANGING BINS"

```
isis> exclude(1);  % ignore dataset #1
isis> include(1);  % bring back with previous binning
```

AREN'T SYSTEMATIC ERRORS *EVIL*?

- ☐ YES! BUT SOMETIMES IT'S ALL WE'VE GOT.
- ☐ PEOPLE ARE WORKING ON FANCY METHODS WITH BAYESIAN STATS, MARKOV CHAIN MONTE CARLO, ...
  - ☐ USEFUL IN SOME SITUATIONS, BUT ...
  - ☐ THEY PRESUME A "DEFINABLE", CORRELATED BUT RANDOM UNCERTAINTY
  - ☐ PRESUME FITS TO AN ISOLATED DATA SET
- ☐ NEITHER IS TRUE FOR PCA+HEXTE: SYSTEMATICS DOMINATE THE FIRST, BUT NOT THE LATTER

HOW DOES THE FIT FUNCTION WORK IN ISIS?

```
fit_fun("convolution(#, funct1(#)*(funct2(#)+funct3(#)))
        + funct4(#)+...");
```

- ISIS TREATS FUNCTIONS AS VECTORS OF VALUES. ANYTHING THAT MAKES MATH SENSE "WORKS".
- MODELS ARE (USUALLY) COUNTS/BIN, SO MATH SENSE IS NOT NECESSARILY PHYSICAL SENSE!

HOW DOES THE FIT FUNCTION WORK IN ISIS?

```
fit_fun("convolution(#, funct1(#)*(funct2(#)+funct3(#)))
        + funct4(#)+...");
```

- #'S ARE USED TO IDENTIFY AN "INSTANCE" OF A MODEL. THUS YOU CAN HAVE GAUSSIAN(1), GAUSSIAN(5), ... THE NUMBERS KEEP THE INSTANCES DISTINCT. EACH CAN HAVE THEIR OWN PARAMETERS.
- "WILD CARD" CHARACTERS CAN BE USEFUL HERE:

  ```
  set_par("gaussian(*).Sigma",0.1,0,0,0.5);
  ```

- PARAMETER NAMES ARE CASE SENSITIVE!

DEFINING THE FIT FUNCTION

```
isis> fit_fun("phabs(1)*powerlaw(1)");
isis>
isis> list_par; % Or just list_free to see the unfrozen parameters.
phabs(1)*powerlaw(1)
 idx  param                tie-to  freeze          value          min          max
  1   phabs(1).nH                0       0              1            0       100000  10^22
  2   powerlaw(1).norm           0       0              1            0        1e+10
  3   powerlaw(1).PhoIndex       0       0              1           -2            9
isis>
isis> % Could use edit_par; to use an editor window, or, on the command line ...
isis> set_par(1,0.6,-1); % Or: set_par("phabs(*).nH",0.6,-1);
isis> set_par(3,1.7,0,1,3);
isis>
isis> () = renorm_counts;
 Parameters[Variable] = 3[1]
             Data bins = 45
          Chi-square = 2210.172
   Reduced chi-square = 50.23119
```

- ☐ USE YOUR KNOWLEDGE OF PHYSICS AND THE PROPERTIES OF THE SOURCE TO GUIDE YOUR INITIAL PARAMETER GUESSES!

HOW DOES THE FIT FUNCTION WORK IN ISIS?

```
fit_fun("convolution(#, funct1(#)*(funct2(#)+funct3(#)))
        + funct4(#)+...");
```

- YOU CAN RENAME ANY MODEL WITH: `alias_fun("old","new");`
- CONVOLUTION MODELS HAVE A UNIQUE SYNTAX.
  - REFLECTION, RELATIVISTIC SMEARING = CONVOLUTION
- THE UNIQUE ISIS VARIABLE, `Isis_Active_Dataset`, CAN BE USED TO IDENTIFY & ISOLATE MODEL "INSTANCES".
  - `Isis_Active_Dataset` = # OF DATA SET BEING EVALUATED.
  - `gaussian(Isis_Active_Dataset)` = `gaussian(1)` FOR DATA SET 1, = `gaussian(2)` FOR DATA SET 2, ETC.

HOW DOES THE FIT FUNCTION WORK IN ISIS?

```
fit_fun("convolution(#, funct1(#)*(funct2(#)+funct3(#)))
        + funct4(#)+...");
```

- THE FUNCTIONS WILL BE EVALUATED ON THE *INTERNAL* ARF GRID (USUALLY FINER THAN THAT FOR PHA).
  - YOU CAN FORCE ISIS TO USE ANY GRID & REBIN TO ARF
- ISIS WILL LOOK AT THE RMF AND ONLY EVALUATE AT ENERGIES WHERE E->PHA BINS THAT ARE "NOTICED".
  - YOU CAN FORCE ISIS TO EVALUATE A WIDER RANGE - SOMETIMES NECESSARY FOR CONVOLUTION MODELS.
- ISIS EVALUATES WHOLE MODEL FOR *EACH* DATA SET

THINGS YOU CAN DO WITH PARAMETERS

- YOU CAN FREEZE OR THAW ANY FIT PARAMETER:

```
isis> set_par("phabs(1).nH",0.6,1);      % 1 Freezes
isis> set_par("phabs(1).nH",0.6,0);      % 0 Thaws
isis> freeze("phabs(1).nH"); freeze(1); % Either works
isis> thaw("phabs(1).nH"); thaw(1);      % Either works
```

- YOU CAN TIE PARAMETER VALUES TOGETHER.

```
isis> tie(1,2,3,4);      % Tie parameters 2-4 -> 1
isis> untie(2,3,4);      % Untie parameters 2-4
```

THINGS YOU CAN DO WITH PARAMETERS

- ANY PARAMETER CAN BE MADE AN ARBITRARY FUNCTION OF ANY OTHER PARAMETER(S)!

```
isis> set_par_fun("phabs(1).nH","sin(powerlaw(1).norm)+2");
isis> set_par_fun(1,"sin(_par(2))+2");
isis> set_par_fun(1,NULL);     % That was stupid, let's not!
```

- USE CONSTANT AS A CONVENIENT DUMMY PARAMETER

```
isis> fit_fun("gaussian(1)+gaussian(2)+0*constant(1)");
isis> set_par_fun("gaussian(2).LineE",
                  "gaussian(1).LineE+constant(1).factor");
```

GETTING ON WITH THE FITTING

- ☐ START OFF WITH "LEAST IMPORTANT" AND "NARROW BAND" PARAMETERS FROZEN.
- ☐ LIMIT PARAMETER RANGES TO "SENSIBLE" VALUES.
- ☐ TRY AN OVERALL RE-NORMALIZATION FIRST.
- ☐ FIT! CHOOSE FROM SEVERAL OPTIMIZATION METHODS:

```
isis> set_fit_method("lmdif");      % Default. Good & fast.
isis> set_fit_method("marquardt"); % Sometimes works better
isis>                               % than lmdif.  Fast.
isis> set_fit_method("subplex");    % Very slow, but can find
isis>                               % minima that others miss
isis> set_fit_method("plm");        % parallelized Levenberg-
                                    % Marquardt for multi-core
                                    % machines
```

THINGS TO KNOW ABOUT FITTING METHODS

```
isis> set_fit_method("lmdif");
isis> set_fit_method("marquardt");
```

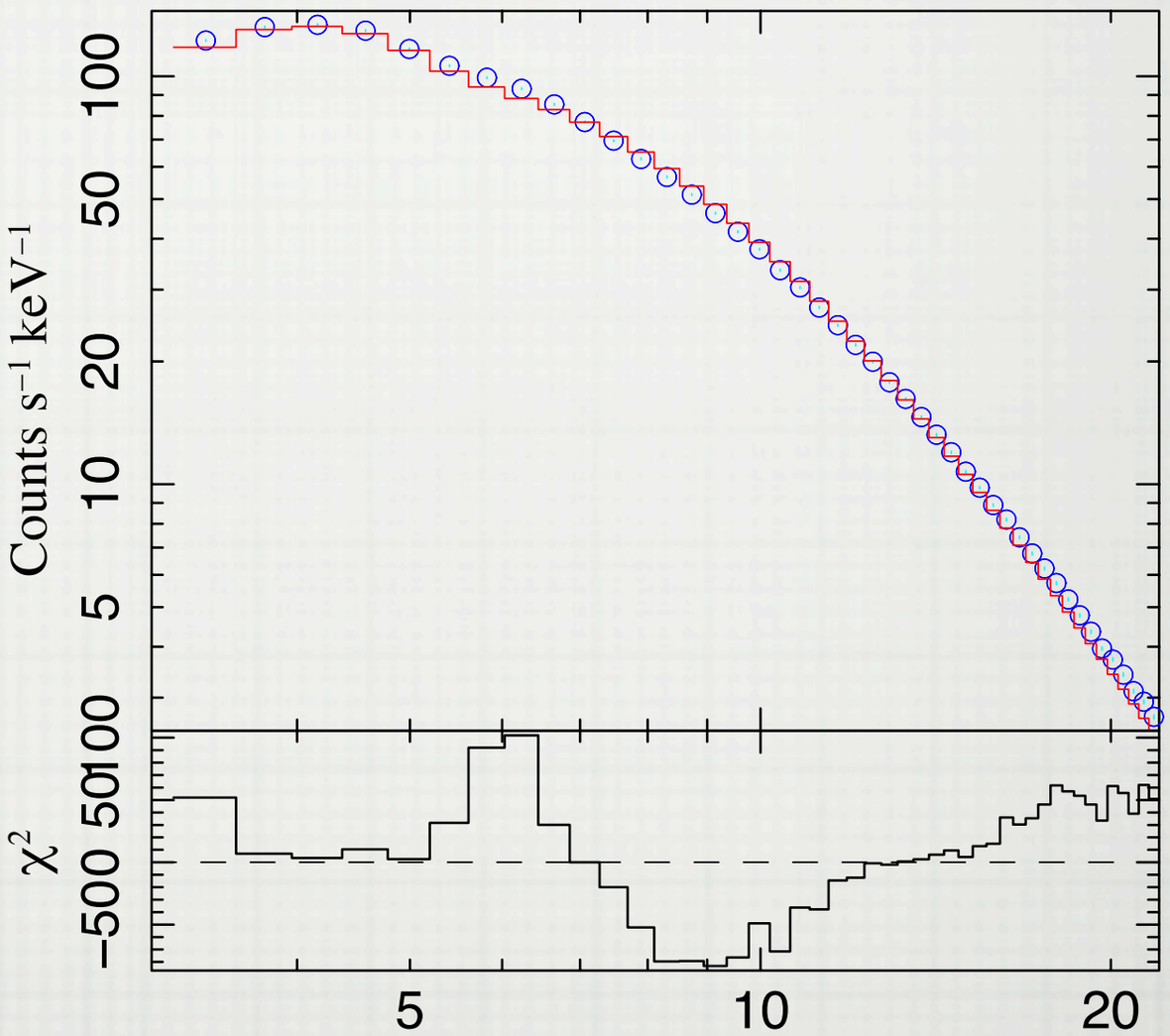
- LEVENBERG-MARQUARDT METHODS EXPECT CHI²-TYPE STATISTICS, I.E.: $\sum_i (D_i - M_i)^2 / \sigma_i^2$

```
isis> set_fit_method("subplex");
```

- SUBPLEX CAN BE USED ON CASH STATISTICS. (NOTE- ISIS CASH IS THE SAME AS XSPEC/SHERPA CSTAT.)

GETTING ON WITH THE FITTING

UNTIL YOU
REALLY KNOW
WHAT YOU'RE DOING,
PLOT RESIDUALS
WITH CHI²!

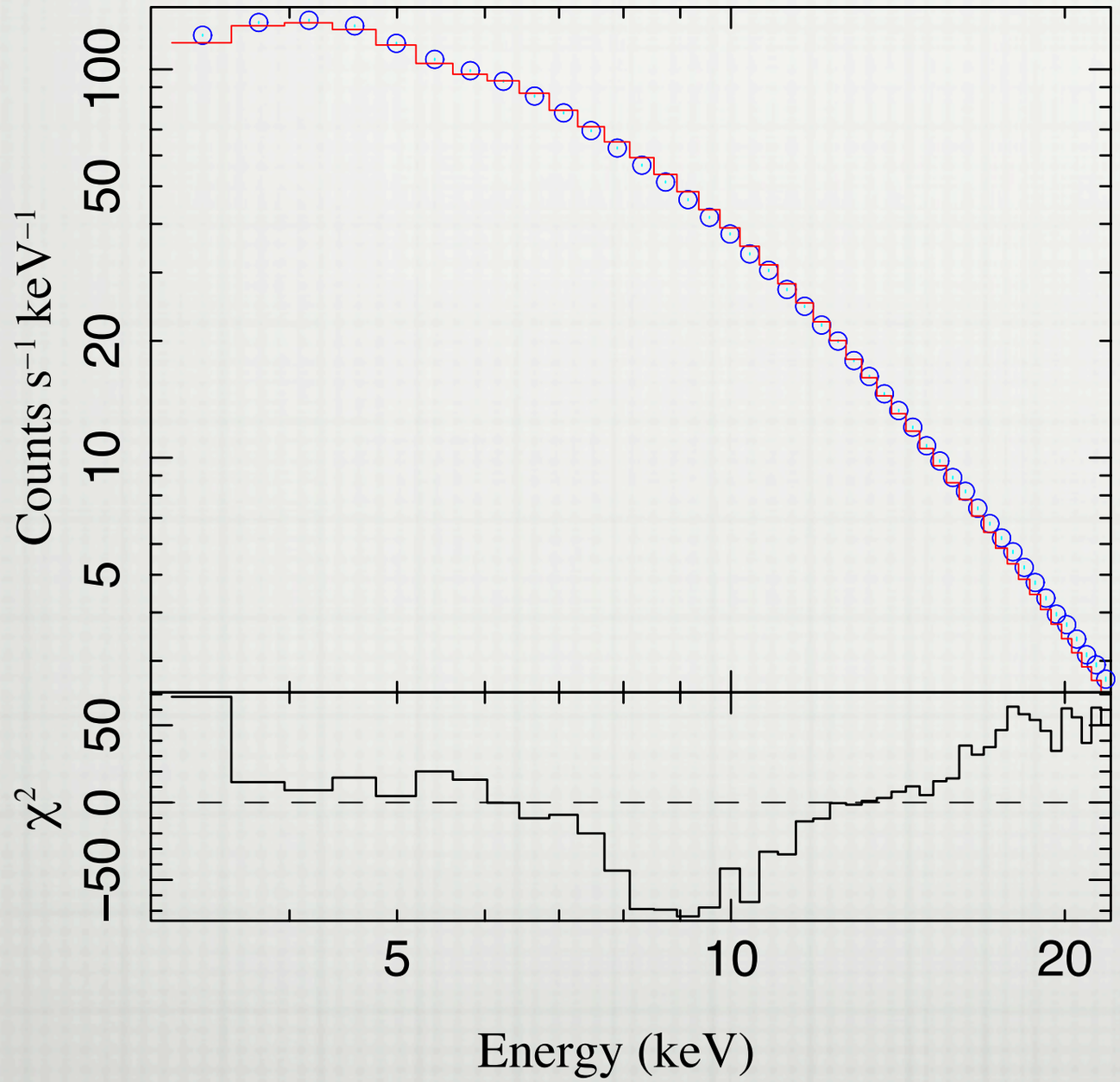



```
isis> () = fit_counts;
 Parameters[Variable] = 3[2]
             Data bins = 45
          Chi-square = 672.4607
   Reduced chi-square = 15.63862

isis> plot_data(pca;dsym=4,dcol=4,decol=5,xrng={3,22},res=2);
```

LINE? ADD ONE AND REFIT

```
isis> fit_fun("phabs(1)*(powerlaw(1)+gaussian(1))");
isis> list_par;
phabs(1)*(powerlaw(1)+gaussian(1))
 idx  param                tie-to  freeze          value          min          max
  1   phabs(1).nH               0     1                0.6            0       100000  10^22
  2   powerlaw(1).norm          0     0           1.539388            0        1e+10
  3   powerlaw(1).PhoIndex      0     0           1.588304            1            3
  4   gaussian(1).norm          0     0                  1            0        1e+10
  5   gaussian(1).LineE         0     0                6.5            0      1000000  keV
  6   gaussian(1).Sigma         0     0                0.1            0           10  keV
isis> set_par("gaussian(1).LineE",6.4,1,6,7);
isis> set_par("gaussian(1).norm",0.1,0,0,1);
isis> () = fit_counts;
 Parameters[Variable] = 6[3]
            Data bins = 45
          Chi-square = 623.9285
   Reduced chi-square = 14.85544
isis> thaw(6);
isis> () = fit_counts;
 Parameters[Variable] = 6[4]
            Data bins = 45
          Chi-square = 618.4107
   Reduced chi-square = 15.08319
isis> plot_data(pca;dsym=4,dcol=4,decol=5,xrng={3,22},res=2);
```

SOME, BUT NOT ENOUGH, IMPROVEMENT

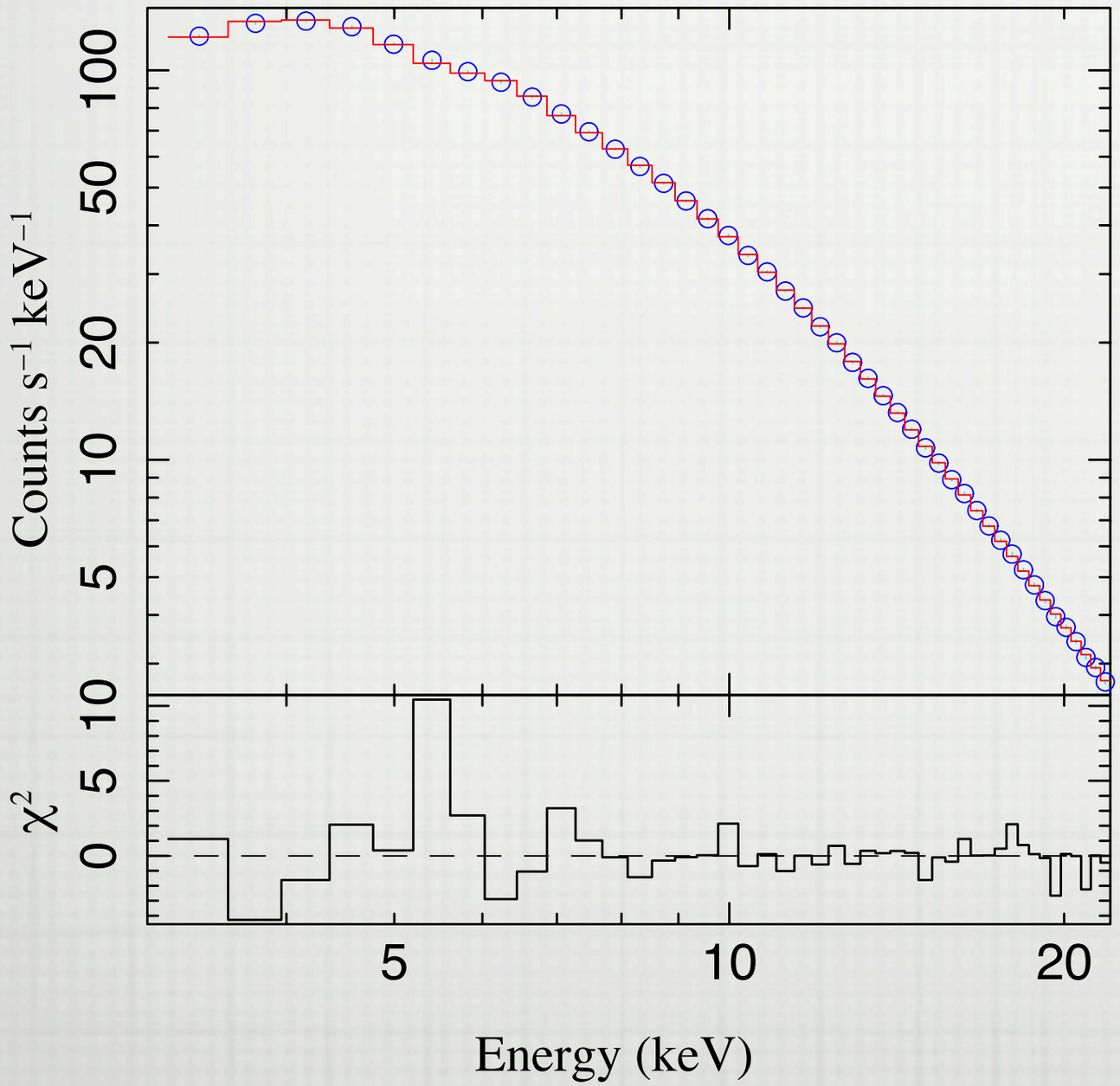


THE RESIDUALS SEEM TO INDICATE THAT A BREAK IN THE POWERLAW WOULD IMPROVE THE FIT.

LET'S ADD ONE.

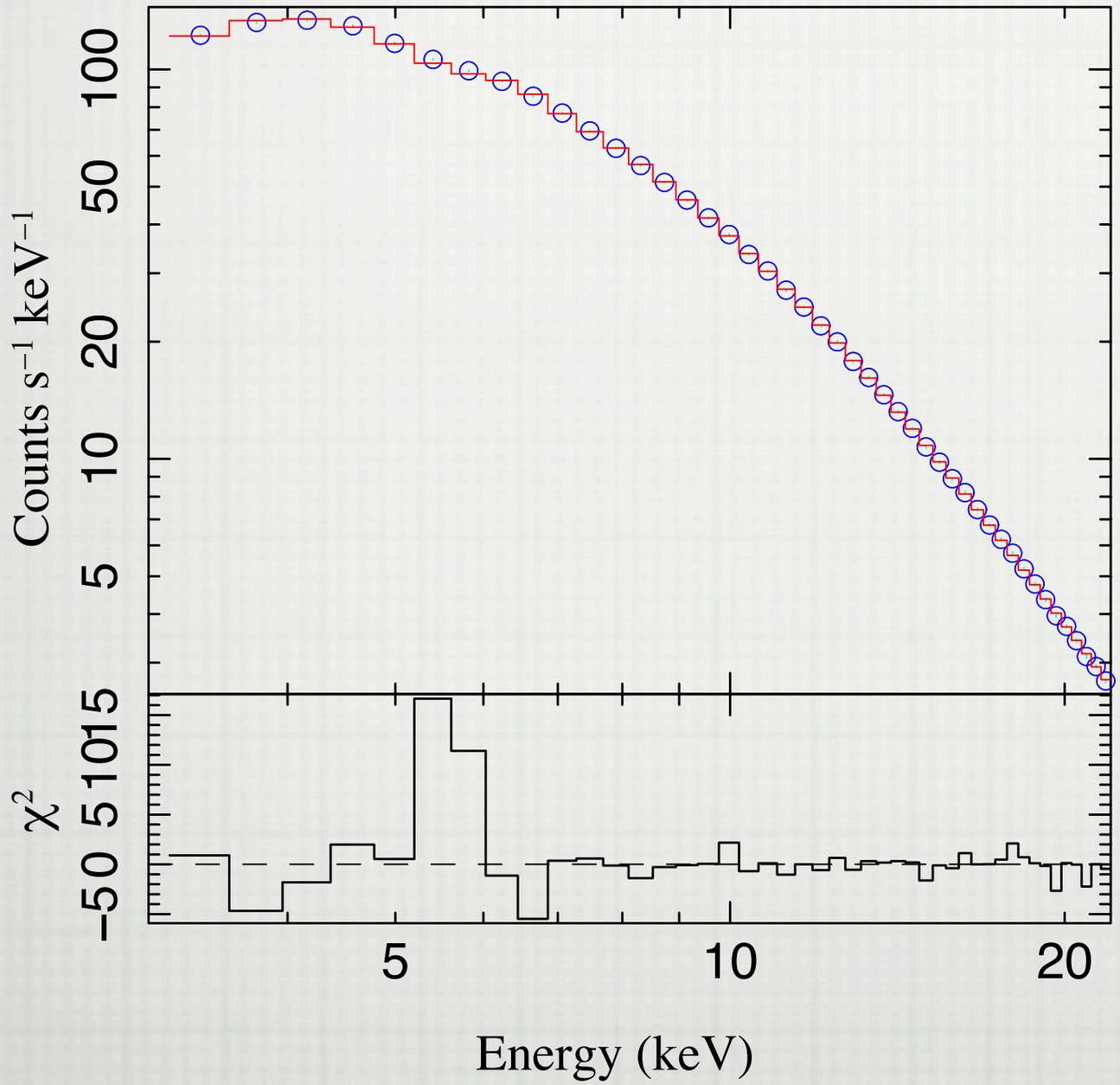
```
isis> fit_fun("phabs(1)*(bknpower(1)+gaussian(1))");
isis> list_par;
phabs(1)*(bknpower(1)+gaussian(1))
 idx  param                tie-to  freeze          value          min          max
  1   phabs(1).nH              0     1                0.6            0       100000  10^22
  2   bknpower(1).norm         0     0                  1            0        1e+10
  3   bknpower(1).PhoIndx1     0     0                  1           -2            9
  4   bknpower(1).BreakE       0     0                  5         0.01      1000000  keV
  5   bknpower(1).PhoIndx2     0     0                  2           -2            9
  6   gaussian(1).norm         0     0        0.006378002            0            1
  7   gaussian(1).LineE        0     1                6.4            6            7  keV
  8   gaussian(1).Sigma        0     0      5.283852e-05            0            1  keV
isis>
isis> set_par(2,1.5,0,0.1,10);  % bknpower has same normalization as powerlaw
isis> set_par(3,1.6,0,1,3);     % slope wasn't too far off
isis> set_par(4,10,0,8,13);     % break looks to be around 9 or 10 keV
isis> set_par(5,1.4,0,1,3);     % second power law looks "harder"
isis>
isis> () = renorm_counts;
 Parameters[Variable] = 8[1]
             Data bins = 45
          Chi-square = 397.054
   Reduced chi-square = 9.023955
isis>
isis> () = fit_counts;
 Parameters[Variable] = 8[6]
             Data bins = 45
          Chi-square = 23.81623
   Reduced chi-square = 0.6106725
isis> plot_data(pca;dsym=4,dcol=4,decol=5,xrng={3,22},res=2);
```

GETTING THERE, BUT WE MAY BE STUCK IN A LOCAL MINIMUM



ERROR BAR SEARCHES ARE OFTEN A GOOD WAY TO GET "UNSTUCK"

MUCH IMPROVEMENT, BUT SOME RESIDUALS REMAIN

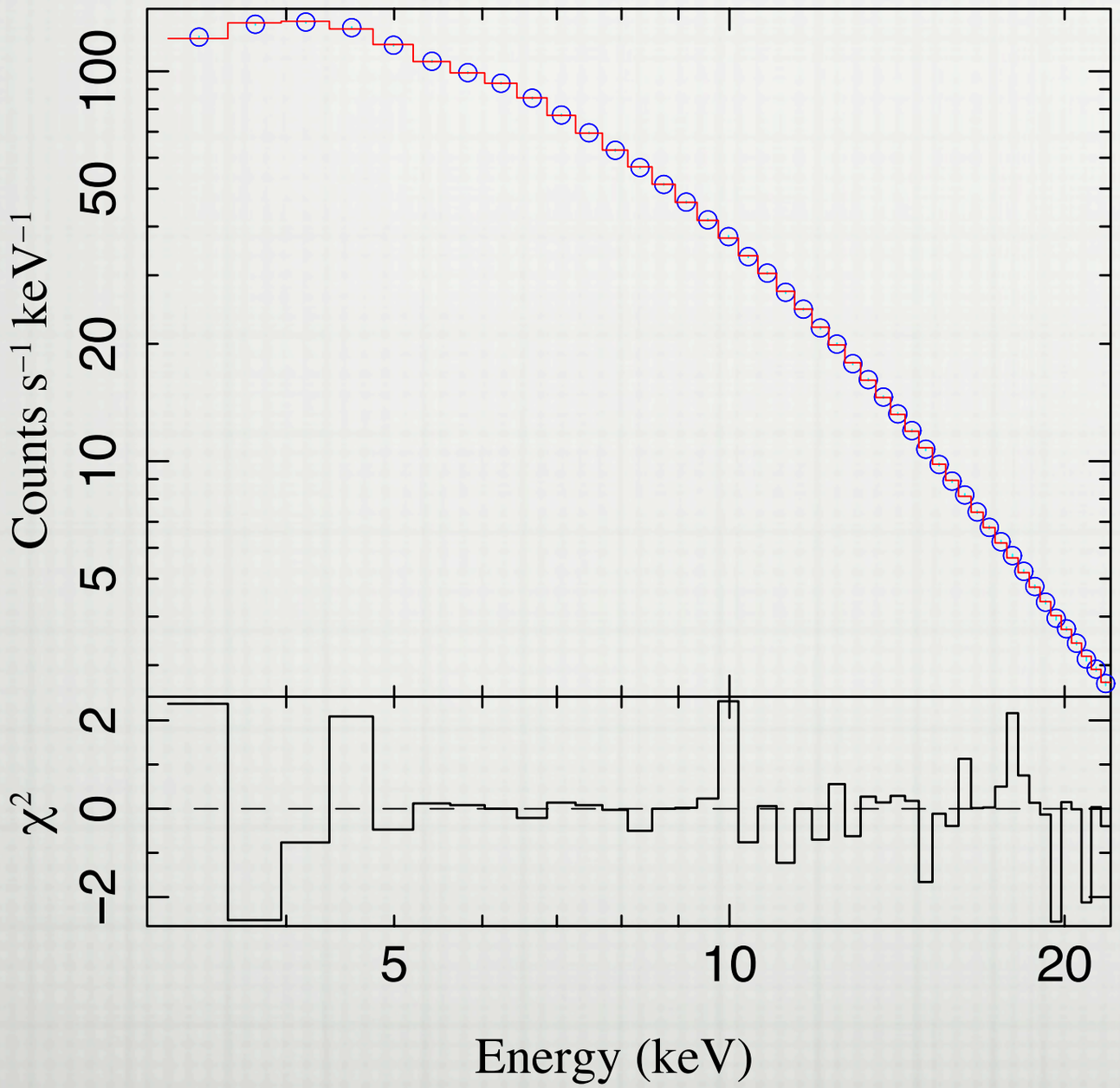


WE HAD FROZEN THE ENERGY OF THE LINE. IMPROVEMENT IF WE LET IT GO FREE?

```
isis> thaw("gaussian(1).LineE");
isis> () = fit_counts;
 Parameters[Variable] = 8[7]
            Data bins = 45
          Chi-square = 23.81605
   Reduced chi-square = 0.6267381
isis> list_free;
phabs(1)*(bknpower(1)+gaussian(1))
 idx  param              tie-to  freeze          value          min          max
  2  bknpower(1).norm        0     0           1.77756          0.1           10
  3  bknpower(1).PhoIndx1    0     0          1.677097            1            3
  4  bknpower(1).BreakE      0     0           9.76005            8           13  keV
  5  bknpower(1).PhoIndx2    0     0          1.408868            1            3
  6  gaussian(1).norm        0     0      0.008118369            0            1
  7  gaussian(1).LineE       0     0                6.4            6            7  keV
  8  gaussian(1).Sigma       0     0     5.283852e-05            0            1  keV
isis>
isis> set_fit_method("subplex");
isis> () = fit_counts;
 Parameters[Variable] = 8[7]
            Data bins = 45
          Chi-square = 20.52235
   Reduced chi-square = 0.5400618
isis>
isis> set_fit_method("lmdif");
isis> () = fit_counts;
 Parameters[Variable] = 8[7]
            Data bins = 45
          Chi-square = 20.39472
   Reduced chi-square = 0.5367033
isis> plot_data(pca;dsym=4,dcol=4,decol=5,xrng={3,22},res=2);
```

```
isis> conf(8);  % Can search for error bars individually
**** Parameter range endpoint 0 is inside the confidence limit
**** Lower confidence limit didn't converge[8]:  allow wider parameter ranges?
Found improved fit, stat= 15.072 for param[8] = 0.500079
**** Found improved fit
0.500079
0.500079
isis>
isis> (,) = conf_loop(,1,0.1;save,prefix="bknpower_gaussian.");
Found improved fit, stat= 14.6825 for param[2] = 1.76451
**** Found improved fit
 Parameters[Variable] = 8[7]
            Data bins = 45
          Chi-square = 14.62909
   Reduced chi-square = 0.3849759
isis>
isis> !more bknpower_gaussian.save
phabs(1)*(bknpower(1)+gaussian(1))
 idx  param                tie-to  freeze          value          min          max
  1  phabs(1).nH                0     1              0.6            0       100000  10^22
  2  bknpower(1).norm           0     0         1.769163     1.736502     1.803789
  3  bknpower(1).PhoIndx1       0     0         1.676277     1.664857     1.688937
  4  bknpower(1).BreakE         0     0         9.626676     9.136459     10.09752  keV
  5  bknpower(1).PhoIndx2       0     0         1.410334     1.393771     1.426189
  6  gaussian(1).norm           0     0       0.01083106  0.008576826   0.01347842
  7  gaussian(1).LineE          0     0         6.160142     6.028331      6.29854  keV
  8  gaussian(1).Sigma          0     0        0.4182171     0.192534    0.6204544  keV
isis>
isis> plot_data(pca;dsym=4,dcol=4,decol=5,xrng={3,22},res=2);
```

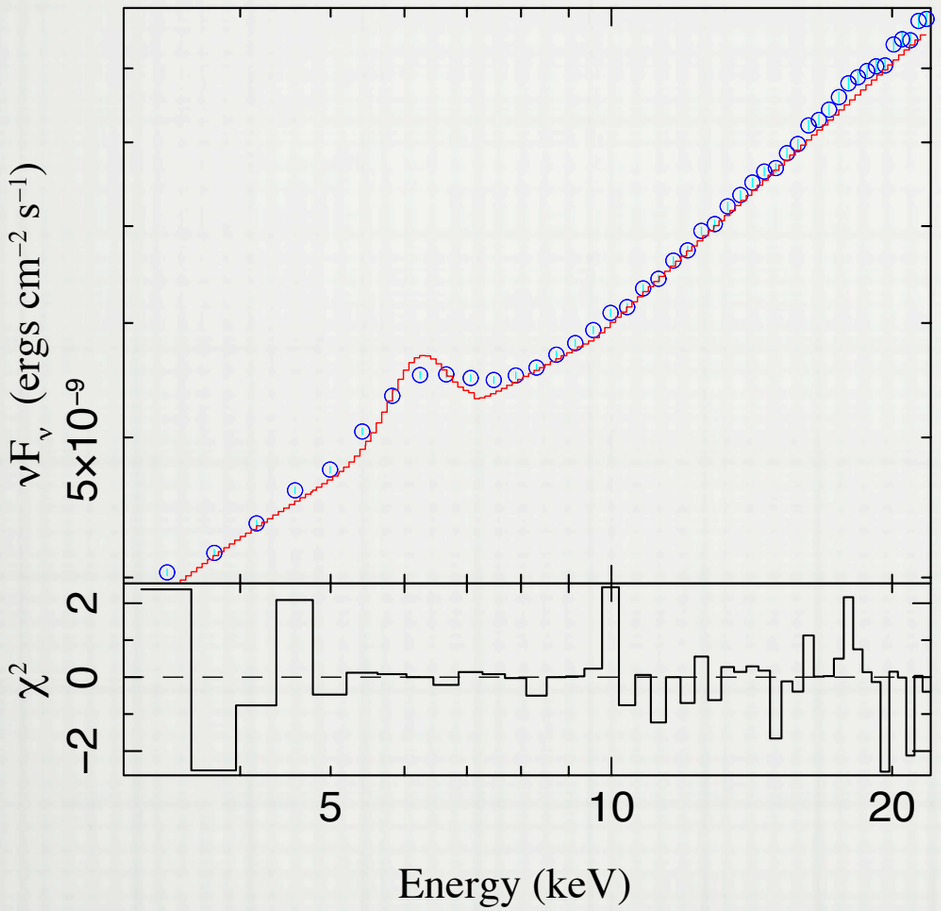
LOW CHI^2: OVERPARAMETERIZED, OR SYST ERRORS TOO LARGE

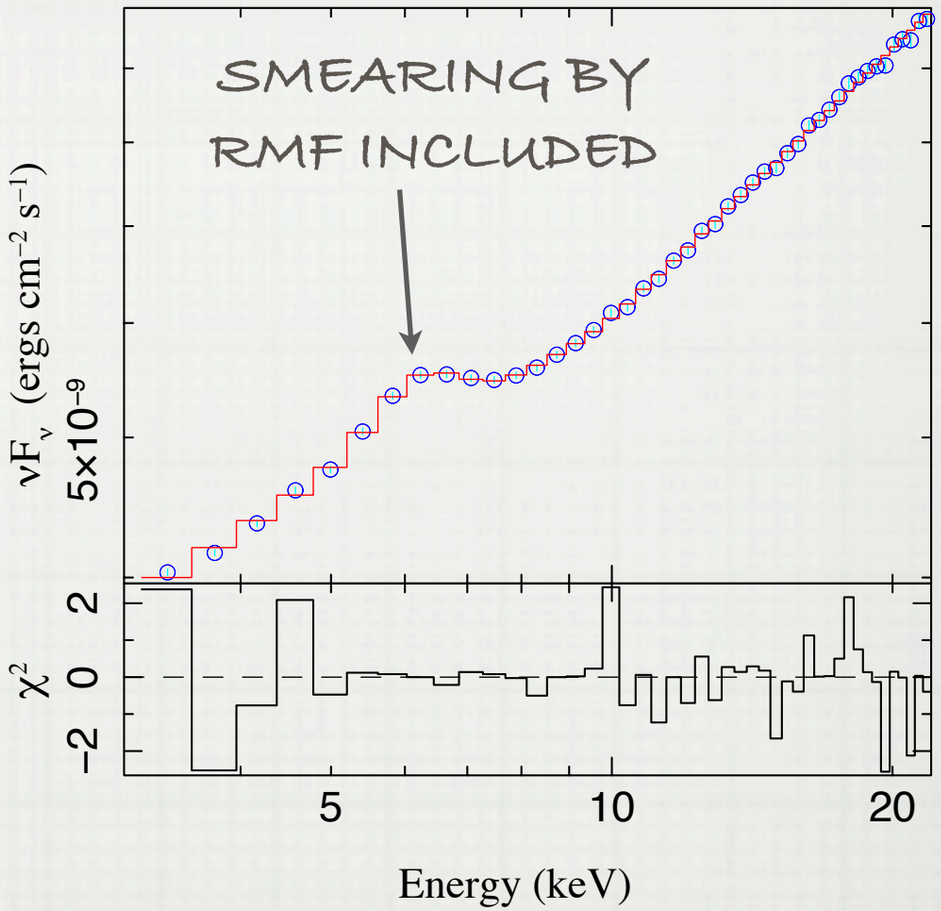


ERROR BARS IN LINE REGION A LITTLE "TOO FLAT".

NOTE ALSO THAT CUSTOM IN X-RAY ASTRONOMY IS TO QUOTE 90% CONFIDENCE LIMIT ERROR BARS.

NOW YOU ARE ALLOWED TO LOOK AT THE UNFOLDED SPECTRA






```
isis> plot_unfold(pca;dsym=4,dcol=4,decol=5,xrng={3,22},res=2,
isis>                 con_mod=0);
isis> plot_unfold(pca;dsym=4,dcol=4,decol=5,xrng={3,22},res=2);
isis>                 con_mod=1);
```

SAVING YOUR RESULTS

- [ ] SAVING AND RELOADING FITS:

```
isis> save_par("my_model.par");
isis> load_par("my_model.par");
```

- [ ] SAVING YOUR PLOTS:

```
isis> id = open_plot("my_model.par");
isis> resize(21,0.85);
isis> plot_data(1,popt);
isis> close_plot("my_model.par");
```

FIT FUNCTIONS THAT YOU MIGHT FIND USEFUL

- `bbody, diskbb, cutoffpl, highecut` - BLACKBODY, DISK, POWERLAW WITH A ROLLOVER, AN EXPONENTIAL ROLLOVER (MULTIPLIES OTHER MODELS)

- `comptt, pexrav` - FOR THE MORE ADVENTUROUS. COMPTONIZATION, AND A "REFLECTED" POWERLAW.

  - THE LATTER MODEL REQUIRES EVALUATION OUT TO ~1 MeV. I.E., ONE HAS TO "EXTEND THE GRID"

  - TRY FORCING THE LINE TO STAY NARROW.

- FOR A LIST OF XSPEC MODELS, SEE:
  http://heasarc.gsfc.nasa.gov/docs/xanadu/xspec/manual/XspecModels.html

ADVANCED TOPICS

- [ ] HOW DO WE CALCULATE FLUX?
- [ ] HOW DO WE CALCULATE EQUIVALENT WIDTHS?
- [ ] HOW DO WE LOOK AT CHI^2 VS. PARAMETER? (I.E., EQUIVALENT TO XSPEC'S STEPPAR COMMAND)
- [ ] HOW DO WE CALCULATE ERROR CONTOURS?
- [ ] HOW DO WE PLOT MODEL COMPONENTS INDIVIDUALLY?
- [ ] EXTENDING THE FIT FUNCTION GRID & CACHING

CALCULATING FLUX

- THREE METHODS DEFINED IN THE SCRIPTS: BASED ON DATA (`data_flux`), OR BASED ON MODEL (`model_flux, calc_flux`)

```
isis> % Just like “unfolding” spectrum
isis> data_flux(1,3,20;unit=”kev”,print);

isis> % Calculated from fit model
isis> model_flux(1,3,20;unit=”kev”,print);

isis> % Uses fit model
isis> calc_flux(1,bin_lo,bin_hi;unit=”kev”,print);
```

- THE FIRST TWO ONLY WORK WITHIN THE RANGE OF THE DATA, THE LAST CAN BE EXTENDED TO ANY ENERGY RANGE.

EQUIVALENT WIDTH

- ☐ EQUIVALENT WIDTH IS A *RELATIVE* MEASURE OF THE STRENGTH OF AN EMISSION LINE, OR THE DEPTH OF AN ABSORPTION LINE.

- ☐ IT IS THE RATIO OF (ΔFLUX)/(FLUX DENSITY), WHERE ΔFLUX=DIFFERENCE WITH/WITHOUT THE LINE, FLUX DENSITY IS EVALUATED AT THE PEAK OF THE LINE.

- ☐ MEASURED IN EV OR MILLI-ANGSTROM.

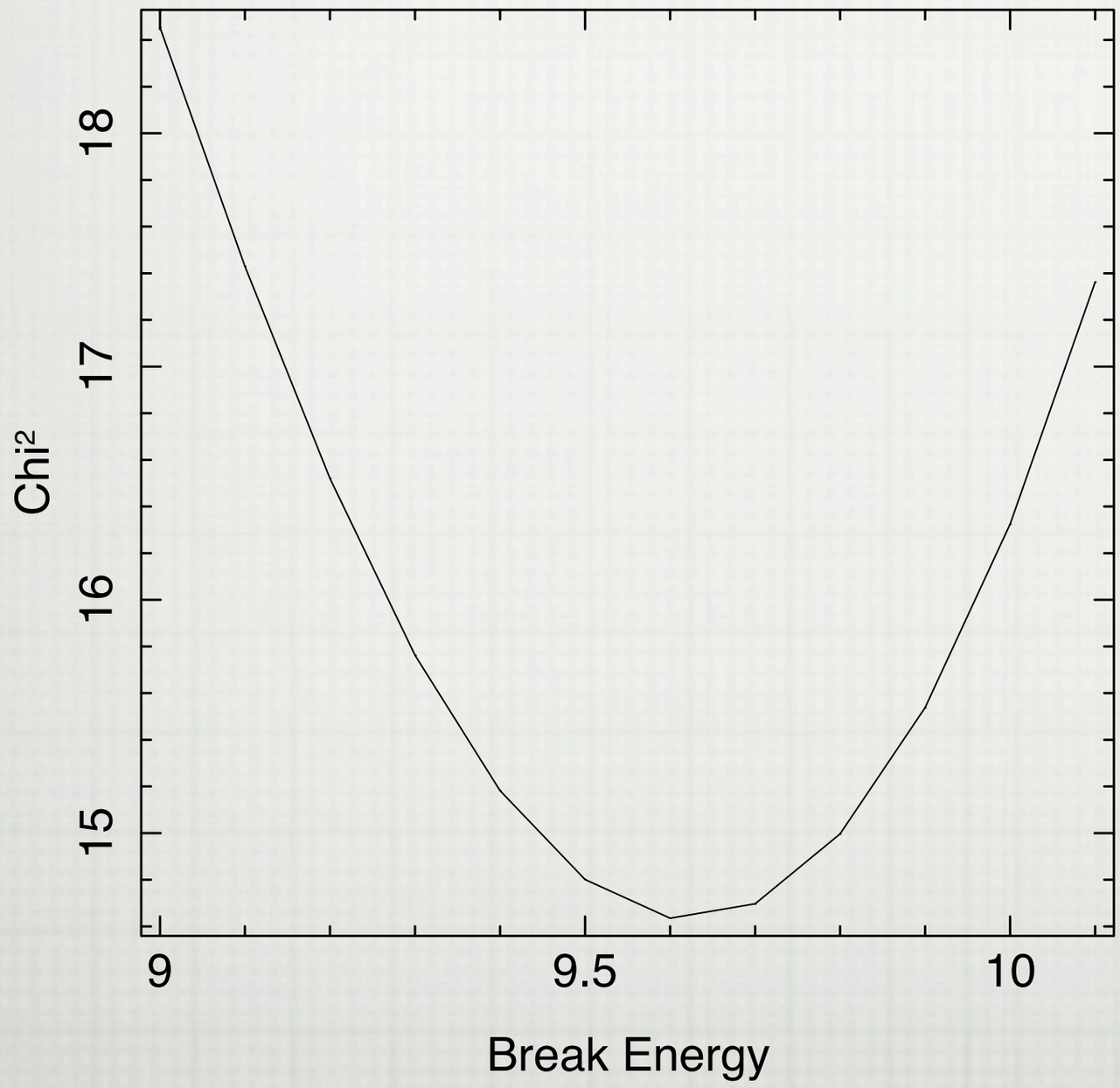
- ☐ EVALUATE MODEL, & USE THE SCRIPT FUNCTION: eqw

```
isis> (eqw_ma,eqw_ev) = eqw(pca,"gaussian(1).norm");
isis> eqw(1,5;print);  % If line norm is parameter #5
```

CHI^2 VS. PARAMETER (I.E., "STEPPAR")

```
isis> % The functions fit_counts; renorm_counts; eval_counts; optionally return info
isis> % about the statistics.  To do this, you must pass a "reference" to a variable
isis> % in which to store these statistics.  The & in front of the variable name below
isis> % says that you are passing this reference.  What you get back is a structure
isis> % with fields for statistic, num_variable_params, and num_bins.  Here's a short
isis> % script that will do what steppar does:
isis> define steppar(par,lo,hi,nsteps)
      {
         variable par_vals=[lo:hi:(hi-lo)/nsteps];
         variable info, stat, par_use;
         stat = Double_Type[length(par_vals)];
         variable i = 0;
         foreach par_use (par_vals)
         {
            set_par(par,par_use,1);  % Freeze parameter
            () = fit_counts(&info);
            stat[i] = info.statistic;
            i++;
         }
         thaw(par); () = fit_counts;
         return par_vals, stat;
      }
isis> (par_vals,stat) = steppar("bknpower(1).BreakE",9,10.2,12);
isis> xrange, yrange; xlin; ylin; xlabel("Break Energy"); ylabel("Chi\\u2\\d");
isis> plot(par_vals,stat);
```

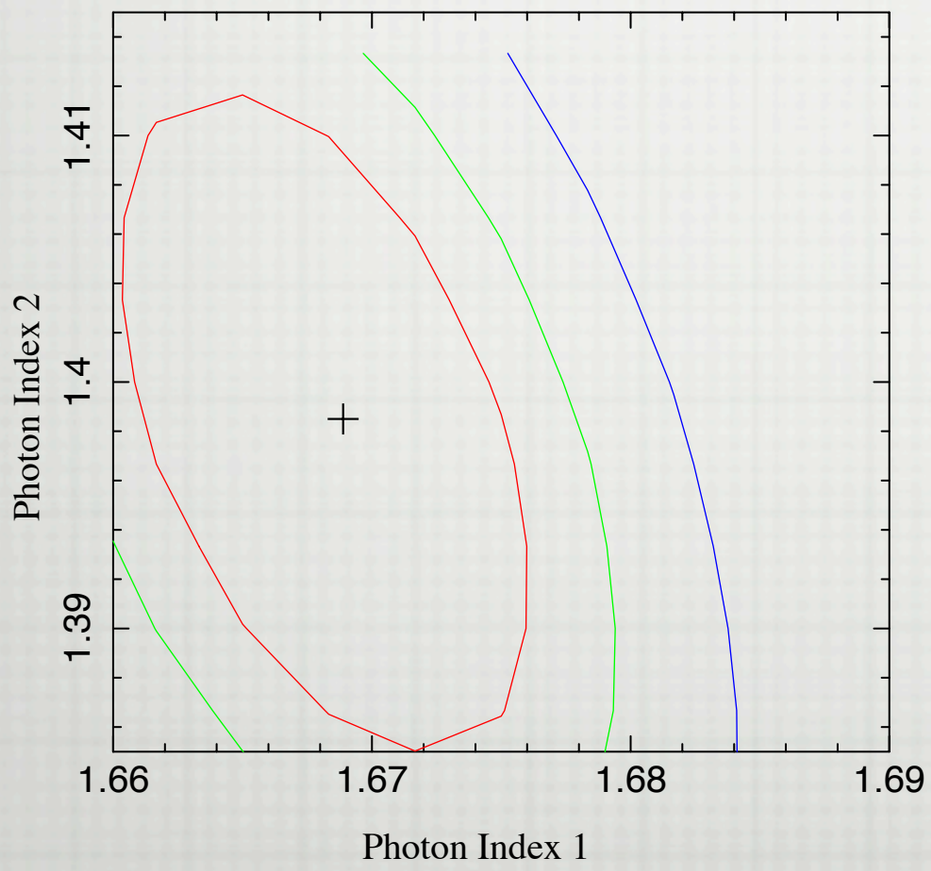
CHI^2 VS. PARAMETER (I.E., "STEPPAR")



HERE ARE THE RESULTS. THIS WAS A QUICK & DIRTY SCRIPT. WITH A LITTLE WORK, WE COULD ADD SOME BELLS & WHISTLES.

CONTOUR PLOTS

```
isis> % Confidence contours can be created with ISIS intrinsic functions.  First you
isis> % Define the x & y grids, then create the contours, then plot and/or save them
isis>
isis> x = conf_grid(3,1.66,1.69,10);   % Photon Index 1
isis> y = conf_grid(5,1.385,1.415,10); % Photon Index 2
isis> contr = conf_map_counts(x,y);    % Create the contours
isis> xlabel("\\frPhoton Index 1"); ylabel("\\frPhoton Index 2");
isis> plot_conf(contr);                % Plot the contours
isis> save_conf(contr,"file");         % Save them (later load them with load_conf;)
```



DEFAULTS ARE 68%, 90%, AND 99% CONFIDENCE FOR 2 PARAMETERS. CUSTOM LEVELS CAN BE CHOSEN INSTEAD. THIS FUNCTION IS PARALLELIZED IN ISIS.

PLOTTING INDIVIDUAL MODEL COMPONENTS

- ☐ ISIS, UNLIKE XSPEC, DOESN'T PLOT INDIVIDUAL MODEL COMPONENTS AS ISIS DOESN'T DIFFERENTIATE BETWEEN "ADDITIVE" AND "MULTIPLICATIVE" MODELS. (THEY'RE ALL JUST VECTORS!) ISIS DOESN'T KEEP TRACK OF THE SEPARATE "PIECES" REQUIRED.
- ☐ I WRITE SCRIPTS THAT CALL MY PLOTTING ROUTINES, ZEROS OUT COMPONENTS, AND OVERPLOTS SERIALLY.
- ☐ FOR "MULTI-PANEL" PLOTS (E.G., DATA & RESIDUALS), A LITTLE BIT TRICKIER THAN USING THE OPLT=1 OPTION.
- ☐ EXAMPLE FOLLOWS.

PLOTTING INDIVIDUAL MODEL COMPONENTS

```
isis> define fancy_plot()
{
   % Save parameters as they initially are
   save_par("/tmp/initial.par");

   % Let's make a 3 panel plot, where the top is counts, the middle is
   % "unfolded" spectra, and the bottom is residuals with and without
   % the lines.

   % 3 panel plot, with vertical ratios of 5:5:2
   multiplot([5,5,2]);

   % pgplot is much happier if you just work on 1 pane at a time.
   % Start with the top most pane.
   mpane(1);

   % For single pane plots, the oplt=1 plot option works fine.  For
   % multi-panel plots, we have to tell pgplot to *not* automatically
   % jump to the next pane after doing a plot.  This global variable
   % does that for my routines.
   no_reset=1;

   % First plot counts/unit/sec; X-axis will be keV.  Y-axis will be
   % proportional to ergs when we plot the unfolded data.
   xlog; ylog; fancy_plot_unit("kev","ergs");
```

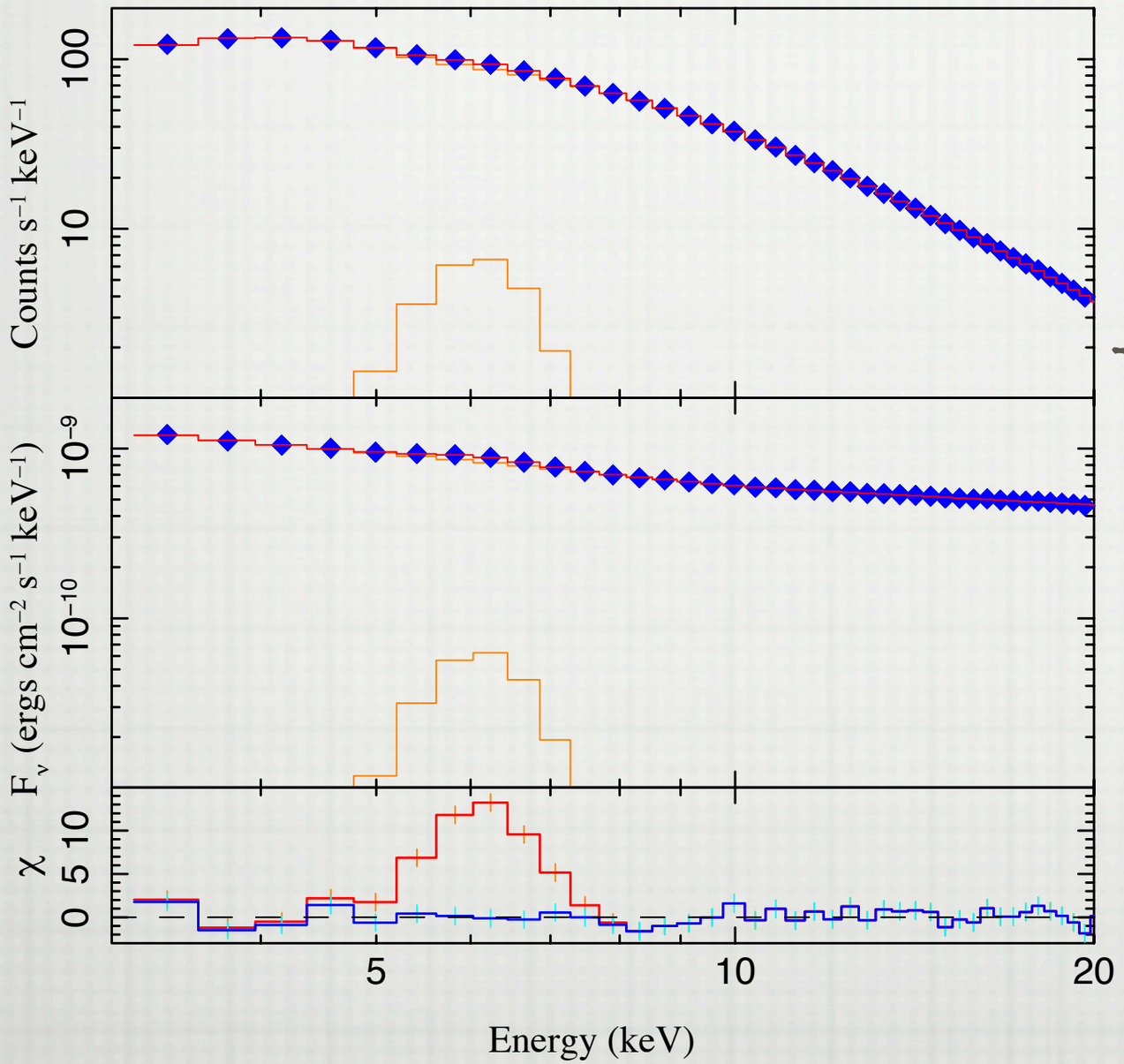
PLOTTING INDIVIDUAL MODEL COMPONENTS

```
% set the gaussian line to 0, then the powerlaw norm to 0, then
% plot the whole model.  Components in orange, whole model in red.

set_par("gaussian(1).norm",0); () = eval_counts;
plot_data(1;dsym=-4,dcol=4,decol=4,mcol=8,xrng={3,20},yrng={1.01,199.9});

load_par("/tmp/initial.par"); set_par("bknpower(1).norm",0,0,0,10);
() = eval_counts;
plot_data(1;dsym=-4,dcol=4,decol=4,mcol=8,oplt=1);

load_par("/tmp/initial.par"); () = eval_counts;
plot_data(1;dsym=-4,dcol=4,decol=4,mcol=2,oplt=1);

% Now do the same thing with the "unfolded" data in the middle panel.
mpane(2);

% Make sure we use the same x-range as above (y-range is different, of course...)
set_par("gaussian(1).norm",0); () = eval_counts;

plot_unfold(1;dsym=-4,dcol=4,decol=4,mcol=8,power=2,xrng={3,20},
                                              yrng={1.01e-11,1.99e-9});
```

PLOTTING INDIVIDUAL MODEL COMPONENTS

```
load_par("/tmp/initial.par"); set_par("bknpower(1).norm",0,0,0,10);
() = eval_counts;
plot_unfold(1;dsym=-4,dcol=4,decol=4,mcol=8,power=2,oplt=1);

load_par("/tmp/initial.par"); () = eval_counts;
plot_unfold(1;dsym=-4,dcol=4,decol=4,mcol=2,power=2,oplt=1);

% Now do the same thing with the residuals in the lower
% panel; however, we only 0 out the gaussian this time.
mpane(3);

% Make sure we use the same x-range as above (y-range is different, of course...)
set_par("gaussian(1).norm",0); () = eval_counts;
plot_residuals(1;res=1,rsym=0,rcol=2,recol=8,xrng={3,20},yrng={-2.99,14.99});

% The next plots after this one we want to go back to "normal" behavior.
no_reset=0;

load_par("/tmp/initial.par"); () = eval_counts;
plot_residuals(1;res=1,rsym=0,rcol=4,recol=5,xrng={3,20},oplt=1);
}
isis>
isis> fancy_plot;
```

PLOTTING INDIVIDUAL MODEL COMPONENTS



YOU PROBABLY WANT TO SAVE "FANCY_PLOT" AS A SCRIPT SO YOU CAN MODIFY IT MORE EASILY.

MORE ON PLOTTING

- [ ] PGPLOT IS A FAIRLY SIMPLE PROGRAM. IT'S SERVICEABLE, BUT LOTS OF PEOPLE (UNDERSTANDABLY) PREFER THINGS LIKE IDL.
- [ ] WITH MY PLOT ROUTINES, "write_plot" WILL SAVE SIMPLE ASCII FILES FOR THE *LAST* PLOT YOU MADE WITH *MY ROUTINES*. THOSE ARE SUITABLE FOR READING INTO OTHER PROGRAMS.
- [ ] AGAIN, USE SCRIPTS TO MAKE REPRODUCING WHAT YOU DID A LITTLE EASIER.

ADVANCED FUNCTION SYNTAX WITH BLANK ()

```
isis> % Use it as short hand for longer models - Manfred Hanke
isis> % has an example like this that creates a model with nearly
isis> % 100 lines (for gratings spectroscopy of Cyg X-1)
isis> define triple(){ return gaussian(1)+gaussian(2)+gaussian(3); }
isis> fit_fun("phabs(1)*(powerlaw(1)+triple())");
isis>
isis> % Use it to define functions that behave differently for each
isis> % data set you are evaluating (i.e., using Isis_Active_Dataset)
isis> public define line(){
        switch(Isis_Active_Dataset);
        { case 1:
          return gaussian(1); % ... if Isis_Active_Dataset=1
        }
        { case 2:
          return diskline(1); % ... if Isis_Active_Dataset=2
        }
        return laor(1);         % All other cases
      }
isis>
isis> fit_fun("phabs(1)*(powerlaw(1)+line())");
isis> % We won't be doing anything this fancy in this exercise!
```

CUSTOM GRIDS & CACHING

- SOME MODELS HAVE TO BE EVALUATED BEYOND THE NORMAL BOUNDARIES OF THE GRID:

```
isis> set_eval_grid_method(USER_GRID,[1:4],&my_function,1);
isis> % my_function returns lo/hi grid; final 1 says cache
isis> % the evaluation and apply it to each datasets
isis> % Now force ISIS to evaluate the entire grid:
isis> set_kernel([1:4];"std;eval=all");
```

- SIMPLE WRAPPER TO CREATE A LOGARITHMIC GRID:

```
isis> usr_grid([1:4],-9,3,0.01,1);
isis> % Creates a grid from 10^-9-10^3 keV, & caches
isis> % the model evaluation, and sets "eval=all"
```

- MODELS CAN BE CACHED AND REUSED FOR MULTIPLE DATASETS IF THE MODELS ARE *EXACTLY* THE SAME. I.E., NO DEPENDENCE ON `Isis_Active_Dataset`!

CUSTOM GRIDS & CACHING

- FOR SLOW MODELS, CACHING CAN REALLY SPEED THINGS UP. CACHE INDIVIDUAL SLOW COMPONENTS, AND THAT COMPONENT ONLY GETS REEVALUATED WHEN THE PARAMETERS CHANGE!

```
isis> (lo,hi) = linear_grid(0.1,100,2000);
isis> () = cache_fun(agnjet,lo,hi;qualifiers);
isis> fit_fun("agnjet_cache(1)+...");
```

- QUALIFIERS DISTINGUISH BETWEEN ADDITIVE AND MULTIPLICATIVE MODELS.
- CACHING IS ESPECIALLY HELPFUL FOR MULTI-SATELLITE FITS

FOR FURTHER HELP -

- S-LANG & ISIS BUILDING & MANUALS, CHECK THE MAIN WEB-SITES:
  - www.s-lang.org
  - http://space.mit.edu/CXC/ISIS/
- COMPARISONS TO XSPEC & TUTORIAL:
  - http://space.mit.edu/home/mnowak/isis_vs_xspec/index.html

FOR FURTHER HELP -

- ISIS-USERS MAILING LIST:
  - http://space.mit.edu/CXC/ISIS/mailing_list.html
- OR CONTACT US DIRECTLY:
  - FOR ISSUES WITH ISIS-BUILDS, ISIS-MODULES, AND BASIC ISIS-FUNCTIONALITY:
    - houck@space.mit.edu
  - FOR TRANSLATING FROM XSPEC EQUIVALENTS AND ANY SCRIPTS SHOWN IN THIS PRESENTATION:
    - mnowak@space.mit.edu