SPEX Cookbook

J. de Plaa, J. Ebrero, Y. Grange, C. Pinto, J. Kaastra, K. Steenbrugge, L. Gu, M. Mehdipour

December 8, 2017

Contents

Chapter 1

How to run SPEX

SPEX is a spectral fitting program used to fit high-resolution X-ray spectra. The code contains several simple and detailed models that are able to deal with the radiative processes observed in the X-ray band. Because SPEX has a command-line interface, a first-time user should get familiar with the syntax of the commands to be able to work with it. This chapter provides some basic commands and threads to fit X-ray spectra.

1.1 The SPEX data format

The data files containing the spectrum of the source and the response need to be in the correct format. In the SPEX installation, we provide a program called `trafo` to convert OGIP standard fits files into SPEX format (see Chapter 2 for an explanation of trafo). In this chapter, we assume that you already have spectra in SPEX format. For example, the demo spectra from the SPEX web site.

SPEX needs two files per spectrum:

- `<filename>.spo` – This file contains the countrate per energy bin for the source (D_i), as well as the background countrate and the errors (σ_i).
- `<filename>.res` – This file contains the instrumental response: the energy redistribution and effective area (R_{ij} A_j).

In order to calculate the observed model spectrum, SPEX uses this integral equation to account for the imperfections caused by the instrument:

$$D(c) = \int R(c, E) A(E) S(E) dE \tag{1.1}$$

$$D_i = \sum_{j=1}^{n} R_{ij} A_j S_j \tag{1.2}$$

The `.res` and `.spo` files are so-called FITS files. This is a data format widely used in Astronomy. FITS files can contain images as well as data tables. The software package FTOOLS provided by NASA contains a large number of tools to manipulate FITS files (see references). If you are interested, then you can launch `flaunch` to see which tools are available.

1.2 Loading spectra into SPEX

The SPEX program is started by entering `spex` in a linux terminal window. In the following sections we describe one run of the program. To start SPEX do this:

```
user@linux:~> spex
 Welcome user to SPEX version 3.00.00

SPEX>
```

First, we have to load the data files. This is done using the command `data`. It is a general thing in SPEX that filename extensions are not typed explicitly when issuing a command. If you have a file called `filename.spo` and `filename.res` then you type:

```
SPEX> data filename filename
```

The responsefile (.res) is entered first and then the file containing the spectrum (.spo). You can avoid confusion by giving the same filename to both .res and .spo files. Remember that the order of the words in the commands is very important!

To save you from typing a lot, many commands can be abbreviated by typing just the first few characters. For example, `da` is equivalent to `dat` and `data`.

1.3 Plotting the data

If the `data` command was successful, we can now have a look at the spectra. SPEX offers a lot of different plot commands through the PGPLOT package (known from, for example, Fortran programs). Below is a simple example to create a linear-linear plot in Å:

```
SPEX> plot dev xs
SPEX> plot type data
SPEX> plot x lin
SPEX> plot y lin
SPEX> plot ux a
SPEX> plot uy a
SPEX> plot rx 8.:35.
SPEX> plot ry 0.:0.05
SPEX> plot set 1
SPEX> plot cap ut text "This is my plot"
SPEX> plot cap lt disp false
SPEX> plot cap id disp false
SPEX> plot
```

The sequence above opens a PGPLOT window (`dev xs`) and tells SPEX that the plot will contain data with linear axes (`x lin` and `y lin`) in unit Å (`ux a` and `uy a`). `rx` and `ry` are the ranges on the x and y axes, respectively. Then the color of the data, background spectrum and model are set. The last commands beginning with `plot cap` remove some standard titles and other text around the plot. After you define the plot like in the example above, you can plot it with a single `plot` command.

The y-axis in this plot is in counts s^{-1} $Å^{-1}$. Ångstrom is not the only unit used in high-energy astrophysics. Usually, the energy of the photons is expressed in keV. In SPEX you can use keV by writing `k` instead of `a` in all commands. For example, `plot ux k` to use keV for the x-axis.

1.4 Ignoring and rebinning

High-resolution X-ray spectra from Chandra and XMM-Newton are usually oversampled (e.g. the energy bins are much smaller than the spectral resolution) and contain a lot more channels then is useful.

Therefore, it is necessary to remove wavelength intervals which contain bad data and rebin your spectrum. In the next example we bin the spectrum over the 8–35 Å range with a factor of 5 and ignore the rest of the spectrum:

```
ign 0:8 unit a
ign 35:100 unit a
bin 8:35 5 unit a
```

The words `unit a` tells SPEX that the ranges (for example 8.0:35.0) are given in Å. If you work with more than one spectrum (from more than one instrument), you can add an extra instrument statement:

```
ign ins 1:2 0:8 unit a
ign ins 1:2 35:100 unit a
bin ins 1:2 8:35 5 unit a
```

Here, instrument 1 to 2 are binned with a factor of 5 over the 8–35 Å range.

1.5 Defining a model

Now we have a clean and rebinned spectrum that is ready to fit. Before we can start fitting, we first need to define a model. It's equivalent to $S(E)$ in Eq. 1.1. The model can contain one or more of these components:

- `absm` Model for interstellar absorption.
- `reds` Redshift.
- `po` Powerlaw.
- `ga` Gaussian.

And there are more (see the SPEX manual)! The following command sequence defines a simple powerlaw model at a certain redshift and absorbed by the interstellar medium. The individual components of the model are loaded one-by-one with the `com` command:

```
SPEX> com reds
SPEX> com absm
SPEX> com po
SPEX> com rel 3 1,2
```

The last command (`com rel 3 1,2`) tells SPEX that component 3, the powerlaw, is first redshifted by component 1 and then absorbed by component 2. The order of the 1 and the 2 is important! Always think what happens in which order on the way from the source to the telescope.

For most sources the distance is more or less known. To get a right luminosity estimate for the source, the expected distance has to be provided to SPEX:

```
SPEX> dist 0.1 z
Distances assuming H0 =  50.0 km/s/Mpc and q0 = 0.500
Sector    m      A.U.        ly        pc       kpc       Mpc  redshift        cz
-------------------------------------------------------------------------------
1 1.894E+25 1.266E+14 2.002E+09 6.139E+08 6.139E+05  613.8689    0.1000   29979.2
-------------------------------------------------------------------------------
```

With this command, the distance to the source is set to a redshift of 0.1. The derived distances for this cosmology are in the output of the `dist` command.

Now we have to estimate the initial parameters. With the command `par show` we can see which parameters there are:

```
SPEX> par show
-------------------------------------------------------------------------------------
sect comp mod  acro parameter with unit     value      status    minimum   maximum

   1    1 reds z    Redshift              0.000000     frozen   -1.0      1.00E+10

   1    2 absm nh   Column (1E28/m**2)   9.9999997E-05 thawn     0.0      1.00E+20
   1    2 absm f    Covering fraction     1.000000     frozen    0.0       1.0

   1    3 pow  norm Norm (1E44 ph/s/keV)  1.000000     thawn     0.0      1.00E+20
   1    3 pow  gamm Photon index          2.000000     thawn    -10.       10.
   1    3 pow  dgam Photon index break    0.000000     frozen   -10.       10.
   1    3 pow  e0   Break energy (keV)   1.0000000E+10 frozen    0.0      1.00E+20
   1    3 pow  b    Break strength        0.000000     frozen    0.0       10.
   1    3 pow  type Type of norm          0.000000     frozen    0.0       1.0
   1    3 pow  elow Low flux limit (keV)  2.000000     frozen   1.00E-20  1.00E+10
   1    3 pow  eupp Upp flux limit (keV)  10.00000     frozen   1.00E-20  1.00E+10
   1    3 pow  lum  Luminosity (1E30 W)   1.000000     frozen    0.0      1.00E+20

-------------------------------------------------------------------------------
Fluxes and restframe luminosities between   2.0000     and    10.000     keV

 sect comp mod   photon flux   energy flux nr of photons    luminosity
            (phot/m**2/s)      (W/m**2)   (photons/s)          (W)
    1    3 pow    0.00000       0.00000       0.00000       0.00000
```

We can set the parameters using the `par` command. The first "1" in column "sect" can usually be ignored. The commands then look like this:

```
SPEX> par 1 z val 0.1
SPEX> par 2 nh val 2.E-3
SPEX> par 3 norm val 1.E+10
SPEX> par gamm val 1.5
```

The last component number used is saved, so in the last line we can skip typing the number 3 after `par`. Then, we run `par show` again to see what happened:

```
SPEX> par show
-------------------------------------------------------------------------------------
sect comp mod  acro parameter with unit     value      status    minimum   maximum

   1    1 reds z    Redshift              0.100000     frozen   -1.0      1.00E+10

   1    2 absm nh   Column (1E28/m**2)   2.0000001E-03 thawn     0.0      1.00E+20
   1    2 absm f    Covering fraction     1.000000     frozen    0.0       1.0

   1    3 pow  norm Norm (1E44 ph/s/keV)  1.000000E+10 thawn     0.0      1.00E+20
   1    3 pow  gamm Photon index          1.500000     thawn    -10.       10.
   1    3 pow  dgam Photon index break    0.000000     frozen   -10.       10.
   1    3 pow  e0   Break energy (keV)   1.0000000E+10 frozen    0.0      1.00E+20
   1    3 pow  b    Break strength        0.000000     frozen    0.0       10.
   1    3 pow  type Type of norm          0.000000     frozen    0.0       1.0
   1    3 pow  elow Low flux limit (keV)  2.000000     frozen   1.00E-20  1.00E+10
   1    3 pow  eupp Upp flux limit (keV)  10.00000     frozen   1.00E-20  1.00E+10
   1    3 pow  lum  Luminosity (1E30 W)  5.6014867E+08 frozen    0.0      1.00E+20
```

```
--------------------------------------------------------------------------------
Fluxes and restframe luminosities between   2.0000     and    10.000     keV

 sect comp mod   photon flux   energy flux nr of photons    luminosity
            (phot/m**2/s)      (W/m**2)   (photons/s)          (W)
    1    3 pow    0.00000       0.00000       0.00000       0.00000
```

Finding the right initial values for the parameters is a game of trial and error. To see whether you are going in the right direction, you can calculate the model with the command `calc` and `plot` again. If you see the model appear in your screen, then the model is close enough to be fitted. Especially the normalization of the powerlaw (`3 norm`) can vary a lot depending on the countrate of the source.

1.6 Fitting the data

We are ready to fit the data! SPEX has a nice feature to look at the progress of the fit. To activate this feature you have to give the command `fit print 1`. If your initial parameters were acceptable, you can see the model converge to the data in the plot window after you entered the `fit` command. When the fit is done, then the parameters and χ^2 are printed on screen. If the χ^2 value is close to the number of degrees of freedom, then your fit is acceptable. Sometimes more runs of the command `fit` are necessary after changing some initial parameters. This is especially true when using complex models. Again this is a game of trial and error.

You also might want to fix or free certain parameters to see if they can be constrained. In SPEX fixing is `f` (frozen) and freeing is `t` (thawn). You can free the redshift and fix the $N_{\rm H}$ by the following commands:

```
SPEX> par 1 z stat t
SPEX> par 2 nh stat f
```

1.7 Calculating errors

When the fit is acceptable, you might want to know the uncertainties on your fitted parameters. Errors are determined one-by-one by fixing the parameter to some value and calculate the $\Delta\chi^2$ with respect to the best fit. If you want to know the 1σ error on the parameter, you need to know its values at $\Delta\chi^2 = 1$. This is done by the `error` command. First you have to set the desired $\Delta\chi^2$ in SPEX: `error dchi 1.` After this you can calculate the error for each parameter. For example redshift:

```
SPEX> error 1 z
```

1.8 Making life easier...

In this short manual you have seen a lot of commands, but to avoid typing too much you want to use some identical series of commands every time you fit a certain spectrum. For example, you don't want to type all plot commands again when making a plot. Therefore, the program has a command to solve this problem. With the command `log exe filename` you can execute a number of commands at the same time. The numbers are read from a normal text file with (in this case) the name `filename.com`. Again the extension `.com` should not be typed explicitly. Below is an example to setup a plot for an EPIC spectrum (range 0.2–10.0 keV) with a small frame that shows residuals. Note that you can put any command in such a command file and you can make comment lines by putting a # sign in front of the line.

```
# This is a command file that creates a plot with residuals.
plot dev xs
```

```
plot type data
plot x log
plot y log
plot rx 0.2:10.
plot ry 0.0001:10.
plot back disp t
plot set 1
plot data col 1
plot model col 2
plot back col 1
plot set all
plot frame new
plot frame 2
plot type chi
plot uy rel
plot x log
plot rx 0.2:10.
plot ry -0.5:0.5
plot view def f
plot view x 0.08:0.92
plot view y 0.1:0.3
plot cap y text "Rel. Error"
plot cap ut disp f
plot cap lt disp f
plot cap id disp f
plot frame 1
plot view def f
plot view x 0.08:0.92
plot view y 0.3:0.9
plot cap x disp f
plot cap id disp f
plot cap ut disp f
plot box numlab bot f
```

1.9 Saving your work

There are several ways in SPEX you can save your work. Below you find a few examples to save your commands, output or plots.

1.9.1 Saving a plot

These commands open a PostScript plot device with filename `filename.ps`, then they plot your figure in the PS file and closes the device:

```
SPEX> plot dev cps filename.ps
SPEX> plot
SPEX> plot close 2
```

1.9.2 Saving commands

If you want to save all commands that you execute to an ASCII file (`filename.com`), then type `log save filename`. Do not forget to close the file at the end of the session by typing `log close save`. The saved commands in the textfile can be executed again by the `log exe filename` command.

1.9.3 Saving output

In the same way as in the previous example, you can also save the output on your screen by typing `log out filename` (the file will be an ASCII file called `filename.out`). You can close the file with `log close out`. This command is very useful to save your parameters and errors.

1.10 Quitting the program

Just type `quit`...

1.11 Tips & Tricks

- If you make a typo in a command or you want to do the same command again, then push the `arrow-up` button on your keyboard. There is an entire history of your commands there.
- The `Tab` key is able to automatically complete the command you are typing. In case there are more possibilities, it shows them all.

1.12 References

- The full SPEX manual can be found here:
  `http://www.sron.nl/files/HEA/SPEX/manuals/manual.pdf`
- FTOOLS software package to open and manipulate FITS files:
  `http://heasarc.nasa.gov/lheasoft/ftools/ftools_menu.html`

Chapter 2

How to run trafo

SPEX uses its own format for spectra and response matrices. The main advantage is that the SPEX format is more memory efficient than the common OGIP format. This means that SPEX can reach a high performance level, especially for high-resolution spectra that consume a lot of memory. It is even possible to rearrange response matrices to reduce the CPU time spent on the convolution of the model spectra with the instrument response. A slight disadvantage is that your OGIP spectra have to be converted into SPEX format before SPEX is run. The SPEX package therefore contains a tool to do this conversion: `trafo`.

2.1 Input files

Before `trafo` is started, you need the OGIP spectra and responses first. Please read the documentation regarding your dataset for more information on how to create OGIP files. The minimum that `trafo` needs is a spectrum (.pi or .pha) and a response matrix (.rmf or .rsp). If you want to subtract background spectra or if you have an additional arf file, then please also collect these files in your working directory.

If applicable, you can use the FTOOL `grppha` to group the spectra before trafo is run. Note that most problems with `trafo` occur when spectra or responses do not exactly conform to the OGIP standard. There is a large number of OGIP varieties, which makes it difficult to support all OGIP spectra.

2.2 Running the program

The easiest method is to run trafo in the directory where your OGIP files are. It is an interactive program, so it will ask the user for information when the program is run.

```
user@linux:~> trafo
Program trafo: transform data to SPEX 2.0 format
This is version 1.02, of trafo

 Are your data in OGIP format            : type=1
 Old (Version 1.10 and below) SPEX format: type=2
```

The first questions are quite straightforward. In the case of OGIP spectra, the type is always 1. In principle, it is possible to put more than one spectrum in a spo and res file, but for most simple cases transforming a single spectrum is sufficient.

```
Enter the type: 1
Enter the number of spectra you want to transform: 1
Enter the maximum number of response groups per energy per spectrum: 100000
```

The maximum number of response groups should just be large in nearly all cases, unless there is a special reason to put a small number here.

2.2.1 Re-arranging the response matrix

The following feature is present in trafo since version 1.02 (SPEX version 2.02). It allows the user to re-arrange the response matrix to increase speed. There are three options: 1. Keep the matrix as provided. 2. Try to re-arrange the matrix into contiguous groups. The program tries to identify physically distinctive components and avoids overlapping data. 3. Split the matrix into N equal-sized components. This is particularly useful for grating spectra (RGS) and allows for efficient matrix multiplication on multi-core processors. Any power of 2 between 8 and 32 should provide a fast response matrix. In the terminal, this option is provided in the following way:

```
How should the matrix be partioned?
Option 1: keep as provided (1 component, no re-arrangements)
Option 2: rearrange into contiguous groups
Option 3: split into N roughly equal-sized components
Enter your preferred option (1,2,3): 1
```

Option number 1 is the safest option to choose, but also the slowest. Option 2 and 3 can provide a significant increase in performance, but results should be carefully checked. More information about re-arranging response matrices can be found in the SPEX Manual.

2.2.2 Reading the spectra

Then, `trafo` asks for the filenames of the source and background spectra. First, provide the file name of the source spectrum. `trafo` will return some of the basic properties of the spectral file, like exposure time and values of the most important FITS keywords.

```
Enter filename spectrum to be read: PN-source.pi
Exposure time (s): 2.10571992E+04
Assuming Poissonian Errors
Areascal: 1.00000000E+00
Backscal: 1.00000000E+00
No BACKFILE keyword found
Corrscal: 1.00000000E+00
No CORRFILE keyword found
No RESPFILE keyword found
No ANCRFILE keyword found
No background specified in pha-file.
```

A background spectrum can be provided (optional), which will be subtracted from the source spectrum. If a background file is already specified in the FITS header of the source spectrum, this question will not be asked.

```
Read nevertheless a background file? (y/n) [no]: y
Enter filename background spectrum to be read: PN-background.pi
Exposure time (s): 2.10572832E+04
Assuming Poissonian Errors
```

```
Areascal: 1.00000000E+00
Backscal: 1.00000000E+00
No BACKFILE keyword found
Corrscal: 1.00000000E+00
No CORRFILE keyword found
No RESPFILE keyword found
No ANCRFILE keyword found
```

2.2.3 Bad channels and grouping

Depending on the instrument used, there is a chance that the spectrum contains bad channels. This is especially true for grating spectra. Sometimes the background spectrum can have a different number of bad channels than the source spectrum. It is therefore important that a particular bad channel in either of the two spectra is ignored. In this example, there are no bad channels, so either yes or no will do.

```
Checking data quality and grouping ...
Ogip files have quality flags. Quality 0 means okay
Your spectrum file has          0 bins with bad quality
Your background file has        0 bins with bad quality
Your combined file has          0 bins with bad quality
Shall we ignore bad channels? (y/n) [no]:y
```

If `grppha` has been used on the spectrum, `trafo` will also ask whether the spectra should be binned according to the groups defined in the PHA file.

2.2.4 Read response and effective area files

In the next step, the response matrix is read. Sometimes, the response matrices start at channel 0, which can be somewhat confusing. Especially when some arrays start at channel 0 and others at channel 1. If both data sets start at zero, it is best to shift the channel numbers with 1 unit. For most instruments this is fine, however, there are situations when this does not apply. In that case, please check your energy grid by loading a delta line component in SPEX and check the energy of the line manually. Then, compare the output with a delta line defined in XSPEC.

```
Determining background subtracted spectra ...
No response matrix file specified in pha-file.
Enter filename response matrix to be read: PN.rmf
Reading response matrix ...
Warning, ebounds data started at channel    0
Warning, response data started at channel 0
Possible response conflict; check xspec/spex with delta line!
Enter shift to response array (1 recommended, but some cases may be 0):1
No effective area file specified in pha-file.
```

Sometimes, also an effective area file needs to be provided separately:

```
Read nevertheless an effective area file? (y/n) [no]: y
Enter filename arf-file to be read: PN.arf
Reading effective area ...
Determining zero response data  ...
Total number of channels with zero response:      373
Original number of data channels                                :         4096
Channels after passing mask and omitting zero response channels:         3723
Rebinning data where necessary ...
Rebinning response where necessary ...
old number of response elements:       435950
```

```
 new number of response elements:       435950
 old number of response groups  :         1481
 new number of response groups  :         1481
 Correcting for effective area ...

 Determine number of components ...
 Found     1 components
 Enter any shift in bins (0 recommended): 0
 order will not be swapped ...
```

If there are bins with zero response, then they are excluded from the resulting file. Also here a shift in bins can be set, but the recommended value is 0.

2.2.5 Writing res and spo files

The final step is writing the spectra in SPEX format. The file names should be provided without an extension. The .spo and .res extension will be added automatically.

```
 Enter filename spectrum to be saved (without .spo): PN
 Enter filename response to be saved (without .res): PN
 Final number of response elements:   435950
```

2.3 Scripting

If one needs to transform a lot of similar spectra, it can be very worthwhile to write a script. The BASH example below basically fills in the questions of trafo automatically. Of course, one needs to know the answers beforehand. Every line in the script is connected to one answer in trafo. In this case, we loop over spectra from different extraction regions, which is common for extended source analysis. But one can also loop over spectra obtained in different epochs. Here we have three extraction regions with files named `PN_1.pi`, `PN_2.pi`, and `PN_3.pi`.

```
#!/bin/bash

# Loop over all extraction regions

for region in 1 2 3
do
/opt/bin/trafo << EOF
1
1
100000
1
PN_${region}.pi
y
PN_${region}_background.pi
y
PN_${region}.rmf
1
y
PN_${region}.arf
0
PN_${region}
PN_${region}
EOF
done
```

We assume for the moment that the trafo binary is installed in the /opt/bin directory. Please use the path to your local SPEX installation here.

2.4 Special situations

Since the OGIP standard is not always strictly followed, `trafo` may ask additional questions or even crash with some FITS files. Below there is a small list of questions or errors that you may encounter.

2.4.1 XMM-Newton EPIC MOS spectra

The first bin in EPIC MOS data has a zero bin boundary. Therefore, `trafo` asks an additional question after reading the response matrix:

```
Reading response matrix ...
Lower model bin boundary for bin 1 must be positive; current values: 0.00E+00  5.00E-03
Enter new bin boundary values manually: 3.E-5 5.E-3
```

Since this is the first bin, which is usally ignored, the values that are put in here are not very important, as long as the values are contiguous. Here, the lower boundary is raised from 0. to 3.E-5. The value of 3.E-5 is arbitrary, as long as it is non-zero and smaller than 5.E-3.

Chapter 3

Combining RGS spectra using rgsfluxcombine and rgsfmat

Author: K. C. Steenbrugge (katrien.steenbrugge@gmail.com)
Keywords: RGS spectra – combining – effective area – errors

3.1 Goal

For longer exposure times it quite often happens that the data were obtained over several orbits and thus separate RGS spectra are produced when reducing the data. The purpose of `rgsfluxcombine` is to correctly add the RGS 1 and 2 for both spectral orders of one or multiple observations to obtain a higher signal-to-noise spectrum. `rgsfmat` produces the corresponding, but much smaller, response matrix. This allows for much faster fitting of the spectrum compared to fitting the different RGS spectra simultaneously. The program avoids creating artificial emission or absorption lines in spectra with varying emission properties due to the presence of bad pixels. As an option one can correct for the unreliable χ^2 values obtained for high signal-to-noise spectra due to the smoothing of the data when the binning on detector pixels is transferred to the binning in wavelength scale. The program also allows to correct for the difference between the effective area between RGS 1 and 2 and second order.

3.2 SPEX solution

To combine the fluxed RGS spectra obtained by `rgsfluxer` of the XMM-Newton SAS software using the `rgsfluxcombine` combine program, which is used instead of `trafo` before running SPEX to fit the data. The output of `rgsfluxcombine` is the input for `rgsfmat` program, which creates the response matrix and assures the spectrum and response matrix are in the SPEX readable format. For normal bins the exposure time is used as weights in combining the spectra. However, for bins that have at least in 1 spectrum a bad pixel or fall in a CCD gap, the weights are determined assuming that the spectral shape across the bad bin did not change between spectra. `rgsfluxcombine` uses the neighboring good pixels to determine the total flux and the flux contribution per spectra, and thus the weights. This avoids creating artificial emission or absorption features due to a varying flux of the source and the presence of bad pixels. This procedure does go wrong if the spectral shape between the good neighboring pixels did change between

the observations, for instance if the bad pixel is located at a line of varying equivalent width. To be able to still combine spectra in this case one has the option to ignore all the spectral bins that have a bad pixel in at least 1 spectrum. This does lead to many spectral bins without data, some of the data gaps will be close to or at astrophysical important features, limiting the results one can obtain from fitting the spectra. To optimize the number of bins retained without biasing the results, `rgsfluxcombine` gives the option to enter a minimum fractional exposure time that the pixels need to have to be included in the combined spectrum. Choosing 0 means that all spectral bins will be included, while for a value of 1 only spectral bins for which there is no bad bin in any of the spectra will be included.

`rgsfluxcombine` gives an option to correct for the differences in the effective area between the first order RGS1 and RGS2 as well as second order spectra. The assumed correct effective area is the average effective area. In the same step the program also applies a correction for the smoothing caused by the slightly different binning between the detector space and wavelength space. This leads to a too low χ^2 value for an acceptable fit in high signal-to-noise spectra. To counter this, the error bars are multiplied 0.85. Both these corrections are only noticeable for high signal-to-noise spectra. For a detailed description of both corrections as well as how the spectra are combined, please see Kaastra et. 2011b.

Using `rgsfluxcombine` and `rgsfmat` one does not need to run `trafo`, however, it is always a good idea to run `trafo` on at least the 1^{st} order spectra of the different observations to ensure that the spectra are similar, to check for variability and that the final spectrum is consistent with expectations from the individual spectra.

3.2.1 Data preparation

Fluxed RGS spectra are automatically produced by either the `rgsproc` or `xmmextract` commands of the XMM-Newton SAS software. However, to be able to use `rgsfluxcombine` `rgsfluxer` must be run with 3400 bins and with a minimum and maximum wavelength of 5 and 38 Å. One can either alter the standard input .xml file using odfParamCreator or by rerunning the `rgsfluxer` command in the SAS software. Assuming you have already run `xmmextract` we will show how to rerun `rgsfluxer`. We assume you have done the start-up treat for the SAS software and that the spectrum and background spectrum are called P0655590201R1S004SRSPEC1001.FIT and P0655590201R1S004BGSPEC1001.FIT; and the response matrix is P0655590201R1S004RSPMAT1001.FIT. We will call the new file rgs1_o1_obs1_flux.fits. One can run `rgsfluxer` with the following command:

```
rgsfluxer pha=P0655590201R1S004SRSPEC1001.FIT \
bkg=P0655590201R1S004BGSPEC1001.FIT \
rmf=P0655590201R1S004RSPMAT1001.FIT \
file=rgs1_o1_obs1_flux.fits mode=wavelength min=5 max=38 bins=3400
```

The above command must be done for all the RGS spectra, first and second order that one wants to combine. Thus if we want to combine RGS1 and RGS2 and 1^{st} and 2^{nd} order for 2 observations, `rgsfluxer` needs to be run 8 times.

Another required input for `rgsfluxcombine` is the exposure times, which can be obtained from the header named SPECTRUM of the spectrum files using `fv`. Note that the exposure times for RGS1 and RGS2 and 2^{nd} orders are different, and that we need the exposure time in seconds and specified up to a hundredth of a second.

3.2.2 Combining data from the same RGS

`rgsfluxcombine` has 2 options, both of which you need to use if you want to combine RGS data from different observations. We will assume that we are combining the RGS spectra for an AGN for 2 different observations taken in consecutive orbits. We thus do not expect spectral differences across narrow bands in the spectrum. One first needs to combine the spectra of the same RGS and same order. Thus if we want to have 1 spectrum containing RGS1 and RGS2 for both orders, we need to run `rgsfluxcombine` four times using option 1. This is independent of how many observations one wants to combine. Option 1 requires an ascii file as input which contains on the first line the number of spectra to be combined, i.e. the number of different observations you want to combine. Then there should be a line per observation

containing the exposure time using the F9.2 format a space and the filename. In our example the filenames are rgs1_o1_obs1_flux.fits and rgs1_o1_obs2_flux.fits, so the file, which we will call com_rgs1_o1.txt, should look like:

```
2
084081.36 rgs1_o1_obs1_flux.fits
053751.80 rgs1_o1_obs2_flux.fits
```

Note that due to the required format both exposure times start with 0. With this we are ready to run `rgsfluxcombine` by typing at the prompt `rgsfluxcombine`. As we are combining spectra from the same instrument and order, we chose option 1. Because we know that the spectral shape across bad bins did not vary between the different observations, we use a minimal exposure time of 0. Finally, we decide to do the effective area correction and correct the error bars, specify that we are combining the 1^{st} order RGS1 data and name the new file rgs1_o1.fits.

```
Welcome to rgs_fluxcombine. This program has two basic options:
Option 1: combine fluxed RGS spectra from the same RGS and spectral order
Option 2: combine fluxed RGS spectra from different RGS and/or spectral orders
          For this option, there is a maximum of 1 spectrum for each combination.
          If you have multiple data, first combine them for the same RGS & order
          using this program with option 1, then rerun with option 2.
Enter your option (1/2): 1

You should provide an ascii file containing the following info:
First line: number of spectra that must be combined
Next lines: for each observation, a line with the following info:
Exposure time (s) and file name (outpur of SAS task RGSFLUXER
This MUST have format (F9.2,1X,A) (xxxxxx.xx aaaaaaa.... )
It is REQUIRED that you have run RGSFLUXER with 3400 bins from 5-38 Ang
This also holds for second order spectra!

Now give filename: com_rgs1_o1.txt

VERY IMPORTANT: if the spectral SHAPE varies, there is NO guarantee
that data bins with incomplete exposure are accounted for properly.
Therefore, you can enter a minimum exposure fraction. If you put
this to 1, you will have no bias but may miss important lines.
If you put this to 0, you have most lines but are in trouble if these
are variable in equivalent width
Now give mininum exposure (0-1): 0
 Total exposure time is: 137832.80 s

You can apply some preliminary RELATIVE effective area corrections, that bring the
individual spectra of RGS1 versus 2 and/or order 1 versus order 2 in better
agreement. Results are not yet perfect, but it helps in several situations.
You also can apply a NOMINAL correction to the error bars, to compensate the
smoothing effect of the binning procedures applied within SAS.
Do you want to apply these corrections? (y/n): y
Now effective area corrections; enter RGS (1/2) and order (1/2): 1 1
Enter file name combined fluxed spectrum: rgs1_o1.fits
```

Please note down the total exposure time as we will need it in the final step with `rgsfluxcombine` using option 2. As we want to combine the RGS1 and RGS2 and 1^{st} and 2^{nd} order, we need to repeat the above steps three more times, but with a different ascii input file, the name of the output file, and changing the input for RGS (1/2) and order (1/2) accordingly.

3.2.3 Combining different instruments

Once we have run the `rgsfluxcombine` for the different RGSs and orders, we need to rerun `rgsfluxcombine`, but now with option 2. This is also the option needed if you want to combine the data from the different instruments and orders for one observation. Before we run `rgsfluxcombine` we need to create another ascii file. This file should contain on the first line the number of spectra to be combined: 4, because we want to combine both orders and both RGSs. The next 4 lines should contain the number of the RGS, of the order, exposure time (same format as before) and filename, which is the filenames given during the runs of `rgsfluxcombine` with option 1. An example is `com.txt`:

```
4
1 1 137832.80 rgs1_o1.fits
2 1 138045.20 rgs2_o1.fits
1 2 137833.16 rgs1_o2.fits
2 2 138045.20 rgs2_o2.fits
```

We then run `rgsfluxcombine`. As we are combining different RGSs and orders, there should be no spectral variability between the spectra to be combined, and thus we chose again a minimum fractional exposure time of 0.

```
Welcome to rgs_fluxcombine. This program has two basic options:
Option 1: combine fluxed RGS spectra from the same RGS and spectral order
Option 2: combine fluxed RGS spectra from different RGS and/or spectral orders
          For this option, there is a maximum of 1 spectrum for each combination.
          If you have multiple data, first combine them for the same RGS & order
          using this program with option 1, then rerun with option 2.
Enter your option (1/2): 2

You should provide an ascii file containing the following info:
First line: number of spectra that must be combined
Next lines: for each observation, a line with the following info:
RGS nr (1/2), spectral order (1/2), exposure time (s) and file name
(output of SAS task RGSFLUXER)
This MUST have format (I1,1X,I1,1X,F9.2,1X,A) (xxxxxx.xx aaaaaaa.... )
It is REQUIRED that you have run RGSFLUXER with 3400 bins from 5-38 Ang
This also holds for second order spectra!

Now give filename: com.txt

You can enter a minimum exposure fraction. If you put
this to 1, you will have full exposure but may miss important lines.
If you put this to 0, you have the maximum number of lines but
you may have poor statistics
Now give mininum exposure (0-1): 0
Enter file name combined fluxed spectrum: mrk359_rgs.fits
```

We now have the combined spectrum which includes the RGS data of 2 observations, the 2 RGS instruments and both orders. We need to create an appropriate response matrix and convert the mrk359_rgs.fits file into a .spo file that SPEX can read in.

3.2.4 Creating the response matrix

To create the response matrix and convert the combined RGS spectrum into SPEX readable format, we run the `rgsfmat` program:

```
Enter file name fluxed spectrum: mrk359_rgs.fits
Enter file name output spectrum (with .spo): mrk359_rgs.spo
Enter file name output response (with .res): mrk359_rgs.res
 Final number of response elements:  1008432
```

We have now the necessary files to analyze the combined spectrum in SPEX. For a description of the different models, please see the other chapters in this cookbook.

3.3 References

Kaastra, J. S., de Vries, C. P., Steenbrugge, K. C. et al., 2011b, A&A, 534, 37

Chapter 4

Modeling absorption spectra

Author: C. Pinto (c.pinto@sron.nl)
Keywords: Absorption spectra – Point-like sources – RGS spectroscopy

4.1 Goal

The light coming from astronomical sources suffers from several absorption mechanisms before arriving to the Earth and being measured. The absorption could occur in regions near the source, for example, the emission of the central engine inside the active galactic nuclei is partially obscured by gas and dust found around the accretion disk of the AGN, just as the light emitted by the central regions of a supernova can be absorbed by the matter found in the expanding shell. However, the most common origin of absorption in the spectra of the astronomical sources is the interstellar medium of our Galaxy (ISM). The ISM is a mixture of multi-phase gas, molecules and dust, it is highly inhomogeneous and has to be carefully analyzed. In this chapter, we show a general method to carefully model absorption due to the ISM using the SPEX package.

4.2 SPEX solution

A typical difficulty in the analysis of the X-ray spectra is to distinguish among the several environments which act as absorber: the Galactic gas has usually low velocity dispersion and zero-shift lines, while expanding shells give rise to broad blue-shifted lines. Of course, the things can be much more complicated. The absorption components in SPEX provide a solution to most of the problems in this field.

4.2.1 Running trafo

We start from two first order RGS spectra (RGS1 and RGS2) of the low-mass X-ray binary (LMXB) GS 1826-24 extracted according the standard procedure of the XMM-Newton SAS Guide. We separately convert the spectra in the SPEX 2.0 format through `trafo`. Here we show the `trafo` run for the first order of RGS1. The run for the first order of RGS2 is performed in the same way.

```
Program trafo: transform data to SPEX 2.0 format
This is version 1.02 of trafo
```

```
Are your data in OGIP format            : type=1
Old (Version 1.10 and below) SPEX format: type=2
Enter the type: 1
Enter the number of spectra you want to transform: 1
Enter the maximum number of response groups per energy per spectrum: 10000
How should the matrix be partioned?
Option 1: keep as provided (1 component, no re-arrangements)
Option 2: rearrange into contiguous groups
Option 3: split into N roughly equal-sized components
Enter your preferred option (1,2,3): 3
Enter number of components N (preferably a power of 2, between 8 and 32): 16
Enter filename spectrum to be read: P0150390301R1S004SRSPEC1001.FIT
Read nevertheless a background file? (y/n) [no]: y
Enter filename background spectrum to be read: P0150390301R1S004BGSPEC1001.FIT
Shall we ignore bad channels? (y/n) [no]: n
Enter filename response matrix to be read: P0150390301R1S004RSPMAT1001.FIT
Read nevertheless an effective area file? (y/n) [no]: n
Enter any shift in bins (0 recommended): 0
Save the spectrum by providing convenient names for the res and spo les.
Enter filename spectrum to be saved: rgs1_order1.spo
Enter filename response to be saved: rgs1_order1.res
```

4.2.2 Running SPEX

Once we have converted the spectra and created .res and .spo files, we can load both spectra and response matrices in SPEX and plot the data in Ångstrom units. We skip the plotting commands and refer to Chapter 1 for the SPEX overview.

```
SPEX> data rgs1_order1 rgs1_order1
SPEX> data rgs2_order1 rgs2_order1
```

We restrict the spectral fit to the region where the effective area is large and the source is brighter than the background, which in our case is between 7-31 Å. We also rebin the data by a factor of 2, i.e. about 1/3 of the FWHM, because this is the suitable binning factor for most detectors. We also remove the regions where the detector chips are out of order:

```
SPEX> ign inst 1:2 0:7 u a
SPEX> ign inst 1:2 31:40 u a
SPEX> ign inst 1 10.5:14 unit an
SPEX> ign inst 2 19.95:24.05 unit an
SPEX> bin inst 1:2 7:31 2 u an
```

4.2.3 The cold gas

First of all, we model the ISM with simple neutral gas. For this purpose, we use the *hot* component in SPEX, which describes the transmission through a layer of a collisionally ionized plasma. At low temperature this model mimics very well the neutral gas of the ISM. The continuum emission of the X-ray binary is usually fitted by a combination of a black body (*bb* model in SPEX) plus a comptonization component (*comt* model). The former comes from the LMXB accretion disk and the latter represents the emission of the accretion disk corona. At the end we relate both the emission components to the absorber:

```
SPEX> com hot
SPEX> com bb
SPEX> com comt
SPEX> com rel 2:3 1
```

We temporarily remove two small regions near the Fe L-edge and O K-edge: 17.2-17.7 Å and 22.5-23.2 Å, respectively. Indeed, these regions are affected by fine structure features due to dust grains and molecules, therefore they can not be fitted with a pure-gas model. For further analysis see §4.2.5.

```
SPEX> ign inst 1:2 22.5:23.2 u a
SPEX> ign inst 1:2 17.2:17.7 u a
```

From the literature we know that the hydrogen column density of the source is within $2-5\times10^{25}\,\mathrm{m}^{-2}$, thus for the *hot* component we choose the following starting values: $N_\mathrm{H} = 4\times10^{25}\,\mathrm{m}^{-2}$ and $t = 5\times10^{-4}\,\mathrm{keV}$, which is the minimum temperature in SPEX. The velocity dispersion is frozen to 1 km s^{-1}, which is typical of the cold gas in the ISM. We also choose as reference the abundances suggested by Lodders (2003), which are suitable for the ISM. The temperature of both the *bb* and *comt* components (seed photon temperature for the latter) are put to 0.1 keV. The electron temperature of the *comt* model is set to 20 keV and its optical depth to 1, because these are standard values for the LMXB.

```
SPEX> par 1 1 nh v 4e-3
SPEX> par 1 1 t v 5e-4
SPEX> par 1 1 v v 1
SPEX> abun lodd
SPEX> par 1 2 n v 1
SPEX> par 1 2 t v 0.1
SPEX> par 1 3 n v 4000
SPEX> par 1 3 t0 v 0.1
SPEX> par 1 3 t1 v 20
SPEX> par 1 3 tau v 1
```

We fit the spectra with the suggested values for the parameters and obtain a reasonable fit with $\chi^2 \sim 1.6$. The high resolution of the spectra enables us to fit the elemental abundances, thus we leave the abundances of O, Ne, Mg and Fe in the *hot* component free, because their absorption edges fall inside the RGS energy band. The spectral fits are plotted in Figure 4.1. We also display the ignored regions for clarity purposes.

```
SPEX> par 1 1 08 s t
SPEX> par 1 1 10 s t
SPEX> par 1 1 12 s t
SPEX> par 1 1 26 s t
```

There are large residuals near the regions where we expect dust effects, as well as near 23.35 Å and 21.6 Å. Here we could expect the O II and O VII absorption lines due to the warm and hot ionized gas. These two plasmas represent the partially and fully ionized phases of the gas in the ISM and should be analyzed separately.

4.2.4 The warm-hot gas

To search for warm gas, we make a fit to the spectra adding columns of O II and Ne II into our model through a *slab* component. The *slab* model calculates the transmission of a layer of plasma with arbitrary composition. Free parameters are the intrinsic velocity dispersion and the column densities of the individual ions. Both O II and Ne II are first set to $1\times10^{20}\,\mathrm{m}^{-2}$, because the ionized gas usually contributes to 1-10% of the total column density. The velocity dispersion of the *slab* component is set to a nominal value of 50 km s^{-1} and should be a free parameter in the fit, but this is not necessary being the ISM lines unresolved. Each time an emission or absorption component is introduced, every component has to be related to the other once again.

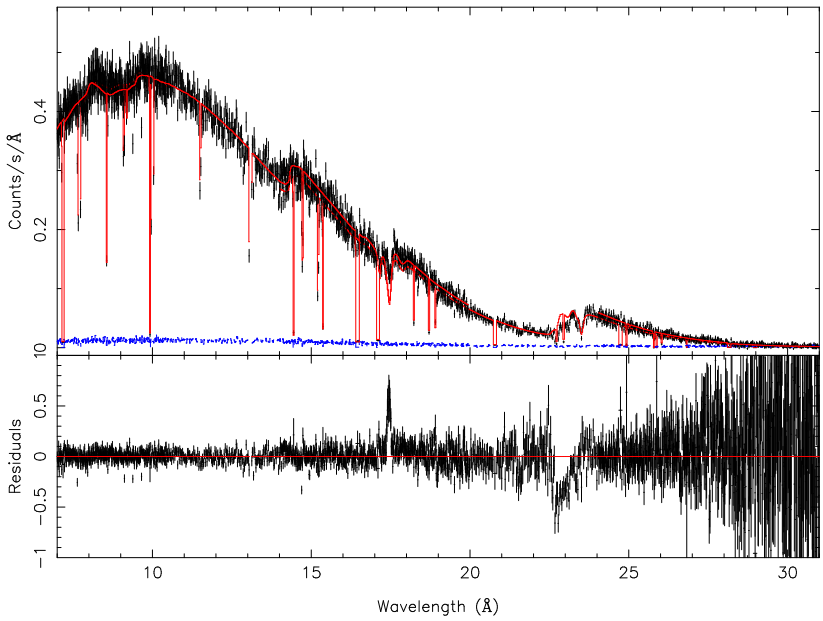

Figure 4.1: *RGS spectral fit for the LMXB. The model used is the simple one: hot * (bb + comt). The ISM is approximated by neutral gas.*

```
SPEX> com slab
SPEX> com rel 2:3 1,4
SPEX> par 1 4 v v 50
SPEX> par 1 4 o2 v 20
SPEX> par 1 4 ne02 v 20
```

The fit will confirm the presence of the warm gas providing interesting values for the O II and Ne II column densities. Then we can test the presence of hot ionized gas by fitting the O VII and Ne IX absorption lines. Thus we add an additional *slab* component with a higher velocity dispersion, for example 150 km s^{-1}. We leave as free parameters the column densities of O VII and Ne IX and choose a starting value of 1×10^{20} m^{-2} for both them.

```
SPEX> com slab
SPEX> com rel 2:3 1,4,5
SPEX> par 1 5 v v 150
SPEX> par 1 5 o7 v 20
SPEX> par 1 5 ne09 v 20
```

If we would deal with more physical models and obtain more information about the warm and hot phases of the gas, we can substitute the two *slab* components with two respective *hot* components. The *hot* components concerning the warm and hot gas should have column densities of about 1 and 2 orders of magnitude lower than the cold gas, respectively. The temperatures of the warm and hot gas should be around 1 and 2 orders of magnitude higher than the temperature of the cold gas. The 3-*hot* model gives results equivalent to the previously used 1-*hot* + 2-*slab* model.

4.2.5 Dust and Molecules

In §4.2.3 we have temporarily ignored the spectral regions between 22.5-23.2 Å and 17.2-17.7 Å, because they suffer from dust and molecular effects. Indeed, Figure 4.1 shows that the pure-gas model can not reproduce the features seen in these regions. Now, to solve the problem, we introduce two additional components in SPEX: *dabs* and *amol*. The *dabs* model accounts for the shielding of X-ray photons by

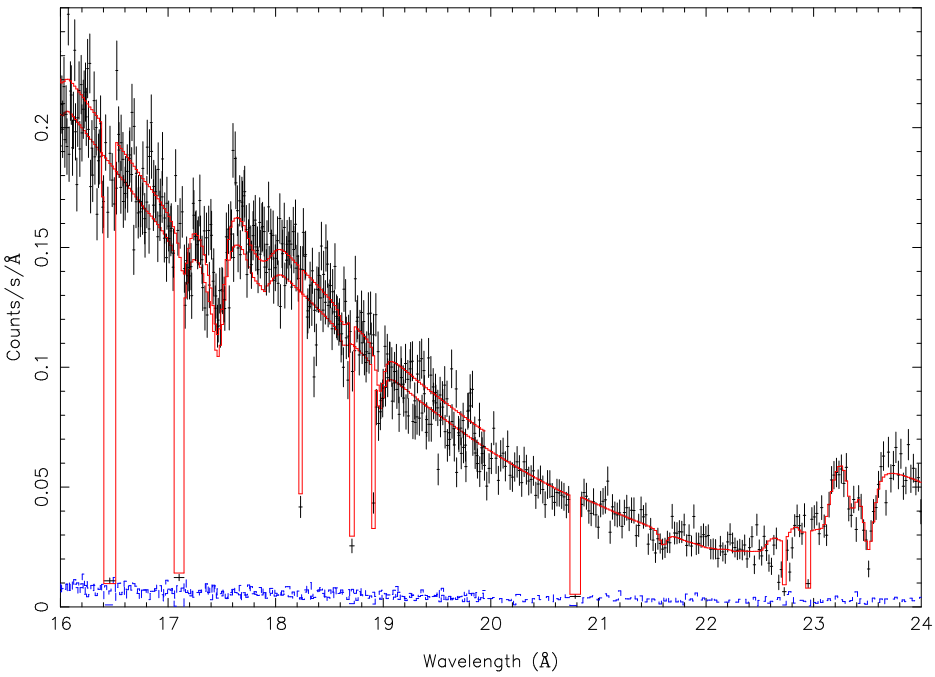

Figure 4.2: *Best fit model to our absorption spectrum. We focus on the spectral region spreading between the Fe L-edge and the O K-edge. The model used is the complete (3-hot + dabs + amol)*(bb + comt) model, that takes into account absorption from both gas, dust and molecules.*

dust grains, but uses the edge structure for the atomic gas. The *amol* model takes into account the modified edge structure around the O K-edge using measured cross sections of various compounds. We use the *amol* model to accounts for the oxygen solid phase, because in that case the shielding effect is weaker than the fine structure effects. Then we remove any contribution to the oxygen column from the *dabs* model by setting its respective abundance (i.e. depletion factor) equal to zero. The other parameters in the *dabs* model, such as the Fe and Mg abundances, are left as default. We also choose, as starting values, column densities of $7 \times 10^{25}\,\mathrm{m}^{-2}$ for the *dabs* component and $7 \times 10^{21}\,\mathrm{m}^{-2}$ for the *amol*. As oxygen compound we test the amorphous water ice, which refers to molecular index number 7 in the *amol* model:

```
SPEX> com dabs
SPEX> com amol
SPEX> com rel 2:3 1,4,5,6,7
SPEX> par 1 6 nh v 7e-3
SPEX> par 1 6 08 v 0
SPEX> par 1 7 no v 7e-7
SPEX> par 1 7 imol v 7
```

In Figure 4.2 we show the final result of the complete model in the spectral region that spread between the iron and oxygen edges. The complete model we have finally built takes into account all the main phases in the ISM that contribute to the absorption in the X-ray energy band. The cold gas gives the strongest contribution to the absorption and it is easy to be analyzed, while the warm and hot phases are not constrained at all. Dust and molecular features are more difficult to reproduce, especially because at the moment there are not complete models to account for all kinds of effects. *Dabs* model takes care only of the shielding effect by dust grains and works well with Fe edge. *Amol* model is currently good to fit the molecular effects inside the oxygen edge. Therefore both models are to be used carefully.

Chapter 5

Fitting RGS spectra of a group of galaxies

Author: Y. Grange (y.g.grange@sron.nl)
Keywords: Extended sources – RGS spectroscopy

5.1 Goal

XMM-Newton RGS spectra of extended sources usually show lines that have been broadened due to the design of this grating instrument. In this example, we show how SPEX can account for this effect using spectra from a group of galaxies. RGS operates without a slit. This means that all photons from within the field of view end up in the final spectrum, but not necessarily at the right wavelength. Only photons which are emitted in the centre of the group end up at the dispersion coordinate which corresponds to the correct wavelength. If the photon originates from the outskirts at an angle θ (projected on the dispersion axis) from the cluster centre, then the instrument will register it at a different dispersion coordinate and assign a wavelength to it which is shifted with respect to the true wavelength. Because of this effect, the line-emission appears to be broadened depending on the spatial extent of the source along the dispersion direction. This effect is illustrated in Figure 5.1.

To fit the spectra of these objects, this broadening needs to be taken care of by the fitted model. Since the broadening depends on the structure of the source emission, observations of the same source taken under different pointing angles (or: roll angles) do yield different broadening of the spectral lines. The *XMM-Newton* Users Handbook gives the equation for the broadening of a line in wavelength due to the spatial extent of the source. The equation is

$$\Delta\lambda = 0.138\Delta\theta/m. \tag{5.1}$$

Here, $\Delta\theta$ is the angular distance in arc minutes and m is the order of the spectrum. The wavelength shift, $\Delta\lambda$, is in Ångstrom.

In this example, we show how to model this broadening and how to simultaneously fit multiple observations with multiple position angles of the telescope. For this example, we will read in three different observations of the same source, of which two have the same roll angle and the third one having a different one. The goal of this example is to show a general method to deal with these kind of problems

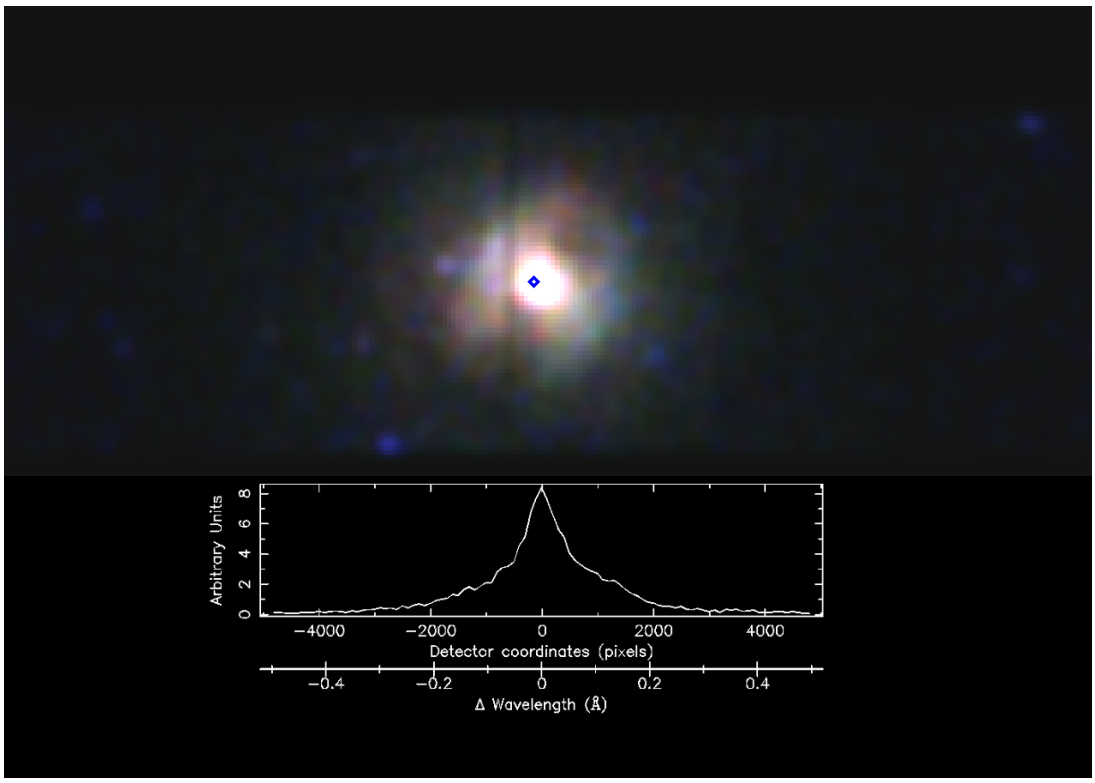

Figure 5.1: *EPIC image of NGC 5813 along the RGS extraction region. The plot shows the intensity versus detector coordinates. The bottom X-axis shows this coordinate converted to wavelength broadening. The diamond shape indicates the position at which the zero point of the RGS wavelength scale is taken.*

in SPEX. There may be better methods available for your specific case. Also the choice of models may need to be different in other cases.

5.2 SPEX solution

We actually have to solve two problems here. The first one is to model the line broadening of an extended source. The second problem to solve is that we want to simultanously fit two observations for which the line broadening differs. The first problem can be solved by using a model for the line broadening. In SPEX, this is taken care of by using the *lpro* model.

The input of the *lpro* model is an ascii file with the cumulative intensity profile of the source over the RGS band. For this we make an image of the MOS1 detector in the region $|DETX - SRCX| < 600$. Here $DETX$ is the X detector coordinate and $SRCX$ is the X detector coordinate of the source. From this we make a histogram (i.e. an intensity profile along the $DETY$ coordinate). This histogram should then be centered around the peak value and converted to wavelength broadening, using Eq. 5.1. From this data, we make a normalised cumulative profile (see fig. 5.2) and save it in an ascii file. We will later refer to this as the `vprof`-file.

To fit the data, we need to make a cumulative profile for each different roll angle whenever the source is not perfectly rotationally symmetric. In the case of clusters and groups of galaxies, such a symmetry is mostly not present. To simultanously fit the data obtained under diffferent roll angles, we use sectors in SPEX. Sectors are essentially model groups representing different areas or different components on the sky. In this case, we will create two sectors: one for each roll angle. The physical model will be simultanously fitted to the data sets, using a different `vprof`-file for each.

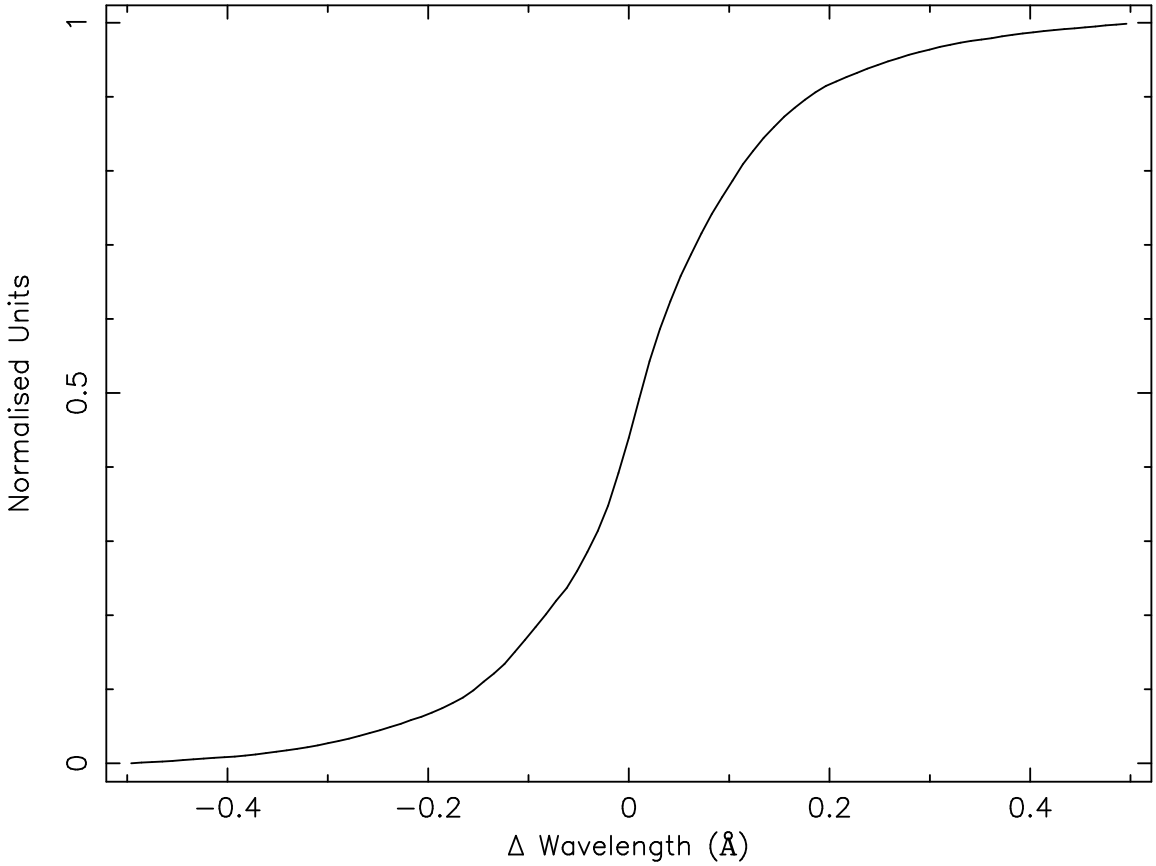

Figure 5.2: *Plot of the normalised cumulative profile of the example observation.*

5.2.1 Running trafo

In this trafo run, we will distribute the three spectra over the two sectors. Here we use the RGS, using 95% of the pulse height and excluding 95% in the background region. The background used for the fits is the RGS model background as produced by the standard RGS pipeline. After starting `trafo` we have to tell it that we want to transform three spectra in two sectors:

```
Program trafo: transform data to SPEX 2.0 format
This is version 1.01 of trafo

 Are your data in OGIP format            : type=1
 Old (Version 1.10 and below) SPEX format: type=2

 Enter the type: 1
 Enter the number of spectra you want to transform: 3
 Enter the maximum number of response groups per energy per spectrum: 1000000
 Enter the number of sectors you want to create: 2
```

The region number respresents the spectral data that we will fit. Because we have three different data sets, we will have to point each data set to a different region.

First, we enter the spectra for the first sector. We only show the most relevant input/output lines here.

```
 Enter the sector and region number: 1 1
 How should the matrix be partioned?
```

```
 Option 1: keep as provided (1 component, no re-arrangements)
 Option 2: rearrange into contiguous groups
 Option 3: split into N roughly equal-sized components
 Enter your preferred option (1,2,3): 3
 Enter number of components N (preferably a power of 2, between 8 and 32): 16

 Enter filename spectrum to be read: P0554680301R1S004SRSPEC1003.FIT
 Read nevertheless a background file? (y/n) [no]: yes
 Enter filename background spectrum to be read: P0554680301R1S004MBSPEC1000.FIT
 Enter filename response matrix to be read: P0554680301R1S004RSPMAT1003.FIT
 Read nevertheless an effective area file? (y/n) [no]: no
```

The first spectrum is now read in. We will now read in another spectrum, which is from data taken under the same roll angle. This data should therefore reside in the same sector, but in a different region.

```
 Enter the sector and region number: 1 2
 Enter filename spectrum to be read: P0554680201R1S004SRSPEC1003.FIT
 Read nevertheless a background file? (y/n) [no]: yes
 Enter filename background spectrum to be read: P0554680201R1S004MBSPEC1000.FIT
 Enter filename response matrix to be read: P0554680201R1S004RSPMAT1003.FIT
 Read nevertheless an effective area file? (y/n) [no]: no
```

Now we will add the observation that was taken using a different roll angle. The data of this observation should be put into a new region. Like for the other data sets, this data set will also be placed in a region of its own.

```
 Enter the sector and region number: 2 3
 Enter filename spectrum to be read: P0302460101R1S004SRSPEC1003.FIT
 Read nevertheless a background file? (y/n) [no]: yes
 Enter filename background spectrum to be read: P0302460101R1S004MBSPEC1000.FIT
 Enter filename response matrix to be read: P0302460101R1S004RSPMAT1003.FIT
 Read nevertheless an effective area file? (y/n) [no]: no
```

Save the spectrum by providing convenient names for the res and spo files.

```
 Enter filename spectrum to be saved (without .spo): all_R1
 Enter filename response to be saved (without .res): all_R1
```

5.2.2 Running SPEX

If the res and spo files are created, we are ready to run `spex`. In this description, we skip some very basic commands about, for example, plotting. See Chapter 1 for an overview of a basic SPEX session. First, we load the spectrum and plot it:

```
 Welcome user to SPEX version 2.02.01

SPEX> data all_R1 all_R1
...
SPEX> plot
```

Figure 5.3 shows a plot of the spectrum. For presentation purposes we rebin the spectrum here with a factor of 8 with the `bin` command. If C-statistics are used, binning is not strictly necessary. We will ignore some data at the low and high ends of the wavelength space. The spectrum, as presented by SPEX is shown in Figure 5.3.

Now we will set the model for both sectors. First we create a new sector, then we put in the model components.

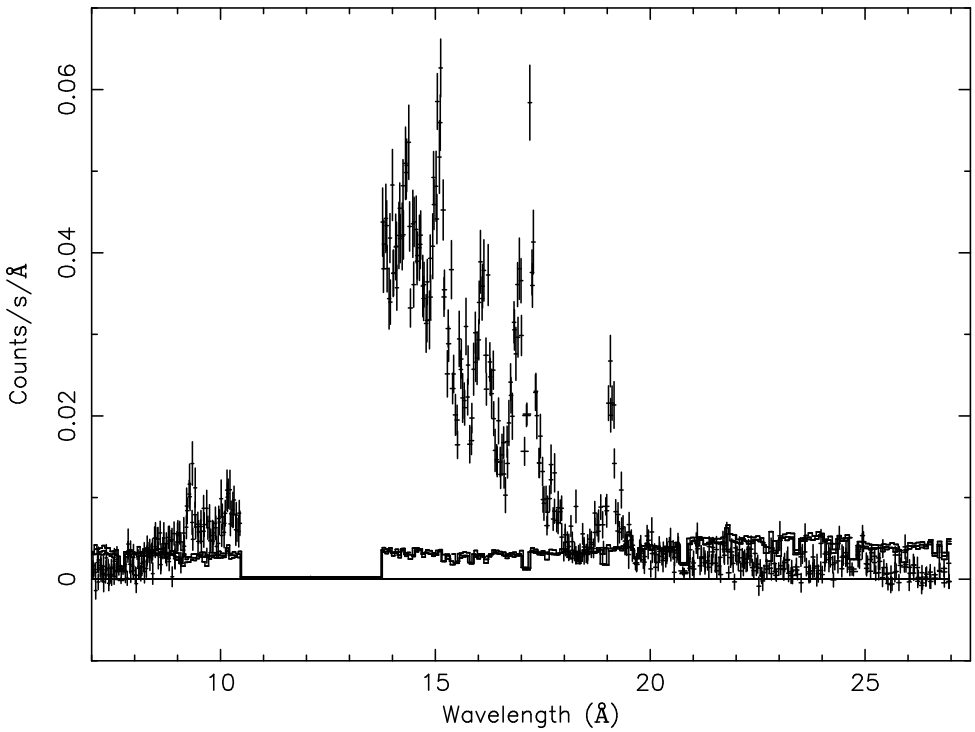

Figure 5.3: *The three RGS spectra without any fit. The crosses represent the data points, the histogram represents the model backgrounds.*

```
SPEX> sector new
There are 2 sectors
SPEX> com reds
SPEX> com hot
SPEX> com cie
SPEX> com lpro
```

Note that for each component, SPEX will tell you twice that you created this component (you created it for two sectors). In this model, we use a cosmological redshift, interstellar absorption, and a single-temperature model to describe the group emission. We plug in and fix the group redshift and hydrogen column density. We put the temperature of the hot model to its minimum (fixed). We also put in the relation between the different components in the same order as they affect the measured data. Furthermore we couple all parameters of all physical models, which we assume to be the same.

```
SPEX> distance 0.0064 z
SPEX> par 1 1 z v 0.0064
SPEX> par 1 2 n v 4.37e-4
SPEX> par 1 2 t v 5e-4
SPEX> par 1 2 n:t status f
SPEX> comp relation 3 1,2,4
SPEX> par 2 1 z:fl couple 1 1 z:fl
SPEX> par 2 2:3 n:file couple 1 2:3 n:file
```

Finally, we need to link the `lpro` models to the two `vprof`-files. This is done using

```
SPEX> par 1 4 file av 301.vprof.dat
SPEX> par 2 4 file av 101.vprof.dat
```

We use the same `vprof`-files for both observation under the same roll angle. The `lpro` model also contains two parameters which account for the fact that the emission peak used for the line bradening might be

slightly off. These parameters can also be let free to fit. Parameter `s` represents a scale factor for the width of the line broadening and `dlam` represents a shift in wavelength space.

```
SPEX> par 1:2 4 s status t
SPEX> par 1:2 4 dlam status t
```

Because we are fitting different RGS observations in different epochs, we should also leave the instrument normalisations (except for 1) thawn for the final fit.

```
SPEX> par -2:-1 1:3 n status t
SPEX> par -1 1 n status f
```

In Figure 5.4, the best fit using this method is compared to the same model without the line broadening model.

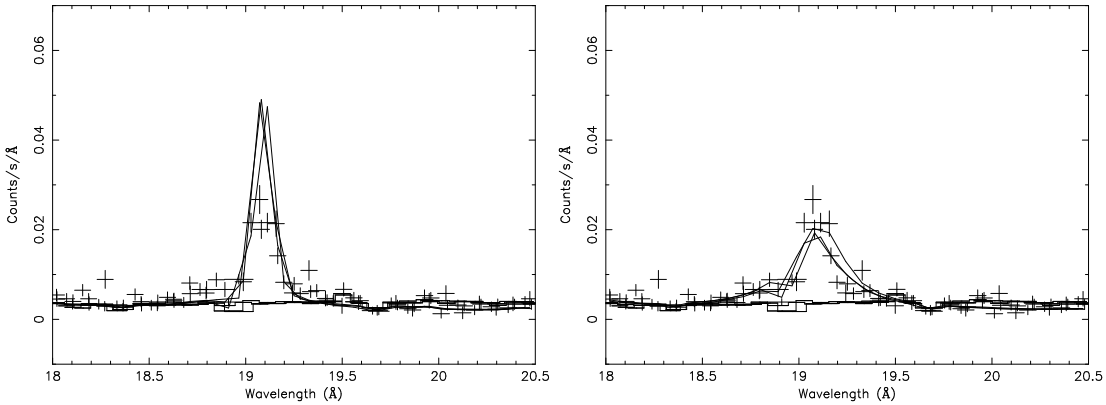

Figure 5.4: *Fit to the example data without (left) and with (right) the use of a line broadening model. The fits are shown around the* `O VIII Ly`α `line at 18.98 Å`.

Chapter 6

Sectors in AGN spectroscopy

Author: J. Ebrero (j.ebrero.carrero@sron.nl)
Keywords: Sectors – Active Galactic Nuclei – High-resolution spectroscopy

6.1 Goal

It is common to find in spectroscopy of active galactic nuclei (AGN, hereafter) several observations of the same source, or even different observations of different sources for which the observer would like to simultaneously fit to the same model (or different models) or couple certain parameters between observations. In this chapter we show how to create different sectors in SPEX and how to define and manipulate parameters in the particular case of an AGN. As in the majority of cases in this cookbook, this one is intended to be a general example on how to work with sectors in the context of SPEX. Note that, in general, the models used in other cases will be different from the ones used in this example.

6.2 Creating sectors

6.2.1 Introduction

In this example we will see how to analyze simultaneously three XMM-*Newton* RGS observations of an AGN using SPEX. The advantage of the following procedure is that the same physical model can be applied to all of the observations of the source, although they can be generally different, and that we could manipulate common parameters simultaneously and even couple them during an spectral fit. This can be easily done by using sectors, which represent different areas or components on the sky. Sectors must not be mistaken with detector regions that do not necessarily coincide with the sectors.

In the context of this particular example we will create 3 sectors (one for each observation of the AGN) and 6 regions (RGS1 and RGS2 for each observation). This is firstly done by running `trafo`.

6.2.2 Running trafo

In this trafo run we will actually create two single spectrum and response files that will be loaded by SPEX afterwards. The spectrum (and the response) file will contain the information of the different sectors and regions.

After starting `trafo` we tell the program that we want to transform six spectra in three sectors (that is, we will create a sector for each observation and the RGS1 and RGS2 spectrum of each of them will be stored together in the same sector):

```
Program trafo: transform data to SPEX 2.0 format
This is version 1.01 of trafo

 Are your data in OGIP format            : type=1
 Old (Version 1.10 and below) SPEX format: type=2

 Enter the type: 1
 Enter the number of spectra you want to transform: 6
 Enter the maximum number of response groups per energy per spectrum: 1000000
 Enter the number of sectors you want to create: 3
```

Note that it could be perfectly possible to create a single sector for each RGS spectrum but we decided to place both RGS spectra in their respective observation sector to increase the signal-to-noise ratio while fitting. Any other combination of spectra and sectors are also possible depending on the scientific interests of the user.

Now the program will ask for the sector and region number that will be assigned to the first spectrum. We will have to provide the spectrum file name as well as the background and the response matrix files:

```
 Enter the sector and region number: 1 1
 Enter filename spectrum to be read: obs1_R1_spec.FIT
 Read nevertheless a background file? (y/n) [no]: y
 Enter filename background spectrum to be read: obs1_R1_bkg.FIT
 Enter filename response matrix to be read: obs1_R1_rsp.FIT
 Read nevertheless an effective area file? (y/n) [no]: n
```

The first spectrum is read in and now we proceed to the second one, which will be the RGS2 spectrum of the first observation. Therefore, we will assign the first sector and the second region (as it is a different detector region) to this spectrum:

```
 Enter the sector and region number: 1 2
 Enter filename spectrum to be read: obs1_R2_spec.FIT
 Read nevertheless a background file? (y/n) [no]: y
 Enter filename background spectrum to be read: obs1_R2_bkg.FIT
 Enter filename response matrix to be read: obs1_R2_rsp.FIT
 Read nevertheless an effective area file? (y/n) [no]: n
```

We now proceed to read in the spectra of the second observation in the second sector:

```
 Enter the sector and region number: 2 3
 Enter filename spectrum to be read: obs2_R1_spec.FIT
 Read nevertheless a background file? (y/n) [no]: y
 Enter filename background spectrum to be read: obs2_R1_bkg.FIT
 Enter filename response matrix to be read: obs2_R1_rsp.FIT
 Read nevertheless an effective area file? (y/n) [no]: n
 Enter the sector and region number: 2 4
 Enter filename spectrum to be read: obs2_R2_spec.FIT
 Read nevertheless a background file? (y/n) [no]: y
 Enter filename background spectrum to be read: obs2_R2_bkg.FIT
 Enter filename response matrix to be read: obs2_R2_rsp.FIT
 Read nevertheless an effective area file? (y/n) [no]: n
```

And we finish with the spectra of the third observation:

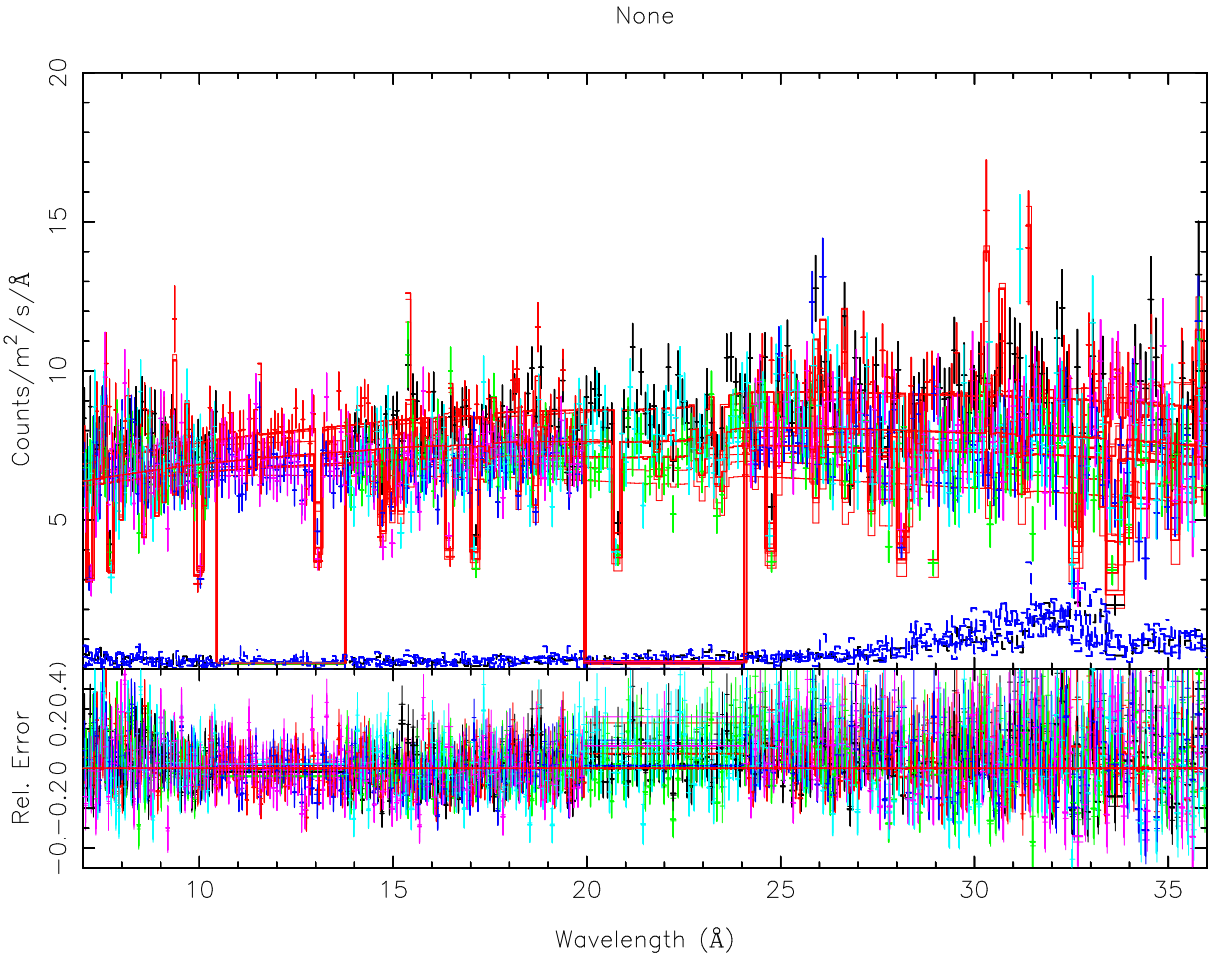

Figure 6.1: *Best fit to a single power law.*

```
Enter the sector and region number: 3 5
Enter filename spectrum to be read: obs3_R1_spec.FIT
Read nevertheless a background file? (y/n) [no]: y
Enter filename background spectrum to be read: obs3_R1_bkg.FIT
Enter filename response matrix to be read: obs3_R1_rsp.FIT
Read nevertheless an effective area file? (y/n) [no]: n
Enter the sector and region number: 3 6
Enter filename spectrum to be read: obs3_R2_spec.FIT
Read nevertheless a background file? (y/n) [no]: y
Enter filename background spectrum to be read: obs3_R2_bkg.FIT
Enter filename response matrix to be read: obs3_R2_rsp.FIT
Read nevertheless an effective area file? (y/n) [no]: n
```

After all of the spectra have been read in, we now save the spectrum by providing convenient names for the res and spo files that will be provided to SPEX:

```
Enter filename spectrum to be saved: AGN.spo
Enter filename response to be saved: AGN.res
```

6.3 Running SPEX

Once the res and spo files have been created we will proceed to run `spex`. As in the rest of this cookbook we will skip some basic commands, such as plotting commands, that might be found either in Chapter 1 or in the SPEX manual.

First of all we launch SPEX and load the spectrum. Since we have created 3 different sectors in the previous `trafo` run we must tell SPEX to do the same. Loading the spectrum creates one sector by default so we have to create two more:

```
 Welcome user to SPEX version 2.01.05

SPEX> data AGN AGN
SPEX> sector new
There are 2 sectors
SPEX> sector new
There are 3 sectors
```

We now rebin the spectrum for presentation purposes. Note that if C-statistics are used no binning is strictly necessary. However, it is customary to rebin by a factor of at least 3 in order to avoid oversampling of the spectrum. Furthermore, we will ignore those parts of the spectrum below 7 Å and above 36 Å, where the signal-to-noise ratio is too low. In general the user may need to re-define the good part of the spectrum depending on the quality of the data and the specific purposes of the observation.

```
SPEX> ig 1.e-4:7 u a
SPEX> ig 36:1.e4 u a
SPEX> bin 7:36 3 u a
```

Now, we set up the distance the source is at, and the basic shape of the continuum which is a redshifted power-law. We can just load the components but note that we must explicitly say that we want them to be loaded in all three sectors; otherwise, the components would be loaded only in sector 1. We also specify an initial guess for the normalization and slope of the power-law:

```
SPEX> dist 0.0305 z
SPEX> com red
SPEX> par 1:3 1 z v 0.0305
SPEX> com po
SPEX> com rel 2 1
SPEX> par 1:3 2 n v 2.e8
SPEX> par 1:3 2 ga v 2.
```

We may want to see what happens if we fit the spectrum now. To do that we set C-statistics and the verbosity of the fit:

```
SPEX> fit method cstat
SPEX> fit print 1
SPEX> cal
SPEX> fit
```

After the last command the redshifted power-law is fitted simultaneously in all three sectors and the results for each observation will be displayed on the screen. In Figure 6.1 we also see that the spectrum is not correctly fitting several absorption features. In the particular case of this source, we know from the literature that some of the absorption lines are caused locally in our own Galaxy and can be modeled by two `hot` components:

```
SPEX> com hot
SPEX> com rel 2 1,3
```

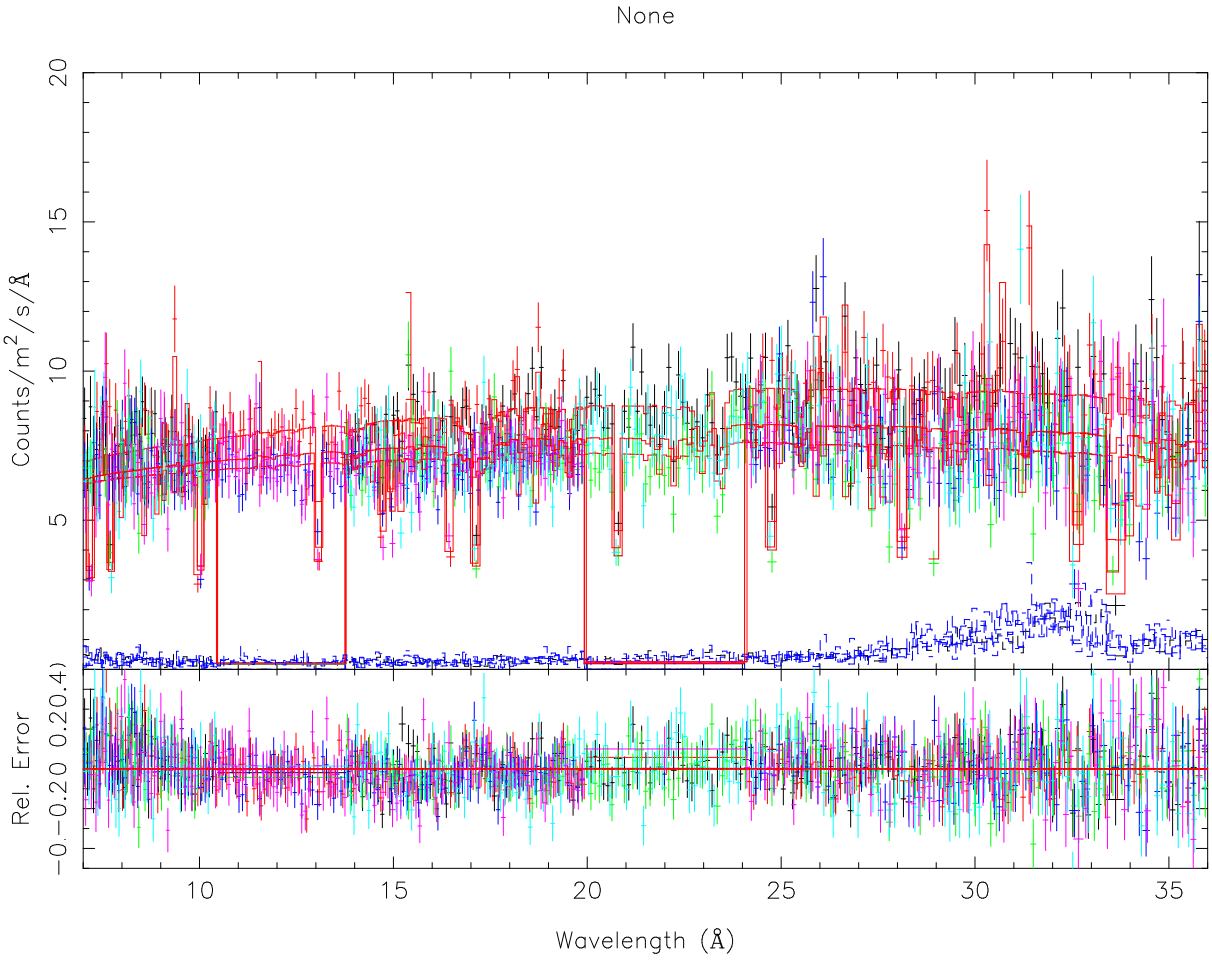

Figure 6.2: *Best fit to a single power law plus two* `hot` *and one* `xabs` *components.*

```
SPEX> par 1:3 3 t v 5.e-4
SPEX> par 1:3 3 nh v 1.41e-4
SPEX> par 1:3 3 t s f
SPEX> par 1:3 3 nh s f
SPEX> com hot
SPEX> com rel 2 1,3,4
SPEX> par 1:3 4 t v 7.e-3
SPEX> par 1:3 4 nh v 1.92e-5
SPEX> par 1:3 4 t s f
SPEX> par 1:3 4 nh s f
SPEX> cal
SPEX> fit
```

After the last sequence of commands we have added two `hot` components. Since the values for the temperature and the column density are known from the literature we have frozen these parameters so that we do not have unnecessary free parameters.

We will fit the rest of absorption features using the photoionised absorption model `xabs`. This component requires an ascii-file as input with pre-calculated ionic column densities versus the ionisation parameter. If not provided, `xabs` will use a default list but be aware that in this case the model could fail to provide an acceptable fit (see `SPEX` manual). In this example we have used an ascii-file obtained after a Cloudy run with the spectral energy distribution of Mrk 279:

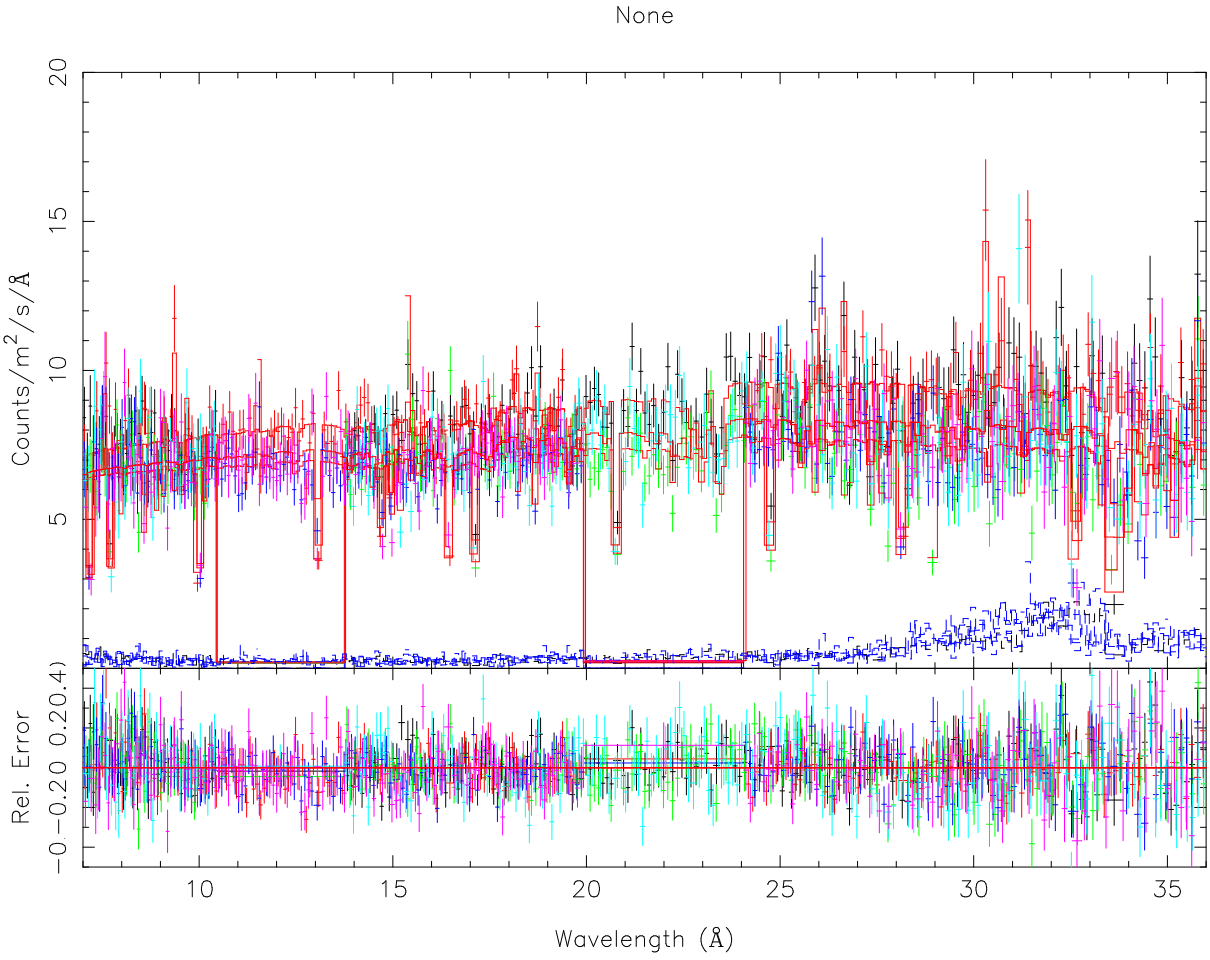

Figure 6.3: *Best fit to a single power law plus two* `hot` *and two* `xabs` *components.*

```
SPEX> com xabs
SPEX> com rel 2 5,1,3,4
SPEX> par 1 5 xi v 0.5
SPEX> par 1 5 nh 7.e-5
SPEX> par 1 5 zv v -50.
SPEX> par 1 5 zv s t
SPEX> par 1 5 v v 50.
SPEX> par 5 col avalue crespex_mrk279.dat
```

We may choose to fit the model in each observation (sector) independently. However, in this particular case we will couple the parameters of the `xabs` model of the sectors 2 and 3 to those of sector 1 that we have defined in the sequence of commands shown above. We have assigned initial values to the ionisation parameter `xi`, the column density `nh`, and the outflow velocity `zv`. The latter is frozen by default but we have decided to thawn it. The root mean square velocity of the lines `v` is also frozen and taken from the literature. Coupling parameters between sectors is easy:

```
SPEX> par 2:3 5 v v 50.
SPEX> par 2:3 5 zv s t
SPEX> par 2:3 5 nh couple 1 5 nh
SPEX> par 2:3 5 xi couple 1 5 xi
SPEX> par 2:3 5 zv couple 1 5 zv
```

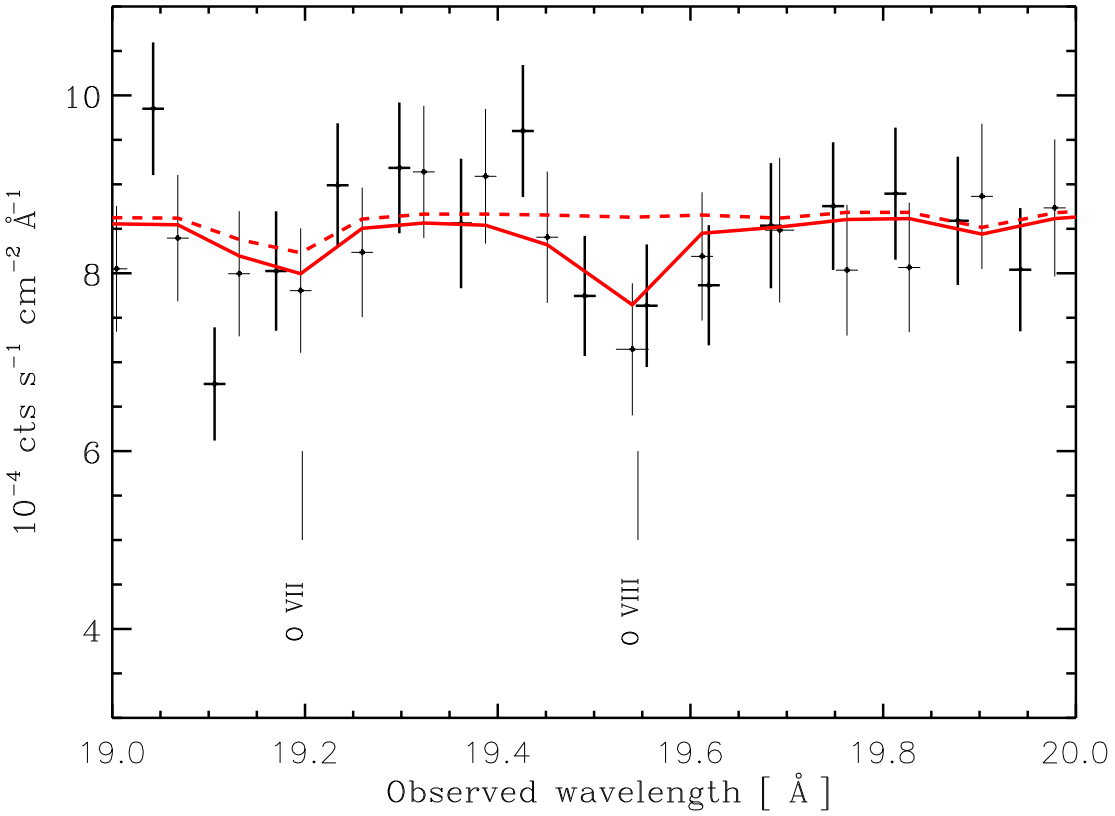

Figure 6.4: *Detail of the spectrum in the* O VIII *area. The dashed line is the best fit model using only* `xabs` *component, while the solid line corresponds to the fit with two* `xabs` *components.*

```
SPEX> cal
SPEX> fit
```

The fit has improved but it is not good yet (see Figure 6.2). Some of the lines are not well fitted, which might be a hint that perhaps and additional high-ionisation component is needed. Therefore, we add an extra `xabs` model:

```
SPEX> com xabs
SPEX> com rel 2 6,5,1,3,4
SPEX> par 1 6 xi v 2.5
SPEX> par 1 6 nh 4.e-4
SPEX> par 1 6 zv v -350.
SPEX> par 1:3 6 zv s t
SPEX> par 1:3 6 v v 50.
SPEX> par 6 col avalue crespex_mrk279.dat
SPEX> par 2:3 6 xi couple 1 6 xi
SPEX> par 2:3 6 nh couple 1 6 nh
SPEX> par 2:3 6 zv couple 1 6 zv
SPEX> cal
SPEX> fit
```

After adding the second `xabs` component the fit has greatly improved (see Figure 6.3); the most relevant absorption features are now correctly fitted (see Figure 6.4). However, it is still obvious that

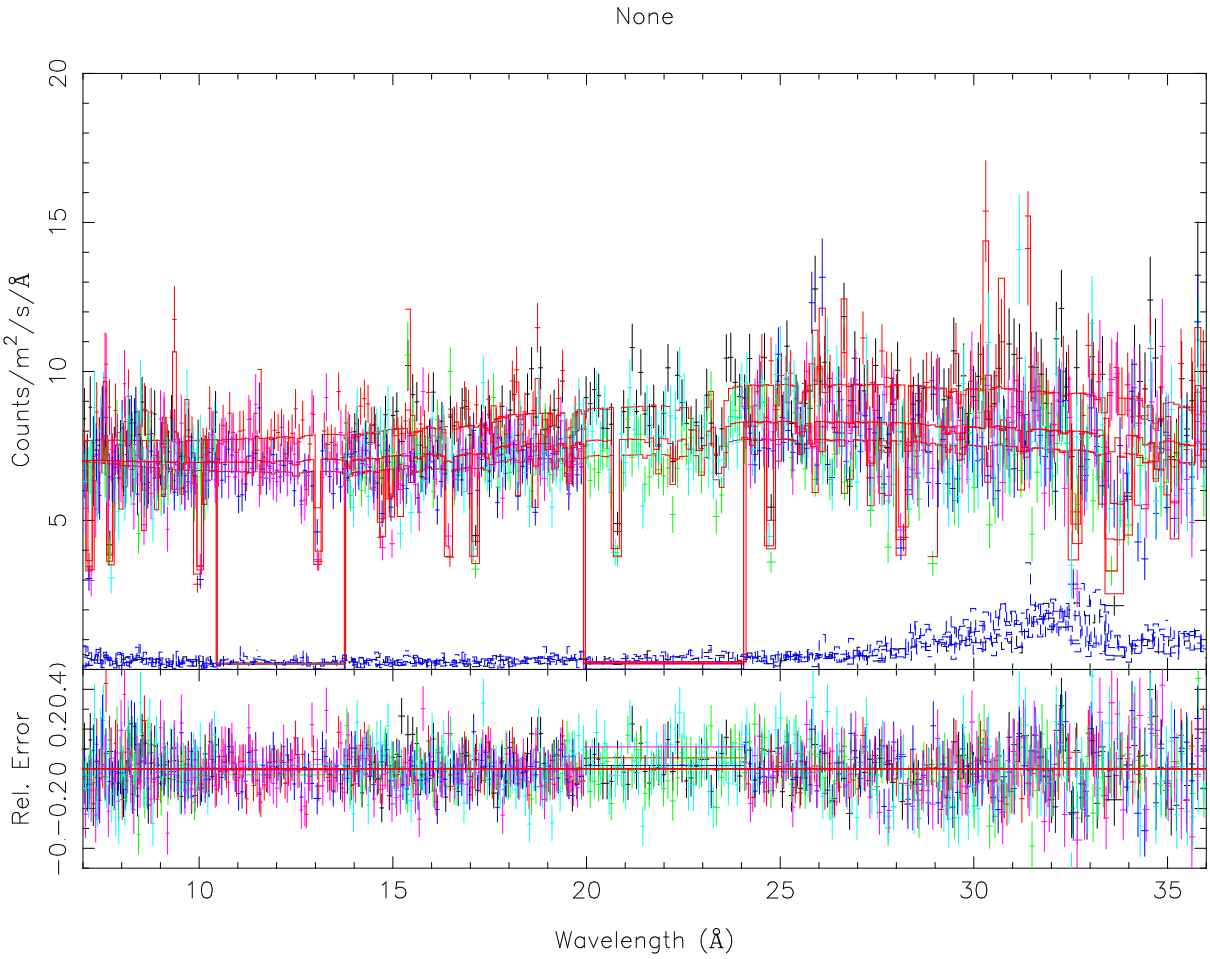

Figure 6.5: *Best fit to a single power law with a modified black body plus two* `hot` *and two* `xabs` *components.*

we are missing an excess at softer X-rays, frequently found in AGN spectra, that is usually modeled as a black body. Here we will use a black body spectrum modified by coherent Compton scattering (`mbb` model):

```
SPEX> com mbb
SPEX> com rel 7 6,5,1,3,4
SPEX> par 1:3 7 n v 7.e6
SPEX> par 1:3 7 t v 0.14
SPEX> cal
SPEX> fit
```

The last sequence of commands add the `mbb` component to all three components with the same initial guess for the normalization and temperature of the black body.

Chapter 7

High-energy exponential cut-off power-law in SPEX

Author: M. Mehdipour (M.Mehdipour@sron.nl)
Keywords: Active Galactic Nuclei – High-resolution spectroscopy

The `etau` model in `SPEX` calculates the simple transmission $T(E) = \mathrm{e}^{-\tau(\mathrm{E})}$, where the optical depth $\tau(E) = \tau_0 E^a$. Note that τ_0 is the optical depth at $E = 1$ keV and is represented by the `tau0` parameter in the model. In `SPEX` by multiplying the `etau` model by a simple power-law (`pow`), a high-energy exponential cut-off power-law (like the `cutoffpl` model in `XSPEC`) can be produced. In order to do this, the value of the τ_0 parameter needs to be specified while $a = 1$. In this case, for an e-folding energy of exponential roll-off (β) in keV, $\tau_0 = 1/\beta$. So for example, to obtain a power-law with an exponential roll-off at 100 keV, just set τ_0 to 0.01 and a to 1. Figure 7.1 shows such a power-law produced in `SPEX` with photon index $\Gamma = 2$ and $\beta = 100$ keV.

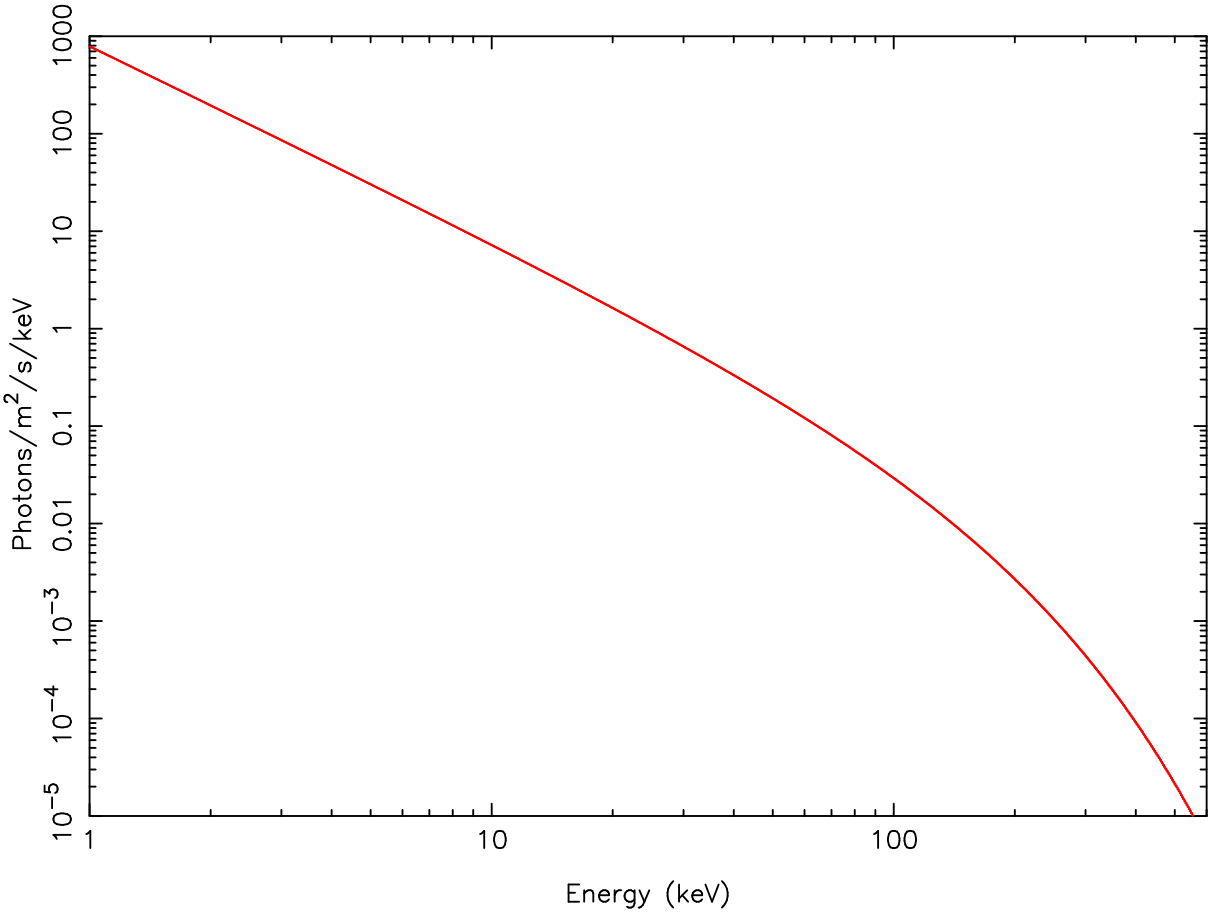

Figure 7.1: *A power-law with* $\Gamma = 2$ *and an exponential roll-off at 100 keV, produced by multiplying the* `pow` *and* `etau` *models in* `SPEX`.

Chapter 8

Modeling particle background

Author: J. de Plaa (j.de.plaa@sron.nl)
Keywords: Background analysis – Extended sources – CCD spectroscopy

8.1 Goal

Time dependent particle backgrounds in X-ray spectra are very difficult to correctly subtract, especially for extended sources. Many times, a quiescent particle background remains present in the spectrum after flare filtering. In this example, we show how to model the quiescent soft-proton contribution in a spectrum extracted from an annulus around the core of a cluster of galaxies. The difficulty here is that the effective area for soft protons is very different from the effective area for X-rays. Please note that the example provided is not necessarily scientifically correct. The goal of this example is to show a general method to deal with these kind of problems in SPEX. The choice of models probably needs to be different in other cases.

8.2 SPEX solution

The basic problem we have to solve here, is that we need a number of model components that are folded through the ARF (the cosmic X-rays) and a few components describing the particle background, which are not folded through the ARF. In SPEX, this can be solved using sectors. Sectors are essentially model groups representing different areas or different components on the sky. In this case, we will create two sectors: one for the cosmic X-rays and one for the particle background. The second sector should not be folded through the ARF. To archieve this, we have to create a special spectrum and response file with `trafo` in which we define the sectors.

8.2.1 Running trafo

In this trafo run, we will actually load the same spectrum twice. One for every sector. Here we use a MOS1 spectrum extracted from an annulus between 6 and 9 arcmin from the cluster core. The background spectrum was extracted using the XMM Extended Source Analysis Software by Snowden & Kuntz. After starting `trafo` we have to tell it that we want to transform two spectra in two sectors:

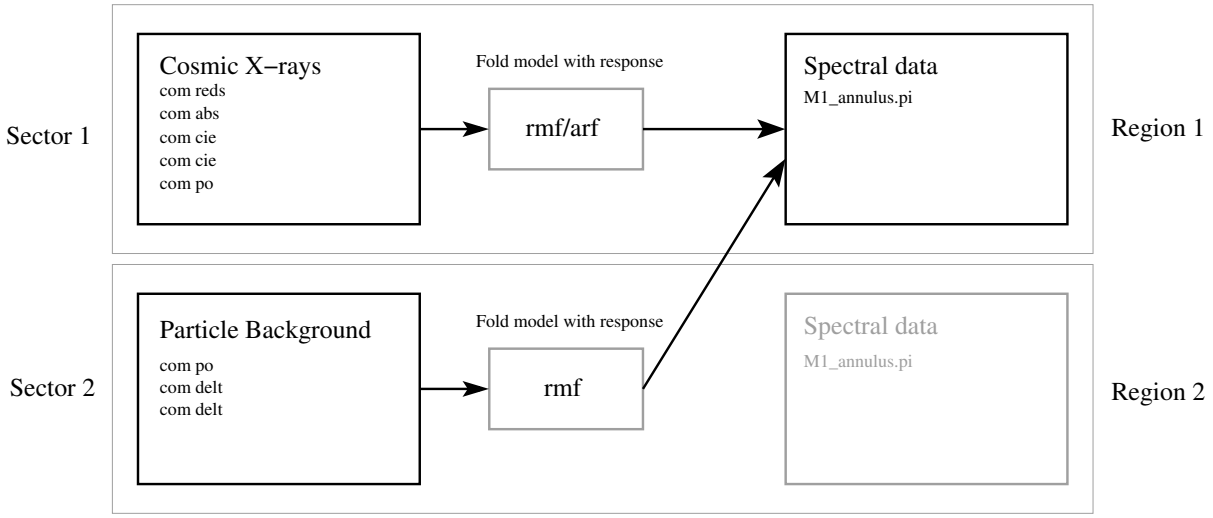

Figure 8.1: *Schematic representation of the sectors and regions in this example. We load two spectra with trafo and we define two sectors (left). The exact model components are later defined in SPEX. The models for sectors 1 and 2 are folded through the response matrix separately. The result of the folding is added and applied to the first spectrum (region 1 on the right) only.*

```
Program trafo: transform data to SPEX 2.0 format
This is version 1.02, of trafo

 Are your data in OGIP format            : type=1
 Old (Version 1.10 and below) SPEX format: type=2

 Enter the type: 1
 Enter the number of spectra you want to transform: 2
 Enter the maximum number of response groups per energy per spectrum: 1000000
 Enter the number of sectors you want to create: 2
```

The region number respresents the spectral data that we will fit. Because we want to add the cosmic X-ray spectrum and the particle background spectrum, we want both sectors to point to region 1. First, we enter the spectra for the first sector. We only show the most relevant input/output lines here.

```
 Enter the sector and region number: 1 1
 How should the matrix be partioned?
 Option 1: keep as provided (1 component, no re-arrangements)
 Option 2: rearrange into contiguous groups
 Option 3: split into N roughly equal-sized components
 Enter your preferred option (1,2,3): 1
 Enter filename spectrum to be read: M1_annulus.pi
 Read nevertheless a background file? (y/n) [no]: y
 Enter filename background spectrum to be read: M1_annulus_bkg.pi
 Shall we ignore bad channels? (y/n) [no]:y
 Enter filename response matrix to be read: M1_annulus.rmf
 Enter new bin boundary values manually: 3.E-5 5.E-3
 Enter shift to response array (1 recommended, but some cases may be 0):1
 Read nevertheless an effective area file? (y/n) [no]: y
 Enter filename arf-file to be read: M1_annulus.arf
```

The first spectrum is now read in, including an ARF file. Now we enter the same spectrum again, but now without ARF. The region number here is 1, because we want the models in this sector to be added

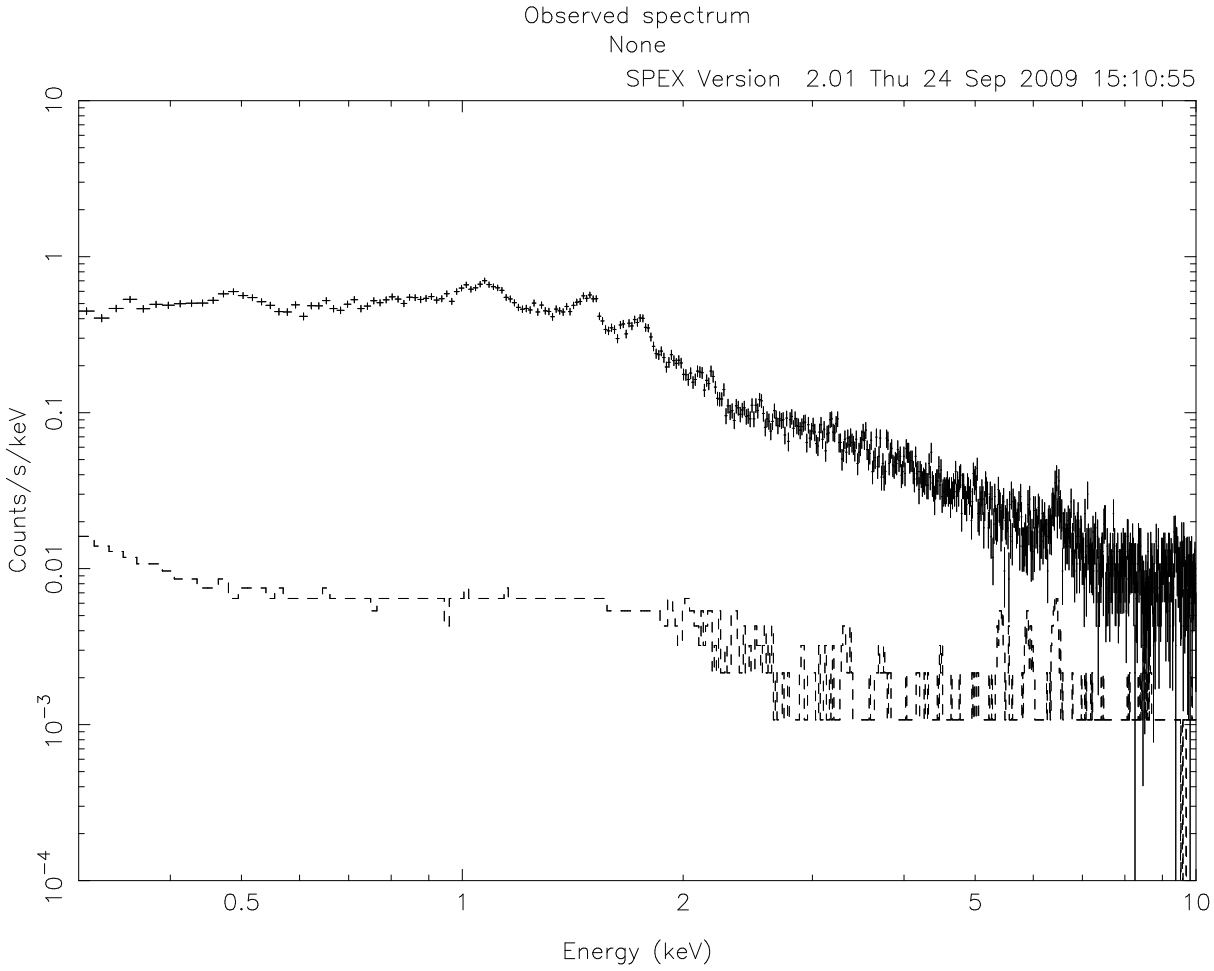

Figure 8.2: *XMM-Newton MOS1 spectrum extracted from a 6–9 arcmin annulus around a cluster of galaxies.*

to the models of sector 1.

```
Enter the sector and region number: 2 1
Enter your preferred option (1,2,3): 1
Enter filename spectrum to be read: M1_annulus.pi
Read nevertheless a background file? (y/n) [no]: y
Enter filename background spectrum to be read: M1_annulus_bkg.pi
Enter filename response matrix to be read: M1_annulus.rmf
Read nevertheless an effective area file? (y/n) [no]: n
```

Save the spectrum by providing convenient names for the res and spo files.

```
Enter filename spectrum to be saved (without .spo): M1_annulus
Enter filename response to be saved (without .res): M1_annulus
```

8.2.2 Running SPEX

If the res and spo files are created, we are ready to run `spex`. In this description, we skip some very basic commands about, for example, plotting. See Chapter 1 for an overview of a basic SPEX session. Note that the procedure for SPEX version 2.02 is exactly the same as for SPEX version 2.01, which is shown here. First, we load the spectrum and plot it:

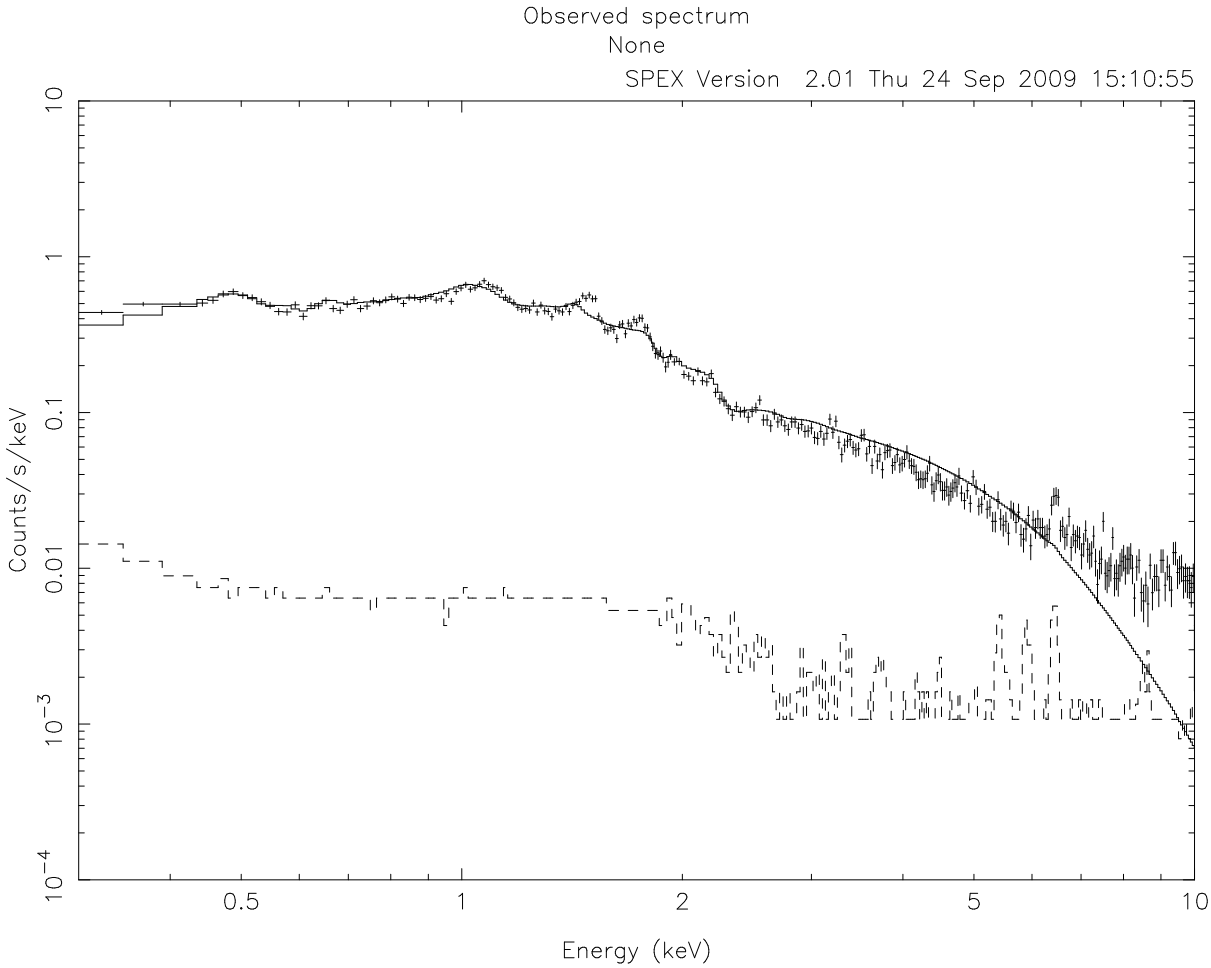

Figure 8.3: *A fit without modeling the particle background is not successful. Especially, the high-energy region in the spectrum is not fitted well due to soft protons.*

```
 Welcome user to SPEX version 2.01.05

SPEX> data M1_annulus M1_annulus
...
SPEX> plot
```

Figure 8.2 shows a plot of the spectrum. For presentation purposes we rebin the spectrum here with the `obin` command. If C-statistics are used, binning is not strictly necessary. An important thing to remember at this point is to ignore the spectrum in region number 2:

```
SPEX> ign reg 2 1:1000000
```

We ignore region 2 from channel 1 to 1000000, which should be more then enough to make sure no data is left in the region. Of course, some data at very low and high energies also need to be ignored in region 1.

Now, we set up the cosmic X-ray model for sector 1. We can just load the components normally, because they are automatically added to the first sector:

```
SPEX> com reds
SPEX> com abs
SPEX> com cie
```

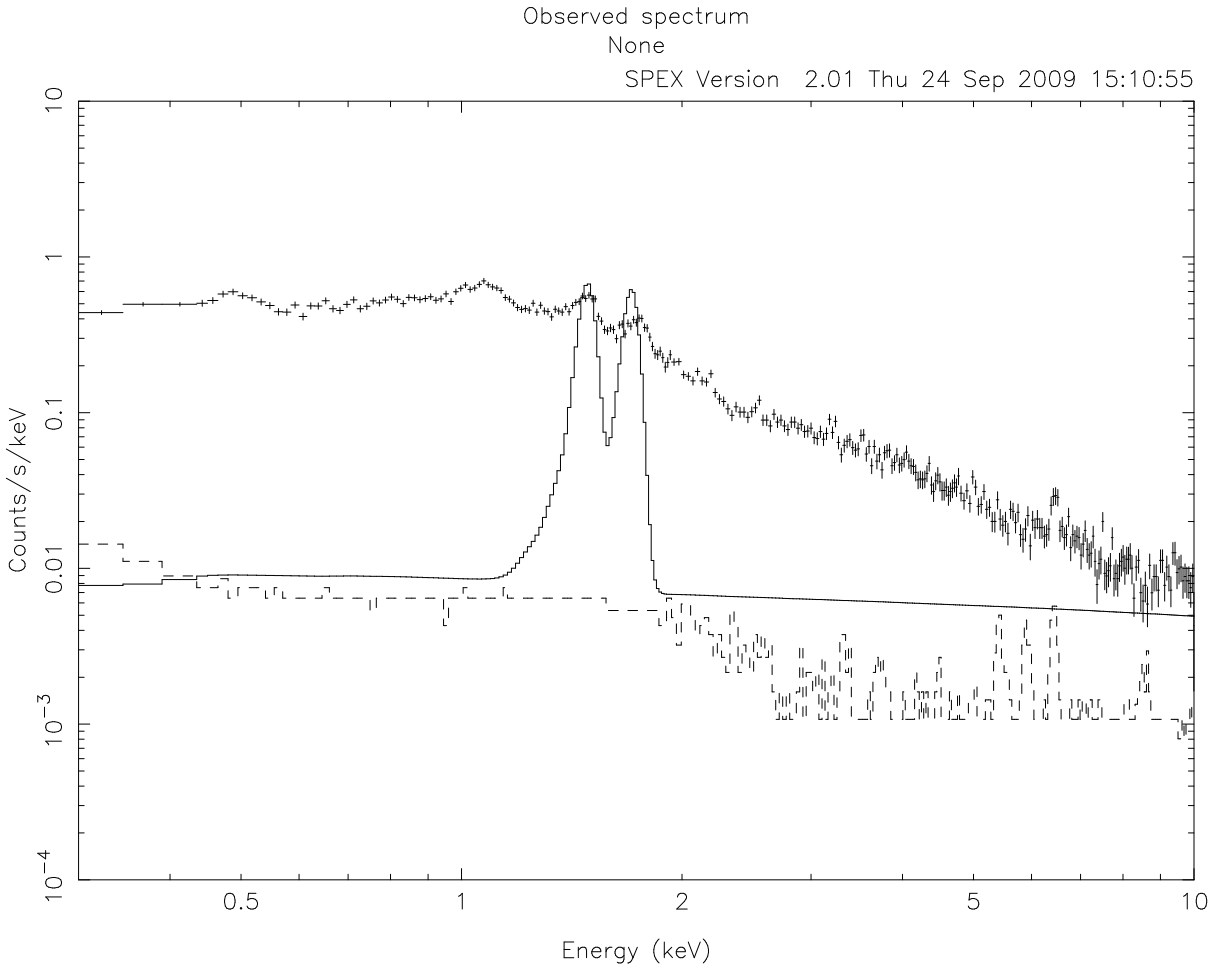

Figure 8.4: *Here, we plot the particle background model. We ignore the cluster model components for now. It is clear to see that the power law is not folded through the arf.*

```
SPEX> com cie
SPEX> com po
SPEX> com rel 3 1,2
SPEX> com rel 5 1,2
```

In this model, we put a cosmological redshift, interstellar absorption, and a single-temperature model to describe the cluster emission. In addition, we put a single-temperature model with a fixed temperature of 0.2 keV to model the emission from the local hot bubble, and a power law with a gamma value of 1.41 to account for the Cosmic X-ray Backgound (CXB) due to unresolved point sources.

```
SPEX> par 1 4 t v 0.2
SPEX> par 1 4 t s f
SPEX> par 1 5 gamm v 1.41
SPEX> par 1 5 gamm s f
```

Just to show what happens if we fit the data now, we plot the result in Figure 8.3. It is clear that the spectrum is not well fitted at low and high energies. A contribution of soft protons is visible at the high-energy end of the spectrum. In addition, we see that the instrumental fluoresence lines of Al and Si at $\sim$1.49 and $\sim$1.75 keV are not fitted. To model these features, we need to use the second sector and define an additional model there.

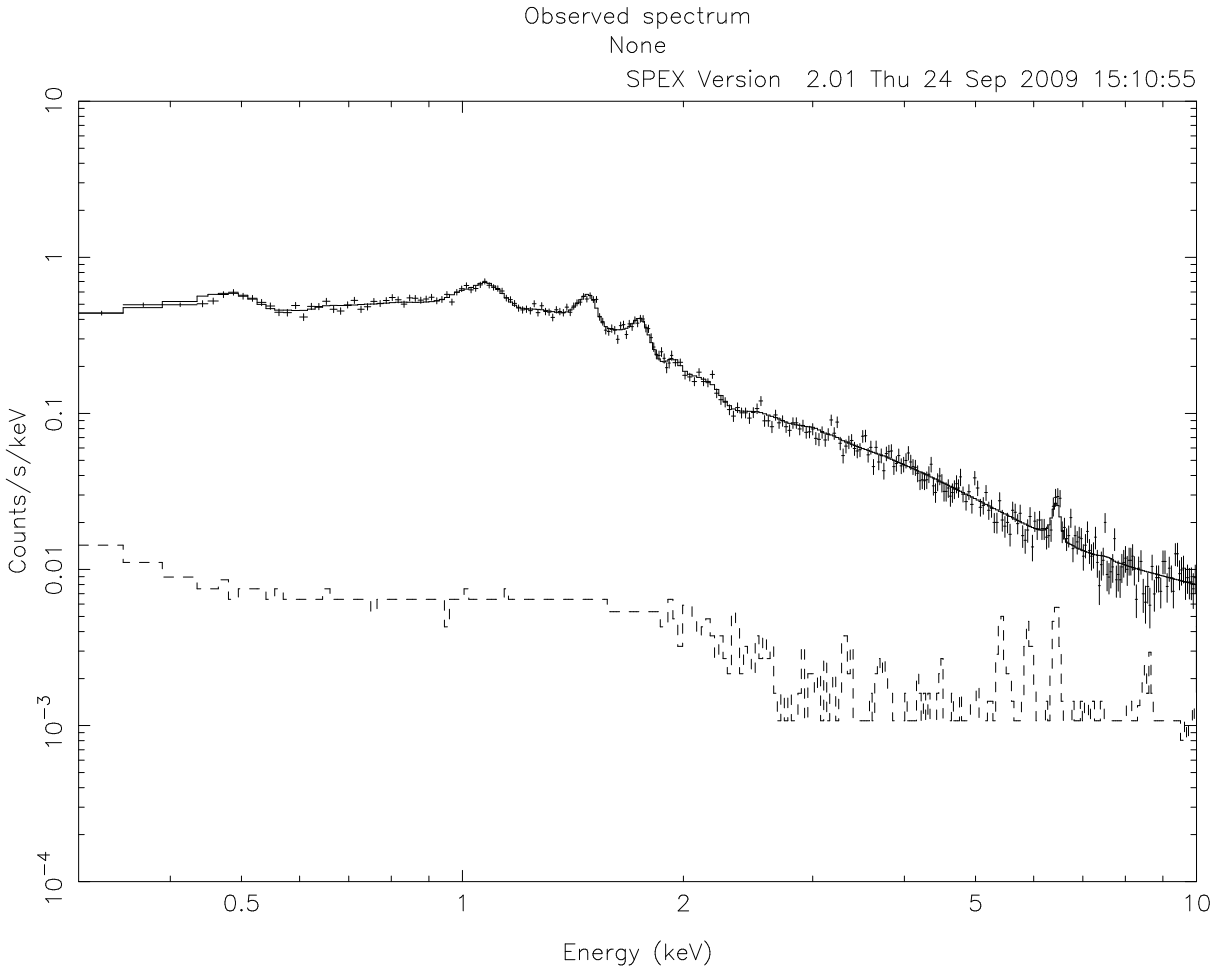

Figure 8.5: *Best fit model to our example spectrum. The particle background model has been able to fit the discrepancies at high energies.*

```
SPEX> sector new
SPEX> com 2 po
SPEX> com 2 delt
SPEX> com 2 delt
SPEX> par 2 1 gamm v 0.2
SPEX> par 2 2 e v 1.49
SPEX> par 2 2 e s f
SPEX> par 2 3 e v 1.75
SPEX> par 2 3 e s f
```

In this sequence of commands, we define a new sector (number 2) and add a power-law and two delta-line components to it. The slope of the gamma-parameter is initially set to ∼0.2. In Figure 8.4, we put the components in sector 1 to zero to show the particle background model that we have just defined. The flat shape of the power-law model confirms that these components are not folded through the arf.

When we reset the components in sector 1 to their initial values we can start fitting. In Figure 8.5, we show the best fit using this model. The contribution of soft-protons at high energies is now being accounted for by the power law.

WARNING! *The example above uses a simplified model of the X-ray background. Background subtraction for extended sources is complicated and subject of continuous research. Please be very careful in selecting model components and deciding which parameters can be left free.*

Chapter 9

User model

Authors: J. de Plaa & Liyi Gu
keywords: User models – Xspec integration

9.1 Goal

Some users would like to add their own models to the SPEX program, similar to the local model option in Xspec. Since the developers of SPEX do not have the man power to implement all possible models out there in a SPEX variety, we have devised a quick way of running external models in SPEX.

9.2 SPEX solution

SPEX contains two user models 'user' and 'musr' for additive and multiplicative models respectively. These model components communicate with the external model through an input and output file containing the model parameters, the energy grid and the calculated spectrum. By creating a program that reads in the input parameters and energy grid, calculates the spectrum, and writes the result to an output file in the right format, users have a very flexible way of using any model in SPEX.

Since this setup offers many possibilities, we can only show a few examples of how these user model components can be used. We can think of the following options, but there are more:

- Write a small wrapper to plug-in an Xspec local model into SPEX (see example).
- Create an interface between the SPEX user model and Xspec through pyXspec or TCL (see example).
- Write a small program to interpolate a (given) table model.
- Write a custom model yourself.

All options require a basic level of programming, but usually nothing too fancy. In principle, the user model can run any executable that can read and write ASCII files and accepts command line arguments. This allows for great flexibility in the choice of programming language.

The amount of programming can be reduced when you use the example programs that we provide on the SPEX web site. In this cookbook we provide a few examples.

9.2.1 Importing an Xspec local model written in Fortran (ismabs)

In this example, we include the Xspec local model called ismabs (Gatuzz et al., 2014) into a SPEX user model executable. The local model consists of a FITS file with atomic data, a `ismabs.f90` Fortran file, and a parameter definition file called `lmodel_ismabs.dat` where the parameters of the model are defined. Since the local model is written in Fortran 90, the easiest way to use it in a SPEX user model is to write a Fortran 90 program that will be the interface between SPEX and the model.

In this case, we need the following files and libraries:

- The cfitsio library should be installed on the system.
- Module `moduser.f90` as provided on the SPEX web site.
- Example program `loc-xspec.f90` as provided on the SPEX web site.
- Ismabs local model files as provided on the XSPEC web site.

Write a small user program

The `moduser.f90` file is a library with user callable functions to make it easy to write a user model. This file does not need any editing. The user should only edit the example program `loc-xspec.f90` and sometimes also the local XSPEC model, which will be explained later. The example program `loc-xspec.f90` looks like this:

```
program locxspec
use moduser
implicit none

integer             :: i

! Get input and output filenames
call getfilenames(fin,fout,ier)

! Read input file
call readprm(trim(fin))

! Allocate output arrays
call allopar()

! Call local XSPEC model
call ismabs(ipar%eg,ipar%neg,ipar%par,1,opar%sener)

! Do not use wener values for now
do i=1,opar%neg
  opar%wener(i)=0.0
enddo

! Write result to output file
call writespc(fout)

! Clean up memory
call deallpar()

end program
```

Explanation of the called routines:

- **getfilenames(fin,fout,ier)** is a routine to read the file names from the command line. The format of the files and the order on the command line is defined by the user model in SPEX. This routine returns the file names the program needs to read the input parameters and write the result.
- **readprm(trim(fin))** is the routine that reads the input parameters and the input energy grid. It allocates and fills the structure `ipar` with the needed numbers.
- **allopar()** allocates the memory for the output arrays based on the input file.
- **ismabs(ipar%eg,ipar%neg,ipar%par,1,opar%sener)** is the actual call of the XSPEC local model. The parameters from the `ipar` structure contain the input parameters and the `opar` structure contains the output spectrum. For other local models than `ismabs`, simply change the name of the routine on this line.
- **writespc(fout)** writes the resulting spectrum to the output file
- **deallpar()** deallocates all the allocated variables from the moduser module.

Compile the executable

In principle, the source files can now be compiled into an executable that the SPEX user model can use. Make sure you have all the necessary files in one directory (see above) and execute the following commands in a terminal:

```
linux:~/lmodel>  gfortran -g -c -o moduser.o moduser.f90
linux:~/lmodel>  gfortran -g -c -o ismabs.o ismabs.f90
linux:~/lmodel>  gfortran -g -c -o loc-xspec.o loc-xspec.f90
linux:~/lmodel>  gfortran -g -o loc-xspec loc-xspec.o ismabs.o moduser.o
```

In the last step, it will be clear whether the executable has access to all the necessary functions. In this case, the `ismabs` model needs cfitsio to read the fits file with atomic data. It also needs a few XSPEC internal functions to read the path for the FITS file. The cfitsio library can be easily linked by adding `-lcfitsio` to the last command in the sequence above. For the internal Xspec calls, we need to adapt `ismabs.f90` slightly. The few calls to the XSPEC routines can be removed and with a slight modification we can also make sure it finds the fits file. This step needs a little programming experience to do it right. Always keep a backup of the original routine.

When you are done, repeat the following commands to create the executable:

```
linux:~/lmodel>  gfortran -g -c -o ismabs.o ismabs.f90
linux:~/lmodel>  gfortran -g -o loc-xspec loc-xspec.o ismabs.o
                    moduser.o -L/path/to/cfitsio -lcfitsio
```

The '-L/path/to/cfitsio' is optional. You may need to adapt it in case the compiler cannot find libcfitsio.so in the library path. In this flag, you can specify the correct path to libcfitsio.so.

Use the Xspec local model in SPEX

Start SPEX in a directory where the `loc-xspec` executable that we just made is located. Since the `ismabs` model is a multiplicative model, we need to load the `musr` component. In the example below, we show how a power-law model is absorbed by `ismabs` in SPEX:

```
SPEX> com po
SPEX> com musr
SPEX> com rel 1 2
# Link the new loc-xspec executable to the musr component
SPEX> par 1 2 exec av ./loc-xspec
# The number of parameters is found in 'local_ismabs.dat', supplied by the Xspec model
SPEX> par 1 2 npar v 31
```

The file `local_ismabs.dat` also describes the parameters and their limits in order. It is advisable to write a SPEX command file to set the parameters and their ranges to their default values. The order of the parameters should be the same in the musr model and in the `local_ismabs.dat` file. If the bookkeeping is right, you should be able to issue a calculate command in SPEX and show the absorbed power law in a plot.

9.2.2 Connecting the SPEX user model to Xspec

The most general way to incorporate Xspec models in SPEX is to call Xspec itself directly by a SPEX user model executable. Here we provide a fortran+tcl example to use an Xspec built-in `comptt` model. It is also possible to incorporate Xspec local models, e.g., ismabs as described in §9.2.1, by this method, while it would be slightly less efficient than the procedure in §9.2.1, since this one need to access the data files of local model via Xspec.

It requires the following files:

- Xspec software as provided in HEASARC web site.
- Module `moduser.f90` as provided on the SPEX web site.
- Example program `comptt-xspec.f90` as provided on the SPEX web site.

Write a program to call Xspec

The example program `comptt-xspec.f90` creates a tcl script and runs it in Xspec.

```
program compttxspec
  use moduser
  implicit none

  integer                          :: i

  ! Get input and output filenames
  call getfilenames(fin,fout,ier)

  ! Read input file
  call readprm(trim(fin))

  ! Allocate output arrays
  call allopar()

  ! Create Xspec tcl
  open(1,file="comptt.tcl",status="unknown")
  write(1,*) "model comptt & ", ipar%par(1), " & ", &
       ipar%par(2), " & ", ipar%par(3), " & ",      &
       ipar%par(4), " & ", ipar%par(5), " & ",      &
       ipar%par(6)
  write(1,*) "dummyrsp", ipar%egb(1)-ipar%deg(1),   &
       ipar%egb(ipar%neg), ipar%neg
  write(1,*) "set ff [open user.dat a]"
  write(1,*) "tclout plot model x"
  write(1,*) "puts $ff $xspec_tclout"
  write(1,*) "tclout plot model y"
  write(1,*) "puts $ff $xspec_tclout"
  write(1,*) "quit"
  close(1)
```

```
    ! Run Xspec tcl
    call system("rm -rf user.dat")
    call system("xspec comptt.tcl >> /dev/null")

    ! Read Xspec output
    open(1,file="user.dat",status="old")
    read(1,*)
    read(1,*) opar%sener
    close(1)

    ! Do not use wener values for now
    do i=1,opar%neg
       opar%wener(i)=0.0
    enddo

    ! Write result to output file
    call writespc(fout)

    ! Clean up memory and temporary files
    call deallpar()
    call system("rm -rf user.dat")
    call system("rm -rf comptt.tcl")

  end program compttxspec
```

Compile the executable

The program will be compiled in a similar way as the one described in 9.2.1.

```
linux:~/lmodel>  gfortran -g -c -o moduser.o moduser.f90
linux:~/lmodel>  gfortran -g -c -o comptt-xspec.o comptt-xspec.f90
linux:~/lmodel>  gfortran -g -o comptt-xspec comptt-xspec.o moduser.o
```

Use the Xspec model in SPEX

Here are the SPEX commands to calculate the Xspec-version comptt model.

```
SPEX> com user
SPEX> par 1 exec av ./comptt-xspec
SPEX> par 1 npar v 6
# The number of parameter needed by the Xspec model
SPEX> par 1 p01 v 0
...
SPEX> calc
```

9.2.3 Interpolate an APEC-like table model

Several atomic codes, such as APEC and MEKAL, are provided as pre-calculated tables usually in FITS format. These tables often contain a series of grids representing a main physical parameter, e.g., electron temperature, and give the spectral properties (e.g., line emissivity, continuum level) for each grid. Here we describe a two-step approach to utilize directly an APEC table by SPEX user model. Indeed the

method in §9.2.2 would provide the same result, the only difference is that this one can run without Xspec.

It requires the following files:

- APEC files (`apec_v3.0.2_coco.fits` and `apec_v3.0.2_line.fits`) provided on the ATOMDB web site.
- Module `moduser.f90` as provided on the SPEX web site.
- Example script `apec-dump.sh` as provided on the SPEX web site.
- Example program `apec-int.f90` as provided on the SPEX web site.
- FTOOLS as provided on the HEASARC web site.

Two-step program to run APEC

Here the example is divided into two steps: dumping APEC files into ASCII tables, and interpolating the related grids to get spectrum. The first step is achieved by using FTOOLS and the second one is by a fortran program. For advanced users, the two steps can be easily merged into one by incorporating fitsio library in fortran. For simplicity, redshift is fixed to 0 in this exercise. The current model has three parameters, `P01`, `P02`, and `P03` are temperature, average abundance, and normalization.

- `apec-dump.sh`: convert the APEC files into ASCII tables in a local directory `apec_files`.

```
# Initialize files
rm -rf apec.info
rm -rf apec_files
mkdir apec_files
mkdir apec_files/line
mkdir apec_files/coco
export linefile=apec_v3.0.2_line.fits
export cocofile=apec_v3.0.2_coco.fits

# Get grid information
fdump ${linefile}[1] columns=kT rows=- outfile=apec.tmp prhead=no clobber=yes

for loop in `seq 2 1 52`
do
# Get lines
export ktloop=`tail -n +5 apec.tmp | head -$((loop-1)) | tail -1 | awk '{print $2}'`
fdump ${linefile}[${loop}] columns="Lambda, Epsilon" rows=- outfile=linesave.tmp prhead=yes sho
export hdl=`sed -n '/A               photons cm^3 s^-1/=' linesave.tmp`
tail -n +$((hdl+1)) linesave.tmp > apec_files/line/${loop}.dat
export linenum=`wc apec_files/line/${loop}.dat | awk '{print $1}'`

# Get continuum
export savename=cocosave.tmp
fdump ${cocofile}[${loop}] columns="E_Cont, Continuum" rows=- outfile=cocosave.tmp prhead=yes sl
export hdc=`sed -n '/keV             photons cm^3 s^-1 keV^-1/=' cocosave.tmp`
tail -n +$((hdc+1)) cocosave.tmp > apec_files/coco/${loop}.dat
export coconum=`wc apec_files/coco/${loop}.dat | awk '{print $1}'`

# Write informations
echo ${ktloop} \"apec_files/coco/${loop}.dat\" ${coconum} \"apec_files/line/${loop}.dat\" ${lin
echo ${ktloop} \"apec_files/coco/${loop}.dat\" ${coconum} \"apec_files/line/${loop}.dat\" ${lin
```

```
# Clear up
rm -rf linesave.tmp
rm -rf cocosave.tmp
done
rm -rf apec.tmp
```

- `apec-int.f90`: read the ASCII files and interpolate based on input values.

```
program apec-int
  use moduser
  implicit none

  integer                          :: i,j,k
  integer                          :: ngrid,codim1,codim2,lidim1,lidim2
  character*100,allocatable        :: cocofile(:),linefile(:)
  integer,allocatable              :: ncoco(:),nline(:)
  integer,parameter                :: nele = 30
  real*8,allocatable               :: kt(:)
  real*8,allocatable               :: liwav1(:),liwav2(:)
  real*8,allocatable               :: liem1(:),liem2(:)
  real*8,allocatable               :: cokev1(:,:),cokev2(:,:)
  real*8,allocatable               :: coem1(:,:),coem2(:,:)
  real*8                           :: co_eg
  real*8,allocatable               :: emi1(:), emi2(:)
  ngrid = 0

  ! Get ascii file info
  open(1,file="apec.info",status="old")
  do
     read(1,*,end=2)
     ngrid = ngrid + 1
  end do
2 close(1)
  allocate(cocofile(ngrid))
  allocate(linefile(ngrid))
  allocate(kt(ngrid))
  allocate(ncoco(ngrid))
  allocate(nline(ngrid))
  open(1,file="apec.info",status="old")
  do i=1,ngrid
     read(1,*) kt(i), cocofile(i), ncoco(i), linefile(i), nline(i)
  end do
  close(1)


  ! Get input and output filenames
  call getfilenames(fin,fout,ier)

  ! Read input file
  call readprm(trim(fin))

  ! Allocate output arrays
  call allopar()
```

```
! Read related ascii files: low-kt case
if (ipar%par(1) .lt. kt(1)) then
   allocate(cokev1(ncoco(1)/nele,nele))
   allocate(cokev2(ncoco(1)/nele,nele))
   allocate(coem1(ncoco(1)/nele,nele))
   allocate(coem2(ncoco(1)/nele,nele))
   allocate(liwav1(nline(1)))
   allocate(liwav2(nline(1)))
   allocate(liem1(nline(1)))
   allocate(liem2(nline(1)))
   codim1 = ncoco(1)/nele
   codim2 = ncoco(1)/nele
   lidim1 = nline(1)
   lidim2 = nline(1)
   open(1,file=trim(cocofile(1)),status="old")
   do j=1,nele
      do i=1,codim1
         read(1,*) cokev1(i,j), coem1(i,j)
      end do
   end do
   close(1)
   cokev2 = cokev1
   coem2  = coem1
   open(1,file=trim(linefile(1)),status="old")
   do i=1,lidim1
      read(1,*) liwav1(i), liem1(i)
   end do
   close(1)
   liwav2 = liwav1
   liem2  = liem1

! Read related ascii files: high-kt case
else if (ipar%par(1) .ge. kt(ngrid)) then
   allocate(cokev1(ncoco(ngrid)/nele,nele))
   allocate(cokev2(ncoco(ngrid)/nele,nele))
   allocate(coem1(ncoco(ngrid)/nele,nele))
   allocate(coem2(ncoco(ngrid)/nele,nele))
   allocate(liwav1(nline(ngrid)))
   allocate(liwav2(nline(ngrid)))
   allocate(liem1(nline(ngrid)))
   allocate(liem2(nline(ngrid)))
   codim1 = ncoco(ngrid)/nele
   codim2 = ncoco(ngrid)/nele
   lidim1 = nline(ngrid)
   lidim2 = nline(ngrid)
   open(1,file=trim(cocofile(ngrid)),status="old")
   do j=1,nele
      do i=1,codim1
         read(1,*) cokev1(i,j), coem1(i,j)
      end do
   end do
   close(1)
   cokev2 = cokev1
```

```
      coem2  = coem1
      open(1,file=trim(linefile(ngrid)),status="old")
      do i=1,lidim1
         read(1,*) liwav1(i), liem1(i)
      end do
      close(1)
      liwav2 = liwav1
      liem2  = liem1
   end if

   ! Read related ascii files: normal-kt case
   do i=1,ngrid-1
      if (ipar%par(1) .lt. kt(i+1) .and. ipar%par(1) .ge. kt(i)) then
         allocate(cokev1(ncoco(i)/nele,nele))
         allocate(cokev2(ncoco(i+1)/nele,nele))
         allocate(coem1(ncoco(i)/nele,nele))
         allocate(coem2(ncoco(i+1)/nele,nele))
         allocate(liwav1(nline(i)))
         allocate(liwav2(nline(i+1)))
         allocate(liem1(nline(i)))
         allocate(liem2(nline(i+1)))
         codim1 = ncoco(i)/nele
         codim2 = ncoco(i+1)/nele
         lidim1 = nline(i)
         lidim2 = nline(i+1)
         open(1,file=trim(cocofile(i)),status="old")
         do j=1,nele
            do k=1,codim1
               read(1,*) cokev1(k,j), coem1(k,j)
            end do
         end do
         close(1)
         open(1,file=trim(cocofile(i+1)),status="old")
         do j=1,nele
            do k=1,codim2
               read(1,*) cokev2(k,j), coem2(k,j)
            end do
         end do
         close(1)
         open(1,file=trim(linefile(i)),status="old")
         do k=1,lidim1
            read(1,*) liwav1(k), liem1(k)
         end do
         close(1)
         open(1,file=trim(linefile(i+1)),status="old")
         do k=1,lidim2
            read(1,*) liwav2(k), liem2(k)
         end do
         close(1)
      end if
   end do

   ! Add continuum and line components
   allocate(emi1(ipar%neg))
```

```
allocate(emi2(ipar%neg))
emi1 = 0
emi2 = 0
do i=1,ipar%neg
   do j=1,nele
      do k=1,codim1-1
         if (ipar%eg(i) .lt. cokev1(k+1,j) .and. ipar%eg(i) .ge. cokev1(k,j)) then
            co_eg   = coem1(k,j) + (coem1(k+1,j) - coem1(k,j)) &
                 * (ipar%eg(i) - cokev1(k,j)) / (cokev1(k+1,j) - cokev1(k,j))
            emi1(i) = emi1(i) + co_eg * ipar%par(2) * ipar%deg(i) * ipar%par(3)
         end if
      end do
   end do
   do j=1,lidim1
      if (12.4/liwav1(j) .lt. ipar%eg(i) + ipar%deg(i)/2. &
           .and. 12.4/liwav1(j) .ge. ipar%eg(i) - ipar%deg(i)/2.) then
         emi1(i) = emi1(i) + liem1(j) * ipar%par(2) * ipar%par(3)
      end if
   end do
end do
do i=1,ipar%neg
   do j=1,nele
      do k=1,codim2-1
         if (ipar%eg(i) .lt. cokev2(k+1,j) .and. ipar%eg(i) .ge. cokev2(k,j)) then
            co_eg   = coem2(k,j) + (coem2(k+1,j) - coem2(k,j)) &
                 * (ipar%eg(i) - cokev2(k,j)) / (cokev2(k+1,j) - cokev2(k,j))
            emi2(i) = emi2(i) + co_eg * ipar%par(2) * ipar%deg(i) * ipar%par(3)
         end if
      end do
   end do
   do j=1,lidim2
      if (12.4/liwav2(j) .lt. ipar%eg(i) + ipar%deg(i)/2. &
           .and. 12.4/liwav2(j) .ge. ipar%eg(i) - ipar%deg(i)/2.) then
         emi2(i) = emi2(i) + liem2(j) * ipar%par(2) * ipar%par(3)
      end if
   end do
end do

! Intepolate the two neighbor grids
if (ipar%par(1) .lt. kt(1)) then
   opar%sener = emi1
else if (ipar%par(1) .ge. kt(ngrid)) then
   opar%sener = emi1
end if

do i=1,ngrid-1
   if (ipar%par(1) .lt. kt(i+1) .and. ipar%par(1) .ge. kt(i)) then
      opar%sener = emi1 + (emi2 - emi1) * (ipar%par(1) - kt(i))/ &
           (kt(i+1) - kt(i))
   end if
end do

! Do not use wener values for now
do i=1,opar%neg
```

```
      opar%wener(i)=0.0
    enddo

    ! Write result to output file
    call writespc(fout)

    ! Clean up memory
    call deallpar()
    deallocate(cokev1)
    deallocate(cokev2)
    deallocate(coem1)
    deallocate(coem2)
    deallocate(liwav1)
    deallocate(liwav2)
    deallocate(liem1)
    deallocate(liem2)
    deallocate(emi1)
    deallocate(emi2)


  end program apec-int
```

Prepare the ASCII files and executable

All related files, including `apec-dump.sh`, `apec-int.f90`, `moduser.f90`, and APEC files (`apec_v3.0.2_coco.fits` and `apec_v3.0.2_line.fits`) should be placed in the same directory,

```
linux:~/lmodel>  source apec-dump.sh
```

It would create a sub-directory called `apec_files`.

Then, the fortran program is compiled by including `moduser.f90`.

```
linux:~/lmodel>  gfortran -g -c -o moduser.o moduser.f90
linux:~/lmodel>  gfortran -g -c -o apec-int apec-int.f90
linux:~/lmodel>  gfortran -g -o apec-int apec-int.o moduser.o
```

Use the APEC model in SPEX

Below we incorporate the APEC model with the SPEX user function.

```
SPEX> com user
SPEX> par 1 exec av ./apec-int
SPEX> par 1 npar v 3
# Three parameters are kT, abund, and norm
SPEX> par 1 p01 v 1
...
SPEX> calc
```

9.2.4 Troubleshooting

My XSPEC local model has more parameters then the SPEX user model allows

By default, the `user` and `musr` components allow about 40 free parameters. This value can be changed without compiling SPEX. The model parameters are defined in an ASCII file called `in_par.dat` and is located in the SPEX `data` directory.

Warning! Make a backup copy of in_par.dat before you proceed! An erroneously formatted file can render SPEX unusable!

The `in_par.dat` file contains the definitions of all the SPEX models. Find the `musr` model in the file. It should look like this:

```
"musr" T F F  42
"exec" "User executable     " 4 0      0          0         0
"npar" "Number of parameters" 3 0      1          1         40
"p01 " "Parameter 01        " 0 1      1          0         1E20
"p02 " "Parameter 02        " 0 0      1          0         1E20
...
```

The first line contains the name of the model, some basic properties, and the number of lines that should be read. If you want to add, for example, 10 lines, increase the number 42 to 52 and add 10 lines starting with `"p41 " "Parameter 41          "` at the end of the sequence. Make sure that every line contains a unique parameter name for this model, so "p41 " to "p50 " should be fine. In principle, it is also possible to rename the parameters for convenience, but then the names will hold for every `musr` or `user` model that are opened!

In case SPEX gives errors when using the new `in_par.dat`, just copy the backup version back into the data directory.

Chapter 10

Calling SPEX from Fortran

Author: Jelle Kaastra (J.S.Kaastra@sron.nl)
Keywords: SPEX general – Programming

10.1 Goal

Sometimes you may want to combine SPEX with some of your own code. For instance, you may have a hydrodynamical code calculating the evolution of a cluster of galaxies or a supernova remnant, and you want SPEX to calculate the corresponding spectrum. Or you need to make realistic simulations for a large sample of sources or a broad range of parameters. In that case it can be useful to call SPEX from a fortran program.

10.2 Solution

The basic flow of the solution is as follows:

- Start your fortran program and calculate whatever is needed
- create a command file for SPEX
- call SPEX through a system call
- read the output of SPEX
- continue with your program

To explain how you can do this, we give below a sample Fortran 90 program. The program first creates a command file called spex.com that contains the input for the run with SPEX. In your case, you can replace the "/usr/local/bin/spex" to whatever path is appropriate for you to call SPEX. The "<<STOP" is essential for linux to tell when the input stream is finished.

In the command file, it tells how to read the pn spectrum of a cluster of galaxies with roughly metallicity 0.3, temperature 2 keV, redshift 0.01 and Galactic foreground absorption of 10^{24} m^{-2}. It ignores data at the lowest and highest energies, sets the initial model and parameters for SPEX. It will

then perform a fit with SPEX. Next we tell SPEX to open an output file named "spex.out", that will contain the results of the next part of the SPEX run, an error search on the temperature. Note also that we have set-up the plotting device to "null", to speed up the calculation (but implying that you do not get to see the fit itself, so use with care). After the error run, we stop SPEX, first by writing the "quit" command for SPEX and then the end of input signal "STOP" for linux. Next we simply close the command file spex.com.

The fortran program then continues by making spex.com executable, and then by executing that file, thereby doing the run with SPEX. When SPEX is finished, fortran takes over control again and reads the spex.out file created by SPEX. The program grabs any info it needs from that file (here the errors on the temperature) and then closes the file.

As a last step, the intermediate outputfile "spex.out" is deleted, as we do not need it longer.

```
real nh,z,t
character*128 s

z = 0.01    !redshift 0.01
nh = 1e-4   !1E20 cm**-2 absorption column density (SPEX uses units of 1E28 m**-2)
t = 2       !temperature 2 keV

open(unit=12,file='spex.com',status='replace') !open a spex command file called spex.com
write (12,'("/usr/local/bin/spex <<STOP")')    !call spex
write (12,'("abu lodders")')                   !set abunances to Lodders scale
write (s,'("dis ",f6.4," z")') z
write (12,'(a)') trim(s)                       !write the distance
write (12,'("data pn pn)')                     !read some spex data set: pn.res & pn.spo
write (12,'("ign in 1 r 1 1:60")')             !ignore first 60 channels
write (12,'("ign in 1 12:10000 u kev")')       !ignore E>12 keV region
write (12,'("com cie")')                       !define a CIE component
write (12,'("com reds")')                      !define a redshift component
write (12,'("com abs")')                       !define galactic absorption
write (12,'("com rel 1 2,3")')                 !couple the components
write (s,'("par 1 2 z v ",f6.4)') z
write (12,'(a)') trim(s)                       !set the redshift
write (s,'("par 1 3 nh v ",1pe12.3)') nh
write (12,'(a)') trim(s)                       !set the galactic absorption
write (12,'("par 1 3 nh s f")')                !freeze NH
write (12,'("par 1 1 06:28 v 0.3")')           !set abundances of C to Ni to 0.3 time solar
write (s,'("par 1 1 t v ",f5.2)') t
write (12,'(a)') trim(s)                       !set the temperature
write (12,'("fit meth cstat")')                !use C-statistics for fitting
write (12,'("plot dev null")')                 !no visible plots
write (12,'("plot type data")')                !plot type data
write (12,'("fit prin 1")')                    !show each fitting step
write (12,'("fit")')                           !do a fit
write (12,'("err dchi 1.")')                   !for error search, do Delta chi**2 = 1
write (12,'("log out spex o")')                !open a file for the SPEX output
                                               !file will be called "spex.out"
                                               !the option "o" means overwrite
write (12,'("err 1 1 t")')                     !do an error search on T
write (12,'("quit")')                          !done with the spex run, so quit
write (12,'("STOP")')                          !last command needed to close the command file
close (12)                                     !close the file spex.com and save

call system("chmod +x spex.com")               !make file spex.com executable
call system("spex.com")                        !execute the file spex.com

open (unit=12,file='spex.out',status='old')    !open the outputfile spex.out
do k = 1,100000                                !loop over all the lines in this file
```

```
     read (12,'(a)',end=20) s                    !read the text of the line; put it in string s
     if (s(2:11).eq."Parameter:") then           !search for the line with text Parameter:
       read (s(34:),*) e1,e2                     !read from this line the errors on T
       err = 0.5 * (-e1 + e2)                    !calculate average pos and neg error
       !..... do further all you wish
     endif
   enddo                                         !close loop over lines file spex.out
20 close(12)                                     !close spex.out
   call system("rm spex.out")                    !remove the file spex.out
   end
```