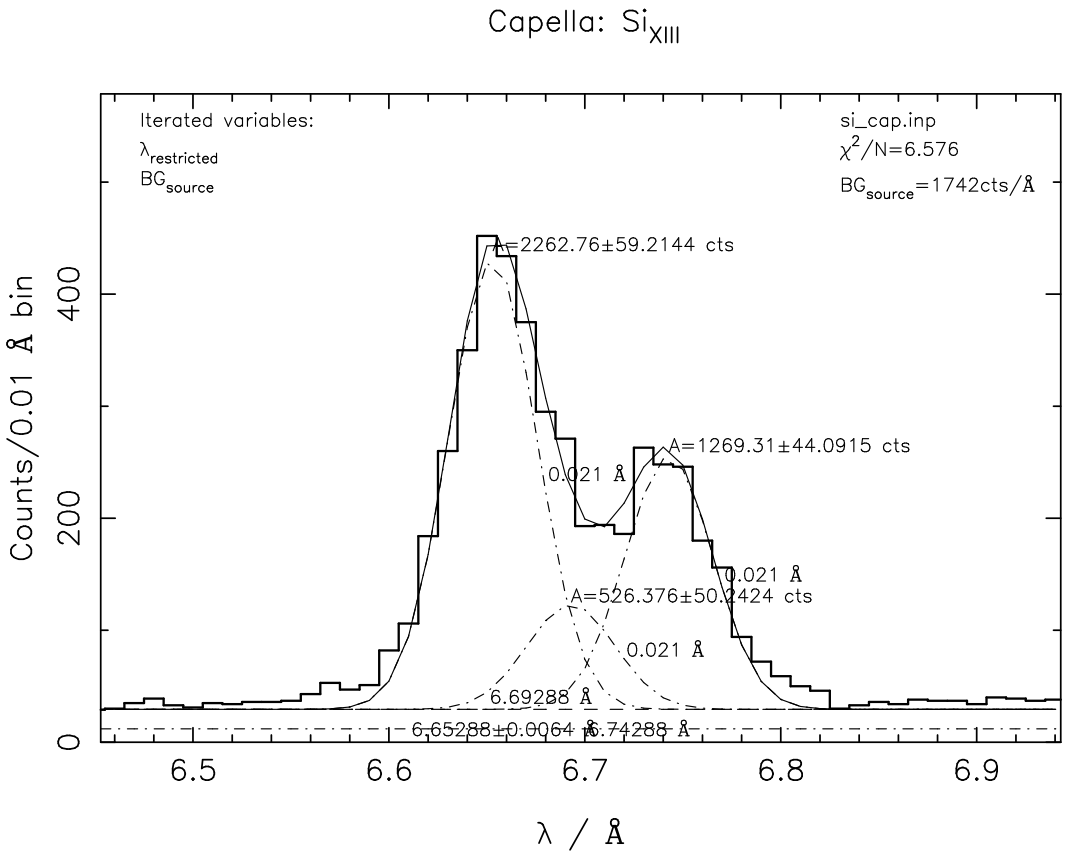
Manual for Cora line fitting tool, Version 4.0

Jan-Uwe Ness – Jan-Uwe.Ness@asu.edu
Rainer Wichmann – rwichmann@hs.uni-hamburg.de

June 4, 2006



This manual is also available at the URL
http://erica.la.asu.edu/ ness/Cora/

Contents

1 Version history

1.1 New in version 1.2/1.3

- Multi-line-fits with blends: The contribution from each line is visible in the plot
- Error analysis of amplitudes: The correlated errors are calculated, i.e. in line blends the errors are larger than the formal errors (cf. A.W. Strong, Astron. & Astrophys. **150**, 273, 1985).
- The fit can be forced to satisfy linear restricting conditions in the form (Sect. 7.1): $\sum_{i=1}^{nlines} f_i \times A_i = v$ with weight factors f_i for each line.
- New input parameter *spec_only* and *fit_only* for suppressing plot of fitted curve (without fitting) and plot of the spectrum, respectively
- More Options:
  - Gauss/Lorentz profile function
  - Iteration of λ and FWHM (in versions prior 3.2 σ) can be turned off individually
  - Lock switch locking the wavelength of a line to the value of the previous line
    Multiplet fitting is preserved
  - Pseudo-Voigt (weighted sum of Gauss and Lorentz, with a common linewidth also calculated from individual Gauss and Lorentz line-widths cf. Sect. 7.3).
  - The log-file [e.g. cora.dat] can be converted into a LaTeX table
- Parameter files are backward compatible!

1.2 New in version 2.0

- Additional plotting options: a) Plot only the spectrum without fit
  b) Plot only the fit, but not the spectrum
- Data file can also read spectra with only two columns instead of five.
  The instrumental background will then be assumed zero in all bins, and the Poissonian errors (i.e. $\sqrt{cts}$) are used.
  A three-column data file is interpreted as: wavelength grid, count spectrum, instrumental background. Errors, again, all Poissonian.
- New program `cora_rgs` applicable from console, converting fits files returned by the XMMSAS task *rgsproc*, into the `Cora` format, using the header information for exposure time and aspect data. The cfitsio library is required, when compiling this program.
- Script for reducing XMM-RGS data. User has to define environment variables which are used for naming convention
- Scripts are provided customizing the installation, for either Tc-shell (cora_setup.tcsh) or K-shell (cora_setup.ksh). The environment variables must be set in a manner corresponding to the individual system.
- Tooltips are included for each button, entry field, etc. in the GUI giving a little explanation.
- Removing an individual line from the linelist is possible by setting the wavelength value to 0 or negative values.
- The X-axis can be plotted in units keV.
- All FWHM (in versions prior 3.2 σ) values except for the first value can be set to zero
- Version number can be requested with -V option.

1.3 New in version 3.0

- Individual profile function can be given as ASCII
- β Profile added (Lorentz$^{\beta}$)
- Additional options to estimate source continuum
- Exposure time can be given in spectrum file
- Style options for nice plot in IDL version
- Line widths can be iterated as same values

1.4 New in version 3.2 and 3.3

- discontinued `cora_rgs`
- Line width parameter must given and is returned in terms of **FWHM** (Full Width at Half Maximum) for all profile types, however, in this manual it is still sometimes labelled σ. In particular names of parameters, e.g., dsig, refer to the old convention and are not renamed.

1.5 New in version 4.0

- Updated this manual
- Minor tidying up of IDL scripts
- Combination of cora_rgspipe.sh and read_rgs.pro do reduction of XMM-RGS data as well as conversion to `Cora` format.

2 Summary

`Cora` is a program package that implements a line fitting procedure based on the Maximum Likelihood technique proposed by Cash 1979 (ApJ **228**, 939) and has been developed for emission line spectra, as introduced in Ness & Wichmann 2002 (Astronomische Nachrichten **323**, 129).

The program measures the number of counts contained in emission lines. Energy fluxes or intensities are not calculated during the procedure, since the response from the specific instrument would have to be processed. This would go beyond the general purpose of this tool. However, the line counts can be converted to photon fluxes by use of the effectives areas that can be delivered with the spectrum. The point-spread function is not obtained from the response but is approximated analytically by use of parameterized Gaussian/Lorentzian profile functions. A Pseudo-Voigt profile is implemented consisting of a weighted sum of a Gaussian- and a Lorentzian profile, with common line widths, also calculated from individual Gaussian and Lorentzian line widths. More profiles can be provided as ASCII files containing a non-analytical line template that represents any type of line shape and line width. We provide some extra tools to calculate fluxes from given effective areas. The program is especially applicable to spectra with low count numbers like UV spectra or X-ray spectra. It has been developed in the context of analyzing Chandra LETGS data, and is especially useful for these data, but sufficiently general to be applicable for other data as well. Supplemental IDL scripts are provided to convert the fits output produced by the Chandra CIAO software and the XMMSAS software into the CORA format.
Most flexible application is provided by the IDL version, but the C-Version is fully functional and the two versions require and produce compatible input/output. This manual focuses on the C-Version and some short description about the IDL usage is given in Sect. 9.
In Sect. 3, installation instructions are given. The basic fit routine is delivered with a Graphical User Interface (GUI) supporting the specification of the input parameters. An interactive graphic is available to choose lines to be fitted directly from the spectrum. The acceptable ranges of the input parameters are described in Sect. 6.

The output is organized in two ways: a plot showing spectrum and fit and a log file recording the fit parameters. The plot can be produced in different formats, i.e., as postscript file designed for inclusion in publications and as GIF or PPM files for inclusion in, e.g., power point presentations or html. In addition the resulting fit parameters with their errors can be written into an ASCII file for further processing. This log-file can be converted into a LaTeX table for publication etc. A detailed description can be found in Sect. 8.

We would like to point out that the program is still being developed and improved. Some new features that are provided may not yet be included in the manual.

2.1 The `cora` program package

The `cora` line fitting package consists of eight programs. Help on command line options can be obtained with the option `-h`.

- `cora_fit` – this is the main fitting program. It can be used as a standalone command-line application, or it can be used from the graphical user interface.
  `cora_fit` requires a **parameter file**, and a **spectrum**.
- `cora` – this is the *optional* graphical user interface, which acts as a graphical front-end to `cora_fit`, and significantly facilitates the use of the latter. Basically, with `cora` you can:
  - browse the spectrum and interactively choose lines to fit
  - interactively set all parameters required by `cora_fit`,
  - easily switch between different parameter settings,
  - call `cora_fit` with the currently selected parameter settings
  - obtain updated parameters and graphical output (i.e. plots).
- `cora_inp` – this is a helper application that writes a parameter file (see below) that is specifically adapted to a particular spectrum. This program is useful if you do not use the GUI. It is also used by the GUI itself.
- `cora_spec` – creates an artificial spectrum with a few lines; for testing purposes.
- `cora_tex` – converts the log file into a LaTeX table
- `cora_ftex` – same as cora_tex calculating fluxes from given effective areas

- `cora_flux` – creates a data file in the `Cora` format with fluxes using a given file with effective areas (two columns: wavelengths, area in cm^2) and exposure time.

In addition, two scripts are provided with the package.

- `cora_setup.ksh` and `cora_setup.tcsh` can be modified according to your system and executed in order to set environment variables. This customizes the installation.
- `cora_rgspipe.sh` will reduce XMM-RGS data using the SAS software. The user has to modify environment variables indicating where SAS is installed and where the data to be processed are located.

2.2 Copyright and license

3 Installation

The `cora` package uses the standard GNU autoconfig build system. Installing the program thus requires the following three commands:

`./configure` *[options]*
`make`
`make` *install*

The `./configure` script can take a number of optional arguments. You can use `./configure` *–help* to get a list. Most interesting perhaps is the *–prefix=DIR* argument, which can be used to change the default installation directory prefix (the default prefix is `/usr/local`, i.e., the executable would be installed in the directory `/usr/local/bin`).
The pgplot and GTK+ libraries will be searched for automatically, but the search can fail in some cases. In order to customize the specification of where these libraries are installed on your system, a script (`cora_setup.ksh` or `cora_setup.tcsh`) is delivered with a number of environment variables, which are read by the configure script, and should be modified to your needs. With the correct setting, the script must simply be executed before running `./configure` (the file *config.cache* might have to be deleted, if an earlier configure execution failed to find libraries).
The locations of libraries and include file can also be given on the command line with `./configure`, e.g. `./configure --with-pgplot-lib=PFX`.

3.1 System requirements

3.1.1 PGPLOT

For configuration and compilation it is necessary to have the PGPLOT plotting library installed. PGPLOT is a standard tool in astronomy, that is available for a large range of UNIX systems, as well as for Linux, and is free for non-commercial usage. If it is not installed on your system, it can be downloaded from

`http://astro.caltech.edu/~tjp/pgplot/`

(ask your system administrator for assistance). Some Linux distributions (e.g. Debian) have precompiled binary packages.

- PGPLOT is a Fortran library, while `Cora` is a C program. Therefore the C wrapper library `cpgplot` that comes with PGPLOT **must** be installed as well (compiled with `make cpg`).
- PGPLOT uses a font file (grfont.dat), whose location **must** be known at runtime. For this purpose, PGPLOT uses an environment variable PGPLOT_DIR, that points to the directory where the font file is installed. Failure to set this environment variable will at least result in crappy plots, if not worse.
- If ./configure fails to find your installed PGPLOT library files, you can use the following options:
  --with-pgplot-include=DIR
  --with-pgplot-lib=DIR
  to make the location of the include file (cpgplot.h) and the library files (cpgplot.a and pgplot.a, or cpgplot.so and pgplot.so) known to ./configure.

3.1.2 GTK+

If you want to use the graphical user interface, you need the Gimp Tool Kit (GTK+) library to be installed. GTK+ is distributed under the terms of the GNU General Public License, and can be freely obtained from

`http://www.gtk.org/ .`

Supported architectures include Linux as well as most Unix systems. These days, all popular Linux distributions come with GTK+ included. However, GTK+ is often split into a *runtime* package and a *development* package. To compile the graphical user interface, the *development* package is required.

On installation, GTK+ also installs a shell script `gtk-config`, that can be used to locate the installed library files. You should make sure that the directory where `gtk-config` is located is in your PATH environment variable.

3.2 Portability

`Cora` is written in standard ANSI C, and thus should compile with any ANSI C compiler. The PGPLOT library that `cora` uses for graphical input/output, is available for a large range of architectures, as well as the GTK+ library for the GUI.

If the GTK+ library is not available, it is still possible to compile and use the command-line `cora_fit` application without the GUI.

To build the `cora` package, the source files must be compiled with a **C compiler**, but linking with the PGPLOT libraries typically requires a **Fortran compiler**. Evidently, the two must 'fit together', i.e., the Fortran compiler must understand the compiled object code produced by the C compiler. The combination of the GNU gcc/g77 C/Fortran compilers works well. Other combinations (i.e., proprietary compilers of your system vendor) may require arcane command-line switches. The `configure` script knows some of them, but if your compilers are not among the supported ones, you may need to dig in your man pages ...
The configure script is tested for Suse Linux and Solaris, but should be compatible with other platforms. In case of problems, please contact us, we will be very willing to add your platform to the list of tested platforms.

4 Quickstart

The program is designed to be learned by intuitive use. We give a description how to get started and refer to more detailed description in section 5.

- Prepare a spectrum datafile.
  - For the required format, see Sect. 5.1.1.
  - If you have XMM-RGS fits files, see Sect. 5.1.3.
  - For testing, run `cora_spec` to create a test spectrum (see Sect. 5.1.1).
- Name the spectrum datafile `spectrum.dat` and move it to your current working directory.

- In your current working directory, start `cora`. If you have not installed it already, you can run it from the source directory with `./cora`.
- In the main window, choose Options/Graph from the top menu to view the spectrum and select lines for fitting.
  - Use the right mouse button to select lines.
  - Hit 'q' when you are done.
  - Check out explanations at the top of the graph window.
- Use the Go button to start the fit.
- Read on. Experiment.

5 Slow and detailed start

5.1 Input

5.1.1 The spectrum

The purpose of the program is to analyze a user-supplied spectrum. The spectrum must be provided as an ASCII text file containing **six, five, three**, or **two** columns of float or integer numbers.

If **five** columns are given, these will be interpreted as follows: the first column contains wavelength values, the second lists the corresponding count values, and the third column contains the errors for the count values. The last two columns must contain the corresponding count values of the instrumental background with their errors. If errors are not measured explicitly, Poisson errors may be included, i.e., the square root of the count values. Errors given in the data file are only used for plotting error bars. The fitting routine will *always* use Poisson errors implicitly without using the given errors.
An extra column may be added giving the effective areas on the same wavelength grid in cm^2. These are not used by the C-Version but are very useful for plotting in the IDL version.
Any number of trailing comments may be given, starting with a #. The first line may contain the exposure time (in seconds, also starting with #). By default, the program will expect the spectrum in a file named `spectrum.dat` (in the current working directory).

Data files with **two** or **three** columns (supported in `cora` 2.0 and later) are interpreted as follows: The first two columns are always the wavelengths and counts. The instrumental background can be given in a third column. I.e., if the data file contains only three columns, the last column will be considered the instrumental background, instead of the count errors, which in turn are considered Poissonian, i.e., $\sqrt{cts}$. When only two columns are given, the instrumental background will be considered zero in all bins.
Test data: the program `cora_spec [-l` *nlines*`] [-a` *amp*`] [-c` *continuum*`] [-s` *FWHM*`]` creates a synthetic example spectrum in the desired format with *nlines* lines, amplitude *amp*, continuum background *continuum*, and line width *FWHM*, stored in a file `spectrum.dat`.

5.1.2 Converting data files from Chandra

In order to be a general-purpose tool we programmed `Cora` to require ASCII input which can be produced from any type of other format, which is much more difficult the other way around. The output of the Chandra software CIAO is in fits format with two layers representing the plus and minus order spectra. We provide an IDL script `read_letg.pro` that reads in the CIAO output and creates a co-added spectrum in CORA format. In IDL this script is called with
IDL> read_letg,/all
The output can be found in a file leg.dat and the keyword /all includes the effective areas as a sixth column in this file (see Sect. 5.1.1). The script requires `mrdfits` (part of the astrolib) and interp_header (part of the atomdb package). The IDL script `read_rgs.pro` contains the function interp_header and can be used by compiling this script
IDL> .compile read_rgs
before starting read_letg. Another script is available for HETG data, read_hetg.pro, which works in the same way and produces two output files, meg.dat and heg.dat for the MEG and HEG spectra, respectively. Note that co-added spectra are produced, i.e., the two opposite dispersion orders are co-added and the respective effective areas are co-added. Individual spectra can be produced by adding the keywords /plus and /minus to `read_hetg`.

5.1.3 Converting data files from XMM-RGS

The conversion from XMM-RGS spectra is a bit more complicated and requires a few more steps. The reason is that the output from XMMSAS does not provide wavelengths and counts in one file but either fluxes versus wavelengths or counts versus channels. However, if fluxes are imported, then we loose the Poissonian statistics. With `Cora` we provide scripts that start with the original odf files, applying the XMMSAS tool `rgsproc` and then converting the output into the `Cora` format.
The shell script `cora_rgspipe.sh` does the complete reduction from odf data to reduced RGS data. The first block can be modified by the user, providing the location where XMMSAS is installed, the location of the odf files and the observation identification number. There is also an exposure number for RGS1 and RGS2, which in most cases is 004 and 005, respectively, however, it may in some cases be different and can also be provided. The output files will be written into a directory that can be specified with the environment variable outdir.

Also the name of a text file must be given in which a number of commands to be carried out in xspec later on will be stored. When completed the script can be executed from the command line with `./cora_rgspipe.sh`. The next step will be to convert the output into the `Cora` format which has to be done in two steps. The script `cora_rgspipe.sh` has produced a text file with xspec commands that is adapted to the naming convention of the specific case. First, xspec has to be set up by initializing the heasoftware. Then the commands given in the script have to be carried out in an X-terminal (that is setup for the heasoftware). They can certainly be typed, but copy-and-paste is much easier, as one can copy the complete content of this file into the buffer memory and drop it into the terminal. The second step is to run the IDL script `read_rgs.pro` as also included in the generated script file. The IDL script requires the astrolib library to be installed. The output consists of two files, rgs1_m1.spec and rgs2_m1.spec, for RGS1 and RGS2 in first dispersion order, respectively. Higher dispersion orders can also be produced by editing read_rgs.pro and the generated script.

We summarize the required steps below.

- The user must modify the environment variables SAS_DIR, SAS_ODF, and obs by editing the script file `cora_rgspipe.sh`. SAS_DIR indicates the directory where the SAS software is installed on your system, SAS_ODF indicates the directory containing the odf files, and obs is the observation ID.
  The directory where the outputfiles will be stored can be indicated with setting outdir. Also the name of a script with further instructions (specific for the individual case) needs to be given by setting outfile.
- The specific usage of the SAS commands can be found under

  `http://xmm.vilspa.esa.es/sas/documentation/threads/`

- The directory specified with outdir in the script is created. All output from `cora_rgspipe.sh` is written into this directory. The messages returned by rgsproc are written into the file my_rgsproc_logfile.
- The file specified with outfile is written into the output directory. Change into this directory and set up the heasoftware in an X-terminal.
- Copy the complete content of the script file into the buffer memory and paste the content into the X-terminal.
- start the IDL script read_rgs (command also included in the script).
- output is rgs1_m1.spec and rgs2_m1.spec

We found the Lorentzian profile to optimally fit the instrumental line profiles. This can be seen from Fig. 1 were an XMM-RGS spectrum of Capella is shown as an example.

5.1.4 The parameter file

The performance of the program can be determined with the settings in a parameter file. The parameter file can be edited by writing in an ASCII file by hand or by using the GUI. A commented default file can be created automatically in the present working directory using the `cora_inp` program. If no file is supplied `cora_inp` will be launched automatically, so for the first run one does not need to bother about it. Some parameters depend on the what kind of spectrum is going to be analyzed (e.g., start values for line widths), so `cora_inp` needs a spectrum to be provided. Detail description of each parameter can be found in Sect. 6.
Cora can be started without specifying an input file, it is then automatically generated and named `fit.inp` (in the current working directory).

5.1.5 Session management

Session management is provided as follows: The location of files selected from the file selection dialogs is stored across sessions in `$HOME/.cora` . The last saved parameter file is stored in `$HOME/.cora/fit.inp` .

5.2 Starting the program

5.2.1 With the graphical user interface

The graphical user interface is simply called with

`cora`.

If the spectrum is not named `spectrum.dat` the option *-s* `filename` has to be added with `filename` the name

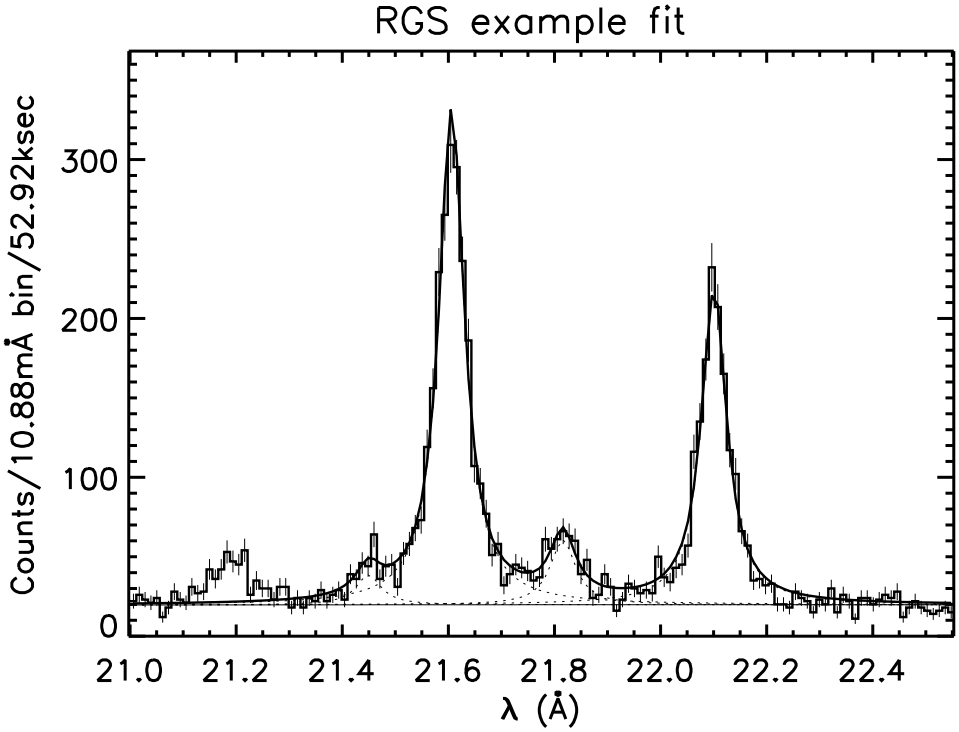


Figure 1: Fit result of an XMM-RGS spectrum of Capella. Obviously the Lorentz profile function is well suited for fitting the emission lines.

of the file containing the spectrum in the right format (cf. Sect.5.1.1). Two windows will pop up, the main window which contains the input parameters controlling the fit algorithm and the plot procedure and a smaller window that contains a table with line wavelengths and -widths. The parameters displayed in the windows are extracted from the parameter file.

By default (if no parameter file is specified), the program will search for a parameter file in the following order: (i) `$PWD/fit.inp`, (ii) the last one used, as stored in `$HOME/.cora/parameters`, (iii) `$HOME/.cora/fit.inp` which is a backup of the last one saved, and (iv) compiled-in defaults. To specify another file, use the command-line option

`cora` *-s* `spectrum.dat` *-i* `parameter_file`.

You can also change to another parameter file during the session with File/Open in the GUI.

While the *parameter file* will be created if it is not present, provision of a *spectrum* is mandatory. By default, the program will expect the spectrum in a file named `spectrum.dat`. To specify another file, use the command-line option

`cora` *-s* `spectrum_file`.

If the *spectrum* cannot be found at startup, you will be informed by a pop up dialog. You can then use the File->Open Data File option to read in a spectrum. Details about the right format are given in Sect. 5.1.1.

The layout of the main window is like a *notebook* with two *folders*, the first one holds the fit parameters, and the second one the plot parameters. At the bottom of the GUI there are three buttons:

Go calls the fit program `cora_fit` with the current parameter settings, reads the output and updates the line wavelengths/widths and the source background *sbg* with the fitted values. Pressing this button is equivalent with calling

`cora_fit` *-i* `parameter_file` *-s* `spectrum_file`

from the command line (description in next part of this section).

Save Values saves the current parameter settings to the currently active parameter file. This is useful, when e.g. using the GUI in combination with IDL (cf. Sect. 9).

Panic immediately terminates the application without saving the current settings.

At the top of the window, a 'File' menu provides the following functions:

New	Create a new parameter file.
Open Parameter File	Open a parameter file.
Open Data File	Open a data file
Save	Save settings to current parameter file.
Save As	Save settings to a new parameter file.
Quit	Quit the application.

Next, there is an 'Options' menu for the following options:

Graph	Open an interactive X window with a graph of the spectrum.
Go	Same as the Go button.
Remove line	Remove a line from the line list (asks for line number)
Replot Last	Re-plot the latest computed spectrum without re-computing it.
Replot From File	Re-plot the stored computed spectrum without re-computing it.
Create Standard ..	Create a standard parameter file.
Set FWHM zero	Set all FWHM values except for the first one to zero. This is implies that all values FWHM will be the same.
Create TeX file	Create a LaTeX file from the log-file
ShowLog	Show the log window.

Selecting the spectral lines to be fitted can be done in two ways. The lines can be entered directly into the line display window (see next paragraph). First, one has to provide the number of lines which then launches an updated line window with the appropriate number of rows. A more convenient way is to selected from an interactive graphical view of the total spectrum. This is done with Options/Graph. An interactive pgplot environment is launched displaying the total spectrum. Pressing the left mouse button will set left and right borders in order to zoom into smaller parts of the spectrum. Lines are selected with the right mouse button. It is also possible to select a background value pressing the middle mouse button when having the horizontal line running through the cursor position at the desired value of background. Further commands to be used from the keyboard are described on top of the graph. Line widths must be entered by hand in the line display window. If non-positive values are present, all other values will be set to the first positive value. This is useful when dealing with a large number of lines with all the same line widths. Only the first value needs to be set, all others (default zero) will be set to the first value. A GUI option is provided with which all FWHM values can be set to zero, the first value will be kept.

The line display window The line window (title string: "Start values") is the window where you can define start values for the lines you want to fit. It will pop up on start of the GUI, as well as whenever you change the number of lines in the main window.
You can set initial values for *lambda* (wavelength), *FWHM* (line width), and *weight* (usually 0.0, but see Sect. 7.1). In addition, there are fields to

1. select the profile type (Gauss/Lorentz),
2. enable/disable fitting of the wavelength
3. enable/disable fitting of the line width, and
4. lock the wavelength to that of the preceding line.
5. set "*Beta*" values for powers of Lorentzians

Note that locking the wavelength to that of the preceding line has a special meaning if the profiles are different: it will cause the program to use a (Pseudo-)Voigt profile with weighted Gaussian and Lorentzian contributions having (starting) values of the line widths as specified (see Sect. 7.3).

5.2.2 From the command line

Since not all platforms support the GTK+ library, but also for scripting, the fit procedure can also be started from the command line. This is done by calling:

```
cora_fit
```

If no parameter file is supplied, a standard file `fit.inp` is created with `cora_inp` (cf. Sect. 6) before starting. A different parameter file name can be specified by typing:

`cora_fit -i parameter_file` .

A spectrum file in the right format (Sect. 5.1.1) is mandatory and is by default expected to be named `spectrum.dat`. If you want to specify the spectrum with another file name, this can be done by typing:

`cora_fit` *-i* `parameter_file` *-s* `spectrum_file`

Another option is the re-plotting option *-r* `resfile`, where `resfile` indicates a text file with seven columns containing line data with wavelengths ± errors, line widths ± errors, amplitudes ± errors and a source background value in units [counts/bin]. Such a file in the correct format is created as fit output when activating the parameter *ires* to write the fit results into an ASCII file (default `cora.dat`); cf. Sect. 8. Another output file name can be specified with the option *-o* `outfile`.

When experimenting, it may be desirable to have a quick way of selecting lines for fitting without having to change the parameter file. This is provided by calling the program with

`cora_fit` *-g*.

After loading the spectrum, an interactive X-window with the complete spectrum will pop up the same way as with the GUI option Options/Graph (Sect. 5.2.1). The left mouse button is used to set boundaries for zooming into smaller parts of the total spectrum. Selecting lines is done with the right mouse button. Further commands to be entered from the keyboard are explained on top of the graph.

5.2.3 Command-line options

-help	Print out a list with the command-line options.
-v	Verbose output; repeat for more verbosity.
-V	Print version number
-g	Open a plot window for interactive line selection (`cora_fit`)
-s `spectrum_file`	Read spectrum from `spectrum_file`.
-i `parameter_file`	Read parameters from `parameter_file`.
-o `result_file`	Output computed spectral line parameters to `result_file` (only if *ires* is activated).
-r `result_file`	Re-plot computed model spectrum from line parameters in `result_file` (`cora_fit`).
-q	(GUI only) Send output to log window only.

5.3 Environment variables

Environment variables are system variables whose values are interpreted by our program or by the `PGPLOT` library. The run of the program can thus be influenced by setting certain environment variables. The setup scripts `cora_setup_ksh` and `cora_setup_tcsh` give some examples. In Sect. 5 the environment variable PGPLOT_DIR is mentioned which is required by PGPLOT. Further environment variables influencing the run of PGPLOT are listed in

`http://phobos.caltech.edu/~tjp/pgplot/chapter1.html#ENV` .

As an example we would like to focus on the environment variables PGPLOT_PS_HEIGHT (default value: 10500) and PGPLOT_PS_WIDTH (default value: 7800). In some LaTeX packages like includegraphics, an included postscript file returned by PGPLOT may appear rotated by 90°. To obtain a postscript file with the proper rotation, you can use *iportrait=1*, producing the plot in portrait format, and swap the values of PGPLOT_PS_HEIGHT and PGPLOT_PS_WIDTH.

5.4 Settings

The GUI will create a directory $HOME/.cora to store the location of the current spectrum datafile and parameter file. The stored locations are used in the file selection dialog (only).

6 Description of input parameters

Two ways are provided for specifying input parameters. The most direct way is to write a parameter file in ASCII format. When starting the program, the name of the parameter file can be specified with (cf. Sect. 5)

`cora_fit` *-i* `filename` or `cora` *-i* `filename`

In order to simplify this procedure, a program `cora_inp` is delivered, writing a default parameter file taking into account the properties of a particular spectrum (the spectrum must be specified with option *-s* and must be an ASCII file as described in Sect. 5.1.1; `spectrum.dat` is the default filename, cf. Sect. 5).

The resulting file, to be specified with the option *-i* (default `fit.inp`), can then be edited for the individual requirements. Example call:

`cora_inp -s spectrum.dat -i fit.inp`

which is identical with just typing `cora_inp`, taking the default filenames. `cora_inp` simply finds the strongest line and determines good starting parameters for fitting this line. Detailed explanations of the parameters are added at the end of the parameter file to help with modifications by hand.

The other way is much more comfortable, but requires the GUI version of the program. The program is called with (cf. Sect. 5)

`cora -s spectrum_file -i parameter_file`

As default the spectrum is assumed to be named `spectrum.dat`, a parameter file must not be provided if default setting are sufficient. The above described procedure `cora_inp` will be called with the specified parameter file if it is not present.

In the parameter file, which is a plain ASCII text file, values are given just by entering the numbers. Switches are activated by entering the integer value 1 and deactivated by entering 0. The default parameter file contains descriptions for each parameter. Calling `cora` will give a graphical user interface where parameters can be entered in entry fields and switches can be activated/deactivated by mouse clicks.

6.1 Fit parameters

Version number The first line of the ASCII parameter file must contain the version number (4.0). The default program `cora_inp` extracts the correct version number from the file `config.h`.

Title In the second line a title, consisting of up to 128 letters, can be given. It will appear in the title of the plot. (The maximum number of letters can be changed at compile time in `Cora.h` and `cora_fit.h`, with the variable MAXTEXTL.)

Spectral lines The number of lines to be fitted decides how many lines follow in the parameter file: For each line a wavelength value λ (in units Å), a line width value (FWHM; units either Å or km/s; unit system for FWHM is explained later), a weight factor for eventual auxiliary conditions (default 0), flags for the profile function (0: Gauss, 1: Lorentz), fitting λ or FWHM, eventual lock a line to the previous line, and a "*Beta*" value must be given. Therefore *nlines* must be followed by a list of *nlines* lines with start values of λ_i/FWHM$_i$, f_i and flags pro_typ_i, $FitLam_i$, $FitSig_i$, $LockLam_i$, and "*Beta*" in eight columns separated by blank or comma.

Apertures Apertures can be indicated for the source spectrum and the background spectrum separately, listed on the same line. Since only ratios are used no measured values must be given, e.g., in the case of identical apertures it only matters that the values *app_s* and *app_b* are the same. This is useful, when different extraction regions for source and background were used during the reduction of, e.g., Chandra grating spectra.

Background Besides the instrumental background a constant source background *sbg* can be added (units [counts/Å]) accounting for eventual continuum radiation or unresolved weak lines ("Pseudo continuum") originating from the source. If the switch *sbgit* (on the same line of the parameter file) is activated, then the given value for *sbg* is ignored, and the source background will be determined by an iteration of median values, applied to the part of the spectrum under consideration (see below).

The third value on this input line, the *bgsmooth* switch, decides whether the instrumental background will be smoothed, i.e., that a polynomial of 2$^{\text{nd}}$ order is used to model the instrumental background (a polynomial of lower order is used in the case that the 2$^{\text{nd}}$ order polynomial fails). Only the part of the spectrum under consideration is smoothed in this way.
More refined methods to estimate the source continuum are provided (see Ness et al. 2003, A&A **407**, 347 and 2004, A&A **427**, 667). A χ^2 method is programmed using the model parameters given by start values in order to generate a model spectrum. This is added to a source background value to be iterated by minimizing χ^2. This method is a good approach in cases when line wings are broad (thus, too many bins belong to emission lines) and when no lines features in the spectrum are excluded from the fit. With the fourth number on the

input line in the parameter file this procedure can be activated. The last entry in that line ($n\sigma$) is a value used for a refined median method. Since the median has to rely on that at least half of all bins have to belong to the background, too crowded line regions can be problematic. This can be solved by removing all bins which obviously belong to lines from the list of bins used to calculate the median value. In this way median values are iteratively calculated after removal of all bins with count values higher than $n\sigma \times \sqrt{\text{median}_{\text{last}}}$. Small values of $n\sigma$ will more critically remove high-count bins resulting in lower source background values. Usage of this parameter represents a parameterized choice of source continuum values by eye.

Spectral ranges and units The part of the spectrum to be under consideration for the procedure is chosen to include all lines to be modeled and is extended on both sides in order to collect sufficient background. The parameter *dlam*, units Å, gives the range below the lowest and above the highest line. The value of FWHM for each line is limited by the range indicated by *dsig*. The unit system for FWHM can either be Å or km/s depending on the parameter *sigunit* (0/I). The wavelengths must be in units Å, but the x-axis can be plotted in units keV depending on the parameter *ikev* (0/I). The four parameters are to be written on the same input line of the parameter file in the order *dsig, sigunit, dlam, ikev*.

Line parameters and spectral order The goal of the procedure is to obtain line fluxes. This is done most accurately when wavelengths and line widths are carefully chosen. Since this is not always possible, each of these two parameters can be optimized by iteration from given start values. With switches the iteration of λ (*ilamit*) and the iteration of FWHM (*isigit*; name refers to naming convention prior to version 3.2) can be activated. When modeling multiplets, the distance between the individual lines is often well known and only the absolute position of the template of lines is to be iterated. Especially when the multiplet contains weak lines, it can be desirable to treat the multiplet as a unit and iterating only the position of the multiplet (*idlam=1*). The line width FWHM can be iterated as the same value for all lines by setting *idsig* to 1. This is useful when the line width is determined by the instrument and is identical for all lines.

The parameter *iord* is a factor with which all wavelength values are multiplied before starting the fitting tool. This is useful when systematic shifts of the spectrum are expected, e.g., higher dispersion orders or reddening. These parameters must appear in the parameter file by putting *ilamit*, *idlam*, *iord*, and *idsig* on the same line and putting *isigit* on the next line.

Logging A log of the program's activity is written to the terminal. The verbosity of this log can be increased by the *-v* switch on the command-line, which can be repeated for even more verbosity. The verbosity can also be indicated from the GUI.

The GUI has an internal log buffer that can be displayed with the *Options->ShowLog* option. By default, a duplicate of the log will still be written to the terminal. To stop terminal output, you can use the *-q* switch on the command line.

The log window allows to run the GUI without associated terminal, e.g., from a window manager menu.

6.2 Output parameters

Title of plot Besides the text-output with *ires* a plot of the wavelength interval including the fitted lines can be created titled with *fittitle*. The title can be encoded in the pgplot-style as described in the PGPLOT User Manual; cf.

```
http://astro.caltech.edu/~tjp/pgplot/chapter4.html
http://astro.caltech.edu/~tjp/pgplot/hershey.html .
```

Format The output format is selected by the *iformat* parameter. Plotting devices supplied are: X-Window, PostScript (PS), Graphic Interchange Format (GIF) and Portable Pixel Map (PPM). The orientation can be set to portrait, if *iportrait* is activated. (Note, that PGPLOT must be compiled with the appropriate drivers for these plotting devices).

Outfit of plot If no plot is desired, the plot procedure can be switched off by deactivating *iplot*. With the switch *ilab* labels can be included. This will put the numerical fit results at the proper positions in the plot, e.g., the line fluxes on top of the corresponding line (cf. Fig. 2). With *ierrplot* error bars are plotted with the spectrum. Logarithmic plotting is provided with *ilog=1* scaling the y axis logarithmic. Finally the data can be binned with *nbin*. The bin-size in Å can be deduced from the y-label. If the user wishes to plot only the

spectrum without any model, the parameter *spec_only* can be used. The fit procedure, labeling etc. will then be switched off and only the spectrum, binned with *nbin*, with title *fittitle* etc. will be created.
Otherwise only the fitted curve can be plotted by suppressing the plot of the spectrum (*fit_only*=1).
Two more parameters are provided in the parameter file, but are only supported in the IDL version (cf. Sect. 9).
In Fig. 2 a PS file, created with the plot routine, is shown, using the parameter setting shown in Fig. 3. The fit was constrained, as described in Sect. 7.1, granting a certain line ratio to be kept, such that reasonable temperatures are consistent with the fit result. In this way a de-blending was attempted.

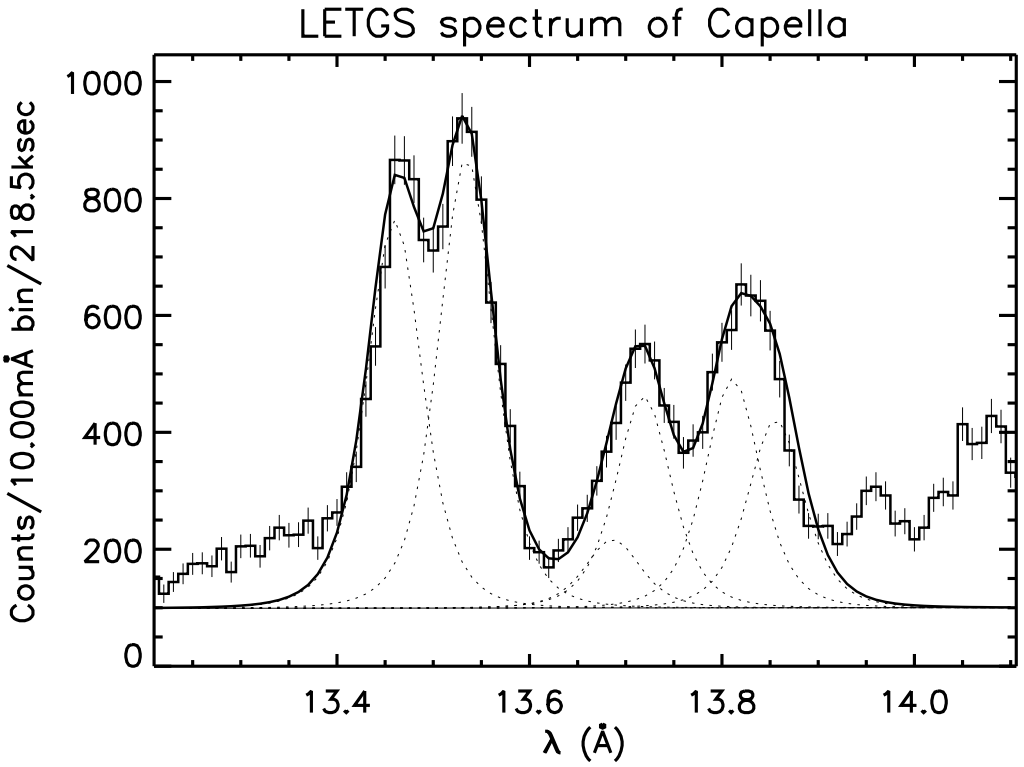


Figure 2: Result of a fit carried out with the settings in Fig. 3, applied to an LETGS spectrum of Capella. The line fluxes are fixed to fulfilling $(A_2 + A_3)/A_1 = 0.8$, granting a realistic temperature of the emitting plasma. With this constraint, the problems arising from the blend at 13.52 Å is avoided.

Output In Sect. 8 handling output is described. When the output of a text file with the fit results printed therein is desired, the switch *ires* must be activated. The filename can be specified in the call of the program with the option *-o*, i.e., `cora` *-o* `outputfile` or `cora_fit` *-o* `outputfile`. This file can be used to create a new plot with different plotting options without repeating the calculation. For this purpose the option Options/Replot Last OR ..From File is provided in the GUI. The command-line program can be started with `cora_fit` *-r* `outputfile` with `outputfile` the file generated with *-o* or with a hand written file (cf. Sect. 8).

LaTeX file The outputfile thus created can be converted into a LaTeX table with the the programs `cora_tex` and `cora_ftex` and options *-i* logfile *-l* lpp *-t* texfile *-a* arfile *-e* exp.time *-s* source name, where `logfile` indicates the outputfile from cora (default cora.dat), `lpp` indicates the number of lines per page (default 40), and `texfile` indicates the name of the LaTeX-file to be created (default cora.tex). With `cora_tex` the counts will be converted to fluxes, the exposure time must be entered after *-e* and a table containing effective areas (units cm^2) on any wavelengths grid (must be sorted!) can be indicated after *-a*. All options are optional, without options the default values will be used and the raw counts are listed. With the program `cora_ftex` a LaTeX table with both counts and fluxes is created. For personal settings we recommend to modify the source programs cora_tex.c or cora_ftex.c.

6.3 Format summary

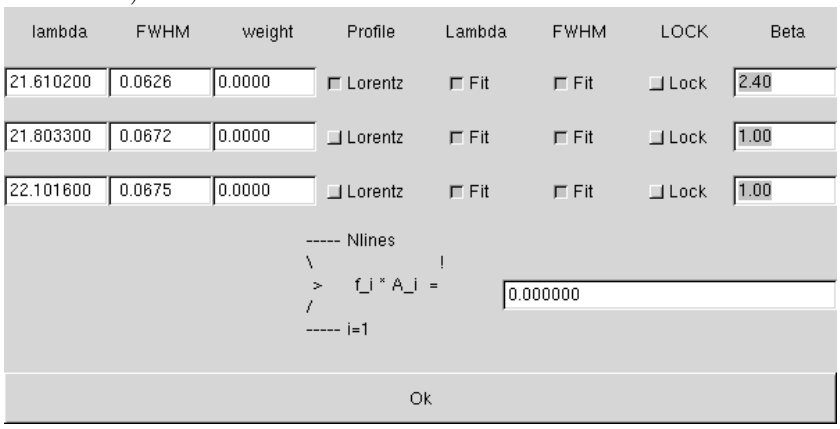
The following listing shows the required format of the parameter file. Parameters in italic are switches (0/1).

`version_number`	The version number.
`nlines`	The number of lines.
`wavelength1 width1 [...]`[a]	The line parameters for exactly `nlines` lines.
`...`	
`app_s app_b`, v, *fixflag*	relative apertures for source and background, right side in auxiliary condition (Sect. 7.1), fill empty bins with zeros
`sbg` *sbgit bgsmooth* sbgchi, nσ	sbg value, iterate and smooth switches, fit sbg with χ^2, median sbg range.
`dlam` *sigunit* `dsig` *ikev*	Spectral range parameters and units.
ilamit idlam `iord` *idsig*	Fit λ, fit λ with const. line distances, fit lines in higher order, fit const. FWHM
isigit	Fit FWHM switch.
ires	Write results into file.
iformat *iportrait*	Select file format, plot portrait switch.
iplot ilab ierrplot ilog nbin *spec_only fit_only*	Switches for creating plot, adding labels, plotting error bars, plotting y-logarithmic. Bin parameter. Switches for plotting only the spectrum/fit.
nlik *ispline*	supported only in IDL version; (Sect. 9).

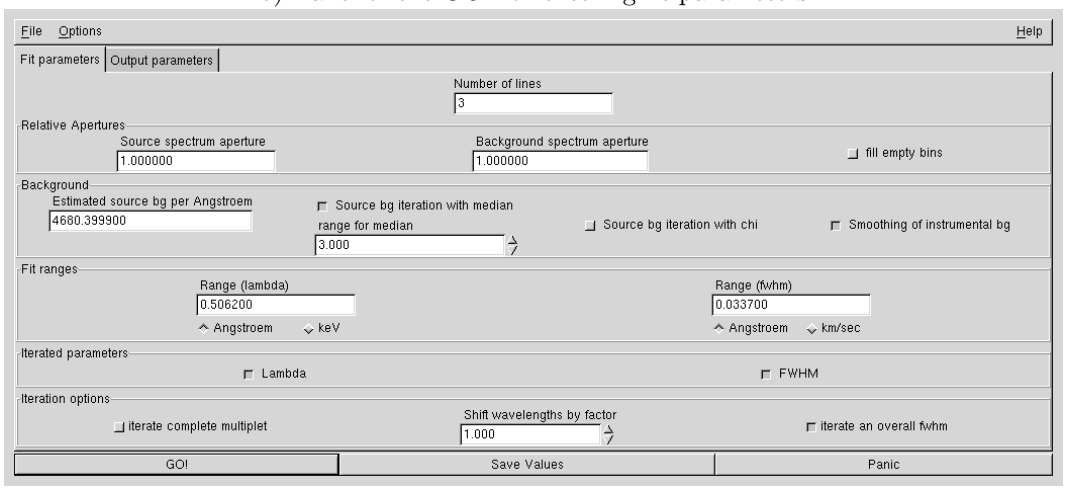
[a]Each line may have optionally the following additional parameters/flags:

`f_N`	(float)	weights (Sect. 7.1; default: 0.0)
`flag_profileN`	(0/1)	profile type (0 = Gauss, 1 = Lorenz, default 0)
`flag_lambdaN`	(0/1)	fit lambda (0 = FALSE, 1 = TRUE, default TRUE)
`flag_FWHMN`	(0/1)	fit FWHM (0 = FALSE, 1 = TRUE, default TRUE)
`flag_lockN`	(0/1)	lock lambda (0 = FALSE, 1 = TRUE, default FALSE)
β`_N`	(float)	value for exponent applied to Lorentzian profiles

a) Panel with line data



b) Panel of the GUI for entering fit parameters



c) Panel of the GUI for entering plot parameters

Figure 3: View of the Graphical User Interface.

7 The procedure

The procedure is based on the Maximum Likelihood method, assuming all noise to be Poissonian. This is necessary in all cases where the number of counts that constitute the spectrum is very small (Cash 1979, ApJ **228**, 939). From this assumption the probability for a model of an emission line spectrum to represent the measured spectrum is derived. The likelihood is used as a criterion for optimizing the parameters of the theoretical spectrum in order to obtain extremal likelihood values.
The point of this program is that the instrumental background is never subtracted thus conserving the basic assumption of Poissonian statistics. Instead, the theoretical spectrum is derived by summing up background and lines and comparing the sum with the measured spectrum without background subtraction.

The theoretical emission line spectrum consists of the designated number of lines with the designated profile function $g_{i,j}$ (presently only Gauss, Lorentz or combinations of these). The line positions and widths must be indicated, and will be optimized with an ordinary minimizing procedure (Powell), if desired. Best line fluxes are always calculated iterating the fixed point equation:

$$a_{j,_{new}} = \sum_{i=1}^{N} n_i \frac{a_{j,_{old}} g_{i,j}}{c_{i,_{old}}} \ . \tag{1}$$

with line fluxes a_j, the measured spectrum n_i and the theoretical spectrum $c_i = sbg + bg_i + \sum_{j=1}^{M} a_j g_{i,j}$ with a constant source background sbg and the instrumental background bg_i. For more details the draft of a paper is enclosed in the distribution.

7.1 Constraining the fit

The progress of obtaining best fits can be manipulated by applying restricting conditions to the procedure. This is done by indicating parameters for the linear condition

$$\sum_{i=0}^{nlines} f_i \times a_i = v \tag{2}$$

with a weight factor f_i for each line and a right hand side v. When all weight factors are zero, no restricting conditions will be applied. Only lines with weight factors $f_j \neq 0$ will have to meet the restrictions during the iteration, all lines with $f_j = 0$ are free, will have to adapt, though, to the restrictions of the other lines. For example, with Eq. 2 an expected count number for a line can be set fixed, by setting the corresponding weight factor $f_i = 1$ and setting v to the expected line counts. All other lines will then have to adapt to the one line having the desired line counts. Also line ratios can be fixed, by setting, e.g., $f_i = 0.5$, $f_j = -1.$, and $v = 0$. The ratio of a_j/a_i will then be 0.5 at the end of the fit, and the rest will end up with the best fit under this restriction.

7.2 Beta profiles

The two implemented profile functions can be modified for more refined profiles. The Lorentzian profile in combination with a parameter β as exponent is called Beta profile. For Chandra LETGS spectra $\beta = 2.5$ represents the instrumental profile better than the Gaussian profile. The implementation is

$$P_{\lambda,\mathrm{FWHM}}(x) = \left[\frac{\mathrm{FWHM}/2\pi}{[(x-\lambda)^2 + \mathrm{FWHM}^2/4}\right]^{\beta} \tag{3}$$

7.3 (Pseudo) Voigt profiles

Voigt profiles are provided as pseudo-Voigt approximation, consisting of a weighted sum

$$Voigt_{\mathrm{pseudo}} = (1-\eta)Gauss(\Gamma) + \eta Lorenz(\Gamma)$$

of a Gaussian and a Lorenzian. Here, the FWHM Γ is computed as

$$\Gamma = (\Gamma_{\mathrm{G}}^5 + 2.69296\Gamma_{\mathrm{G}}^4\Gamma_{\mathrm{L}} + 2.42843\Gamma_{\mathrm{G}}^3\Gamma_{\mathrm{L}}^2 + 4.47163\Gamma_{\mathrm{G}}^2\Gamma_{\mathrm{L}}^3 + 0.07842\Gamma_{\mathrm{G}}\Gamma_{\mathrm{L}}^4 + \Gamma_{\mathrm{L}}^5)^{1/5}.$$

The mixing parameter η is computed as

$$\eta = 1.36603(\Gamma_{\mathrm{L}}/\Gamma) - 0.47719(\Gamma_{\mathrm{L}}/\Gamma)^2 + 0.11116(\Gamma_{\mathrm{L}}/\Gamma)^3.$$

(For reference, see P. Thompson, D.E. Cox, J.B. Hastings, J. Appl. Cryst. 1987, **20**, 79.)

To use a Voigt profile, you need to define two lines, one with a Gaussian, the other with a Lorenzian profile, and lock their wavelengths. The required parameters in the parameter file are listed in Sect. 6.3. In the GUI, use the `Profile` and `LOCK` toggles in the line definition window (note that `LOCK` will lock to the wavelength of the preceding line, thus you should toggle this only for the second line in your Gaussian/Lorenzian pair). The Voigt profiles are not supported in the IDL version.

8 Description of output

Two ways are provided for retrieving information about the result of the procedure. A text file containing the fit results can be created and a plot of the desired part of the spectrum can be created. The decisive parameters are *ires* for creating a text file with results and *iplot* for creating a graph.

If called with the option *-v* the program will print the fit results together with performance information on the screen. More performance information can be obtained by typing the *-v* option up to three times, e.g., `cora` *-v -v* or `cora_fit` *-v -v* .

8.1 Text file with results

A text file with the fit results for each fitted line is created only when the switch *ires* is activated. The name of the file can be specified with the calling option *-o* `filename`; if no option is given, the default file name `cora.dat` will be used. If the file already exists the results are appended. This is, e.g., useful when the program is used for scanning a spectrum for lines and listing all results in a log-file.

The first line of the text file contains a header explaining the meaning of each column. Units are given in brackets, e.g. [cts]. When errors with value -1.0 are listed, this means that the variable is not iterated or that the value of the variable lies close to its iteration limits, set by *dsig* and *dlam*. The last column contains the result of the source background used for the calculation. It is given in units [cts/Å].

This text file can be used for creating a new graph with different plotting options without having to repeat the calculation. For this purpose simply change the parameter file either directly or with the user interface, and re-run the program with the option *-r* :

`cora_fit` *-i* `parameter_file` *-s* `spectrum_file` *-r* `textfile` .

From the graphical user interface this function is called from Options/Replot Last_OR_From File.

It can be converted into a LaTeX table with:

`cora_(f)tex` *-i* `logfile` *-l* `lpp` *-t* `texfile` *-a* `arfile` *-e* `exp.time` *-s* `source name` ,

where `logfile` indicates the outputfile from cora (default cora.dat), `lpp` indicates the number of lines per page (default 40), and `texfile` indicates the name of the LaTeX-file to be created (default cora.tex).

If the measured counts are to be converted to fluxes, the exposure time can be entered after *-e* and a table `arfile` containing effective areas (units cm^2) on any wavelengths grid (must be sorted!) can be indicated with *-a* `arfile`.

All options are optional, without options the default values will be used. From the graphical user interface this function is called from Options/Create TeX file from log-file. Corrections for effective areas cannot be done from the user interface, this has to be done from the command line.

8.2 Graph

The parameter *iplot* decides whether a graph will be created. The filename can be specified with the calling option *-p* :

`cora_fit` *-p* `plotfile` .

The default filename `cora.[ps,gif,ppm]` is used, when no option *-p* is specified. The suffix depends on the selected format (*iformat=[1,2,3]*); no file is created, of course, when *iformat=0*, i.e., X-Window is selected. When specifying a filename with *-p*, the suffix can be omitted. When not omitted, it should correspond to the suffix of the selected format, otherwise filenames like `cora.gif.ps` will result. The graph will contain the original spectrum, binned with *nbin* (bold line, histogram style), the fitted spectrum (continuous thin line for sum of lines, dotted line for individual lines), the total background (dashed line) and the instrumental background (line-dotted).

The wavelength range depends on the position of the lines, the line widths and the allowed ranges *dlam* and *dsig*. The appearance of the plot is determined by the input parameters *ilog* for logarithmic plotting (only y-axis), *ierrplot* for inclusion of error bars from the measured spectrum and *ilab* for putting labels into the plot quoting the fitted line parameters.

9 Usage with IDL

In addition to the program written in ANSI C an IDL version is delivered. This version is the predecessor of the presented program and is still the preferred working version by the developers. Using the IDL version allows the user to interactively process the fit results (as IDL is the *Interactive* Data Language). Also, the graphic is more flexible in IDL than in pgplot and better plots can be obtained without much trouble using IDL.

The IDL package contains the main program `fit_main.pro`, a file containing all functions and routines `fit_procs.pro`, an include file `fit.inc` that defines COMMON blocks and style settings, and a short script `do.pro` customizing the application.

Before using the IDL routines, the include must be 'sourced' and the routines must be compiled:

```
IDL> @fit.inc
IDL> .compile fit_procs
IDL> .compile fit_main
```

For the case that the spectrum was prepared in IDL it can be written into a file with the correct format using the procedure `writefile`:

```
IDL> writefile,wave,sp,err,bkgd,bkgderr,fname,exp=exp,eff=eff,/unix
```

where `wave` contains the wavelength vector, `sp` the spectrum, `err` the errors of the spectrum, `bkgd` the instrumental background, and `bkgderr` the errors of the background. All these vectors must have identical dimensions. In addition, the exposure time can be given (exp) and the effective areas (eff). The name of the resulting file, e.g., `spectrum.dat`, is indicated with the variable `fname`. If no full IDL version is available the keyword /unix can be used to still write the ASCII with the demo version of IDL. This takes longer than without the keyword.

A parameter file must be provided; it can be created with the GUI (cf. Sec. 5.2.1) or the command-line program `cora_inp` (Sect. 5.1.4). It follows the same format as the parameter file used by the C program. The parameter file can be edited by hand or one can run the GUI in parallel to the IDL session and use it just for editing the parameter file.

After compilation the program is started with

```
IDL> fit_main,[dtime=exposure time,]'spectrum.dat','fit.inp'[,psfile=psfile] .
```

Terms in [..] can be omitted, the default names 'spectrum.dat', 'fit.inp' can be replaced by any name corresponding to correct and existing names of the spectrum and the parameter file.

The last two parameters, listed in the last line of the parameter file, can be used to make IDL plot a graph of the likelihood curve in dependence of the iterated variables. This is useful in order to get an impression of the shape of each likelihood curve and thus the degree of the minimization problem. The parameter *nlik* represents the number of points calculated for such a graph, and with the second parameter, *ispline*, Spline interpolation can be activated ($ispline = 1$) or deactivated ($ispline = 0$). This function is also carried out by the C-program for the first line only, but is not supported by the GUI.

Format parameters controlling line styles, character size etc. can be set in `fit.inc`. Default values are given in the file `fit.inc` delivered with the program, but can be ignored. If no specification is given, the defaults are still used. Fit results and other variables and vectors used by the program are stored in COMMON blocks, listed in `fit.inc`, so after calling the fit routine, the variables can be used for further processing. This feature was, e.g., used by Ness, J.U., Mewe, R., Schmitt, J.H.M.M. et al. 2001 (A&A, **367**, 282) to disentangle first and third order of their Chandra LETGS spectrum of Capella. The third dispersion order lines were modeled in their first order, and these lines were then added to the instrumental background at the position of their third order on the detector plate. This new spectrum with the manipulated background spectrum was then used for the analysis of the remaining first order lines, recognizing the third order lines as part of the instrumental background.

A short example session is delivered with the program `do.pro`, where compiling and executing is done. Note that closing the plot device is necessary after running the program. This is left to the user for the case that the user wants to add plotting elements not provided by the program like labels before closing the device. The program is called with

```
IDL> @do
```