SED fitter v4 manual

Thomas Robitaille

13 June 2008

Contents

Introduction

The SED fitter was originally written to help analyze GLIMPSE data, and can now be used with any set of photometric data. The concept is very simple: consider a set of sources to be studied. For each source, the SED fitter can fit model stellar photospheres, YSO model SEDs, as well as galaxy and AGB templates to the multi-wavelength photometry measurements of this particular source using linear regression. The scale factor S (which is related to the luminosity and distance of the sources) and the extinction A_V are used as free parameters in the fitting process. The result for any given source is a value for the goodness of fit and best-fit values S and A_V for every single model. These fits can then be analyzed to derive properties of the source.

To quantify the goodness/badness of each fit to each source, we calculate the χ^2 value:

$$\chi^2 = \sum_{i=1}^{n} \left(\frac{F_i - M_i}{\sigma_i^2} \right)^2$$

where F_i are the flux values at a given wavelength λ_i, σ_i are the flux errors, and M_i are the extincted and scaled model values. In this manual we sometimes refer to the χ^2 per datapoint, $n_{\rm data}$, where the number of datapoints does *not* include upper and lower limits.

There are two kinds of models that can be used with the SED fitter:

- Models for which there is no aperture dependence on the flux and for which the absolute distance cannot be determined (e.g. stellar photosphere models)
- Models for which the flux depends on the aperture chosen and/or for which the distance can be found from the scalefactor of the fit. (e.g. YSO models). In this case, the fitting procedure is more complex as it involves computing the aperture-dependent SEDs for a fine grid of distances, and optionally removing models that would clearly be extended relative to the aperture chosen. This is described in more detail in Paper II.

Two binaries are available for fitting, optimized for each case: `fit_stellar` should be used for models with no aperture dependence and no distance information, while `fit` should be used for models with aperture-dependent fluxes and/or absolutely scaled models. Unlike the previous versions of the fitter, only one set of models can be used at a time, however, it is easy to produce a new set of models without changing a single line of code in the fitter distribution. If you wish to produce your own package of models, see the `model_packages.pdf` file.

Chapter 1

Installation

1.1 Requirements

The fitter requires a Fortran 95 compiler. The current source code has been successfully tested with g95-0.90.6, gfortran-4.4.0, and ifort-10.1. Also required are the `PGPLOT` library:

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

and the `cfitsio` library:

http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html

More information on configuring these can be found in Appendix **??**. The main SED fitter webpage can be found at the following url:

http://caravan.astro.wisc.edu/protostars/private/fitter/

1.2 Before the first installation

Before continuing, you need to ensure that the `$PGPLOT_DIR` and `$LD_LIBRARY_PATH` environment variables are set up correctly in your shell. Make sure that you are consistently using either all 32-bit or all 64-bit libraries/compilers, or you will get `invalid architecture` errors when compiling.

Before installing this fitter distribution for the first time on a computer, you will need to set up the following environment variables:

```
FITTER_COMPILER - compiler
FITTER_PGPLOT - compile flags for PGPLOT
FITTER_CFITSIO - compile flags for cfitsio
```

The compiler should be `ifort`, `g95`, or `gfortran` (for other compilers, contact me). These environment variables can be set using `setenv` in `csh`, and `export` in `bash`.
To check that these are set correctly, open a new terminal and type the following `echo` commands:

```
user$ echo $FITTER_COMPILER
ifort
user$ echo $FITTER_PGPLOT
-L/usr/X11R6/lib64 -lX11 -L/d/glimpse62/tom/lib/pgplot64 -lpgplot
user$ echo $FITTER_CFITSIO
-L/usr/local/lib64 -lcfitsio
```

1.3 Installing a new fitter directory

To create a fitter directory anywhere on your computer, first ensure that there is no directory named `sedfitter` in the current directory, and type:

```
git clone git@github.com:astrofrog/sedfitter-legacy.git sedfitter
```

Once this has completed, there should be a new directory called `sedfitter`.

To compile the source code, go inside the `sedfitter` directory, and enter `make`. This should compile the source code without any errors. The directory `bin` should contain the compiled binaries.

1.4 Installing a new set of models

To install a new set of models (e.g YSO SEDs, stellar photospheres, etc.) simply download the appropriate tar file[1], and expand it in a central repository on your computer. Then, create a symbolic link to it from your various fitter directories, e.g.:

```
cd /my/models/repository
tar xvzf models_r06.tgz
cd /where/I/keep/the/fitter
ln -s /my/models_repository/models_r06
```

1.5 Updating an existing fitter directory

To update, go inside the `sedfitter` directory (not any of the sub-directories) and type:

```
git pull origin master
```

Then, recompile the source code by typing

```
make clean
```

followed by

```
make
```

This should compile the source code without any errors. The directory `bin` should contain the updated binaries.

[1] http://sedfitter.readthedocs.org/en/stable/common_model_packages.html

Chapter 2

Preparing the data

2.1 The data file

The data file you will need to feed to the fitter should contain the name, coordinates, fluxes, and errors for each source that you wish to fit model SEDs to, with one source per line.

The fortran format for each source/line is:

`'(A30,2(1X,F9.5),n(1X,I1),2n(1X,ES11.4))'`

We refer to this format as the **fitter data format**, and we adopt the naming convention for all fitter data files of **data_name**, where **name** is the name you wish to use to refer to the data (for example you could call the data files `data_rcw79` or `data_taurus`). The various elements in the format are:

- `n` is the number of data points
- `A30` is the source name (e.g. GLIMPSE catalog name)
- `2(1X,F9.5)` corresponds to the two coordinates (ℓ/b, or α/δ, in degrees)
- `n(1X,I1)` corresponds to an integer for each data point which can take one of five values:
  - `0` for data points which are not valid or used
  - `1` for valid data points with fluxes and errors specified in mJy
  - `2` for lower limits specified in mJy.
  - `3` for upper limits specified in mJy
  - `4` for valid data points (as for `1`) but where the fluxes are expressed as $\log_{10}$[flux] and $\log_{10}$[error]. This is useful for sources which have a magnitude with a large error, which can then be converted to $\log_{10}$[flux] and $\log_{10}$[error] without worrying about the conversion from magnitude and magnitude error to linear fluxes and errors.
- `2n(1X,ES11.4)` corresponds to the data with fluxes and errors alternating: `flux(1), error(1), flux(2), error(2), ....` When the `valid` flag is set to `4`, $\log_{10}$[flux] and $\log_{10}$[error] should be specified. For lower and upper limits, the error should be set to the confidence placed on the limit, with 0 corresponding to 0% and 1 corresponding to 100%. For example, in the case of an upper limit, setting the confidence to 1 effectively forbids any models to be larger

than the upper limit, while setting the confidence between 0 and 1 allows such models, but introduces a χ^2 penalty dependent on the confidence. Setting the confidence to 0 effectively disables the upper limit.

Such a file can also be generated automatically from a GLIMPSE/GLIMPSE II/GLIMPSE 3D catalog using the `catalog2data` utility. This utility extracts source names, coordinates, fluxes, errors, as well as many flags and values which are added at the end of the line. The `catalog2data` command is used as follows:

Syntax

```
Usage: catalog2data [arguments]

Required arguments :
  input=filename     - the GLIMPSE catalog or archive file
  output=filename    - the output fitter file
  coords=value       - the coordinate system (gal/equ)

Optional arguments :
  box=yes/no          - whether to constrain coordinates (default no)
  xmin=value          - the lower x coordinate of the box
  xmax=value          - the upper x coordinate of the box
  ymin=value          - the lower y coordinate of the box
  ymax=value          - the upper y coordinate of the box
  rejectsingle=yes/no - reject single detections (default no)
```

This utility can only be used with `v2.0` GLIMPSE I/II/3D Catalogs and Archives.

The following example shows the simplest way to run `catalog2data`:

Example 1 - simple extraction

```
bin/catalog2data catalog=GLIMPSE_l319-320_Archive.20061201.txt \
                 output=data_319 coords=gal
```

The above command will extract all sources from

`GLIMPSE_l319-320_Archive.20061201.txt`

and the coordinates in the fitter file will be Galactic coordinates. It is also possible to specify only a range of coordinates to extract:

Example 2 - simple extraction with constraints on coordinates

```
bin/catalog2data catalog=GLIMPSE_l319-320_Archive.20061201.txt \
                 output=data_319 coords=gal box=yes xmin=319 xmax=319.5 ymin=0 ymax=0.5
```

The above command will extract all sources with $319^\circ < \ell < 319.5^\circ$ and $0^\circ < b < 0.5^\circ$ and the coordinates in the fitter file will be Galactic coordinates.
Finally, the `rejectsingle` option can be used to remove fluxes that are based on single detections. This is recommended, as most red sources with single detections are unreliable.

2.2 The data parameter file

The data parameter file is one containing information about the data you wish to fit. That is the number of wavelengths, and the filters and apertures the fluxes were measured in.

An example of data parameter file is provided in `config/data_glimpse_ex.par`. This file contains the parameters you would need for a GLIMPSE, GLIMPSE II, or SAGE catalog for example (i.e. one created with the `extract_catalog` program). The format of the file is:

```
############### DATA-DEPENDANT PARAMETERS ###############
7                       = Number of wavelengths
2J 2H 2K I1 I2 I3 I4    = Filters (format: xx xx xx xx xx)
3. 3. 3. 3. 3. 3. 3.    = Aperture radii in arcseconds
```

The filters are those the fluxes were measured in. These should be in the same order as in the data file. Filters are specified by two or more characters separated by a space. For a list of all the filters available, see Appendix A. The radii are those of the apertures used to compute the fluxes (in arcseconds). This can be used for example to eliminate SED models of YSOs which would have been well resolved in a given aperture when using the aperture-dependent SED fitter. In that case, the outermost radius at which the surface brightness falls to 50% of the peak, $r_{1/2}$, is calculated for all models. If the user has specified that the source looks smaller than the aperture at all wavelengths, then models where $r_{1/2} >$ aperture in at least one aperture are discarded. This is because if the surface brightness of a model is half of the peak surface brightness at the edge of the aperture, then this means that it is clearly resolved with respect to the aperture size. For these reasons, the aperture size should be specified to the best of one's knowledge.

2.3 Signal-to-noise cuts

The `sn_filter` utility can be used to reject flux measurements which do not meet a given signal-to-noise requirement, and is used as follows:

Syntax

```
Usage: sn_filter [arguments]

Required arguments :
  n_wav=value        - number of wavelengths
  par_file=filename  - the parameter file (e.g. config/sn_requirements.par)
  input=filename     - the input fitter data file
  output=filename    - the output fitter data file
```

where `par_file=filename` gives the path to a parameter file with the minimum signal-to-noise required for each filter, one per line (in the same order as the columns in the data file).

For example, to apply signal-to-noise cuts to a typical data file with JHK and IRAC fluxes, in order to keep all fluxes with a signal-to-noise of at least 5 in JHK and IRAC [1] and IRAC[2], and at least 10 in IRAC [3] and IRAC [4], you would need a file which contains the following:

```
5
5
5
5
5
5
5
```

You can add comments to the right of the numbers provided that you leave a space in between. For example, you could write:

```
5   = minimum signal-to-noise of J flux
5   = minimum signal-to-noise of H flux
5   = minimum signal-to-noise of K flux
5   = minimum signal-to-noise of IRAC [1] flux
5   = minimum signal-to-noise of IRAC [2] flux
5   = minimum signal-to-noise of IRAC [3] flux
5   = minimum signal-to-noise of IRAC [4] flux
```

Any source which does not meet any of these signal-to-noise requirements is not included in the new data file.

An example of such a parameter file is provided in `config/sn_requirements_ex.par`. To apply signal-to-noise requirements to the `data_319` file created in §2.1, you would need to type the following:

Example

```
bin/sn_filter n_wav=7 par_file=config/sn_requirements_ex.par \
              input=data_319 output=data_319_filtered
```

This would produce a new data file in fitter format called `data_319_filtered`, which would contain only fluxes with sufficient signal-to-noise.

2.4 Resetting flux errors

The `reset_errors` utility can be used to set a lower limit on the errors for fluxes in a fitter data file, and is used as follows:

Syntax

```
Usage: reset_errors [arguments]

Required arguments :
  n_wav=value        - number of wavelengths
  par_file=filename  - the parameter file (e.g. config/min_frac_error.par)
  input=filename     - the input fitter data file
  output=filename    - the output fitter data file
```

where `<par_file>` is a parameter file with the minimum fractional error desired for each filter, one per line (in the same order as the columns in the data file).

For example, to reset the errors in a typical GLIMPSE data file (JHK+IRAC) to at least 5% for the JHK fluxes and to at least 10% for the IRAC data, you would need a file which contains the seven following lines:

```
0.10
0.10
0.10
0.10
0.10
0.10
0.10
```

You can add comments to the right of the numbers provided that you leave a space in between. For example, you could write:

```
0.10   = minimum fractional error on J flux
0.10   = minimum fractional error on H flux
0.10   = minimum fractional error on K flux
0.10   = minimum fractional error on IRAC [1] flux
0.10   = minimum fractional error on IRAC [2] flux
```

```
0.10   = minimum fractional error on IRAC [3] flux
0.10   = minimum fractional error on IRAC [4] flux
```

These settings would result in any errors on the JHK flux errors being reset to 5% if they are smaller than 5%, remaining unchanged if they are larger than this, and any errors on the IRAC fluxes being reset to 10% if they are smaller than 10%, remaining unchanged otherwise.

An example of such a parameter file is provided in `config/min_frac_errors_ex.par`. To reset the errors, using these settings, to the fluxes in the `data_319` file created in §2.1, you would need to type the following:

Example

```
bin/reset_errors n_wav=7 par_file=config/min_frac_err.par \
                 input=data_319 output=data_319_reset
```

This would produce a new data file in fitter format called `data_319_reset`, which would contain the same sources as the input file, but with modified errors in cases where the errors were smaller than the lower limits specified.

Note: if you want to reset errors and apply signal-to-noise cuts, you should apply signal-to-noise first, and then reset the errors (as resetting the errors changes the signal-to-noise).

Chapter 3

The parameter files

3.1 A general note on parameter files

Parameter files are made up of lines with syntax:

```
value = key = comment
```

When reading in parameters, the various programs look for the required parameters based on the `key`, so it is possible to re-order the parameters, to insert blank lines or comment lines. For the three plotting programs discussed in Chapter 6, it is possible to split the parameters into three files, one for each program. On the other hand, it can be advantageous to leave all parameters in the same file to control the number of fits to use uniformly for the three programs. One ubiquitous pair of parameters is the output format and number, which consists of a single character for the format, and a real or integer number. The options are outlined in Table 3.1.

3.2 The fitter parameter file

The fitter parameter file is one containing parameters for the fitting process. Two examples of such a file are the `fitter_stellar_ex.par` and `fitter_yso_ex.par` files in the main `fitter` directory. The former is an example of parameter file for aperture and scale-independent models (e.g. Kurucz stellar photospheres) and the latter is an example of parameter file for aperture and scale-dependent models (e.g. R06 YSO models). The parameters are:

- `config/data_glimpse_ex.par = dform = data format`
  `data_comparison = dfile = data file`

  These filenames refer to the data parameter file and the input data file in fitter format.

- `4 = drequ = minimum number of datapoints`

  This is the minimum number of datapoints that you require to be able to fit a source. Any source with a number of valid datapoints less than this will be 'dropped', and will not be considered for fitting. I would suggest never to set this value any lower than `4`.

- `models_r06 = modir = models directory`

  Which models to use in the fitting.

Table 3.1: Output options for fitting and plotting programs

oform	onumb	model fits selected
N	n	n best fits
C	v	all fits with $\chi^2 <$ v
D	v	all fits with $\chi^2 - \chi^2_{\rm best} <$ v
E	v	all fits with $\chi^2/n_{\rm data} <$ v
F	v	all fits with $\left(\chi^2 - \chi^2_{\rm best}\right)/n_{\rm data} <$ v

- `3.` `= minav = Minimum interstellar extinction`
  `22.` `= maxav = Maximum interstellar extinction`

  These extinction values refer to interstellar extinction, and in the case of the YSO models do not take into account the extinction caused by the circumstellar environment (disks, envelopes, etc...) which is intrinsic to the models.

- `2.` `= mind = minimum distance in kpc`
  `2.5` `= maxd = maximum distance in kpc`

  If the aperture-dependent SED fitter is used, a distance range can be specified. This distance range (specified in kpc) is used to rule out any models which do not have the right luminosity to fit the data. These options are not required for the aperture-independent fitter as shown in the `fitter_stellar_ex.par` file provided.

- `config/ex_law_gl.par` `= exlaw = extinction law`

  The extinction law to use

- `output.fits` `= ofile = output filename`
  `F` `= oform = what to output (N/C/P/D/E/F)`
  `6.` `= onumb = number relating to the above option`

  Output options - the `oform` and `onumb` options specify the maximum number of fits to be stored in the fitter output file (see Table 3.1).

- `Y` `= oconv = Output convolved fluxes`

  Whether to output convolved fluxes to the output file (used for plotting the convolved fluxes with `src/plot`.

- `N` `= kpext = Any sources larger than aperture? (Y/N)`

  Whether the source is larger than the aperture at any wavelength. If set to `N`, models with $r_{1/2} >$ aperture in at least one aperture are discarded (see §2.2). This option is not required for the aperture-independent fitter as shown in the `fitter_stellar.par` file provided.

Chapter 4

Running the fitter

To run the fitter, use the `fit` or `fit_stellar` commands:

Syntax

```
Usage: fit [arguments]

Required arguments :
  par_file=filename  - the parameter file (e.g. config/sn_requirements.par)

Optional arguments :
  output=filename    - the output directory
  input=filename     - the input fitter data file
  models=directory   - the models to use
  best=yes/no        - whether to output only the best fit
```

For example if your parameter file is named `fitter.par`, and you are fitting stellar photosphere models, type

Example 1

```
bin/fit_stellar  par_file=fitter.par
```

The fitter should now run! An output FITS file will be created, containing information about the sources and the model fits. This output can be further processed with the programs described in the following chapters.

To make it easier to write pipeline scripts to fit a region several times with different models, several options can be specified after the fit command. These options **override** the values set in the `fitter.par` file:

- `output=filename` tells the fitter what to call the output FITS file (the filename should end in `.fits`)
- `input=filename` tells the fitter which input file to use

- `models=directory` tells the fitter which models to use
- `best=yes` or `best=no` tells the fitter to output only the best SED fit or all SED fits respectively (effectively sets `oform=N` and `onumb=1`).

You can use any combination of these. If you do not specify one of these options, the value in the parameter file will be used instead. For example, to use the Robitaille et al (2006) YSO models, output the results to `myoutput.fits`, and always output the best fit, regardless of the settings in `fitter.par`, you would need to type:

Example 2

```
bin/fit  par_file=fitter.par  output=myoutput.fits  models=models_r06
```

provided that a symbolic link to `models_r06` exists inside the current directory.

Chapter 5

Output files

5.1 The output file

Unlike previous versions of the fitter, only one output file is initially produced, containing all the information about the sources and the fits. Furthermore, the output file conforms to the FITS format, making it directly readable by utilities such as `fv`. Therefore, there is no obligation to further process the file to make it human-readable.

Note: for the output modes `C`, `D`, `E`, and `F`, the FITS file always contains one extra fit so as to ensure that the χ^2 boundary specified has been passed.

5.2 Splitting the output file

Once the above FITS file has been produced, it is possible to split it to separate well and badly fit sources. To do this, the `filter_output` command can be used:

Syntax

```
Usage: filter_output [arguments]

Required arguments :
  input=filename     - the input file

then one and only one of the following:
  chi=value          - constrain using total chisquared
  cpd=value          - constrain using chisquared per datapoint
```

The various ways to split the output are:

- `chi=value`: this means that any source whose best fit (on the basis of the χ^2 value) has a total (not reduced) χ^2 less than `value` will be considered as well fit, and any other source will be considered as badly fit.
- `cpd=value`: this means that any source whose best fit (on the basis of the χ^2 value) has a χ^2 per data point value less than `value` will be considered as well fit, and any other source will be considered as badly fit.

Only one option should be specified.

This command will create two new files in the original output directory, one for the well-fitted sources, and one for the badly-fitted sources. For example, to split the `myoutput.fits` directory created in Chapter 4, into sources which are well fit with a χ^2 value per datapoint less than 3, and the remaining sources, you would need to type:

Example

```
bin/filter_output  input=myoutput.fits  cpd=3
```

This will create `myoutput_good.fits` and `myoutput_bad.fits`

5.3 Making a data file from an output FITS file

A utility named `fits2data` is available to produce a data file suitable for input to the fitter from an output file. It is used as follows:

Syntax

```
Usage: fits2data [arguments]

Required arguments :
  input=filename     - the input fitter FITS file
  output=filename    - the output data file
```

Chapter 6

Plotting the results

6.1 Overview

There are three tools available to plot the results. `plot` produces plots of the SED fits, `plot_params_1d` produces histograms of parameter values, and `plot_params_2d` produces plots with one parameter on each axis (useful to distinguish real from artificial correlations between parameters). Two examples of plotting parameter files are provided: `plot_stellar.par` is set to show only the best fit (useful for stellar photosphere models for example) and `plot.par` is set to show a large number of good fits, making use of the greyscale feature and interpolating the SED to different apertures at different wavelengths (useful for the YSO models for example). Both these files contains all the parameters required by the three plotting programs. Either of the files can technically be split into three different files, but there are advantages of not doing so, for example to make it easy to consistently change the number of model fits shown in the different plots (via the `oform` and `onumb` parameters. The parameters are split into **General** and **Advanced** parameters, and in each case, whether they apply to all programs, or one in particular.

The **general** options required by all programs are:

```
N       = oform = what to plot (N/C/P/D/E/F)
1000    = onumb = number relating to the oform option
```

These are the output options - the `oform` and `onumb` options specify the maximum number of fits to be plotted (see Table 3.1).

The **advanced** options required by all programs are:

```
Y        = pname = show source name on plot (Y/N)
.eps/VPS = devic = PGPLOT device (change to .eps/VCPS for color)
0.75     = chlab = title and labels character height
0.60     = chaxi = numerical axis character height
2        = lwbox = line width of box, numbers, and labels
1        = lwsed = line width of SEDs
0.75     = shade = shade of grey
2.5      = labdx = x-label displacement
3.0      = labdy = y-label displacement
```

These are very technical options used to change the appearance of the plots. The default values are sensible, but you can always tweak them if needed.

6.2 plot

This is the main plotting program which produced SED plots. The `plot` specific parameters are:

- ```
  A       = pmode = mode - I(ndividual) or A(ll SEDs on one plot)
  Y       = pgrey = if in A mode, whether to plot a greyscale
  ```

  `I` mode means make one plot per source per fit. `A` mode means make one plot per source with all the fits on the same plot. For `A` mode, the best fit is shown in black, and the subsequent fits are shown in grey. However, this may produce `eps` files which take a while to draw - the `pgrey` option fixes this by rasterizing all the grey SEDs into a single greyscale map, which makes loading almost instantaneous.

- ```
  A       = xmode = X values - A(uto)/M(anual)
  0.1     = xminm  = Xmin (if manual)
  100.    = xmaxm  = Xmax (if manual)
  1.      = xmina = Xmin margin in orders of magnitude (if auto)
  1.      = xmaxa = Xmax margin in orders of magnitude (if auto)

  A       = ymode = Y values (A(uto)/M(anual))
  1.e-14  = yminm  = Ymin (if manual)
  1.e-9   = ymaxm  = Ymax (if manual)
  1.      = ymina = Ymin margin in orders of magnitude (if auto)
  2.      = ymaxa = Ymax margin in orders of magnitude (if auto)
  ```

  These options control the minimum and maximum x and y values for the plots. If the `xmode` or `ymode` options are set to `M` for example, then the `xmin` and `xmax` or `ymin` and `ymax` values are used. If the options are set to `A`, then the plotting tool automatically finds the minimum and maximum values from the data, and adds a margin specified in the above parameters. For example, if `xmode=A`, and if the minimum datapoint is at 1μm, and the `xmina` margin is set to `1`, then the minimum wavelength on the plot will be 0.1μm.

- ```
  Y       = pseds = plot SEDs (Y/N)
  N       = pconv = plot convolved fluxes (Y/N)
  Y       = patmo = overplot stellar atmosphere (Y/N)
  interp  = stype = what type of SED to show (see manual)

  Y       = pinfo = show fit info on plot (Y/N)

  N       = pcapt = create color caption (Y/N)  (note: not implemented to date)
  ```

  These are extra options which allow you to further customize the plots. You can either plot SEDs, or the convolved fluxes (requires `oconv=Y` in `fitter.par`), or both, and you can show the stellar photosphere (if `pmode=A` only the stellar photosphere for the best fit is shown). The `stype` parameter sets what kind of SED to plot. The options are:

  - `largest` - the SED for the largest aperture specified for the data
  - `largest+smallest` - the SEDs for the largest and smallest apertures specified for the data
  - `interp` - an SED interpolated to the various apertures as a function of wavelength. This works best if there are no sudden jumps in the aperture as a function of wavelength.
  ```

– `all` - the SEDs for all the different apertures with color coded lines. Note that this is no longer possible when `pmode=A` as this produced confusing plots.

In the case where `stype=all`, a color caption can be produced showing the color of each SED as a function of aperture. Finally, `pinfo` allows the χ^2, A_V and scalefactor to be overplotted on the SED plot.

The following advanced options are available for customizing the look of the plots:

```
400     = greyx = greyscale resolution (x direction)
300     = greyy = greyscale resolution (y direction)
2.5     = greyc = greyscale clipping value
8.0     = greym = greyscale stretch
```

The `plot` command is used as follows:

Syntax

```
Usage: plot [arguments]

Required arguments :
  par_file=filename  - the parameter file (e.g. plot.par)
  input=filename     - the input FITS file
  output=filename    - the output plots directory
```

6.3 plot_params_1d

This plotting program can produce histograms of the parameters of a given number of YSO fits. How many fits are shown is controlled as before using the `oform` and `onumb` parameters. *All the fits which contribute to the histogram are weighted equally.*

As well as the common options specified in §6.1 an advanced option is available to specify the number of bins in the histogram. This number should be a multiple of 10 for best results:

```
30      = histn = number of bins in histogram
```

This tool is used as:

Syntax

```
Usage: plot_param_1d [arguments]

Required arguments :
  par_file=filename  - the parameter file (e.g. plot.par)
  input=filename     - the input fitter FITS file
  output=directory   - the output plots directory
  parameter=name     - the name of the parameter to show (e.g. MDISK)
```

```
  log=yes/no          - whether to plot the parameter on a log scale
  zero=yes/no         - if log=yes, whether to show zero values
```

For a list of possible `parameter` values for the R06 or the Kurucz models, see Appendix B. If you define your own set of models, whatever numerical parameters you include in `parameters.fits.gz` will be available here.

Two histograms are shown in each plot. In grey, the distribution of models in the grid is shown, normalized so that the maximum value is 1. The hashed histogram shows the distribution of the fits, also normalized to its maximum value (the normalization factor is not the same for the two histograms as the good fits only represent a small fraction of all the models in the grid).

6.4 plot_params_2d

This plotting program can produce two-dimensional maps of the parameters. How many fits are shown is controlled as before using the `oform` and `onumb` parameters.
As well as the common options specified in §6.1 a few advanced options are available, although the default values should be sensible:

```
8       = greyspl2d = greyscale resampling
2048    = greybig2d = greyscale resolution (major direction)
136     = greysma2d = greyscale resolution (minor direction)
13.     = greycli2d = greyscale clipping value
40.     = greymax2d = greyscale stretch
0.75    = ch2dpoint = size of points in 2d plots
```

This tool is used as:

Syntax

```
Usage: plot_param_2d [arguments]

Required arguments :
  par_file=filename  - the parameter file (e.g. plot.par)
  input=filename     - the input fitter FITS file
  output=directory   - the output plots directory
  parameterx=name    - the name of the parameter to show on the x axis
  parametery=name    - the name of the parameter to show on the y axis
  logx=yes/no         - whether to plot the x-axis parameter on a log scale
  logy=yes/no         - whether to plot the y-axis parameter on a log scale
  zerox=yes/no        - if logx=yes, whether to show zero values for the x-axis
  zeroy=yes/no        - if logy=yes, whether to show zero values for the y-axis
```

The distribution of models in the grid is shown in grey points. All the fits are shown as black points.

Chapter 7

Custom analysis

The main fitter output files are in FITS format, and should be readable with programs such as `fv`. However, in some cases you may wish to perform some post-processing tasks, such as constraining parameters using existing information, computing average parameters, etc. To do this, there are two options:

- Two `generic_ex?.f90` analysis programs are provided in `src/custom`, that you can modify as you wish. It is **strongly** recommended that you rename these file from `generic_ex?.f90`, as subsequent code updates `will` overwrite the existing `generic_ex?.f90` files. In addition, you will need to modify the Makefile to compile the new file.

  The basic `generic_ex?` programs are used as follows:

  Syntax

  ```
  Usage: generic_ex? [arguments]

  Required arguments :
    par_file=filename  - the parameter file (e.g. plot.par)
    input=filename     - the input FITS file
    output=filename    - the output ASCII file
  ```

  where `par_file` can be any parameter file containing the `oform` and `onumb` parameters (for example `plot.par`). The two examples demonstrate how to work with a specific subset of parameters, or all parameters. The `generic_ex2` program should work on any set of models, while `generic_ex1` will only work with the R06 YSO models as it looks specifically for the `MDISK`, `MDOT`, and `MASSC` parameters.

- The other option is to write your own program in the language of your choice (e.g. IDL) to read in the FITS files. The syntax of all the FITS files is described in the `model_packages.pdf` document. All files conform to the FITS standard and therefore should not pose any problems for reading in.

Chapter 8

Example run-through

The following are examples of the commands that could be typed to analyze a GLIMPSE region, although this is by no means a unique way to proceed.

The starting point is a GLIMPSE catalog, say `GLIMPSE_l064-065_Archive.20050409.txt`. First, we convert it to the fitter data format using:

```
bin/catalog2data input=GLIMPSE_l009-010_Catalog.20061201.txt \
                 output=data_glimpse coords=gal
```

Then, we apply signal-to-noise requirements using the default `config/sn_requirements_ex.par` file:

```
bin/sn_filter n_wav=7 par_file=config/sn_requirements_ex.par \
              input=data_glimpse output=data_glimpse_filt
```

We apply the lower limits on the fluxes using the default `config/min_frac_errors_ex.par` file:

```
bin/reset_errors n_wav=7 par_file=config/min_frac_errors_ex.par \
                 input=data_glimpse_filt output=data_glimpse_new
```

We set the parameters as follows in the `fitter_stellar_ex.par` file:

```
------------------------- Data --------------------------
config/data_glimpse_ex.par = dform = data format
data_glimpse_new        = dfile = data file
4                       = drequ = minimum number of datapoints

------------------------ Fitting ------------------------
models_kurucz           = modir = models directory
config/ex_law_gl.par    = exlaw = extinction law
0.                      = minav = Minimum interstellar extinction
40.                     = maxav = Maximum interstellar extinction

------------------------ Output -------------------------
output_stellar.fits     = ofile = output filename
N                       = oform = what to output (N/C/P/D/E/F)
1                       = onumb = number relating to the above option

----------------------- Advanced ------------------------
Y                       = oconv = Output convolved fluxes
```

We now run the aperture-independent fitter. The parameter file specifies that the Kurucz stellar photospheres should be used, and only the best fit should be output for each source.

```
bin/fit_stellar par_file=fitter_stellar_ex.par
```

Once this is done, we now choose to filter out all the sources which can be fit well by stellar photospheres. We can do this on the basis of the χ^2 per data-point value of the best fit for example. For this example, we use $\chi^2_{\rm best}/n_{\rm data} = 3$ as the limit below which a source is considered to be well fit:

```
bin/filter_output input=output_stellar.fits cpd=3
```

We can now produce SED plots of all the sources which were well fit. First we make a copy of the `plot.par` file called `plot_stellar_ex.par`, and we edit the parameters to show only the best fit:

```
------------------ General - All programs ---------------

N       = oform = what to plot (N/C/P/D/E/F)
1       = onumb = number relating to the oform option

------------------- General - SED plots -----------------

I       = pmode = mode - I(ndividual) or A(ll SEDs on one plot)
N       = pgrey = if in A mode, whether to plot a greyscale

A       = xmode = X values - A(uto)/M(anual)
0.1     = xminm  = Xmin (if manual)
100.    = xmaxm  = Xmax (if manual)
1.      = xmina = Xmin margin in orders of magnitude (if auto)
1.      = xmaxa = Xmax margin in orders of magnitude (if auto)

A       = ymode = Y values (A(uto)/M(anual))
1.e-14  = yminm  = Ymin (if manual)
1.e-9   = ymaxm  = Ymax (if manual)
1.      = ymina = Ymin margin in orders of magnitude (if auto)
2.      = ymaxa = Ymax margin in orders of magnitude (if auto)

Y       = pseds = plot SEDs (Y/N)
N       = pconv = plot convolved fluxes (Y/N)
Y       = patmo = overplot stellar atmosphere (Y/N)
interp  = stype = what type of SED to show (see manual)

Y       = pinfo = show fit info on plot (Y/N)

N       = pcapt = create color caption (Y/N)
```

Then, we run:

```
bin/plot par_file=plot_stellar_ex.par input=output_stellar_good.fits output=plots_stellar
```

We produce a data file for the badly fit sources:

```
bin/fits2data input=output_stellar_bad.fits output=data_nonstellar
```

We edit the `fitter_yso_ex.par` file to set the distance range to set the various parameters (note: these are just example values):

```
------------------------- Data --------------------------
config/data_glimpse_ex.par = dform = data format
data_nonstellar         = dfile = data file
4                       = drequ = minimum number of datapoints

------------------------ Fitting ------------------------
models_r06              = modir = models directory
config/ex_law_gl.par    = exlaw = extinction law
2.                      = mind  = minimum distance in kpc
2.5                     = maxd  = maximum distance in kpc
3.                      = minav = Minimum interstellar extinction
22.                     = maxav = Maximum interstellar extinction

------------------------ Output -------------------------
output.fits             = ofile = output filename
N                       = oform = what to output (N/C/P/D/E/F)
1000                    = onumb = number relating to the above option

------------------------ Advanced -----------------------
Y                       = oconv = Output convolved fluxes
N                       = kpext = Any sources larger than aperture? (Y/N)
```

And we run the aperture-dependent fitter:

```
bin/fit par_file=fitter_yso_ex.par
```

The sources can once again be split into well and badly fit sources using the same criterion as previously:

```
bin/filter_output input=output_yso.fits cpd=3
```

The `plot_yso_ex.par` file is edited to set `oform=F` and `onumb=3.`, `pmode=A`, and `pgrey=Y` and the plotting program is run:

```
bin/plot par_file=plot_yso_ex.par input=output_yso_good.fits output=plots_yso_good
```

```
bin/plot par_file=plot_yso_ex.par input=output_yso_bad.fits output=plots_yso_bad
```

Have fun!

Appendix A

Available filters

BU	Bessel U-Band	M1	MIPS 24
BB	Bessel B-Band	M2	MIPS 70
BV	Bessel V-Band	M3	MIPS 160
BR	Bessel R-Band	S1	IRAS 12 microns
BI	Bessel I-Band	S2	IRAS 25 microns
2J	2MASS J-Band	S3	IRAS 60 microns
2H	2MASS H-Band	S4	IRAS 100 microns
2K	2MASS K-Band	W1	Scuba 450WB microns
UL	UKIRT L-Band	W2	Scuba 850WB microns
LP	UKIRT L'-Band	N1	Scuba 350NB microns
UM	UKIRT M-Band	N2	Scuba 450NB microns
I1	IRAC Band [1]	N3	Scuba 750NB microns
I2	IRAC Band [2]	N4	Scuba 850NB microns
I3	IRAC Band [3]		
I4	IRAC Band [4]		
XA	MSX Band A		
XC	MSX Band C		
XD	MSX Band D		
XE	MSX Band E		

Appendix B

Available R06 YSO parameters

```
TIME      Age of the central source
(used to find Rstar and Tstar from evolutionary tracks, but does
not relate to the circumstellar environment parameters)

MASSC     Mass of the central source [Solar Masses]
RSTAR     Radius of the central source [Solar Radii]
TSTAR     Temperature of the central source [K]
MDOT      Envelope accretion rate [Solar Masses/yr]
RMAX      Envelope radius [AU]
THETA     Envelope cavity angle [degrees]
RMINE     Envelope inner radius [Dust sublimation radii]
MDISK     Disk mass [Solar Masses]
RMAXD     Disk outer radius [AU]
RMIND     Disk inner radius [Dust sublimation radii]
RMIND(AU) Disk inner radius [AU]
RC        Centrifugal Radius
RCHOLE    -
ZMIN      Disk scaleheight [relative to HSEQ]
A         Disk density power
B         Disk flaring power
ALPHA     Disk alpha parameter
RHOCONST  Envelope cavity density [cgs]
RHOAMB    Ambient density around the envelope [cgs]
MDOTDISK  Disk accretion rate [Solar Masses/yr]
INCL.     Inclination [degrees 90=edge-on]
AV_INT    Extinction inside the model down to the stellar surface
TAU60     Visual optical depth at an inclination of 60 degrees
LTOT      Total luminosity
H100      Disk scaleheight at 100AU
```