3.2 Modeling the surface brightness profile

Next we model the above derived surface brightness profile. Outside the core, the surface brightness is often well described by a single beta model (see lecture notes)

$$I(b) = I_0 \left(1 + \left(\frac{b}{r_{core}}\right)^2\right)^{(-3\beta+0.5)} \tag{3.1}$$

(where b is the projected radius, r$_{core}$ is the core radius, β is the slope parameter and I$_0$ is the central surface brightness). Usually the rapid cooling in the dense core creates a brightness peak, which requires a second component to the model, i.e.

$$I(b) = I_{0,1} \left(1 + \left(\frac{b}{r_{core_1}}\right)^2\right)^{(-3\beta_1+0.5)} + I_{0,2} \left(1 + \left(\frac{b}{r_{core_2}}\right)^2\right)^{(-3\beta_2+0.5)} \tag{3.2}$$

In the current implementation, the β parameter is equal in both components, i.e, $\beta_1 \equiv \beta_2 \equiv \beta$

In order to find a reasonable starting point for the fit, vary first the β model parameters in the file **make_surf_par.pro**. Typical parameters for the nearby clusters are $r_{core_1} \sim 2-3$ arcmin, $\beta \sim 0.6-0.7$, and the second component is much narrower, $r_{core_2} < 0.5$ arcmin. The normalization of the narrower component may be 10 times as high as that of the wider component, i.e. I$_{0,2}$ $\leq$ 10$\times$ I$_{0,1}$. Executing **@surbrightmake** plots the model together with the data into a file defined as **testplotfile**. You can control the choice of single or double β model by setting a parameter **profmodtest** to **singlebeta** or **doublebeta**. Once you find a rough agreement with the data, insert the parameter values as initial values into file **fit_surf_par.pro**.

The fitting routine varies the parameter values and computes a new surface brightness profile model for each new set of parameter values. The program compares each model with the data and computes the value of

$$\chi^2 = \Sigma\left(\frac{data(b) - model(b)}{\sigma(b)}\right)^2, \tag{3.3}$$

where data(b), model(b) and σ(b) are the values of the surface brightness data, model prediction and statistical uncertainties at each projected radius b. The goal is to find the parameter combination (= the best-fit parameters) which minimizes the χ^2.

A common problem with model fitting are the local χ^2 minima. This means that even if there is one truly global χ^2 minimum (a parameter combination that gives the smallest χ^2 considering the full parameter space), the fit may be stuck in a significantly different parameter combination that gives locally a best fit. The current implementation tries to avoid the local minimum problem using annealing method whereby the procedure is driven by "thermal" fluctuations, i.e. the actual χ^2 value of each fit is randomly changed by an exponentially distributed deviate. The amplitude (T) of the variation decreases gradually and at the final stage (T = 0) the fit reduces into a simple downhill method around the smallest local χ^2 minimum. The process can be controlled by **niterhot** and **niter0** parameters in the file **fit_surf_par.pro**. **niterhot** is the number of iterations with T $\neq$ 0. Bigger value makes the procedure spend more time in finding the global minimum. The choice of the value depends on the data quality. Preliminary tests showed that a value 100 is adequate for **niterhot**. **niter0** is the maximum number of iterations with T = 0. I recommend using 1000 for **niter0**. The internal criterion for the program determination is that the χ^2 changes by less than 10^{-8} between two consecutive iterations. If this is not met during 1000 T=0 iterations, there is probably something wrong with the fit.

The fit is run by editing first the file **fit_surf_par.pro**. Then start IDL and run **.r idl_imaging_setup**. Then, at IDL prompt, execute a script command **@surfbrightfit**. This command reads the necessary parameters from files **make_surf_par.pro** and **fit_surf_par.pro**, compiles necessary procedures and executes the program **fit_surf.pro**. Make sure that the parameters in the files **make_surf_par.pro** and **fit_surf_par.pro** are consistent since they both are used. When the execution determinates, it prints the best fit values on the screen and into file **fitfile** and plots the data and the best-fit model into file *plotfile* defined in **fit_surf_par.pro**.

3.2.1 Statistical uncertainties

In addition to the best-fit parameters, we need to estimate the statistical uncertainties of the model parameters. The starting point here is the above best-fit combination of parameter values. The routine starts to deviate gradually one parameter at a time from the best fit by the amount given by **dparinit_err** in a file **err_surf_par.pro**. The parameter in question is frozen to the above test value, and a new fit is performed. With the parameters offset from the best-fit, the fit naturally becomes worse. When the χ^2 increases by 1.0, then you have reached a 1 σ confidence limit for the given parameter. The procedure is repeated for each parameter in both upward and downward direction around the best-fit values and the obtained parameter confidence limits are printed on the screen and into a file defined as **errfile** in a file **err_surf_par.pro**. Note that the value of **dparinit_err** determines the resolution of the confidence limit. Smaller value gives a more accurate limit but takes more time. If the step values in **dparinit_err** are $\sim$10% of the best-fit values, the run may take $\sim$10 minutes while step value of $\sim$1% might take hours to run.

Keep **niterhot_err** and **niter0_err** at their default values of 0 and 1000. The parameter uncertainty computation is carried out by executing an IDL script **@surfbrighterr**. Make sure the parameters in the files **make_surf_par.pro, fit_surf_par.pro err_surf_par.pro** are all consistent since these files will be executed during the procedure.