For Galfit 3.0.4 and more recent versions
Preprint typeset using LaTeX style emulateapj v. 7/8/03

GALFIT USER'S MANUAL

Chien Y. Peng[1]

For Galfit 3.0.4 and more recent versions

ABSTRACT

Galfit is a tool for extracting information about galaxies, stars, globular cluster, stellar disks, etc., by using parametric functions to model objects as they appear in two-dimensional digital images. In simplest use, Galfit allows one to fit an ellipsoid model to light profiles in an image. For more complicated situations, it can model highly detailed shapes that are curved, irregular, lopsided, ringed, truncated, or have spiral arms. One can mix and match these features within a single component model, or can add them to other components to create complex shapes. This document describes how to run Galfit and explains its features, but it is not complete without two companion papers (Peng, Ho, Impey, & Rix 2010, AJ, 139, 2097; and Peng, Ho, Impey, & Rix 2002, AJ, 124, 266) which illustrate how the features can be used on real galaxies. There have been a number of upgrades since the original 2002 publication because Galfit continues to evolve. Thus this document supersedes both of the articles whenever there are differences.

Subject headings:

1. INTRODUCTION

A way to characterize the structure of objects in an image is to model their light distribution using analytic functions. For the functions to be useful, they generally have to have free parameters (e.g. size, luminosity) which one can adjust to model a wide variety of different shapes. Galfit[2] is an image analysis algorithm that can model profiles of galaxies, stars, and other astronomical objects in digital images. If successful, the features of interest are summarized into a small set of numbers, such as size, luminosity, and profile central concentration, which one can compare against other objects for doing science. One common application of this technique is to measure global morphology by using a single component, ellipsoidal model. Another use is to "take apart" a galaxy into different constituents like bulge, disk, bar, or to separate overlapping galaxies, by using two or more component models. This document details the features available for use in Galfit and how to use them.

Design Considerations Fitting functions to an image and interpreting what the results mean can be challenging even to skilled analysts because astronomical objects come in many sizes, shapes, and degrees of complexity. In addition, what one wishes to get out of the data depends on the science goal. Therefore there is often not one universal way to do an analysis that will satisfy everyone's needs. Deciding what to do and interpreting what the results mean require one to draw on scientific, technical, and often, artistic, skills and intuition.

The analysis being potentially complex, the design premise of Galfit tries to avoid adding another layer of complexity in its use. If the analysis is to be complex, it ought to be because an object is physically complicated, not because the code is complicated to use. Moreover, the ability to do complicated analysis should not make it burdensome for fitting a single component. These criteria mean that Galfit does only one thing, which is to fit functions. It does not help users do a lot of other things that are useful for analyzing large surveys, like helping users extract a point spread function, determining the initial parameters of the fit, figuring out the image size to fit, locating galaxies in an image, masking out neighboring objects, determining the sky pedestal level a priori, etc. — even though all these things are crucial for doing a correct analysis. Instead, the user must take care of *all* those pre- and post-processing (parameter value conversion, "book-keeping") outside of Galfit. Doing so greatly simplifies the use of Galfit. It gives users more control over the analysis. Because each of the aforementioned steps requires some degree of mastery, by not trying to do everything for users allows Galfit to be less a "black box" than it might seem otherwise. Even though Galfit does not perform critical pre-analysis for people, the project website does provide all the information necessary for users to learn the background necessary to perform proper analysis.

The above considerations mean the Galfit interface has the smallest essential set of controlling parameters needed to perform reliable photometry, located in a concise menu text file. There are no other hidden program knobs to fine-tune behavior. The only things which are not immediately visible to users are fundamental data that ought to be present in FITS image headers: exposure time, instrumental gain, readnoise, and number of images combined to produce the data. They are essential for doing quantitative science, therefore one should reasonably expect or provide them in all FITS images — just as importantly, that they correspond *correctly* [3] *to the data units*. To provide additional controls, an user may define a constraint file. The bare essential interface allows users to focus attention on the analysis itself, rather than on tweaking the program "knobs," in order to produce a reliable outcome.

To emphasize the above philosophy, this manual starts by getting new users quickly going on running Galfit by the next page. A new user should then immediately

[1] NRC Herzberg Institute of Astrophysics, 5071 West Saanich Road, Victoria, British Columbia, Canada V9E 2E7; Chien.Y.Peng@gmail.com

[2] http://users.obs.carnegiescience.edu/peng/work/galfit/galfit.html

[3] "correspond correctly" means that the exposure time must correspond to the image data units (e.g. an image in flux units has an exposure time of 1-second, not total exposure time). The gain factor directly converts the pixel values into electrons, and the readnoise is in units of electrons. See Section 3.

be able to try out complex applications, to hone intuition without being mired in trying to figure out what different program options do.

With the basic hand-shakes out of the way, the quickest way to learn about GALFIT is to read up to Section 2, run the first example provided in the *galfit/EXAMPLE* directory, and experiment with it. The remaining document will then explain how GALFIT determines the goodness of fit (§ 3), what functions are available to fit in GALFIT (§ 4, 5, 6), how one can run GALFIT using an interactive menu (§ 7, which one actually never needs to do), items in the GALFIT menu (§ 8, which is the most important section). Section 9 will explain how one can interact with GALFIT through a green window once it gets going (useful but not crucial). This will give the user some control over when to quit, which is sometimes nice. Once the fit is done, the algorithm will produce several output files (§ 10). Then, Section 11 tries to offer some hints on how to deal with difficult situations.

Lastly, for frequently-asked questions that are not answered by this document, please visit the GALFIT website, which is permanently maintained at (there is no space in the address):

http://users.obs.carnegiescience.edu/peng/work/galfit/galfit.html

2. QUICK START: RUNNING GALFIT FOR THE VERY FIRST TIME ON THE EXAMPLE

The quickest and easiest way to run GALFIT is to have a pre-formatted template file like the one shown in Figure 17 (also look for the *EXAMPLE.INPUT* file in the GALFIT source directory). Skip to § 8 to see what the menu parameters mean. In the *galfit/EXAMPLE* directory there is a simple example that one can try out immediately. After GALFIT has been compiled and aliased/linked, to run the example, go into that directory and simply type:

```
galfit galfit.feedme
```

In fact, *please do so now* before reading any further. If there is a problem at this stage, now is the time to request assistance. If the example runs properly, the residual image *imgblock.fits[3]* should look flat at the center, where there used to be a galaxy. One should display *imgblock.fits[1], imgblock.fits[2]*, and *imgblock.fits[3]* and examine their image headers to understand what information is produced. Note that *imgblock.fits[0]* is blank.

The example only fits one Sérsic model to a real galaxy. It is simple enough that just one will remove all significant residuals down to the noise level. In general, if the galaxy is more complex one can add more components to reduce the residuals. There is no limit as to the number of components or the mix of functional types one can use in a fit. For this particular example, however, adding more components is not useful because a single Sérsic already removes all the galaxy light: adding another model will cause one of the components to be highly suppressed, or be shifted out of the image completely. Either one of these things happening will then make GALFIT crash. Even though this is not a graceful way to exit, the operating philosophy is to provide *no* solution over a false solution when the problem is not well posed numerically.

After GALFIT finishes running it will produce three output files called "galfit.01", "fit.log", and "imgblock.fits". To understand these files, please see Section 10.

You now know how to run GALFIT. The rest of the document will explain the parameters in the input and output files. Please take a look at *EXAMPLE.INPUT* which contains examples of things that can be done in GALFIT, and visit the GALFIT website for answers to questions that may already have come up by now.

3. LEAST-SQUARES MINIMIZATION AND STATISTICS

Least-Squares Minimization. GALFIT is a least-squares fitting algorithm of the "non-linear" type, and uses a Levenberg-Marquardt algorithm to find the optimum solution to a fit. "Non-linear" simply means that the parameters being fitted are not only coefficients (amplitude) to functions, but can be involved in exponents of powerlaws, in fractions, etc.. Non-linear analysis requires iterating to find the best solution, whereas linear minimization involves a matrix inversion.

As with all least-squares algorithms, GALFIT determines the goodness of fit by calculating χ^2 and computing how to adjust the parameters for the next step. If the χ^2 decreases significantly, GALFIT will keep going. When the solution no longer improves by some criterion, it will stop on its own. In GALFIT the indicator of goodness of fit is the *normalized* or reduced χ^2, i.e. χ^2_ν:

$$\chi^2_\nu = \frac{1}{N_{\rm DOF}} \sum_{x=1}^{nx} \sum_{y=1}^{ny} \frac{\left(f_{\rm data}(x,y) - f_{\rm model}(x,y)\right)^2}{\sigma(x,y)^2}, \quad (1)$$

summed over all nx and ny pixels, where $N_{\rm DOF}$ is the degree of freedom (number of pixels - number of free parameters).

As shown in Equation 1, χ^2 minimization requires there to be two input images: the input data, $f_{\rm data}(x,y)$, and the σ image, $\sigma(x,y)$. The model image, $f_{\rm model}(x,y)$, is generated by GALFIT internally on the fly as it tries to find the best match to the data; the user specifies what models are to be used in the fit. $\sigma(x,y)$ is often called either as the "sigma" image or the weight image, i.e. one standard deviation of counts at each pixel (which is related to the Poisson "noise"). In times when the $\sigma(x,y)$ image is not available to the user, GALFIT has a way to automatically generate one internally based on Poisson statistics of the data. This is the only time when the image header information is used. For GALFIT to compute a sigma image reliably, the input data image has to be in the following specific form and there need to be some information in the image header:

What Header Keywords GALFIT *Wants and Why.* There are only 4 standard header keywords that GALFIT normally scans for in a FITS image without prompting: EXPTIME, GAIN (or ATODGAIN), RDNOISE, and NCOMBINE. If these keywords are not found for some reason, GALFIT assumes the default values of 1, 7, and 5.2, 1, respectively which, for historical reasons only, correspond to values for the WFPC2 camera on HST. As of Version 3.0.1 the RDNOISE parameter is not used.

Together with the photometric zeropoint in the input menu file given by the user, EXPTIME is used to calculate the magnitude or surface brightness of a model.

If, for some reason, an image has units of flux (counts $\sec^{-1}$), the EXPTIME parameter value should be 1 second, rather than the total or average exposure time. Side note: it should be noted that it is highly inadvisable to analyze images where the ADUs are in flux units rather than count units because the sky in flux units virtually always appears deceptively small and insignificant.

The GAIN and NCOMBINE parameters are only used by GALFIT to calculate the σ-image (the $\sigma(x, y)$ in Eq. 1) for pixel weighting if the user does not provide one. In the example you ran above, GALFIT generated a σ image internally. Further details will be discussed in Section 8.1 regarding σ images. However, here is the basic idea: GALFIT converts the image ADUs into electrons using the GAIN parameter such that, at each pixel: $ADU \times GAIN$ = *electrons per pixel*. Therefore, if the ADU has units of [counts], then the GAIN has to have a unit of [e^-ADU^{-1}]. This signal term (i.e. with background removed) is then added in quadrature with the background RMS to determine the Poisson noise at each pixel. Finally, this image with units of [electrons] is converted back into the same units as the data image, i.e. in [ADU].

There is one subtlety in this process related to the number of images used to create the data image, because the data image might have been created by averaging or summing several subexposures. If the subexposures were *averaged* into a final image, NCOMBINE equals the number of images used in the average, and the GAIN and RDNOISE need to be that for a single readout (GAIN_0, RDNOISE_0). The RDNOISE parameter should have a unit of [electrons]. On the other hand, if the subexposures were *summed*, then NCOMBINE should be set to 1, the GAIN value should be that for a single image (GAIN_0), and the RDNOISE should be an effective read noise (often $\sqrt{\text{N}_{\text{images}}} \times$ RDNOISE_0).

When creating a σ-image, the sigma at each pixel comes from both the source and from a uniform sky background, summed in quadruture. The sigma of the background is estimated directly from the RMS fluctuation in regions where there are no objects. Sky estimation is automatic and, though rather crude, is nevertheless often sufficient. The user can turn off sky estimation by telling GALFIT:

```
galfit -noskyest <filename>
```

When this happens the σ-image is obtained by scaling the image pixels, as is, directly into electrons based on Poisson statistics without adding a sky RMS term. This is a useful way to see how sensitive the results are due to the accuracy of the σ-image as well as the crude sky determination. The user can also tell GALFIT what sky and RMS to use by specifying:

```
galfit -skyped {value} -skyrms {value} <filename>
```

If the user wants to supply a sigma image, the user may override GALFIT (see § 8.1). GALFIT will then ignore the RDNOISE, GAIN, and NCOMBINE information. But, it is worth emphasizing that **doing so is advisable only for users who are clear about the notion of a sigma image**. Indeed, one of the "problems" most commonly reported could be avoided if users simply allowed GALFIT to create an internal sigma image instead of providing one without making sure it is appropriate (as explicitly defined by Equation 1). GALFIT is not very sensitive to the correctness of a sigma image, as long as it obeys the Poisson statistics, even roughly. Even if the normalization of the sigma image is off greatly, only the χ^2_ν calculation is affected but not the convergence on a solution. Most mistakes in the sigma image are pretty glaring: the sigma images have zero or negative values, have the wrong pedestal level, are not Poisson in scaling, or do not look anything like the data.

If you feel you ought to provide a σ-image, please first read Section 8.1 and refer to the GALFIT website on "Frequently Asked Questions and Advisory." Otherwise, it generally would not hurt to let GALFIT do so, unless the data ADUs have mysterious units that are not easily converted into electrons through the GAIN parameter. **So please make sure that the image header units are such that $ADU \times GAIN$ = *electrons*, and that the object does not dominate the field of view**. Please see Section 8.1 for a more detailed discussion on the σ-image.

The next three sections describe the analytic functions available in GALFIT for fitting light profiles. The functions are divided into three categories: the regular radial profile (§ 4), the azimuthal shape (§ 5), and the truncation function (§ 6).

4. THE RADIAL PROFILE FUNCTIONS

The radial profile functions control the radial fall-off in flux, e.g. the Sérsic, Nuker, exponential models, among others. GALFIT allows for some of the most frequently used functions in literature, and more will likely be added in the future.

The normalization parameter for the radial profiles can be specified by the user, by adding an integer to the end of the name of the functional name, e.g. *sersic2*, *gaussian1*, etc.. The default normalization, which is not indicated using a number, depends on the functional form and is discussed in the individual sections below. For instance, the default normalization for a Sérsic profile is total luminosity, whereas for a Ferrer's profile, it is central surface brightness. Aside from the default option, the numerical suffix has the following meaning for the normalization:

- *function* – default (see documentation below)
- *function1* – central surface brightness
- *function2* – surface brightness at radius parameter 4 (e.g. effective radius for Sérsic, FWHM for Gaussian, r_s for exponential, etc.).
- *function3* – for truncation only: surface brightness at the break radius, i.e. 99% flux radius.

The Sérsic Profile The Sérsic powerlaw is one of the most frequently used to study galaxy morphology, and has the following functional form:

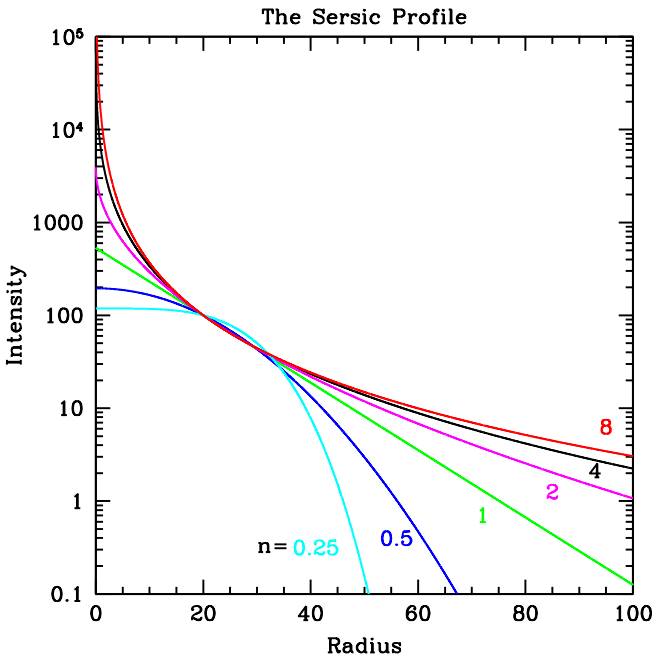


FIG. 1.— The Sérsic profile. Notice that the larger the Sérsic index value n, the steeper the central core, and more extended the outer wing. A low n has a flatter core and more sharply truncated wing. Large Sérsic index components are very sensitive to uncertainties in the sky background level determination because of the extended wings.

$$\Sigma(r) = \Sigma_e \exp\left[-\kappa\left(\left(\frac{r}{r_e}\right)^{1/n} - 1\right)\right]. \qquad (2)$$

Σ_e is the pixel surface brightness at the effective radius r_e. The parameter n is often referred to as the concentration parameter. When n is large, it has a steep inner profile, and a highly extended outer wing. Inversely, when n is small, it has a shallow inner profile and a steep truncation at large radius. The parameter r_e is known as the effective radius such that half of the total flux is within r_e. To make this definition true, the dependent-variable κ, is coupled to n, thus it is not a free parameter. The classic de Vaucouleurs profile that describes a number of galaxy bulges is a special case of the Sérsic profile when $n = 4$ (thus $\kappa = 7.67$). As explained below, both the exponential and Gaussian functions are also special cases of the Sérsic function when $n = 1$ and $n = 0.5$, respectively. As such the Sérsic profile is a common favorite when fitting a single component.

The flux integrated out to $r = \infty$ for a Sérsic profile is:

$$F_{\rm tot} = 2\pi r_e^2 \Sigma_e e^{\kappa} n \kappa^{-2n} \Gamma(2n) q / R(C_0; m), \qquad (3)$$

The term $R(C_0; m_i)$ is a geometric correction factor when the azimuthal shape deviates from a perfect ellipse. As the concept of azimuthal shapes will be discussed in Section 5, we will only comment here that $R(C_0; m_i)$ is simply the ratio of the *area* between a perfect ellipse with the area of the more general shape, having the same axis ratio q and unit radius. The shape can be modified by Fourier modes (m being the mode number) or diskyness/boxyness. For instance, when the shape is modified by diskyness/boxyness, $R(C_0)$ has an analytic solution given by:

$$R(C_0) = \frac{\pi(C_0 + 2)}{4\beta(1/(C_0+2), 1 + 1/(C_0+2))}, \qquad (4)$$

where β is the Beta function. In general, when the Fourier modes are used to modify an ellipsoid shape, there is no analytic solution for $R(m_i)$, and so the area ratio must be integrated numerically.

In GALFIT, the flux parameter that one can use for the Sérsic function is either the integrated magnitude m_{tot} (use "sersic" function), or the surface brightness magnitude μ_e (use "sersic2" function) at the effective radius, corresponding to Σ_e. The integrated magnitude is the standard definition:

$$m_{tot} = -2.5\log_{10}\left(\frac{F_{tot}}{t_{exp}}\right) + \text{mag zpt}, \qquad (5)$$

where t_{exp} is EXPTIME from the image header. Each Sérsic function can thus potentially have 7 classical free parameters in the fit: x_c, y_c, m_{tot}, r_e, n, q, PA. The non-classical parameters, C_0, Fourier modes, bending modes, and coordinate rotation may be added as needed. There is no restriction on the number of Fourier modes, and bending modes, but each Sérsic component can only have a single set of C_0 and coordinate rotation parameters.

The Exponential Disk Profile The exponential profile has some historical significance, so GALFIT is explicit about calling this profile an *exponential disk*, even though an object which has an exponential profile need not be a classical disk. Historically, an exponential disk has a scale length r_s, which is not to be confused with the effective radius r_e used in the Sérsic profile. For situations where one is not trying to fit a classical disk it would be less confusing nomenclature-wise to use the Sérsic function for $n = 1$, and quote the effective radius r_e. But because the exponential disk profile is a special case of the Sérsic function for when $n = 1$ (see Figure 1), there is a relationship between r_e and r_s, given by:

$$r_e = 1.678 r_s \qquad \text{(For } n = 1 \text{ only)}. \qquad (6)$$

The functional form of the exponential profile is:

$$\Sigma(r) = \Sigma_0 \exp\left(-\frac{r}{r_s}\right) \qquad (7)$$

$$F_{\rm tot} = 2\pi r_s^2 \Sigma_0 q / R(C_0, m), \qquad (8)$$

The 6 free parameters of the profile are: x, y, total magnitude, r_s, $\theta_{\rm PA}$ and q.

The Gaussian Profile The Gaussian profile is another special case of the Sérsic function for when $n = 0.5$ (see Figure 1), but here the size parameter is the FWHM instead of r_e. The functional form is:

$$\Sigma(r) = \Sigma_0 \exp\left(\frac{-r^2}{2\sigma^2}\right) \qquad (9)$$

$$F_{\rm tot} = 2\pi\sigma^2 \Sigma_0 q / R(C_0, m), \qquad (10)$$

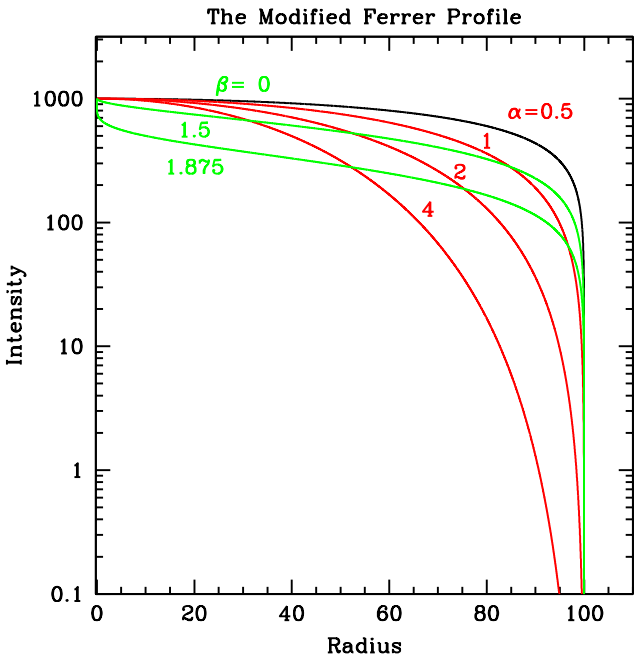


FIG. 2.— The modified Ferrer profile. The black, reference, curve has parameters $r_{\rm out} = 100$, $\alpha = 0.5$, $\beta = 2$, and $\Sigma_0 = 1000$. The red curves differ from the reference only in the α parameter as indicated by the red numbers. Likewise, the green curves differ from the reference only in the β parameter as indicated by the green numbers.

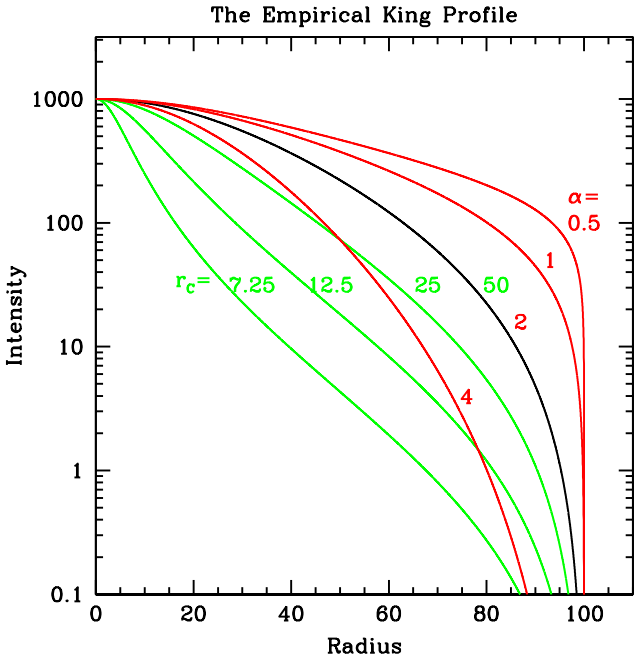


FIG. 3.— The empirical King profile. The black, reference, curve has parameters $r_c = 50$, $r_t = 100$, $\alpha = 2$, $\Sigma_0 = 1000$. The red curves differ from the reference curve only in the α parameter as indicated by the red numbers. Likewise, the green curves differ from the reference only in the r_c parameter as indicated by the green numbers.

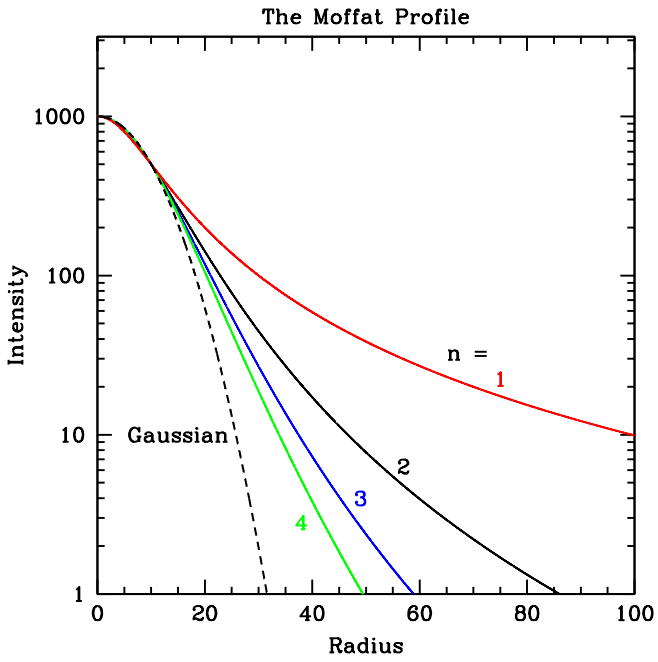


FIG. 4.— The Moffat profile. The black, reference, curve has parameters $n = 2$, FWHM $= 20$, and $\Sigma_0 = 1000$. The other colored lines differ only in the concentration index n as shown by the numbers. The dashed line shows a Gaussian profile of the same FWHM.

where FWHM $= 2.354\sigma$. The 6 free parameters of the profile are: x, y, the total magnitude, FWHM, q, and $\theta_{\rm PA}$.

The Modified Ferrer Profile The Ferrer profile (Figure 2) has a nearly flat core and an outer truncation. The sharpness of the truncation is governed by the parameter α, whereas the central slope is governed by parameter β. Because of the flat core and sharp truncation behavior, it is often used to fit galaxy bars and "lenses."

$$\Sigma(r) = \Sigma_0 \left(1 - (r/r_{\rm out})^{2-\beta}\right)^{\alpha} \qquad (11)$$

The profile is only defined within $r \leq r_{\rm out}$, beyond which the function has a value of 0. The 8 free parameters of the Ferrer profile are: x, y, central surface brightness, $r_{\rm out}$, α, β, q, and $\theta_{\rm PA}$.

The Empirical (Modified) King Profile The empirical king profile (Figure 3) is often used to fit the light profile of globular clusters. It has the following form (Elson 1999):

$$\Sigma(r) = \Sigma_0 \left[1 - \frac{1}{(1+(r_t/r_c)^2)^{1/\alpha}}\right]^{-\alpha} \times \left[\frac{1}{(1+(r/r_c)^2)^{1/\alpha}} - \frac{1}{(1+(r_t/r_c)^2)^{1/\alpha}}\right]^{\alpha}. \qquad (12)$$

The standard empirical King profile has a powerlaw $\alpha = 2$. In GALFIT, α can be a free parameter. In this model, the flux parameter to fit is the central surface brightness, expressed in mag/arcsec^2 form, i.e. μ_0 (see

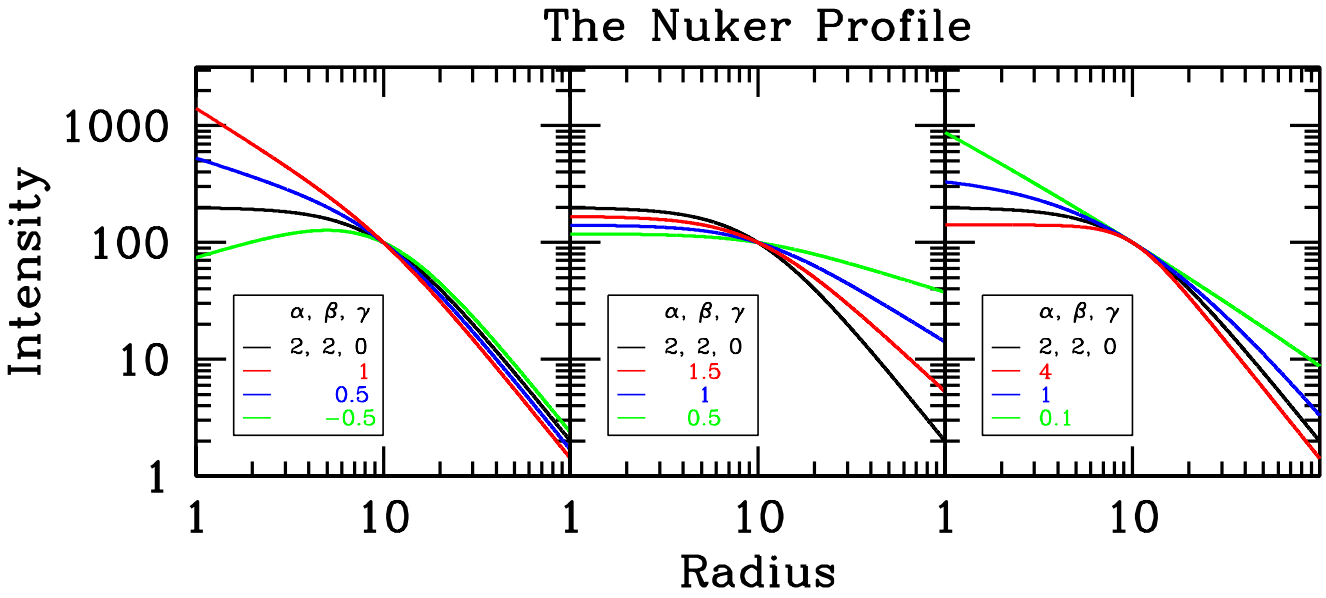


FIG. 5.— The Nuker profile. The black, reference, curve has parameters $r_{\rm b} = 10$, $\alpha = 2$, $\beta = 2$, $\gamma = 0$, and $I_b = 100$. For the other colored lines, only one value differs from the reference, as shown in the legend.

Eq.19). The other free parameters are the core radius (r_c) and the truncation radius (r_t), in addition to the geometrical parameters. Outside the truncation radius, the function is set to 0. Thus the total number of classical free parameters is 8: x, y, Σ_0, r_c, r_t, α, q, $\theta_{\rm PA}$.

The Moffat Profile The profile of the HST WFPC2 PSF is well described by the Moffat function. Other than that, the Moffat function is less frequently used than the above functions for galaxy fitting. The functional profile and the total flux equations are, respectively:

$$\Sigma(r) = \frac{\Sigma_0}{[1 + (r/r_d)^2]^n}, \tag{13}$$

$$F_{\rm tot} = \frac{\Sigma_0 \pi r_d^2 q}{(n-1)R(C_0, m)}, \tag{14}$$

In GALFIT the size parameter to fit is the FWHM, where the relation between r_d and FWHM is:

$$r_d = \frac{\rm FWHM}{2\sqrt{2^{1/n} - 1}} \tag{15}$$

The 7 free parameters are: x, y, $m_{\rm tot}$ (i.e. total magnitude, instead of μ_0) FWHM (instead of r_d), the concentration index n, q, and $\theta_{\rm PA}$.

The Nuker Profile The Nuker profile (Figure 5) was introduced by Lauer et al. (1995) to fit the nuclear profile of nearby galaxies, and it has the following form:

$$I(r) = I_b \, 2^{\frac{\beta-\gamma}{\alpha}} \left(\frac{r}{r_b}\right)^{-\gamma} \left[1 + \left(\frac{r}{r_b}\right)^{\alpha}\right]^{\frac{\gamma-\beta}{\alpha}} \tag{16}$$

The flux parameter to fit is μ_b, the surface brightness of the profile at r_b, which is defined as:

$$\mu_b = -2.5 \log_{10}\left(\frac{\rm I_b}{\rm t_{exp} \Delta x \Delta y}\right) + \text{mag zpt} \tag{17}$$

The Nuker profile is a double powerlaw, where (in Eq. 16) β is the outer power law slope, γ is the inner slope, and α controls the sharpness of the transition. The motivation for using this profile is that the nuclei of many galaxies appear to be fit well in 1-D (see Lauer et al. 1995) by a double powerlaw. However, use caution when interpreting this function, because, for example, a low α value ($\alpha \lesssim 2$) can be reproduced by simultaneously a high γ and a low β, (compare Figure 5*c* with the other two panels), which is a serious potential for degeneracy. In all there are a total of 9 free parameters: x, y, μ_b, r_b, α, β, γ, q, $\theta_{\rm PA}$.

The Edge-On Disk Profile If a flattened disk galaxy is viewed edge on, the projected surface brightness distribution takes on the following form:

$$\Sigma(r, h) = \Sigma_0 \left(\frac{r}{r_s}\right) K_1 \left(\frac{r}{r_s}\right) \text{sech}^2 \left(\frac{h}{h_s}\right), \tag{18}$$

where Σ_0 is the surface brightness profile, r_s is the major-axis disk scale-length, and h_s is the perpendicular disk scale-height, and K_1 is a Bessel function. The flux parameter being fitted in GALFIT is the central surface brightness:

$$\mu_0 = -2.5 \log_{10}\left(\frac{\Sigma_0}{\rm t_{exp} \Delta x \Delta y}\right) + \text{mag zpt}, \tag{19}$$

where $t_{\rm exp}$ is the exposure time from the image header, and Δx and Δy are the platescale in arcsec, which the user supplies (Item K in the GALFIT input file).

Note that if the disk is oriented horizontally the coordinate r is the x-distance (as opposed to the radius) of a

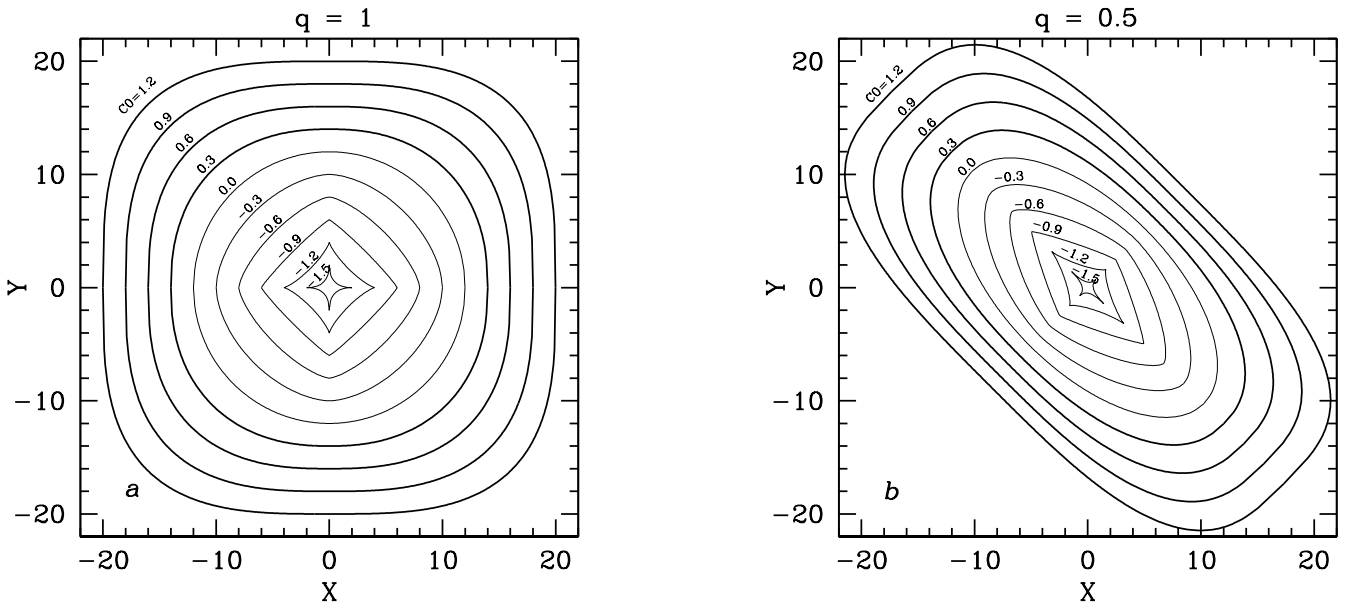


FIG. 6.— Generalized ellipses with *a*) axis ratio $q = 1$ and *b*) axis ratio $q = 0.5$.

pixel from the origin. There are 6 free parameters in the profile model: x_0, y_0, Σ_0, r_s, h_s, and $\theta_{\rm PA}$.

The PSF Profile For unresolved sources, one can fit pure stellar PSFs to an image (as opposed to narrow functions convolved with the PSF). The PSF function is simply the convolution PSF image that the user provides (in Item D of the GALFIT menu), hence there is no prescribed analytical functional form. As the point spread function, this is the only profile that is not convolved. The PSF has only 3 free parameters: x_c, y_c and total magnitude. Because there is no analytical form, the total magnitude is determined by integrating over the PSF image and assuming that it contains 100% of the light. If the PSF wing is vignetted, there will be a systematic offset between the flux GALFIT reports and the actual value.

If one wants to fit this "function", make sure the input PSF is close to, or super-, Nyquist sampled. The PSF shifting is done by a Sinc+Kaiser interpolation kernel, which can preserve the widths of the PSF even under sub-pixel shifting. This is in principle much better then Spline interpolation or other high order interpolants. However, if the PSF is under-sampled, aliasing will occur, and the PSF interpolation will be poor. If the PSF is undersampled, it is better to provide an oversampled PSF if possible, even if the data is undersampled. With HST data this can be done using TinyTim (Krist & Hook 1997) or by combining stars. GALFIT will take care of rebinning during the fitting.

Note that the alternative to fitting a PSF is to fit a Gaussian with a small width, i.e. 0.4-0.5 pixels, which GALFIT will convolve with the PSF. This is generally not advisable if a source is a pure point source because convolving a narrow function with the PSF will broaden out the overall profile, even if slight. The convergence can also be poor if the FWHM parameter starts becoming smaller than 0.5 pixel. However, this technique can still be useful to see if a source is truly resolved.

The Background Sky. The background sky is a flat plane that can tilt in x and y. Thus it has a total of 3 free parameters. The pivot point for the sky is *fixed* to the geometric center (x_0, y_0) of the image, calculated by $(n_{pix} + 1)/2$, where n_{pix} is the number of pixels along one dimension. The tip and tilt are calculated relative to that center. Because the galaxy centroid located at (x, y) is in general not at the geometric center (x_0, y_0) of the image, the sky value directly beneath the galaxy centroid is calculated by:

$$\mathrm{sky}(x,y) = \mathrm{sky}(x_0,y_0) + (x-x_0)\frac{d\mathrm{sky}}{dx} + (y-y_0)\frac{d\mathrm{sky}}{dy} \quad (20)$$

5. THE AZIMUTHAL PROFILE FUNCTIONS

The previous section illustrates the kind of radial profiles that are available to fit galaxies. However, to generate the shapes of a galaxy requires the use of azimuthal functions, which control what a model component looks like in the sky projection, e.g. elliptical, disky/boxy, irregular, curvy, or spiral. Each profile component can be modified by any one or all of the following azimuthal functions. For example, an ellipse can be modified by bending modes, Fourier modes, diskyness/boxyness, and a spiral rotation function – the combination of which would result in a rather odd looking spiral structure.

Generalized Ellipses The simplest azimuthal shape in GALFIT is the traditional generalized ellipse. This is the starting point for all GALFIT analysis: no matter how complex the final outcome is, one should always begin by fitting an ellipsoid, on top of which complications are introduced.

The radial coordinate of the generalized ellipse is defined by:

$$r(x,y) = \left(|x - x_c|^{C_0+2} + \left| \frac{y - y_c}{q} \right|^{C_0+2} \right)^{\frac{1}{C_0+2}} . \quad (21)$$

Here, the ellipse axes are aligned with the coordinate axes, and (x_c, y_c) is the centroid of the ellipse. The ellipse is called "general" in the sense that C_0 is a free parameter, which controls the diskyness/boxyness of the isophote. When $C_0 = 0$ the isophotes are pure ellipses. With decreasing C_0 ($C_0 < 0$), the shape becomes more disky (diamond-like), and conversely, more boxy (rectangular) as C_0 increases ($C_0 > 0$) (see Figure 6). Note that C_0 will not appear in the menu unless the user explicitly asks for it (see *EXAMPLE.INPUT*). The major axis of the ellipse can also be oriented to a position angle (PA – not shown). Thus, there are a total of 4 free parameters $(x_0, y_0, q, \theta_{\rm PA})$ in the standard ellipse, and one: C_0, for diskyness/boxyness parameter. C_0 should not be used until a best fitting ellipsoid model has been found.

Fourier Modes Few galaxies look like perfect ellipsoids, and one can better refine the azimuthal shape by adding perturbations in the form of Fourier modes. The Fourier perturbation on a perfect ellipsoid shape is defined in the following way:

$$r(x,y) = r_0(x,y) \left(1 + \sum_{m=1}^{N} a_m \cos\left(m(\theta + \phi_m)\right) \right). \quad (22)$$

In the absence of Fourier modes in the parenthesis, the $r_0(x,y)$ term is the radial coordinate for a traditional ellipse, and $\theta = \arctan\left((y - y_c)/((x - x_c)q)\right)$ defined in Equation 21. a_m is the Fourier amplitude for the mode m. Defined as such, a_m is the fractional deviation in radius from a generalized ellipse of Eq. 21. The number of modes N is up to the user to decide, and the user may also choose to skip certain modes. See Figure 7 for some examples of how Fourier modes modify a circle and an ellipse into other shapes.

The "phase angle", ϕ_m, is the angle of a mode m, relative to the PA of the generalized ellipse. That is to say, the phase angle is 0 degrees in the direction of the semi-major axis of the generalized ellipse (rather than up), increasing counter-clockwise. Notice that the phase angle is complete within a range of: $-180^o/m < \phi_m \leq 180^o/m$, which is referred to as the *cardinal angle*. The phase angle in GALFIT is always reported to users in the cardinal range.

Unlike the classical shape parameters q and PA, the Fourier modes will only appear in the menu when the user wants to fit them (see *EXAMPLE.INPUT* for how to do so). Otherwise they will remain out of view to avoid clutter. Each mode has 2 free parameters, a_m and ϕ_m, and the number of modes the user can add is unrestricted. However, the most useful modes are low order ones ($m = 1, 3...6$). Also, $m = 2$ should rarely be used if one is already fitting the traditional axis ratio q for an ellipse, since q and $m = 2$ are fairly degenerate with each other. Initially, the parameters of the Fourier modes can be all set to 0 even if the final values greatly deviate from 0.

Properly Interpreting Fourier Mode Parameters A common (but incorrect) first impression is to regard the Fourier modes as a "shapelet decomposition" technique. In shapelet decomposition of an image, an object is broken down into some number of *basis functions*, i.e. fixed 2-D geometric patterns that are mathematically orthogonal functions. The amplitude of a shapelet basis function is the *flux* of that component. By combining the right shape and number of basis functions, anything in the image can be modeled. However, GALFIT *does not do shapelet decomposition.* In GALFIT, the Fourier modes modify the *coordinate system* from a rectilinear grid into something more exotic. By stretching/shrinking it in the radial direction, by an amount that depends on the azimuthal angle, the result is that the *azimuthal shape* of a component is modified, but the *radial profile is not.* Another way to think about it is that, in GALFIT, the concept of using Fourier modes to modify an ellipse is fundamentally the same as using the axis ratio parameter (q) to modify a *circle* into an ellipse – the latter is what all 2-D image fitting algorithms do. Indeed, the axis ratio parameter q is a special case of the Fourier modes. In both instances, only the *shape* of the model changes, whereas the structural parameters (size, concentration index, magnitude) retain their original meaning.

It is worth emphasizing that coordinate stretching by Fourier mode is a self-similar remapping. This means that the form and meaning of the *radial profile functions* being modified do not change by this remapping, i.e. our prior intuitions about the meaning of the parameters (e.g. Sérsic index n, size, etc.) in traditional 2-D fitting still apply. For instance, each model component still has a single peak, and radially the profile falls off according to one of the fitted functions such as Sérsic, exponential, Nuker, etc., *in every direction from the peak*; the peak does not have to be the geometric center if the component is lopsided. The decline is monotonic, so it is still meaningful to talk about, e.g., an *"average" light profile (e.g. Sérsic)*, with, say, *an average Sérsic concentration index n* – no matter what the galaxy may look like azimuthally. In this manner irregular galaxies can be parameterized, because even they have an *average light profile*. For instance, when the average peak of an irregular galaxy is not located at the geometric center, it has a high amplitude $m = 1$ Fourier mode (i.e. lopsidedness). Other high order modes can be used to quantify higher degrees of asymmetry on top of lopsidedness.

The phase angles of the Fourier modes are also useful information to keep in mind. Modes with the following phase angles have the following symmetry properties:

- Symmetry about a central point: $a_1 = 0$, regardless of other mode phase and amplitude.
- For all modes m, there is reflection symmetry at: $\phi_m = 0°, \pm\frac{180°}{m}$. For m =even, this symmetry is about both the major and minor axes. Whereas, for m =odd, the reflection symmetry is only about the major axis.
- For *odd* modes of m, there is additional reflection symmetry about the minor axis at: $\phi_m = \pm\frac{90°}{m}$.

An irregular galaxy has angles that are "out of phase" whereas regular galaxies have angles that are more "in

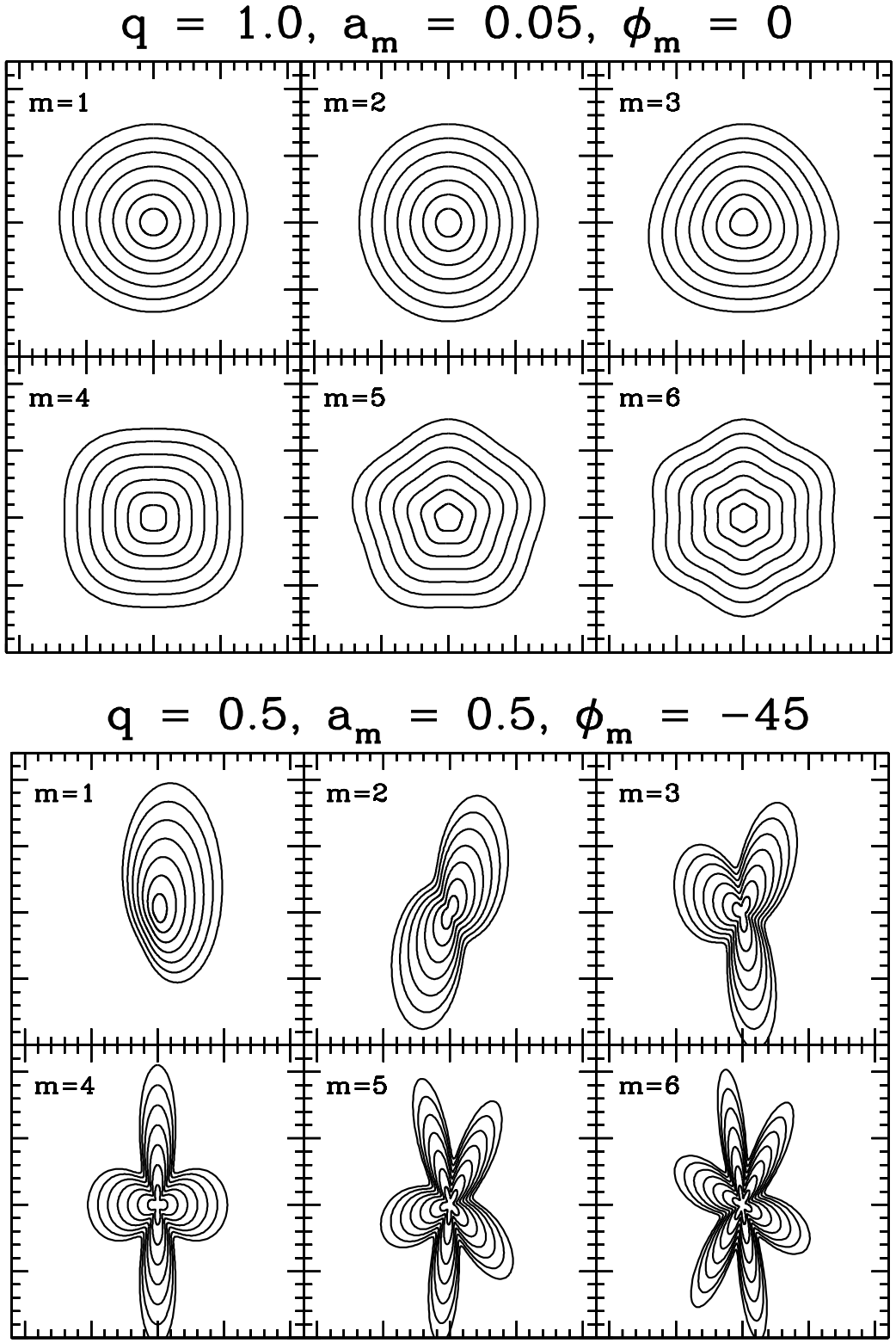


FIG. 7.— Examples of Fourier modes. Top: Low amplitude ($a_m = 0.05$) Fourier modes modifying a circular profile (q=1.0) with phase angle $\phi = 0$ degree. Bottom: High amplitude ($a_m = 0.5$) Fourier modes modifying an elliptical profile (q=0.5) with phase angle $\phi = -45$ degrees.

phase" (i.e. reflectionally symmetric around either minor or major axis). Therefore, it is possible to quantify various forms and degree of asymmetry by constructing indices based on the amplitude and phase angles of the Fourier modes. The most intuitively obvious asymmetry index is the $m = 1$ mode, which captures the lopsidedness (A_L) of a galaxy, i.e. the positioning of the brightest central region relative to the fainter outer region of a galaxy:

$$A_L = |a_1| \,. \tag{23}$$

Asymmetric galaxies are also characterized by overall deviation from an ellipse; thus, another intuitively useful quantity to measure is the sum of the Fourier amplitudes:

$$A_E = \sum_m^N |a_m| \,. \tag{24}$$

Asymmetric galaxies by definition have high A_E. However, it is possible for galaxies with both high A_E and A_L to be reflectionally symmetric; the degree of reflectional symmetry may be an indicator for how well the galaxies is relaxed. Reflection *asymmetry* is given by the index A_R:

$$\begin{aligned} A_R = & \textstyle\sum_{m=\text{even}} |a_m| \ \sin^2\left(\pi m \frac{\phi_m}{180^\circ}\right) + \\ & \textstyle\sum_{m=\text{odd}} |a_m| \ \sin^2\left(\pi m \frac{\phi_m}{90^\circ}\right), \end{aligned} \tag{25}$$

where ϕ_m is in degrees. In this formulation, the higher the reflectional asymmetry, the higher the index A_R. Used together, these three descriptors provide highly useful ways to quantify the degree galaxies are irregular. For instance, high values of A_R and A_L most likely imply high global asymmetry in the intuitive sense. Whereas a high value of A_E with low A_R implies high regularity, but large deviation from an ellipse, such as edge-on disky galaxies or a disky/boxy ellipticals.

Caveat in using Fourier modes. Because Fourier modes are *high order corrections* to some intrinsic shape, care should be taken when fitting them. For instance, Fourier modes can be high order corrections to a perfect circle ($q = 1$), or an ellipse ($q < 1$). In most instances, it is the perturbation on an ellipsoid shape that is the most interesting. Therefore, with the possible exception of the first mode, *high order Fourier modes should be used only after finding an optimal solution with the classical ellipsoid model.* Otherwise, strong residuals can always be locally Taylor expanded into Fourier modes which would have high amplitudes. Likewise, high order modes are quite sensitive to the presence of neighboring objects, such as stars, overlapping galaxies. They ought to be removed either by masking or simultaneous fitting.

Bending Modes Bending modes allows for curvature in the model when the amount of curvature is less than half a circle. The coordinate transformation $(x, y) \Longrightarrow (x', y')$ is obtained by only perturbing the y-axis (in a rotated frame) in the following way:

$$y' = y + \sum_{m=1}^{N} a_m \left(\frac{x}{r_{\rm scale}}\right)^m, \tag{26}$$

where $x' = x$, $r_{\rm scale}$ is the scale radius of a model (i.e. $r_{\rm eff}$ for Sérsic, r_s for exponential, etc.). Some examples of this perturbation are shown in Figure 8. Note that $m = 1$ resembles quite closely to the axis ratio parameter, q. However, the $m = 1$ bending mode is actually a shear term, the effect of which is most easily seen when it operates on a purely boxy profile $C_0 \sim 2$ (Figure 6*a*), shearing it into a more disky shape (see Figure 8*d*). The bending modes can be modified by Fourier modes or diskyness/boxyness to change the higher order shape of the overall model. This kind of coordinate transformation again preserves the original meaning of the radial profiles. Here, the object size parameter refers to the unstretched size, i.e. projected onto the original (x, y) Cartesian frame, as opposed to a length along the curvature.

Coordinate Rotation I: Powerlaw - Hyperbolic Tangent (α-tanh) Sometimes the isophotes of a galaxy may rotate as a function of radius, as in the case of spiral galaxies. To model the light profile, it is now also possible to allow for coordinate rotation in GALFIT. GALFIT allows for two types of coordinate rotation functions, the powerlaw spiral (α-tanh), and the logarithmic spiral (log-tanh). Both of these are coupled to the hyperbolic tangent in order to potentially generate a bar that extends into the center. The exact functional form of the rotation function is lengthy (see Appendix A), but the schematic functional dependence of the powerlaw spiral on the parameters is given by the following:

$$\theta(r) = \theta_{\rm out} \ \tanh\left(r_{\rm in}, r_{\rm out}, \theta_{\rm incl}, \theta_{\rm PA}^{\rm sky}; r\right) \times \left[\frac{1}{2}\left(\frac{r}{r_{\rm out}} + 1\right)\right]^{\alpha}. \tag{27}$$

Figure 9 shows a hyperbolic tangent rotation function for several different bar parameters (left), and a combination of bar parameters and the asymptotic powerlaw slope α (right), where r is the radial coordinate system, $\theta_{\rm out}$ is the rotation angle roughly at $r_{\rm out}$. The bar radius, $r_{\rm in}$, is defined to be the radius where the rotation reaches roughly 20 degrees. This angle corresponds fairly closely to our intuitive notion of bar length based on examining the images, and is not a rigorous, physical, definition. The inclination $\theta_{\rm incl}$ is the line-of-sight inclination of the disk, where $\theta_{\rm incl} = 0$ deg is face on and $\theta_{\rm incl} = 90$ deg is perfectly edge-on. As shown in Figure 10, a face-on model does not necessarily mean that the outer-most isophotes are round. Rather, the ellipticity of the outermost isophotes is related to the asymptotic behavior of the rotation function, which asymptotes to a constant PA beyond a radius of $r_{\rm out}$ for a pure hyperbolic tangent ($\alpha = 0$, Figure 9*a*). The isophotes only appear circular in the main body of the spiral structure when it has a large number of windings. Figure 11 shows several examples of bar and non-barred galaxies, with different α values, sky inclination angle, and rotated to different sky position angles ($\theta_{\rm PA}^{sky}$). When the powerlaw index α is negative, the spiral pattern can reverse course after reaching a maximum value (see Figure 11). In summary, the hyperbolic tangent powerlaw function has 6 free parameters, $\theta_{\rm out}, r_{\rm in}, r_{\rm out}, \alpha, \theta_{\rm incl}, \theta_{\rm PA}^{\rm sky}$. The thickness of the spiral structure is controlled by the axis ratio q of the ellipsoid being modified by the hyperbolic tangent, or by

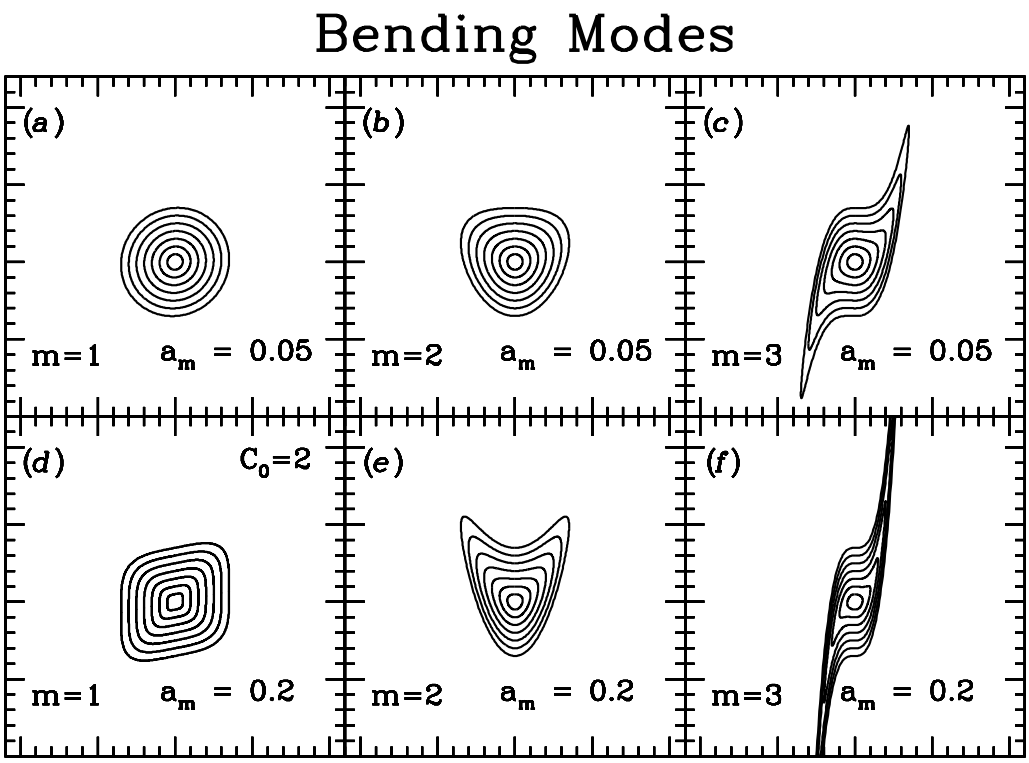


FIG. 8.— Examples of bending modes modifying a circular profile (q=1.0), and $C_0 = 0$ (unless indicated otherwise). Top row: Low amplitude ($a_m = 0.05 r_{\rm scale}^m$) bending modes. Bottom: High amplitude ($a_m = 0.2 r_{\rm scale}^m$) bending modes. Bending modes can be combined with Fourier modes to change the higher order shape.

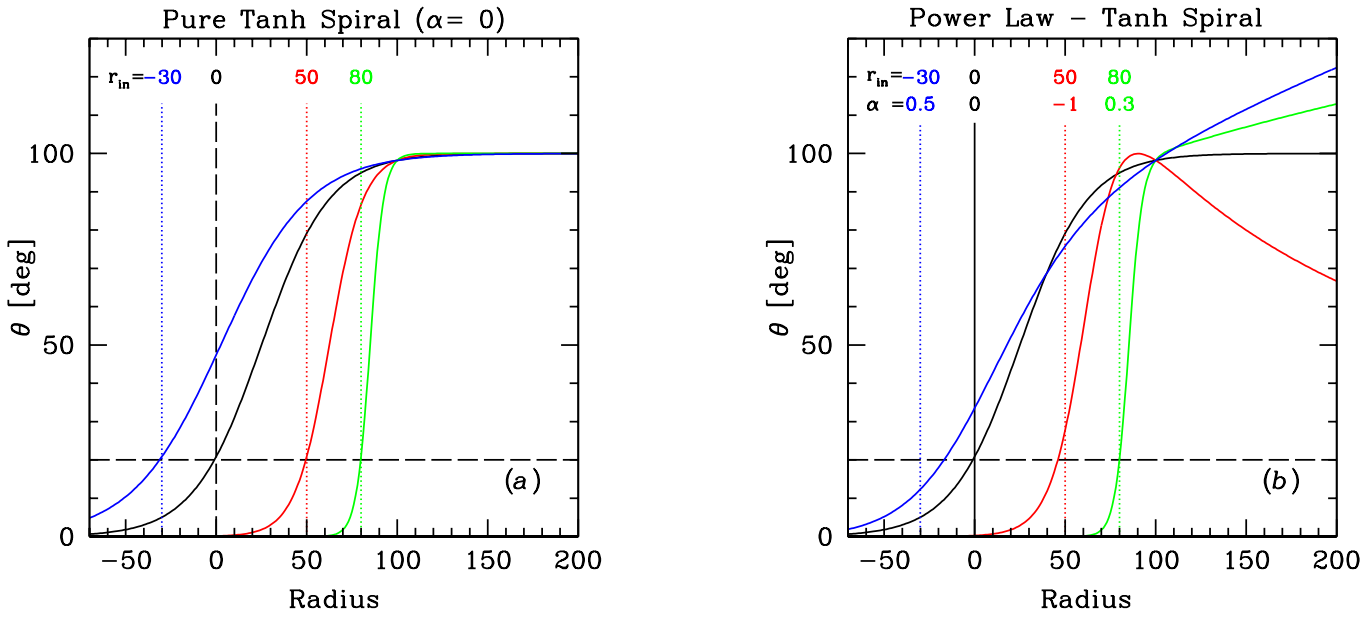


FIG. 9.— Hyperbolic tangent-powerlaw spiral angular rotation functions with outer spiral radius of $r_{\rm out} = 100$ pix. *a*) Examples of a pure hyperbolic tangent spiral ($\alpha = 0$) with different "bar" radii (r_{in}). *b*) Examples with different bar radii and asymptotic powerlaw α as indicated.

the Fourier modes that modify the ellipsoid. To create highly intricate and asymmetric spiral structures, Fourier modes can be used in conjunction with coordinate rotation.

The "bar" radius ($r_{\rm in}$) is a mathematical construct. Note that the "bar" term in the coordinate rotation should be regarded only as a *mathematical construct* to grant the rotation function as much flexibility as possible. This construct *can* reflect reality, but it does not have to, and it often does not. For instance, mathematically, a *negative* $r_{\rm in}$ radius (Figure 9*b*) is perfectly

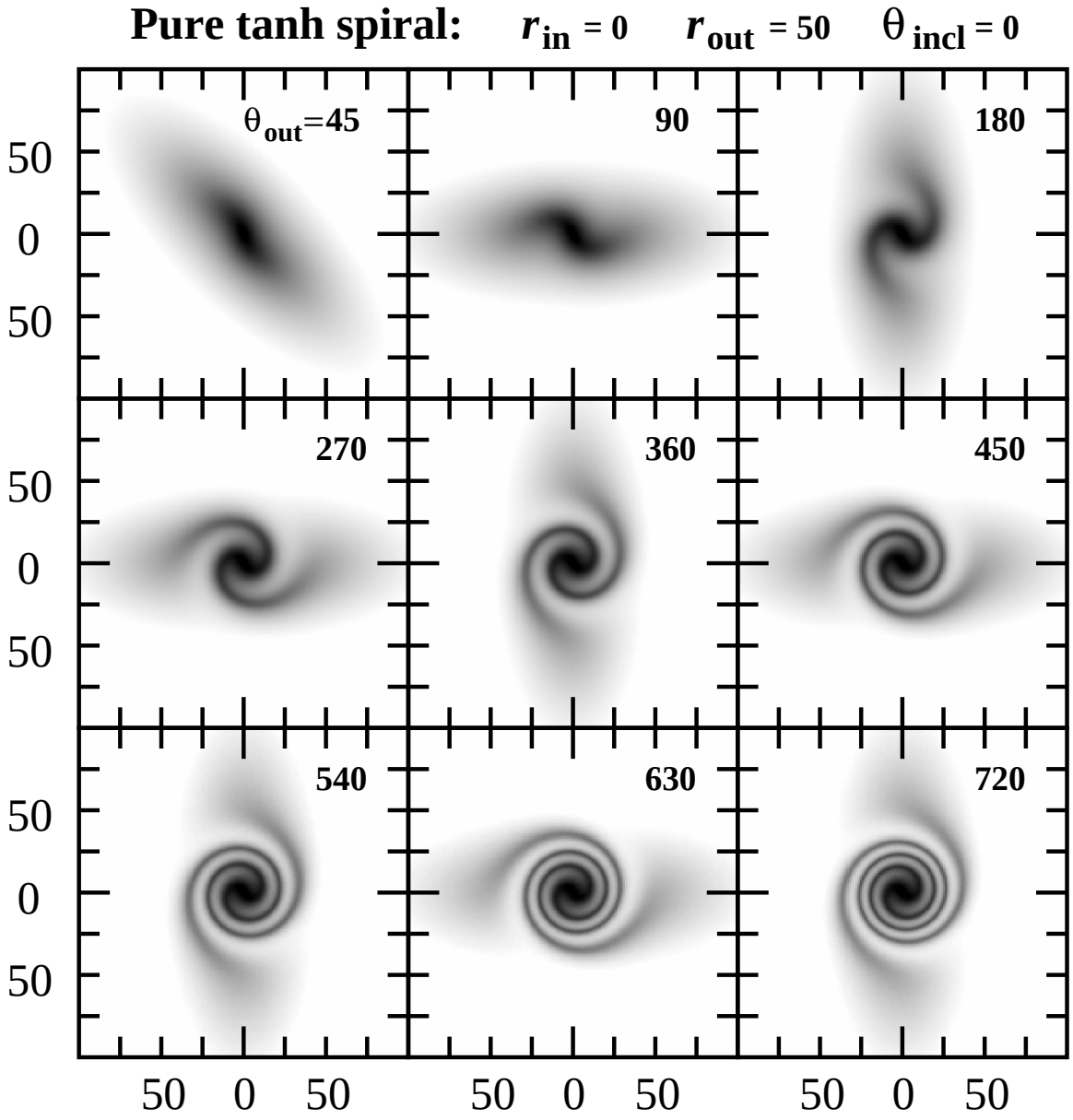


FIG. 10.— Examples of pure (i.e. with powerlaw $\alpha = 0$ or without logarithmic function) hyperbolic tangent coordinate rotation modifying a elliptical profile with axis ratio $q = 0.4$. Note that all the figure panels share the same parameters as shown up top, external to the figures. The spiral model has no bar. The numbers within each panel shows the amount of total winding (units in degrees) at the spiral rotation radius of 50 pixels. Notice that outside $r = 50$ pixels, the rotation angle becomes constant, due to the rotation function being a hyperbolic tangent, thereby creating the appearance of a flattened disk, even though there is not a separate disk component involved in the model.

sensible because of the way Eqs. 27 and 28 (for logarithmic spiral, below) are defined: a negative $r_{\rm in}$ just means that the spiral rotation function has a finite rotation angle at $r = 0$ relative to the initial ellipsoid out of which it is constructed. When there is clearly no bar, the $r_{\rm in}$ parameter can become quite negative; in this case, one should probably just hold $r_{\rm in}$ fixed to 0. Furthermore, often times, one may not wish to create a bar and a spiral out of one smoothly continuous function for various reasons, e.g. they may have different widths, the spiral may not extend into the center, or the spiral may start off in a ring. In these situations, one can "detach" the bar from the spiral by using a truncation function (see § 6 and § 8.3.5), by instead creating a bar with a separate Sérsic, Ferrer, or other functions. When this is done, a "bar radius" is still useful mathematically in the coordinate rotation function, but it may bear no physical relation to the physical bar.

Limitations of the spiral rotation formulation. While the *α-tanh* rotation function works surprisingly well for many spiral galaxies, there are several limitations to the simple formulation. One limitation is that the spiral rotation functions are smooth, so "kinks" in the spiral structure cannot yet be modeled, even though it is possible to do so by allowing for "kinks" in the rotation function. Lastly, the spiral structure cannot wind back onto itself, because that would require the rotation function to be multi-valued. While it is possible

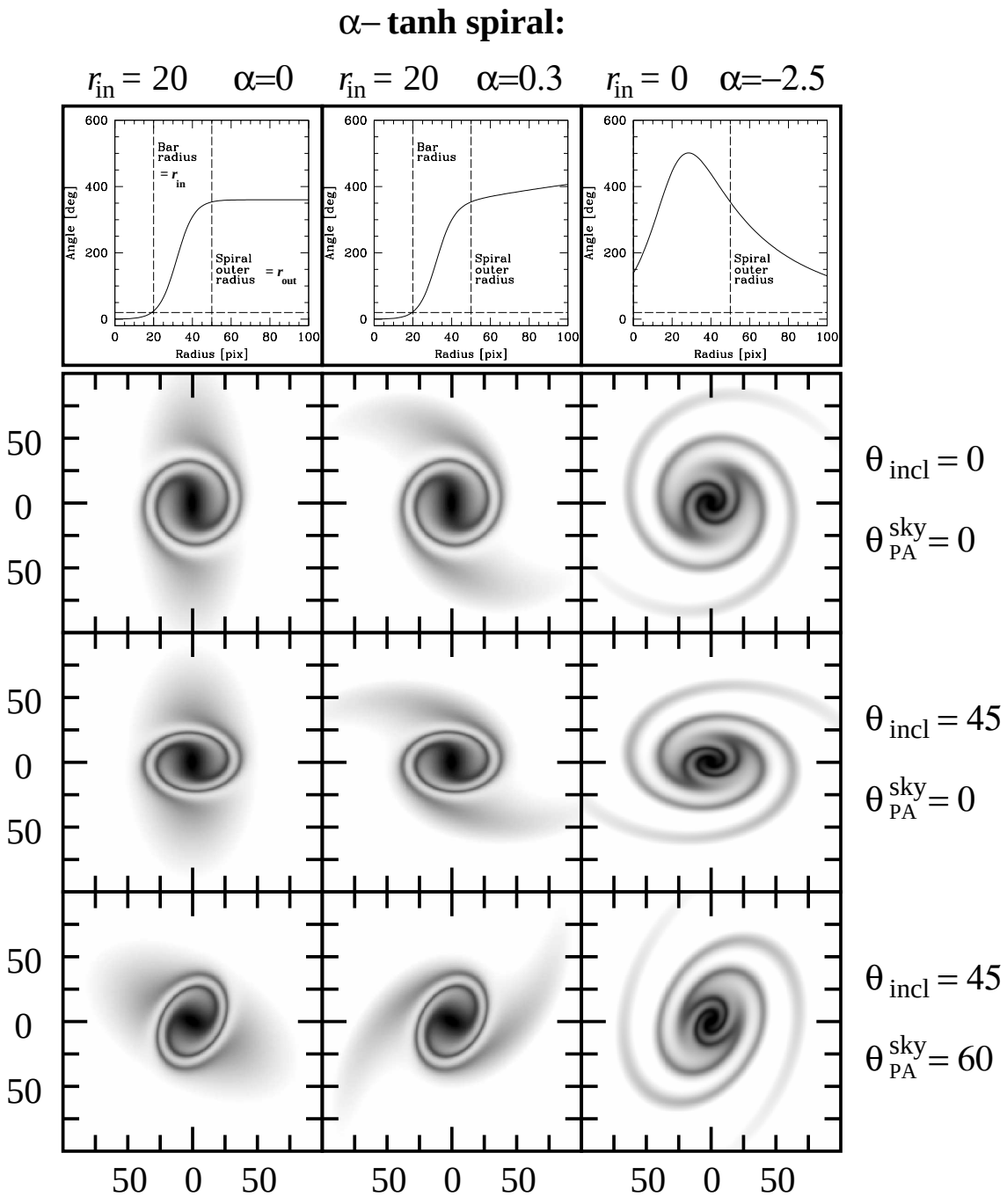


FIG. 11.— Examples of powerlaw - hyperbolic tangent (α-tanh) coordinate rotation modifying a face on (tilt = 0 deg) elliptical profile with axis ratio $q = 0.4$. The parameters of the rotation functions are shown on the top and right hand side of the diagram. The top panels show the spiral rotation angle as a function of radius for the panels in the same column.

to do so, it is not yet implemented.

Coordinate Rotation II: Logarithmic - Hyperbolic Tangent (log-tanh) The winding rate of spiral arms in late-type galaxies is often thought to be logarithmic with radius rather than powerlaw. Thus, GALFIT also allows for a logarithmic-hyperbolic tangent coordinate rotation function, which is defined as:

$$\theta(r) = \theta_{\rm out}\ \tanh\ \left(r_{\rm in}, r_{\rm out}, \theta_{\rm incl}, \theta_{\rm PA}^{\rm sky}; r\right) \times \left[\log\left(\frac{r}{r_{\rm ws}}+1\right)/\log\left(\frac{r_{\rm out}}{r_{\rm ws}}+1\right)\right]. \quad (28)$$

Like the α-tanh rotation function, the log-tanh function has a hyperbolic tangent part that regulates the bar-length and the speed of rotation within $r_{\rm out}$. Beyond $r_{\rm out}$ the asymptotic rotation rate is that of the logarithm function, which has a winding scale radius

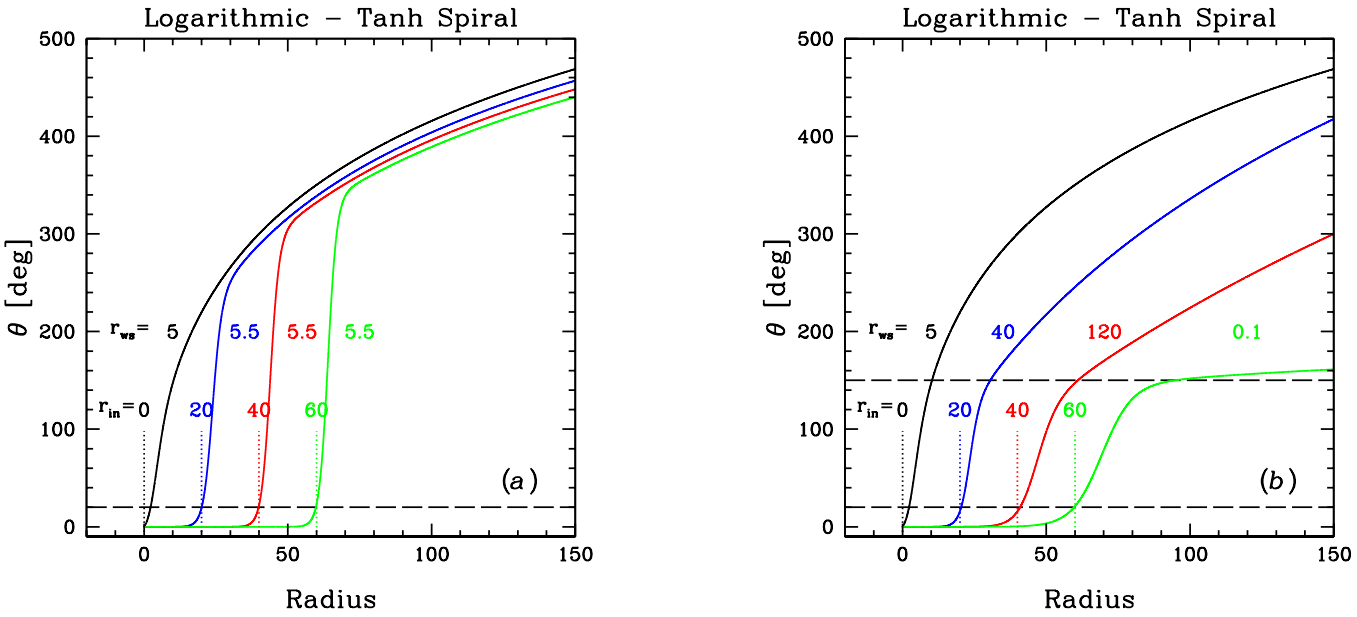


FIG. 12.— Logarithmic - hyperbolic tangent spiral angular rotation functions. *a*) Examples of different bar radius, and where the outer hyperbolic spiral radius is $r_{\rm out} = r_{\rm in} + 10$ pix. The lower horizontal dashed line shows the rotation angle at the "bar" radius ($r_{\rm in}$). *b*) Examples with different bar radii and winding scale radii $r_{\rm ws}$ as indicated, illustrating the degree of flexibility of the spiral rotation rate. The rotation angle at $r_{\rm out}$ is fixed to 150 degrees, as shown by the upper horizontal dashed line. The left-most, black, curve is close to being a pure logarithmic function, recasted so that at $r = 0$, the rotation angle $\theta = 0$.

of $r_{\rm ws}$; the larger the winding scale radius, the *tighter* the winding. Thus, like the α-tanh spiral, the log-tanh spiral rotation function also has 6 free parameters: $\theta_{\rm out}, r_{\rm in}, r_{\rm out}, r_{\rm ws}, \theta_{\rm incl}, \theta_{\rm PA}^{\rm sky}$. Note that in terms of capabilities, the α-tanh function can often reproduce the log-tanh function and more. Therefore, the α-tanh is probably a more useful rotation function in practice.

Note that GALFIT does not allow for a pure logarithmic spiral because such a function has a negative-infinity rotation angle at $r = 0$. Therefore, in GALFIT, at $r = 0$ the rotation function reaches $\theta = 0$ (Figure 12). Lastly, it is also important to keep in mind that the meaning of the "bar radius," just as described in the section for α-tanh rotation function, is a mathematical construct.

6. THE TRUNCATION FUNCTION

Truncation functions allow for a possibility to create rings, outer profile cut-offs, dustlanes, or a composite profile in the sense that the inner region behaves as one function and the outer behaves as another. The truncation function can modify both the radial profile and azimuthal shape. A ring can be created by truncating the inner region of a light profile. Likewise, when a galaxy has spiral arms that do not reach the center, it can be viewed as being truncated in the inner region.

6.1. *General Principle*

In GALFIT each truncation function can modify one or more light profile models. Also, any number of light profile can share *the same* truncation function. The truncation function in GALFIT is a hyperbolic tangent function (see Eq. 7 in Appendix B). Schematically, a truncated component is created by multiplying a radial light profile function, $f_{0,i}(x, y; ...)$, by one or more truncation functions, P_m or $1 - P_n$ (depending on whether the type is an inner or an outer truncation – see Eq. 7 of Appendix B) in the following way:

$$f_i(x, y; ...) = f_{0,i}(x, y; x_{c,i}, y_{c,i} ... \; q_i, \theta_{{\rm PA},i}) \times \prod^{m} P_m \left(x, y; x_{c,m}, y_{c,m}, r_{{\rm break},m}, \Delta r_{{\rm soft},m}, q_m, \theta_{{\rm PA},m}\right) \times \prod^{n} \left[1 - P_n \left(x, y; x_{c,n}, y_{c,n}, r_{{\rm break},n}, \Delta r_{{\rm soft},n}, q_n, \theta_{{\rm PA},n}\right)\right], \quad (29)$$

where, $r_{\rm break}$ is the break radius where the profile is 99% of the original, i.e. un-truncated, model flux at that radius. The parameter $\Delta r_{\rm soft}$ is the softening length, so that $r = r_{\rm break} \pm \Delta r_{\rm soft}$ is where the flux drops to 1% that of an un-truncated model at the same radius (the $\pm$ sign depends on whether the truncation is inner or outer). The inner truncation function (P_m) tapers a light profile in the inner regions of a light profile, whereas the outer truncation function ($1 - P_n$) tapers a light profile in the wings.

The behavior of the hyperbolic tangent function is ideal for truncation because it asymptotes to 1 at the break radius $r \gtrsim r_{\rm break}$ and 0 at the softening radius $r < r_{\rm soft}$, and vice versa for the complement function. Thus, when multiplied to a light profile $f(r)$, the functional behavior exterior of the break radii region have intuitively obvious meanings. For example, as shown in Figure 14*a*, if a Sérsic function with $n = 4$ is truncated in the wings (shown in red), the core has exactly an $n = 4$ profile interior to the $r_{\rm break}$ radius (marked in vertical dashed line), which is a free parameter to fit. Likewise, an $n = 4$ profile truncated in the core (green) has exactly an $n = 4$ profile exterior to the outer break radius. Thus, when one sums two functions of different Sérsic indices n, Figure 14*b*, the asymptotic profiles of the wing and core retain their original meaning, and there is very little

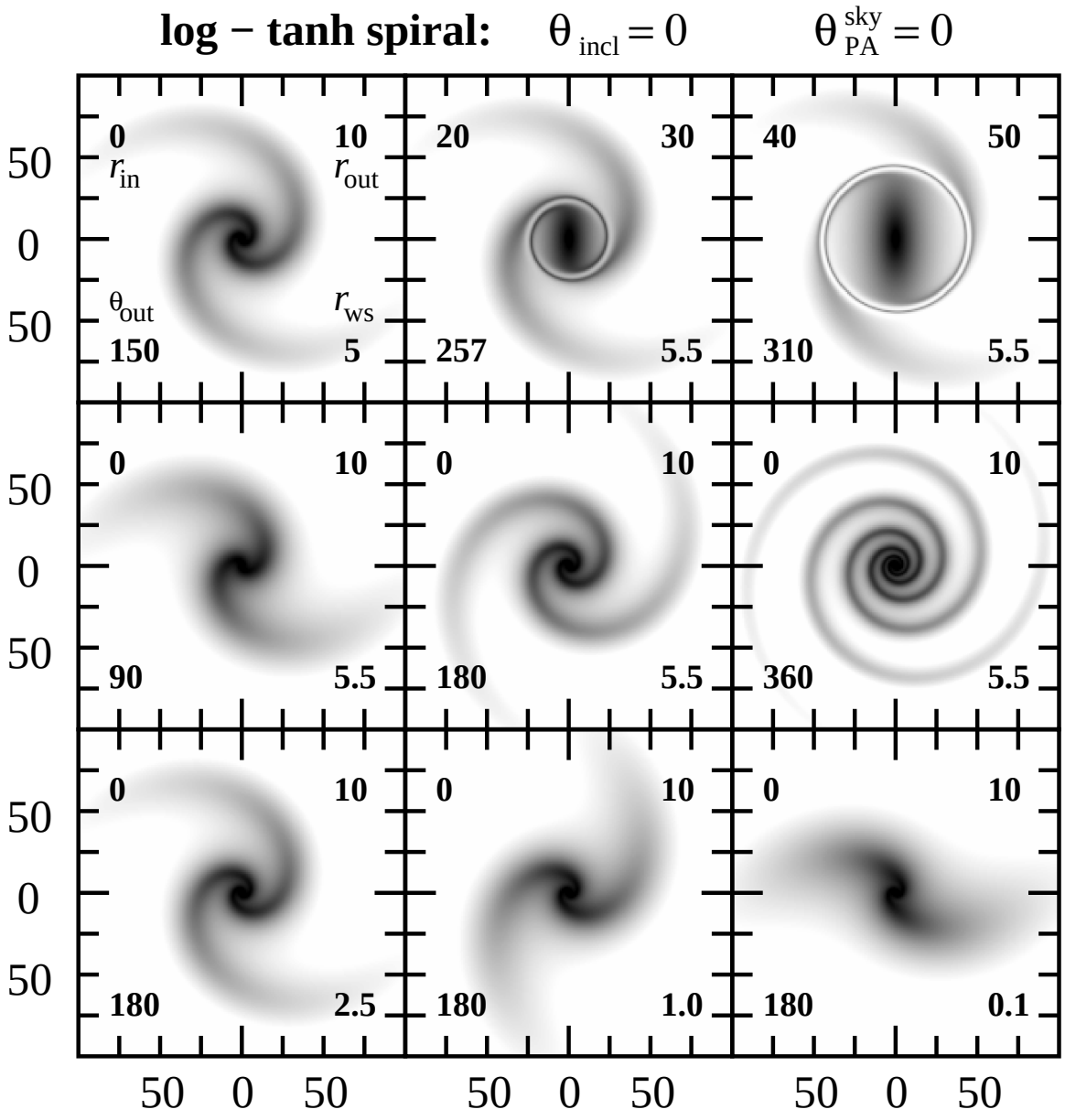


FIG. 13.— Logarithmic - hyperbolic tangent spiral (log-tanh) angular rotation examples, all face-on ($\theta_{\rm incl} = 0$), and $\theta_{\rm PA}^{\rm sky} = 0$ deg. The top left panel shows the meaning of the rotation parameter values at the corners of each box. Like with the α-tanh spirals, the log-tanh spiral can be tilted and rotated to any sky projection angle, or combined with Fourier modes to produce lopsided or multi-armed spiral structures (not shown), and with truncation function to produce an inner ring or an outer taper. The top left panel figure, for all practical purposes, is a pure logarithmic spiral with a winding scale radius $R_{\rm ws} = 5$ pixels.

crosstalk outside of the truncation region (denoted by vertical dashed lines in Figure 14).

The use of the truncation functions is highly flexible. There can be an unrestricted number of inner and outer truncation functions for each light profile model. Furthermore, multiple light profile models can share *the same* truncation functions. This is useful, for instance, when trying to fit a dustlane (inner truncation), in a fairly edge on galaxy that may affect both the bulge and the disk components. Just as with light profile models, the truncation functions can be modified by Fourier modes, bending modes, etc., independent of the higher order modes for the light profile they are modifying.

6.2. *Different Variations of the Truncation Function*

Truncation models appear in many physical contexts, e.g. dustlane, rings, spirals that do not reach the center, joining a spiral with a bar, cut-off of the outer disk, etc.. To allow the truncation function to be intuitively to use as the situations require, GALFIT allows for several variations. In addition to inner and outer truncations, truncation functions can share in the same parameters as the parent light profile. There are radial and length/height truncations, softening radius vs. softening length (default *vs.* Type 2), inclined vs. non-inclined (default *vs.* Type b) truncations, and lastly, 4 different ways to normalize the flux – the most intuitive choice depends on how a profile is truncated. We now discuss each of these

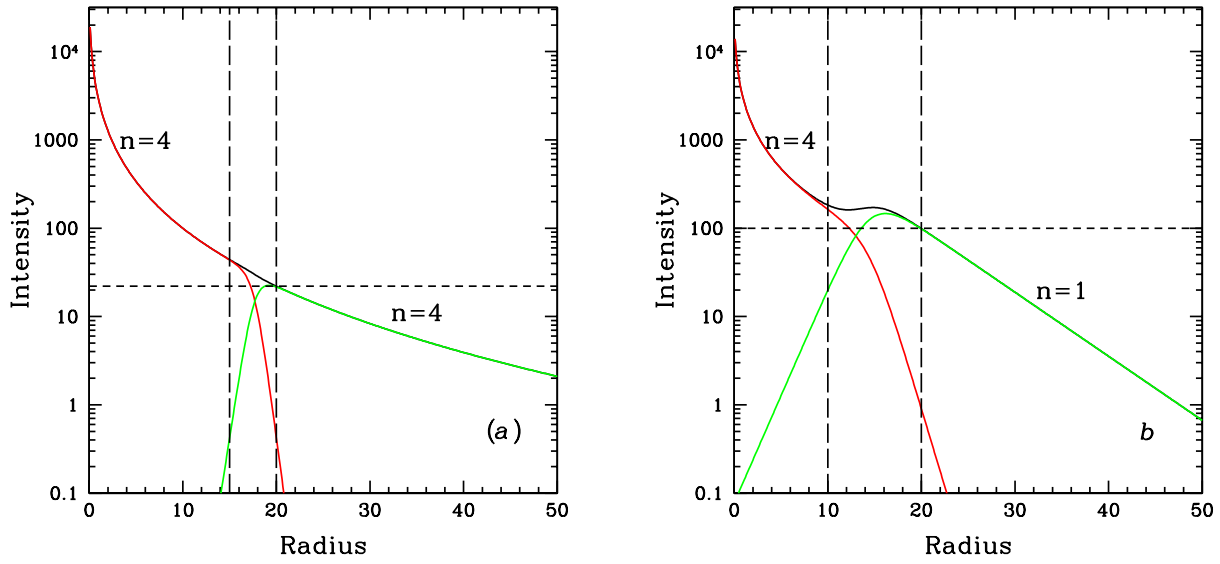


FIG. 14.— Examples of hyperbolic truncation function on $n = 4$ and $n = 1$ Sérsic profiles. *a*) a continuous $n = 4$ model represented as two truncated models of otherwise identical r_e, n, and central surface brightness, with truncation radii at $r = 15$ and $r = 20$ as marked by the vertical dashed lines. The black curve is the sum of the inner and outer functions. This shows that, outside the truncation region, there is very little "crosstalk" between the inner and outer components. *b*) a composite profile made up of an $n = 4$ nucleus truncated in the wings, and an $n = 1$ truncated in the core with truncation radii $r = 10$ and $r = 20$. Note that the hump in the summed model would give rise to a ring in a 2-D model.

variations in more detail, and Section 8.3.5 will discuss how these options are specified in the GALFIT input file (see also EXAMPLE.INPUT).

Parameter sharing. In the most general form, each truncation function has its own set of free parameters: $x, y, r_{\rm break}, \Delta r_{\rm soft}, q, \theta_{\rm PA}$. However, by default, the x, y, q, and $\theta_{\rm PA}$ are tied to the light profile model, and are activated only when the user explicitly specifies a value for them.

Radial ("radial") vs. Length ("length") or Height ("height") Truncations The most useful type of truncation is one which has radial symmetry to first order, i.e. where it has a center, an ellipticity, an axis ratio. However, in the case of a perfectly edge-on disk galaxy ("edgedisk" model), an additional type is allowed that truncates linearly in length or in height. For instance, a dustlane running through the length of the galaxy has an inner height truncation. For the "edgedisk" profile, GALFIT also allows for a radial truncation like with all other functions. The one drawback to height and length truncations is that they cannot be modified by Fourier and higher order modes like the radial truncations.

Softening length ("radial") vs. softening radius ("radial2") Sometimes, instead of softening *length* ($\Delta r_{\rm soft}$), it is more useful for the fit parameter to be a softening *radius* ($r_{\rm soft}$), especially when one desires to hold the parameter fixed. That is also allowed in GALFIT as a Type 2 truncation function, designated, e.g. as "radial2". The default option does not have a numerical suffix.

Inclined (default, "radial") vs. Non-inclined ("radial-b") Truncations A spiral rotation function is an infinitesimally thin, planar, structure. Nevertheless, it should be thought of as a 3-D structure in the sense that the plane of the spiral can be rotated through 3 Euler angles, not just in position angle on the sky. When a truncation function is modifying a spiral model, it is therefore sometimes useful to think about the truncation in the plane of the spiral model. When Fourier modes and radial truncations are modifying a spiral structure, the default ("radial") is for the modification to take place in the plane of the spiral structure. However, there are some instances when that may not be ideal for some reason (e.g. a face-on spiral may actually be ellipsoidal). In those situations, one can choose "radial-b", which would allow a truncation function to modify the spiral structure in the plane of the sky, even though the spiral structure can tip and tilt as needed.

And, yes, the truncation function can be Type 2b (i.e. "radial2-b") as well.

Flux normalization. The most intuitive flux normalization for a truncated is the total luminosity. Unfortunately, both the total luminosity, and the derivative of the free parameters with respect to the total luminosity are especially time-consuming to work out computationally, and there are generally no closed form analytic solutions to the problem. Therefore, the alternative is to allow for different ways to normalize a component flux. The user may choose whichever one is more intuitive, given the situation and the science task at hand. The default way depends on the truncation type:

- Inner truncation: the flux is normalized at the break radius. For a ring model, this represents the outer radius of the ring, near the peak flux.
- Outer truncation: flux normalized at the center.

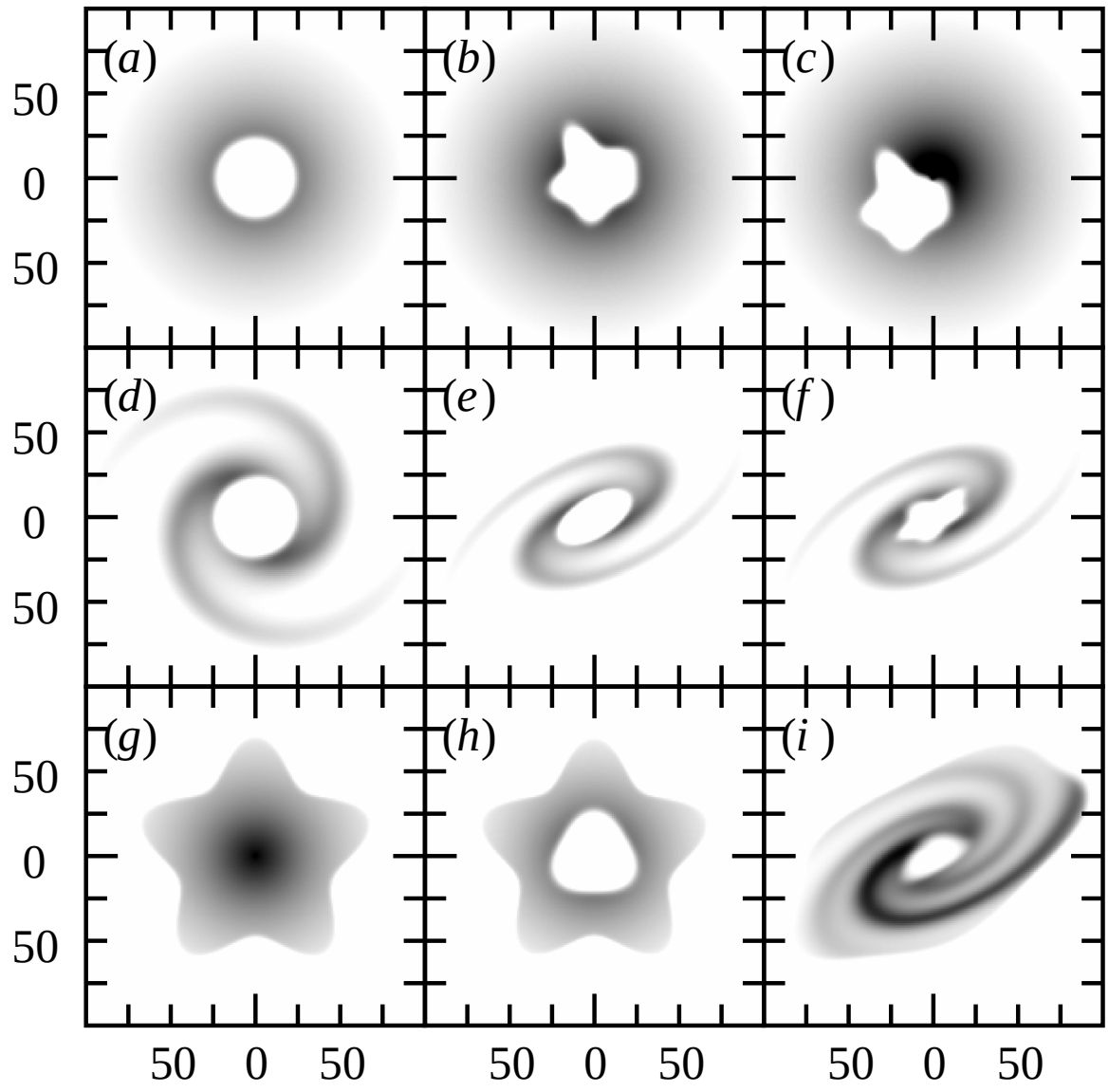


FIG. 15.— Examples of truncation functions acting on a *single component* light profile of various shapes. *a*) Inner truncation of a round profile, creating a ring. *b*) The truncation function can be modified by Fourier modes, just like the light profile. *c*) The truncation function can be offset in position relative to the light profile. *d*) The truncation function can act on a spiral model. *e*) The truncation can tilt in the same way as the spiral. *f*) The truncation function can be modified by Fourier modes while acting on a spiral model. *g*) A round light profile is being truncated in the wing by a pentagonal (Fourier mode 5) truncation function. *h*) A round light profile is being truncated in the inner region by a triangular function (Fourier mode 3), and in the wing by a pentagonal function. *i*) A 3-arm, lopsided, spiral light profile model is truncated in the wing by a pentagonal function, and in the inner region by a triangular function.

- Both inner and outer truncation: same as the case for Inner truncation.

However, there are many situations when the default is not desirable. Instead, the user can choose the radius where the flux is normalized, using the same scheme discussed in the introduction to Section 4. To be pedagogical, we explicitly show here the normalization for just the Sérsic function, and allow an user to infer the analogous relation for other functions:

- *function* : default (e.g. "sersic", "nuker", "king", etc.). See above.
- *function1*: flux normalized at the center $r = 0$ (i.e. Σ_0). A function which is given originally by $f_{\rm orig}(r)$ is now defined as: $f_{\rm mod}(r) = \Sigma_0 \frac{f_{\rm orig}(r)}{f_{\rm orig}(0)}$. For the Sérsic profile (i.e. called by "sersic1"), the profile function is redefined in the following way, written explicitly:

$$f_{\rm mod}(r) = \Sigma_0 \frac{\exp\left[-\kappa\left(\left(\frac{r}{r_e}\right)^{1/n} - 1\right)\right]}{\exp\left[\kappa\right]}. \qquad (30)$$

For the Ferrer, Gaussian, King, Moffat, this corresponds to functions as written in Section 4.

- *function2*: flux parameter is the surface brightness at a model's native size parameter (parameter 4 of the light profile model). For a Sérsic profile, called by "sersic2", this means the effective radius r_e. So, $f_{\rm mod}(r) = \Sigma_{\rm eff} \frac{f_{\rm orig}(r)}{f_{\rm orig}(r_{\rm eff})}$. For example, a Sérsic profile now has the following explicit form:

$$f_{\rm mod}(r) = \Sigma_{\rm eff} \exp\left[-\kappa\left(\left(\frac{r}{r_e}\right)^{1/n} - 1\right)\right]. \quad (31)$$

  For the Sérsic, exponential, and Nuker, profiles, this corresponds to functions as written in Section 4.

- *function3*: flux parameter is the surface brightness ($\Sigma_{\rm break}$) at the break radius $r_{\rm break}$. This is the most useful situation when a truncation results in a large scale galaxy ring, so that the surface brightness parameter corresponds closely to the peak of the light profile model. When the truncation is not concentric with the light profile model, this kind of normalization is not very intuitive. For "radial" truncation, $r_{\rm break}$ is parameter 4, whereas for "radial2", $r_{\rm break}$ is parameter 4 for outer truncation, and parameter 5 for inner truncation. When "sersic3" option is chosen, the $r_{\rm break}$ parameter comes automatically from the *first* truncation component that a certain light profile model is associated.

  In our example of the Sérsic profile, $f_{\rm mod}(r) = \Sigma_{\rm break} \frac{f_{\rm orig}(r)}{f_{\rm orig}(r_{\rm break})}$. For example, a Sérsic profile now has the following explicit form:

$$f_{\rm mod}(r) = \Sigma_{\rm break} \frac{\exp\left[-\kappa\left(\left(\frac{r}{r_e}\right)^{1/n} - 1\right)\right]}{\exp\left[-\kappa\left(\left(\frac{r_{\rm break}}{r_e}\right)^{1/n} - 1\right)\right]}. \quad (32)$$

Figure 15 demonstrates just some of the possibilities allowed when fitting truncations. In addition to the regular ellipsoid shape, the higher order modes like diskyness/boxyness parameters, bending modes, and Fourier modes can also modify the shape of the truncation functions. One can also use the truncation function *on* a spiral model, on models with Fourier and bending modes, and diskyness/boxyness models, some of which are shown in Figure 15*d, e , f, & i*. When a truncation function acts on a spiral component, it can do so either in the plane of the disk ("Type *a*") or in the plane of the sky ("Type *b*", e.g. "radial-b"). While the default is in the plane of the disk, the parameters are more intuitive in Type *b* cases when the disk is tilted and rotated.

6.3. *Caveats about using the truncation function.*

The use of truncation functions should be carefully supervised because unexpected things can happen, such as the size or the concentration index of a component can grow without bound. This behavior is due to the fact that there are degeneracies between the sharpness of truncation and the steepness/size of the galaxy. Therefore, truncation functions should only be used on objects that clearly have truncations.

When two functions are joined by using a truncation function, the cross-talk region is located in between the two truncation radii: it is worth bearing in mind the definition that at the break and softening radii, the fluxes are 99% and 1% that of the same model without truncation, respectively. In other words, the larger the truncation length, the larger the cross-talk region. Therefore, when one (or more) of the parameters $r_{\rm break}$, $r_{\rm break} + \Delta r_{\rm soft}$, or $r_{\rm soft}$ is either too small ($\lesssim$ few pixels) or larger than the image size, it probably indicates that profile truncation parameters are not meaningful. Rather, it more likely reveals the fact that there is a mismatch between the light profile model and the actual galaxy profile.

7. RUNNING GALFIT

There are two ways to run GALFIT, by either providing a template file on the command line or through an interaction menu. Providing a template file to GALFIT on the command line is the easiest way. In the future, an interaction menu may not be available. Once GALFIT starts going, it does not care when and how you quit. To quit abruptly, hit *control-c* at anytime at your pleasure, but note that the results will not be saved. In this section, I will describe several ways to run (and quit).

7.1. GALFIT *Optional Flags*

When starting GALFIT on the command line, several flags are available, and the list of flags are summarized using the command:

```
>  galfit -help
```

The options are:

```
-noskyest   (See Section 8.1, Item C)
-skyped n
-skyrms n
-outsig

-o1         (See Section 8.1, Item P)
-o2
-o3
```

7.2. *The Template File*

GALFIT is completely menu driven, and the menu can either come via a text file template (easy! – this is the way to go), or it can be filled in manually (tedious! – *not* a good way to go). The menu items will be more fully described in § 8.

Figure 17 below shows an example of the GALFIT input (template) file. When you start GALFIT without a template file, you will see a similar screen except the entries are blank. You can enter everything interactively on the command line of GALFIT – this can be quite tedious if you have a long object list, but it is certainly possible (§ 7.4 will show you how). After the menu Item P, the length of the list is flexible, and depends on how many components you want to fit. You can add or remove the number of components as you deem necessary.

In the input file, things after hash marks "#" are comments, and are always ignored by the program. Blank lines are also ignored, and the column alignment you see is optional (purely aesthetic). There is pretty much no error checking to catch problems in the input file, so one

should stick to the following format pretty closely. For example, in the input file, do not modify "3) xxx" (note the spaces), to look like "3)xxx" (bad spacing). GALFIT does not care how the columns are aligned (vertical alignment is only for aesthetics), as long as they are in the proper sequence. The order of the rows is also arbitrary for the image parameters and within each object block (because each row has a unique ID except for the objects). In fact, the image parameters A-P can appear anywhere in any row. Everything else about the format should be pretty intuitive. If there are errors in the input file, GALFIT may not complain about them and may simply crash.

7.3. *The Easiest Way to Run* GALFIT *: Reading in a Template File*

7.3.1. *From the Unix/Linux Prompt*

To facilitate fitting with minimal interaction, once you have a pre-formatted text file shown in Figure 17, the typical GALFIT session is just a single step. On the command line, type:

```
> galfit input_file_name
```

The example shown below (Figure 17 fits a galaxy with a PSF, a Sérsic, exponential disk, and a Nuker model, while holding the sky level constant at 2 counts (ADU).

7.3.2. *From Within* GALFIT *Prompt*

The second way to run GALFIT is by typing at the UNIX/LINUX command prompt:

```
> galfit
```

If you don't have a prepared input file, ignore the first question by hitting return. This brings you to the GALFIT prompt where you can edit object and image parameters manually. The GALFIT prompt looks like:

```
Enter item, initial value(s), fit switch(es) ==>
```

Below, I will not bother to re-show the prompt; all commands issued are assumed to come after that generic prompt.

At the GALFIT prompt, you can read in one or more template files. To do so, simply type:

```
t file_name
```

or

```
t
```

followed by a return. Be careful about reading in two or more input files: while additional models will be *added* to the pre-existing file, the parameters A-P, i.e. input data image, output file name, noise file, etc., will take on values of the newest input file. (Note: most of the time you won't use this option anyway, but you can.)

7.4. *OPTIONAL Interactive Command Line Options While in* GALFIT *– a.k.a. "Running* GALFIT *the Hard Way"*

Note: you can bypass this entire section and go on to § 8.1 if you already have your own input file. The easiest way to run GALFIT *is to directly edit a parameter file and feed it into* GALFIT *on the Unix command line, as in § 7.3.1*

You can run GALFIT entirely via the GALFIT command line. The one possible benefit of using the interaction menu is to check on your input file format, which has been causing GALFIT to crash for no apparent reason. If so, here are the things you can do on the command line:

(Re)displaying the Menu To display or redisplay the menu after new changes have been made, hit "r".

Adding New Objects, Changing Initial Parameters, Deleting Objects To add a new object on the command line, hit 0 or N, followed by the name of the model you want to add. For example, when you get the GALFIT prompt, type:

```
0 devauc
```

will add a de Vaucouleurs function. Initially, all the parameters are set to 0 which you can then change.

To change the value of an item, enter the following on the GALFIT command line in consecutive order, separated by one or more spaces only:

1. the item number/alphabet (without the right parenthesis),
2. the initial value, then
3. optionally followed by whether to hold that parameter fixed or not during the fit. To hold a parameter fixed, use the value 0; to fit, use 1.

Here are 3 examples that produce some entries shown in Figure 17:

```
a v.fits            (See Item A -- Changes the
                     input file name to  v.fits.)

h 300 440 330 470   (See Item H -- Changes the
                     4 corners of the fitting box)

1 369.4 395.3 1 1   (Change the initial x,y position
                     to (369.4,395.3), and allow both
                     to vary)
```

If you have more than one model and, for instance, you want to change parameters for model <*number*>, you have to first designate it by typing *m* <*number*>. Then all parameters you specify to change will only affect that component. So, for example, to change the magnitude (3rd parameter) of object 4 to 19th magnitude, you have to first enter:

```
m 4
```

then:

```
3 19.0
```

Right now the program does not check to see if the parameters you entered make any sense or even if they're valid numbers.

Adding a Non-Classical Fitting Parameter (e.g. C_0, Fourier Mode, Coordinate Rotation, Truncation) The menu items by default only show the so-called "classical" parameters commonly associated with 2-D galaxy fitting. However, there are additional free parameters that can add even more flexibility to GALFIT: the diskyness/boxyness parameter C_0, the Fourier modes, bending modes, coordinate rotation and profile truncation. To minimize clutter, these parameters would only appear in the menu when they are specified to be fit by being given a value. This simplifies the use of GALFIT. Partly, the scheme is to underscore the fact these parameters are higher order corrections to a model. As such, they should only be "turned on" after a solution has first converged for the "classical" parameters.

The non-classical parameters will only appear when the user assigns to them a value (even a value of 0). For instance:

```
c0 0.1 1 1
F3 0. 0. 1 1
r0 power
```

Deleting A Non-Classical Parameter These non-classical parameters can be deleted by setting *both* the values and the toggle flags equal to 0. To disable coordinate rotation, set R0 equal to "none". For instance:

```
c0 0. 0      (Removing diskyness/boxyness)
F7 0. 0. 0 0 (Removing 7th Fourier mode)
r0 none      (Disabling coord. rotation)
```

Once a parameter is deleted or the value is held fixed to 0., it will go back into hiding unless called upon.

Deleting an Object To delete an object, use the x key, followed by the object number. For example,

```
x 3
```

deletes the 3rd model from the fit.

Start Fitting Once you're done with entering/changing all the parameters, to start fitting, hit "q".

8. GALFIT MENU ITEMS

This section describes the menu items (Figure 17) in a little more detail. The GALFIT menu is separated into two sections: the image parameters (the top half of the menu, i.e Items labeled A-P) and the object fitting parameters (the bottom half, i.e. Items numbered 0-10 and Z). There is only one set of image parameters A-P, but there can be an arbitrary number of object parameters depending on what you want to fit simultaneously. In principle there is no limit to the number of fitted components, except by the computer memory and the computation speed.

8.1. *The Image Parameters A-P*

Item A – Input Data Image: The input data image is a FITS file. It must be a single image and not an image block. If the user does not provide an input image, or if the input image is not found, then a model image will be created using the input parameters, having an exposure time of one second.

Item B – Output Image Block: The output image block is a 3-D image cube in FITS, created using CFITSIO (Pence 1999). The image data cube has 3 layers: blahblah.fits[1], blahblah.fits[2], and blahblah.fits[3]. The first image is the postage stamp sized region specified in Item I (see below). Image [2] is the final model of the galaxy in that region. Image [3] is the residual image formed by subtracting [2] from [1]. Image [0] is blank.

The final parameters obtained in the fit are stored in the FITS header of image [2]. If you are dumping the final parameters into a table using a script, this is an easier place to obtain them instead of trying to dissect "fit.log."

Note that if Item Z (see below) is set to 1 for any component, that component will still be optimized, but the model for that component will not be constructed in the output image [2]. The residual image [3] will also reflect that fact, since image [3] is created by subtracting image [2] from [1].

Item C – Input Weight (Sigma) Image: A sigma image is a map of the standard deviations of the data image, as defined by Equation 1, which is derived based on assumptions about Poisson (more precisely, Gaussian) statistics. It is used to give relative weights to the pixels during the fit, and it should not be arbitrary. More precisely, $\sigma(x, y)$ should be one standard deviation of counts at pixel position (x, y) for a *noiseless* underlying parent distribution, i.e. image. That means that, in principle, $\sigma(x, y)$ can be known precisely only if we have an infinite signal-to-noise image of the source. Because we never have this, the sigma image is always an approximation.

This rather subtle point can be made more clear by way of a simple example: an image taken of a constant sky background $\langle sky \rangle$ has only *one value of* σ across the entire image, and that value $\sigma \propto \sqrt{\langle sky \rangle}$. The proportionality factor depends on the units of the ADU. The pixel-to-pixel fluctuation one sees in a sky image comes about because each pixel value is randomly drawn from a Poisson distribution *with the same* σ *for all the pixels*, appropriate to a given mean sky value. So, in this example, the sigma image for GALFIT should be a *single value across the whole image*. For data having real objects, it is somewhat more difficult to obtain the underlying $\sigma(x, y)$ because the flux distribution is no longer constant. Nonetheless, this concept suggests an approach to take for creating a sigma image, which is described below.

If the sigma image is not provided. If a sigma image is not provided or the name is unrecognized, then it is generated internally by GALFIT based on the GAIN (or ATODGAIN) and RDNOISE parameters taken from the image header (see § 2.1). If you let GALFIT do this, the data image ADUs and the image gain should have units such that $[GAIN] \times [ADU]$ = total electrons collected at each pixel.

The background sky is important for estimating the sigma image because it is a source of noise. As far as statistics are concerned, there is no distinction between thermal background due to the telescope environment and that due to a natural sky brightness. To determine what the sky RMS noise is, GALFIT first computes a filtered median after removing ±20% outliers of bright and faint data points. For this to work, the image should in principle have have 60% of empty regions not occupied by a source of any kind. From empty regions, the RMS is computed and added in quadrature to the flux variance of the data, after removing the estimated sky background.

Schematically, therefore, the RMS image is obtained in the following manner:

$$\sigma(x,y) = \sqrt{\left(f_{\rm data}(x,y) - \langle {\rm sky} \rangle\right)/{\rm GAIN} + \sigma_{\rm sky}(x,y)^2} \tag{33}$$

In Eq. 33, the readnoise term is absorbed by the estimation of the sky RMS $\sigma_{\rm sky}(x,y)$ in [ADU], so the RDNOISE parameter is not used.

The user has the option to provide an estimated sky pedestal and sky RMS by specifying on the command line, respectively:

```
galfit -skyped ADUs  <file>
galfit -skyped ADUs -skyrms ADUs <file>
```

The sky pedestal here is an estimated value. Even though this should in principle be the same as the sky parameter one is allowed to fit for, the σ-image must be generated before fitting is done, therefore does not have the benefit of hindsight. Note that when providing the sky RMS, the sky pedestal must also be provided.

The user also has the option to not allow GALFIT to estimate the sky pedestal, by specifying:

```
galfit -noskyest  <file>
```

This is not the same thing as setting "-skyped 0" on the command line. When the sky is not estimated the sky RMS is determined from the image RDNOISE parameter, and added in quadrature sum with the noise of the flux pixel value, scaled directly to [electrons] using the GAIN parameter. This is useful if the user wants to see how much the uncertainty in the sigma image affects the fitting results.

To see the σ-image generated by GALFIT, one can specify the following on the command line:

```
galfit -outsig  <file>
```

GALFIT will then produce a file called "sigma.fits."

Details to keep in mind when providing a sigma image. There are several important things to keep in mind if one intends to create one's own sigma image. Indeed, many of the odd behaviors reported back to me about GALFIT were by people who supplied their own version of the sigma image, which were not appropriate in the sense of Equation 1. Such odd behaviors include simple fits that last for hundreds of iterations, wild swings in parameter values between each iteration, extremely large/small χ^2_ν values, or convergence on solutions that make no sense. These issues may be avoided if one keeps in mind the following when supplying a sigma image:

- *Visual Inspection.* Before using a sigma image, *please display it to see that it bears some resemblance to the data*, modulo weird scaling, since, after all, the σ-image is derived directly from the data! It is known that some algorithms (e.g. Multidrizzle) output a "weight image" that is either an exposure time map, or a map of the number of images combined. Either kind of those "weight image" is incorrect and has been known to cause GALFIT to go haywire.

- *Units.* The σ-image should have the same units as the input data image. If the data pixels are in [counts sec^{-1}] then so must the weight image be in [counts sec^{-1}]. Otherwise, the χ^2_ν value that GALFIT outputs will not make much sense, even if it might not affect the solution or how GALFIT converges on it.

- *Noise.* The σ-image is a noise map of the data. However, if one thinks about it, it usually ought not itself be noisy! Unfortunately, when one creates a sigma image from a noisy image, the map also looks noisy. There is not much one can do about this.

It is important to keep in mind that often one should not have to provide his/her own sigma image if the GAIN and RDNOISE parameters are in the image header, and ADU × GAIN = total electrons detected so that Eq. 33 applies. When in doubt, first let GALFIT *generate a σ-image internally, even if the ADU normalization is weird.* The effect of wrong units by a multiplicative constant (as sometimes the case if the units are in [Counts sec^{-1}]) only makes the normalization of χ^2_ν rather screwy. But it is the relative weights between the pixels that govern convergence, which is not affected by an uncertainty in χ^2 normalization.

Item D – Convolution PSF, and (optional) CCD Diffusion Kernel: The observed PSF image, in FITS format, is required only if one wants to convolve a model with the PSF or fit a PSF to a star. Otherwise, it is optional. The diffusion kernel is also optional, and is mostly associated with *oversampled* HST PSFs that are created using the TinyTim software for CCD imagers such as WFPC, WFPC2, STIS, and ACS. The reason charge diffusion kernel is considered when using TinyTim is that, physically in CCDs, the neighboring pixels are not completely insulated from one another because the potential wells are not infinitely deep. Therefore, when incoming photons excite the release of electrons in a substrate, the electrons may travel into neighboring pixels. The effect of this diffusion causes an image to be slightly blurred. This blurring is entirely a detector artifact, and adds to the blurring caused by telescope and atmospheric optics.

If the PSFs are created through natural stars, or if the TinyTim PSF is created to be 1-time oversampled, one can ignore the CCD diffusion kernel because the kernel is already included. However, if you created a twice (or more) sub-sampled PSF using TinyTim and your data image is only one-time sampled, then you may want to provide the diffusion Kernel (see below).

The peak flux of the PSF image should be at the geometric center when the number of pixels on a side is odd. However, if the number of pixels N on a side is even, the

peak should be located at pixel position $(N/2+1)$. If the peak is anywhere else, the model that GALFIT generates will be systematically offset in position by the difference from the predefined center. This is important to keep in mind when the convolution box is small: you may want to make sure to refit the image with a large convolution box after the solution has first converged.

As mentioned above, the input PSF can be accompanied by a CCD charge diffusion kernel, which is simply a text file. It is usually only needed if one has created an *oversampled* HST PSFs using TinyTim. If the PSF has unit sampling, the diffusion is applied by TinyTim automatically, so GALFIT will not re-apply it even if a kernel is specified. The appropriate charge diffusion kernel can be found under the "COMMENTS" section in the PSF image header created by TinyTim. Here is a typical example of what the diffusion kernel input file should look like:

```
0.012500  0.050000  0.012500
0.050000  0.750000  0.050000
0.012500  0.050000  0.012500
```

Table 1: An example of the WFPC2 CCD charge diffusion kernel:

The charge diffusion convolution is applied only after the model image has been convolved and rebinned (see next Item) down to the original resolution of the HST imagers. Note that if the science data has been drizzled to a platescale other than the original instrumental pixel scale, it makes no sense to apply the diffusion kernel (see Item E for further explanation). Also, note that observed HST PSFs are sometimes slightly broader than the PSFs generated by TinyTim, which may be caused by a small amount of spacecraft jitter.

NOTE: If the PSF image is sitting on top of a non-zero sky background, then the sky should be removed. Or else, when convolved with a model, the region inside the convolution box would seem to be elevated compared to the region outside of the box. The PSF does not have to normalized – GALFIT *will do so automatically.* A DC offset will also appear if the PSF size is too small, because the image cuts the PSF off in the wings. However, in this case, the sky pedestal is not the problem. Instead, one should enlarge the PSF size.

Item E – PSF Fine-Sampling Factor: When the PSF is not Nyquist sampled the Fourier transform is not uniquely invertible. Therefore, in principle all data should be Nyquist sampled (i.e. FWHM $\gtrsim 2$ pixels) for convolution purposes. If one does not have a Nyquist sampled science image, sometimes one can obtain a reasonably accurate oversampled (i.e. more finely sampled) PSF through observational (e.g. dithering) or numerical (e.g. TinyTim) techniques. For HST images, there is a software that can generate oversampled PSFs. The other alternative is to combine dithered sub-exposures of bright stars during an observation, or by interpolating to get an "intrinsic" PSF by using multiple stars in the same image, e.g. with globular clusters.

GALFIT can deal with situations where the PSF provided is more finely sampled than the data, i.e. *the PSF has a finer pixel scale (arcsec/pix) than the data.* GALFIT will internally produce models at the same sampling factor as the input PSF, perform convolution, and finally rebin the results to the data pixel scale.

The PSF fine-sampling factor for Item E can only be an integer value. It is a *ratio between the platescale (arcsec/pix) of the PSF and the DATA, observed under the same seeing condition (i.e. optics convolved with atmosphere).* If they have the same platescale + seeing, the factor is 1.

A TECHNICAL MEMO REGARDING HST CCD DATA, i.e. for WFPC2, STIS, WFPC, and ACS, but not NICMOS: If the PSF being used is created from natural stars, or if the TinyTim PSF is 1-time sampled, there is no need to apply the CCD diffusion kernel. However, if the PSF is created by TinyTim and it is N-times oversampled (N ¿ 1), TinyTim provides a diffusion kernel for you, but it does not actually apply it. The reason is that the kernel should only be applied after you bin the PSF down to *1-time* sampling. So, what do you do about the diffusion kernel if your data are drizzled to 2× sampled so you want to keep your TinyTim PSF on the same grid? One obvious solution is to just transform (i.e. interpolate) the diffusion kernel the new plate scale and apply that. If you decide to do this, you can apply the kernel outside of GALFIT instead of inside, since the data and the PSF have the same sampling factor. This will make GALFIT run faster as it doesn't have to apply the same kernel each time.

Item F – Bad Pixel Mask: Sometimes one may want to exclude pixels from a fit. This bad pixel map can be either a FITS file or an ASCII text file. If the file is an ASCII, it should have 2 columns listing x and y coordinates (without a comma separator) of all the bad pixels. If you want to mark out an irregularly shaped region and have a list of polygon vertices, you can run a program called "fillpoly" to create points inside the polygon. See GALFIT website on Frequently Asked Technical Questions to get a copy. The output file can then be read directly into Item D.

If the dust map is a FITS image, the bad pixels should have a value of > 0. while good pixels have a pixel value of 0.

Item G – Parameter Coupling: Parameter constraint/coupling file (ASCII) is optional. Parameters can be held fixed to within a certain range or can be coupled between different components by providing this file. An example of the format is found in the file *EXAMPLE.CONSTRAINTS*. When constraints are imposed it is unclear what the errorbars mean, if anything. Furthermore, it may prematurely force the solution to wander off into a corner of the parameter space from which it is difficult to wander out. So use constraints at own risk! See § 11 regarding parameter constraints for more details.

Item H – Fitting Region: The image region to fit. GALFIT will cut out a section of the image specified by "xmin xmax ymin ymax" from the original image and then minimize χ^2 only over that region. The fitting region should be large enough to include a significant amount of background sky, especially if the sky is a free parameter in the fit.

Item I – Convolution Box Size: The convolution box size and the PSF image size are the two most important factors in determining the running speed of GALFIT.

Convolution can take up ¿ 80% of total execution time for a small image. There is one box for each model component, centered on the model, so that all components will have their centers convolved with the PSF. This is much more efficient than convolving the entire image. Of course, if one wants to convolve the entire image, one can still set the convolution box to the fitting region size or even larger, but this is often not the most efficient way to go.

In principle the box size should be just big enough so that the seeing does not affect your galaxy profile outside of the box (something like 20 or more seeing diameters, depending on how extended the PSF wing is.)

Item J – Magnitude Zeropoint: The magnitude zeropoint is used to convert pixel values and fluxes into a physical magnitude by the standard definition:

$$\text{mag} = -2.5\log_{10}(\text{ADUs}/t_{exp}) + \text{mag zpt.} \qquad (34)$$

The exposure time is taken from the EXPTIME image header without prompting the user. *So please make sure the EXPTIME header reflects how the data pixel have been normalized.* Unfortunately, it is rather common for the EXPTIME to show the total integration time, but for the image ADUs to be normalized to one second. One reason why this may happen (especially in infrared data) is that not all pixels may have the same exposure time. If the "EXPTIME" keyword is not found, GALFIT assumes the exposure time to be 1 second. If you want GALFIT to generate a $\sigma(x, y)$ image internally and the ADUs are in [counts/sec], please multiply EXPTIME back to the image and update the EXPTIME header accordingly.

Item K – Plate Scale: The plate scale should be in units of arcseconds, and is used only to convert fluxes into surface brightness units [mag/arcsec^2] for the King and the Nuker profiles (see Eq. 35). Note that the *sizes* of objects (r_e, r_s, r_b, and FWHM) in GALFIT are quoted in [pixel] units rather than in [arcsec] units.

$$\mu = -2.5\log_{10}\left(\frac{\text{ADUs}}{\Delta x \Delta y\ t_{exp}}\right) + \text{mag zpt.} \qquad (35)$$

Item O – Interaction Window: Sometimes it is useful to quit out of a fit early (and save results), pause a fit, extend the number of maximum iterations (default = 100), etc.. The interaction window lets you do this. There are three options:

In the "regular" option there is no interaction possible.

In the "both" option, a green xterm window will pop up to show you what commands one can issue to GALFIT during fitting. The commands (see § 6) are issued by typing a single letter in the green window, not in the fitting window (except when the fit is paused, or when one is entering new number of iterations).

In the "curses" mode, one can issue the same commands as the "both" option. But all the interaction is done in the fitting window instead of a separate window.

For Mac OS X users only: To use the "both" mode, you must run GALFIT *in X11 (xterm) mode, otherwise it would complain about not being able to open a window. If not in X11, set this parameter to "regular."*

Item P – Options: If this option is set to 0, GALFIT will run normally. Once it finishes running, it will create the standard image block (data, model, residual) of the best fit[4]. If the option is set to 1, GALFIT will create a model image based on your input parameters and immediately quit. If the option is set to 2, GALFIT will create an image block (data, model, residual) based on the current fitting parameters, and quit. If the option is set to 3, GALFIT will create an image block "subcomps.fits" where each image slice after the first (data) is the individual components used in the fit. Alternatively, one can do these things on the command line by doing:

```
galfit -o1 <file>   (model only)
galfit -o2 <file>   (standard img. block)
galfit -o3 <file>   (subcomps)
```

8.2. *Classical Object Fitting Parameters 0-10, and Z*

GALFIT allows for a simultaneous fitting of arbitrary number of components simply by extending the following object list (0-10, Z) for each component. Items 1-10 are the initial rough guesses at the object parameters, and they don't have to be accurate. But of course, the more complicated a fit is, i.e. the more number of components, the better the initial guesses should be so GALFIT doesn't wander off to never-never-land.

The 2nd column in Items 3-10 are initial guesses for the parameter and the 3rd column is where one can hold the parameters fixed (0) or allow them to vary (1). Note that items 1 and 2 (x_c, y_c) are on the same line (except for fitting the sky).

Below is a more detailed explanation of what each of the parameters means:

Item 0: Object name. The valid entries are: sersic, devauc, nuker, expdisk, moffat, gaussian, sky, psf.

Items 1, 2: X and Y positions of the galaxy in pixels. For the sky, Items 1 and 2 are on consecutive lines instead of the same line. For the sky Item 1 is the DC offset in it ADUs, and Item 2 is the gradient in the X-direction.

Item 3: For Sérsic, de Vaucouleurs, and exponential disk this is the *integrated* magnitude of a galaxy. For Nuker and the King profile this is the *surface brightness* (mag/square arcsec) calculated using the pixel scale from Item K. For fitting the sky, this is the sky gradient in the Y-direction.

Item 4: Scalelength of the fitted galaxy in PIXELS, not arcseconds. The scale length is measured along the semi-major axis.

Item 5: For Sérsic it is the concentration index n. For Nuker, it is the powerlaw α. For King, it is the truncation radius beyond which the fluxes are 0. For all other functions it is ignored.

Item 6: For Nuker, it is the powerlaw β, and for King, it is the powerlaw α. For all other functions, this parameter is ignored.

Item 7: For Nuker, it is the powerlaw γ.

Item 9: The axis ratio is defined as semi-minor axis over the semi-major axis: for a circle this value is 1, for an ellipse this value is less than 1.

Item 10: The position angle is 0 if the semi-major axis is aligned parallel to the Y-axis and increases toward the counter-clockwise direction.

[4] "Best fit" does not necessarily mean a "good fit"

Item Z: If you want this model to *not* be subtracted in the final image, set this to 1. If you want to subtract this model from the data, set this to 0. When this parameter is set to 0 for all the objects, you will get a residual image. The default is 0. Note that this does *not* affect whether this component is optimized, just whether or not it shows up in the output images. This is useful for showing subcomponents in the residual image in a multi-component fit.

8.3. *Non-Classical Fitting Parameters*

Non-classical parameters allow GALFIT to break from ellipsoidal axisymmetry. They are absconded from view if they are not used in the analysis. The following parameters are non-classical: diskyness/boxyness, Fourier modes, bending modes, coordinate rotation, and truncation. Another reason for keeping the non-classical parameters from view is to emphasize the higher-order nature of these parameters. This means that it is often necessary to first let GALFIT converge on a classical, ellipsoidal, solution, before turning on the non-classical ones. In other words, the simplest and most correct way to fathom these parameters is to consider them as higher order perturbations on the best fitting ellipse, even if the perturbations can be quite large. In the menu or template file, the non-classical parameters usually should come at the end of the component that one is interested in modifying. The only exception is the truncation model, which can appear anywhere in the object list, and behaves like a separate component.

Note that a single component model can be altered by any one or *all* of the following functions *simultaneously*. For example, one can modify an ellipse into a spiral, which can be perturbed by Fourier modes, and truncated either in the inner or outer regions, or both. Those truncation functions may even be modified by Fourier modes, bending modes, and/or the diskyness/boxyness parameter. The degree of complexity that is allowed may seem a bit dizzying at first. But, just because such complexity is allowed does not mean that using them all simultaneously from the start is a good idea. Proper use of these capabilities requires some training and human supervision to produce the most physically meaningful solution.

8.3.1. *Diskyness and Boxyness*

The effect of diskyness/boxyness parameter on a perfect ellipse is shown in Figure 6. The diskyness/boxyness parameter can be used with Fourier modes, though it is not advisable because of parameter degeneracy issues.

Item C0 – Diskyness/Boxyness Amplitude C_0: When $C_0 < 0$ the ellipsoid appears disky, and when $C_0 > 0$, it appears boxy. The following specifies its use in the input file:

```
c0) 0. 1
```

or on the command line: c0 0. 1

8.3.2. *Fourier Modes*

The Fourier modes allow the possibility to create complex shapes (Figure 2). Just like diskyness/boxyness, Fourier modes are higher order corrections to a model. As such, it is not wise to "turn them on" when most of the other parameters are still rough, because far away from an optimum solution, large initial residuals would look like high power Fourier modes. This is especially true for the $m = 2$ mode, which is quite degenerate with the axis ratio q. The recommendation for using Fourier modes is to first run GALFIT without them. Once a solution has been found, the Fourier modes may be used to improve the fit, and to quantify the degree of deviation from an optimum model.

Item FN – Fourier mode amplitude and phase angle: Each Fourier mode has two free parameters, the amplitude and phase angle as given in Equation 22. The mode amplitude is in units of the fractional radius, whereas the phase angle has units of degrees. The phase angle is zero in the direction of the semi-major axis of the best fitting ellipse, and is reported only in the cardinal range.

To activate Fourier modes, one can specify something like the following, which enable the 3rd and 2nd Fourier modes:

```
F3) 0. 0. 1 1     #  Fourier mode m=3
F2) 0. 0. 1 1     #  Fourier mode m=2
```

Note that holding the amplitude fixed to 0 for any mode will effectively turn off that Fourier mode, making it disappear from the menu. One can employ an unrestricted number of Fourier modes, and some may be skipped. However, only the first 6-10 modes are useful. The higher the mode one uses, the more likely that the fit is sensitive to neighboring objects.

8.3.3. *Bending Modes*

Bending modes allow the possibility for the model to shear and bend (like a banana), see § 5, Figure 3, which are otherwise not possible to model using Fourier modes.

Item BN – Bending Modes: The free parameters are the model amplitude a_m defined in Equation 26. The bending modes appear like the following in the menu or template file:

```
B1) 0. 1      #  Shear term
B2) 0. 1      #  "banana" term
```

Just like the Fourier modes one can fit for an unrestricted number of bending modes and any number may be skipped. However, only the first 3 are the most useful, and the first mode (B1 = shear), can be somewhat degenerate with the axis ratio of a model or the second Fourier mode, when the amplitude is low. So care should be taken when fitting the B1 mode.

8.3.4. *Coordinate Rotation*

As shown in Section 5, coordinate rotation allows for the creation of spiral structures. There are two types of coordinate rotation allowed, α-tanh and log-tanh. The prefix α and *log* refer to the asymptotic behavior of the angular rotation function beyond $r > r_{\rm out}$: $\theta(r) \propto r^{\alpha}$ or $\theta(r) \propto \log(r)$. The suffix *tanh* refers to the fact that at

small radii, the powerlaw/logarithmic functions are truncated by a hyperbolic tangent. This is useful because the tapering of the *tanh* function toward $r = 0$ allows for the creation of a natural bar. Note that unlike ellipsoid models, a spiral model is actually a 3-dimensional structure that can be inclined and tilted. However, it is an infinitely thin model when viewed at 90^o inclination.

Item R0 – Coordinate rotation function type: The options are "power" for α-tanh rotation or "log" for "log-tanh rotation.

Item R1 – Inner (bar) radius: The radius at which $\theta(r)$ flattens off to 0, thus creating the appearance of a bar (see Figure 7). As Section 5 explains, the "bar radius" is an useful mathematical construct, and may or may not have any bearing on the physical bar.

Item R2 – Outer radius: The radius beyond which the function $\theta(r)$ behaves either like a pure power law or a pure logarithm (see Figure 7).

Item R3 – Total angular rotation out to outer radius: The total angular rotation ($\theta_{\rm out}$) at the outer transition radius, $\theta = \theta_{\rm out}(r/r_{\rm out})^{\alpha}$ or $\theta(r) = \theta_{\rm out}\log(r/r_{\rm out})$.

Item R4 – Asymptotic powerlaw rate (α-tanh) or scale parameter (log-tanh): For α-tanh, this parameter is the α parameter in $\theta \propto r^{\alpha}$. For the log-tanh function, it is the winding scale parameter (r_{ws}) in $\theta \propto \log(r/r_{ws})$.

Item R9 – Inclination angle to line of sight: A face-on spiral can be inclined relative to the line of sight to appear more elliptical. Zero degrees is face on, whereas 90 degrees is edge-on. Note that a 90 degree solution is not allowed, because it corresponds to an infinitesimally thin model.

Item R10 – Position angle in the plane of the sky: An inclined spiral can be rotated in the plane of the sky to match the position angle of the galaxy. A zero degree PA is usually parallel to the x-axis, in contrast to the standard definition of the PA of an ellipsoid model.

8.3.5. *Profile Truncation*

As discussed in Section 6 the profile truncation function can modify both the radial profile *and* the azimuthal shape of an object when the truncation is allowed to have its own set of shape parameters.

The profile truncation is also a little unusual in the sense that it is a pseudo-component: in the GALFIT menu, it is designated as an object component just like a Sérsic or a Gaussian, in contrast to Fourier or bending modes which only modify the component they immediately follow. This is done so that multiple components can be truncated by the same functions of identical parameters without resorting to complicated constraint files. For instance, this allows a bar, a spiral arm, and a ring, when represented as three individual components, to be joined seamlessly when truncated at the transition region by just one function. For this to work, the component being truncated has to be associated with a truncation component, and this is done by attaching one or both of the following to the end of a regular light profile component:

Item Ti – Inner truncation by component number N: The light profile is truncated in the inner region by component number N. This has the effect of making a ring out of a regular light profile, whose outer region (beyond $r_{\rm out}$) is unaffected by truncation. When an object is truncated in the inner region, the flux normalization parameter changes to surface brightness normalized at $r = r_{\rm out}$, as opposed to the total luminosity or central surface brightness. This is done because the flux is *near* peak near that location, thus is more intuitive.

Item To – Outer truncation by component number N: Just like the previous, this parameter is added on to the end of a light profile model to indicate it is truncated in the wings by component number N. When a profile is truncated *only* in the outskirts, the flux normalization parameter is the central surface brightness. If used in conjunction with inner truncation, the luminosity parameter is the surface brightness at $r_{\rm out}$.

The truncation function itself can have the following free parameters, some of which are optional. If not explicitly listed in the template file or GALFIT menu, the optional parameters take on the value of the light profile component to which the truncation function is associated. See Appendix B for details on the how the truncation function depends on the following parameters.

Item T0 – Truncation type: The options are "radial" (for all light profiles except *edgedisk*. For *edgedisk* only, the only options are "length", and/or "width". The truncation type also has options Type 1 vs. Type 2 as well as Type a vs. Type b (for spiral rotations only), as explained below. Type 1 and Type a are default, and have no special designations, whereas Type 2 and Type b are represented, for example, by "radial2" and "radial-b".

Item T1 – (x, y) position (optional): The centroid of the truncation function can be different from the light profile model. When not specified (i.e. when T1 does not exist in the menu), the centroid position is the same as the individual light profile models (see Figure 15c).

Item T2 – Break radius: For an inner truncation, this corresponds to the radius where the flux is 99% of the amplitude of the original model (i.e. at $r_{\rm out}$). For an outer truncation, this corresponds to the inner radius where the flux is 99% of the amplitude of the original model.

Item T3 – Softening length (Type 1) / softening radius (Type 2): This parameter is the length beyond $r_{\rm break}$, over which the flux drops to 1% of the flux normally at that radius. Type 1 models have softening length as the free parameter, so that the smaller the softening length value, the sharper the truncation. Whereas, Type 2 models have softening radius so that the closer in value T2 is to T3, the sharper the truncation.

Item T9 – Axis ratio (optional): The truncation function can have its own elliptical axis ratio parameter independent of the light profile (see Figure 15). When T9 does not exist in the menu, it takes on the value of the light profile model.

Item T10 – Position angle (optional): The truncation function can have its own position angle parameter independent of the light profile (see Figure 15). When T10 does not exist in the menu, it takes on the value of the light profile model.

Type a vs. Type b truncations. For spiral galaxies, the truncation parameter normally follows the tip

and tilt of the coordinate system, because it is generally more intuitive to envision truncation in a face-on configuration. However, if for some reason it is easier to conceive truncation in the plane of the sky instead of in the plane of the disk, then the Type-b truncations are allowed. All the parameters thereby refer to the values as measured in the plane of the sky rather than in the plane of the disk. For non-spiral galaxies, there is no difference between Type-a and Type-b.

As shown in Figure 15, the truncation function, as a pseudo-component, can be modified by diskyness/boxyness, Fourier and bending modes. These terms can be added onto the end of any truncation function component, and identically affect all profiles that are linked to it.

9. THE GREEN POP-UP GALFIT WINDOW

If Item O is set to "both", a green GALFIT window will pop up to indicate that the fit is ongoing. At any time, you can quit, output a menu file, pause the fit, or change the number of iterations by hitting the keys "q", "o","p", or "n", respectively, *in the green pop-up window*. At the first convenient moment (i.e. after the current iteration), GALFIT will do what you asked. The pop-up window has memory, so if you hit a key multiple times, e.g. out of frustration, it will be reissued at the earliest convenience. To get rid of the memory, kill the green window.

The output menu file (option "o"is simply a preformatted file that you can feed back into GALFIT . It stores parameters in the current fit iteration.

If you hit "p" or "n", you will receive a prompt in your iteration window (not the green one). You must answer the question in the iteration window before it would go on. Usually, GALFIT will decide when to quit fitting on its own. But, if GALFIT iterates over 100 times (an artificial limit), it will also quit. You can set the maximum number of iterations you want this to happen by hitting "n" and specifying the number of iterations. Normally, GALFIT should converge between 10 to 30 iterations.

10. OUTPUT FILES

Once GALFIT finishes fitting, it will store the final parameter information into 2 text files. The first file, "fit.log" summarizes all the final parameters and error bars for the fit. This file just keeps getting appended and does not get removed. The errors quoted in "fit.log" are based on diagonalizing and projecting the covariance matrix. Thus the errors are purely statistical – some would say meaningless because the errors are often dominated by systematics due to the real galaxies not being idealized profiles. The columns of numbers in the output file are in the same order as the parameter numbers. Parameters whose values are held fixed are enclosed in square brackets, e.g. [6.27], whereas constrained parameters are in curly braces, e.g. {1.24}. If there is any parameter enclosed in between stars (as of Version 3.0.1), e.g. *0.15*, it may have caused numerical convergence issues such that even if GALFIT produced an output, the entire solution may have been unreliable. One must refit the galaxy and restart/fix that or other parameters to something sensible. A solution should never have a parameter enclosed within stars!

When GALFIT finishes fitting, it will also output a file called "galfit.NN" that contains all the best fit parameters. NN is a value that keeps increasing so it will never overwrite the previous fit. Note that if there are missing numbers in the "galfit.NN" sequence, the gap will eventually be filled. One can modify this file and feed it back into GALFIT to refine the fit.

Lastly, GALFIT will output a data image block, which is described in Item B of § 8.1. The final fit parameters and a few important input parameters are also placed into the FITS header of image[2] for convenience. Like in 'fit.log', parameters whose values are held fixed are enclosed in square brackets, and constrained parameters are in curly braces.

If a user is running GALFIT in a batch fitting mode using an external wrapper script, GALFIT will return error number 1 if it crashes, otherwise it'll output a 0. This code number can be intercepted by the calling script, for example:

```
$errno = system ("\galfit\ $parfile");
```

The calling script can then decide how to proceed, i.e. whether to move on to the next object, to fit this object manually, or to redo the fit by modifying the input file.

Also, in the header of image.fits[2], GALFIT will output a parameter called "FLAGS" which lists all the flags that have been tripped during the iterations. To see what the flags mean, type on the command line:

```
galfit -help
```

Most of the flags will not affect the fit; they are just gathered into one place so the user may be aware of which ones were tripped. The most important flags to watch out for are numbered "1" and "2," as well as the A ones.

11. HINTS ON DIFFICULT FITTING

The above information is all that one needs to know to run GALFIT. This section discusses some rules of thumb when dealing with complicated analysis.

For simple one or two component models GALFIT can usually perform its duty without any interaction. But once in a while one may come across a galaxy that requires some amount of interaction to achieve a reasonable fit. Unfortunately, this situation is often blamed on parameter degeneracies, when in fact it may be due to some other reasons. To be sure, there *are* times when degeneracy and local minima are to blame. However, many of the problems people encounter are caused by either bad priors or bad initial guesses that cause parameters to take on extreme values. With bad priors, no amount of manipulation or parameter restarts is going to instill more meaning to the analysis.

There is also another widely-held belief that numerical degeneracy is mostly caused by the number of parameters – a notion which is overly simplistic. To see why, consider a situation where there are a million stars that are non-overlapping. The three million parameters needed to fit those stars using a Gaussian profile, or the 7 million needed for using a Sérsic model, are completely non-degenerate. Contrast this to a single Nuker model of 9 parameters when r_b is sufficiently small, which would be degenerate in the powerlaw parameters, α, β, and γ.

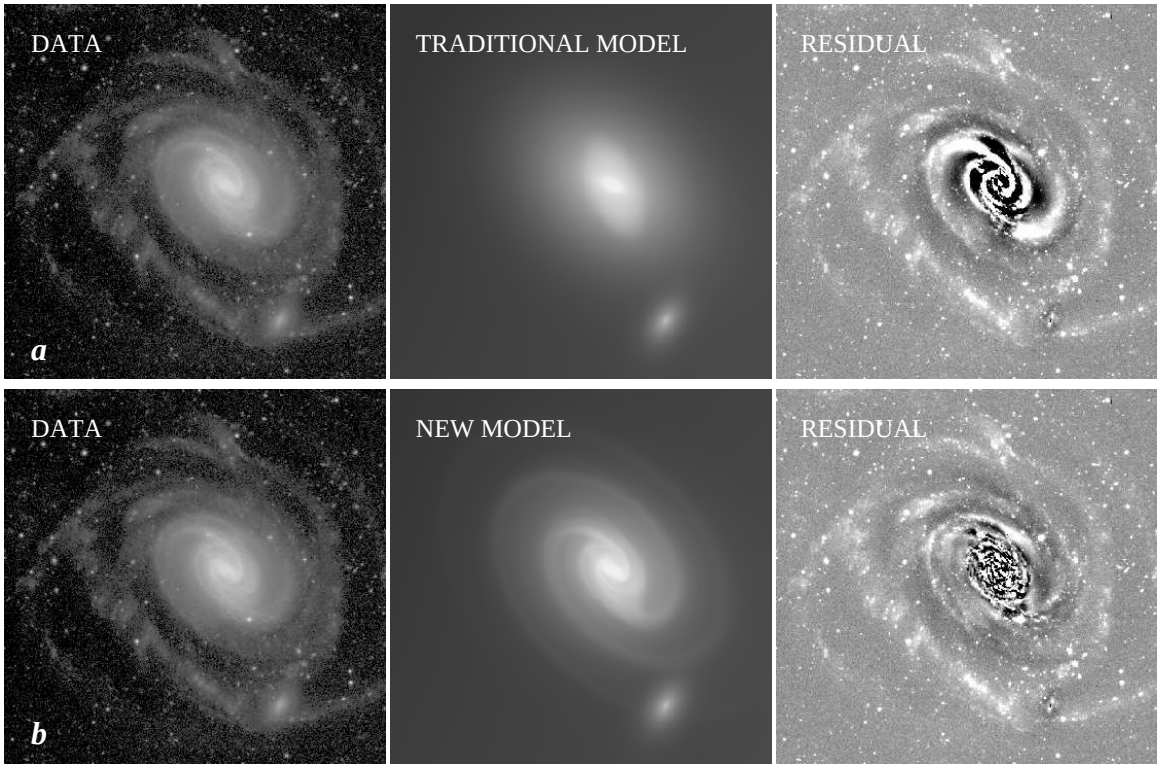


FIG. 16.— Examples of bad priors or numerical degeneracy on image analysis. *a*) a 3 component traditional ellipsoid fit to NGC 289 and a single component fit to a neighboring object. Note that none of the components corresponds to a "bulge" due to the strong spiral arm residuals. *b*) a bulge, disk, bar, and spiral arm fit of the same galaxy. The fitted bulge comes out naturally to have an axis ratio of 0.8, and is an exponential bulge.

It is important for users to learn to tell the difference between situations caused by *bad priors* from those caused by true *numerical degeneracies* when attempting to perform complex analysis, because the solutions to those problems are quite different.

11.1. *What are Bad Solutions Caused by Bad Priors?*

A prior is bad when an user wants GALFIT to perform analysis by using models that are ill-suited to the task. The most common example is when one tries to do a two-component decomposition, allegedly to extract a bulge and a disk, on a galaxy which clearly has three or more components, e.g. bulge, disk, bar, spiral, ring, etc.. The solution here is one which finds the best compromise between the substructures, so that neither model component may represent any physical feature in particular. Here, the notion that one is performing bulge-to-disk decomposition with two components is a flawed prior. Another common situation is when a two component fit might settle on features which are closest to the initial parameter guesses, because features like a ring, bar, or spiral, are spatially localized. This situation is often referred to as a local minimum solution, and it mostly happens when some of subcomponents are strongly localized features. However, it is important to keep in mind that even if a code can break out of such a local minimum, the meaning would still be vague because the model represents a compromise of sorts. Figure 16, *top*, shows such a situation where the prior is a 3 component fit intended to extract a bulge, disk bar. However, even with 3 components it is clear that at least the bulge component bears no visual resemblance to a bulge, and is instead strongly affected by the spiral residuals. Contrasting that to Figure 16, *bottom*, where spiral structures are fitted away, the bulge comes out more sensibly to $q = 0.8$ and $n = 1$.

Another situation where the prior is inappropriate is when one tries to fit *more* components than is present in a galaxy, e.g. a 2 component fit onto a one component galaxy. Because few galaxies are modeled perfectly by a single analytic function, the solution one obtains may only reflect the degree of mismatch between the model profile and the data.

In summary, when a prior is bad, restarting the fit from different initial conditions can yield an equally bad and different solution; no amount of clever fine-tuning or starting initial conditions can rescue the situation from ambiguity. In that sense, it is important to understand that bad solutions due to flawed priors are not true numerical degeneracies. The solution is to apply better priors, and sometimes this means using *more* components rather than fewer. Indeed, often a more stable solution can be achieved by using more components if they can remove all the prominent features, as for the case of Figure 16. In contrast, for true numerically degenerate solutions, no amount of adding new components will produce a more meaningful solution.

11.2. *What are Bad Solutions Caused by Numerical Degeneracy or Local Minima in the χ^2?*

There are situations in galaxy fitting where there are true problems with numerical degeneracy and local minima. This happens when multiple parameters can pro-

duce similar features. For instance, in the Nuker profile (Equation 16), when r_b is sufficiently small, when there is a profile mismatch, or when the data are noisy, the parameters α, β, γ are truly numerically degenerate, because there is an infinite number of ways to achieve the same value for the fraction: $\frac{\gamma-\beta}{\alpha}$. More specifically, as shown in Figure 13, a low α can be reproduced by both a high γ and a low β.

Another example of numerical degeneracy is between the axis ratio (q), the second Fourier mode ($m = 2$), and the first bending mode ($m = 1$, i.e. shear), all of which can simulate ellipticity. Likewise, a combination of the $m = 2$ and $m = 4$ Fourier modes can reproduce the same features as the diskyness/boxyness parameter (C_0).

A third example is in trying to fit a spiral structure. When spiral arms are diffuse and have low inter-arm contrast, it may be difficult for GALFIT to lock onto a solution. In this situation, a model with a large amount of winding of thin spiral arms might reduce the residual just as well as a thick arm with fewer windings.

Some other examples of numerical degeneracy include high Sérsic index values with the sky level, or, the size/central concentration of a truncated model that has a long softening radius. Note that in every one of the above examples, degeneracies happen because, numerically, different parameters are similar in their ability to model some behavior. Therefore, it is worthwhile to keep in mind that some of these parameters should not be used together haphazardly, even though one is allowed to do so in GALFIT.

The bottom line is that true numerical degeneracies often cannot be solved by using better priors, and they represent situations where different starting conditions *can* affect the output parameter values or the physical inference of the components. Indeed, practically the only solution to true numerical degeneracies is to start the fit with different initial conditions, or by choosing a more appropriate function for the desired science task.

Even though real degeneracies may be present between some parameters, it is useful to think carefully about whether the science goals are negatively affected. Here is a concrete example: if a goal is to extract the total luminosity or size of a galaxy bar, then the issue of degeneracy is moot even if one chooses to use both the C_0 and the Fourier modes. For, even though C_0 and Fourier modes are mutually degenerate, they are *not* degenerate with the size and luminosity parameters. The use of high order modes would benefit in this scenario by recovering more of the flux. As another example, as shown Figure 16, sometimes the spiral arm parameters may be degenerate for various reasons. Even so, the use of a spiral component can help to stabilize the measurements of bulge and disk components.

12. RULES OF THUMB FOR IMAGE FITTING AND FREQUENTLY ASKED QUESTIONS

There are certain rules of thumb, i.e. good practices, to understand when doing image fitting analysis. Some of these originate from questions I received in the course of user support over the years, others based on technical knowledge of the how the Levenberg-Marquardt algorithm works, while others still are just based on personal experience.

1. *It is better to work with images where the pixels are in* counts *units rather than counts per second.* Now a days, it is common for reduced images from telescopes to have pixel units of counts per second. However, for image analysis, it is better to work with units in counts because it is easiest to known when the sky parameter is fit incorrectly. In galaxy fitting, the sky parameter is one of the most important parameter to measure accurately. A wrong estimate by even a count or two can lead to significantly wrong total luminosity, concentration, and size of a galaxy. When the image is in counts per second, a significant error in the fit of the sky may be hard to detect when it is divided by the exposure time.

2. *Check to make sure the sigma image is correct.* If the convergence seems a bit weird (large changes in parameters that do not seem to converge toward any particular answer) make sure that the sigma image is reasonable. If you are providing your own sigma image into GALFIT, please take a look at it first to see if the objects show up in that image. If they do not, it is not a sigma image. Please see the GALFIT page regarding why the sigma image is important to get right.

3. *Start simple, build up complexity.* When fitting multiple components, start with single or two components first, then add more components based on need arising from visual inspection of the residuals.

4. *Always start with ellipsoidal models first.* When fitting Fourier modes, spiral arms, or other things more complicated, always start out with simple ellipsoids. Then, take the best fit solution and add complexity to those. One should never introduce Fourier modes until the very last step. For instance, when fitting a spiral galaxy using coordinate rotation, with the exception of the $m = 1$ mode, I always find the best solution without Fourier modes first. Once the spiral component "locks on," I then introduce Fourier modes to create more realistic looking arms.

5. *Keep a sharp eye out for small axis ratio ($q \lesssim 0.1$) or size ($r \lesssim 0.5$ pix) parameters in the model!* When either the size parameter or the axis ratio of a model grows too small at any time while GALFIT is iterating, it will not be able to converge effectively. All the parameters would appear to be frozen, even when the code continues to iterate. The reason this happens is that small sizes and axis ratios often mean that the flux of the model can fit within a single pixel in one or all directions, therefore there is insufficient gradient information for GALFIT to converge. Note that GALFIT often will *not* crash when this happens. The code will just seem to converge extremely slowly. To solve this problem, hold the problematic size or axis ratio parameter fixed to a larger value until the other parameters have converged, then release it to see whether it continues to misbehave. In a similar vein....

6. *Know when to hold 'em (i.e. the parameters fixed).* If a fit is converging on something non-sensical, one can temporarily hold some of the parameters fixed to more sensible values in the main input template file (as opposed to the constraint file). They do not have to be precise. One can set them free later when the other parameters have converged. This technique is useful when parameters like the Sérsic index and size to haywire because of nearby sources, PSF or profile mismatches, sky determination issues, or for other reasons. Note that if one has to do this, often it is because the initial guesses are very far from reality for at least one of the components, or there is something else in the image that should be either be fitted simultaneously or masked out. For instance, large residuals around bright unresolved sources can often cause an underlying Sérsic model to take on extreme values of n or r_e.

7. *Parameter constraint files are NOT good to use.* Often times, people use parameter constraints to make sure that the parameters fall within a sensible range. However, keep in mind that: 1. The reason that parameters may get nonsensical is that other things in the image are affecting the fit. Thus the solution should be to remove the offending objects. 2. Parameters that hit the boundary walls are bad parameters, and should not be trusted/used. 3. When a parameter hits a boundary, often times it hampers the convergence of *other* parameters as well, possibly making those equally unreliable. The only constraints that are good to use are the ones in the GALFIT main menu/template file or the "offset" constraints in the constraint file. For instance, it is fine to use the "offset" constraint to hold the centroids between two components (e.g. bulge & disk) fixed relative to each other. But, in general, boundary constraints are rather finicky to use. In my personal experience, I have never needed to use constraint files except to keep *relative centroids* fixed. So, users please be aware and inspect the results carefully.

8. *Use object masking.* One should always consider masking out objects that are not being fitted, especially if an area is too dusty or irregular in shape. For example, the jet in M87 will give GALFIT problems because, even though they are local features, the knots are much brighter than the diffuse underlying star light. Masking is especially important when fitting high order modes such as Fourier modes, because by definition higher order modes are sensitive to small perturbations. To create a mask, I have a program which can help. If one needs to mask a lot, please visit the "Frequently Asked Question" section on the GALFIT website or use SExtractor. On my website, I provide programs to expedite the process of creating masks.

   A better way to mask out a lot of objects (e.g. stars) is to use the SExtractor software, and choose the option to output a "segmentation image." The segmentation image is a mask of all the objects that are detected by SExtractor where, instead of fluxes, the pixel values correspond to unique ID numbers of objects. To convert the segmentation image into an object mask for GALFIT, all one has to do is to locate the object of interest to fit, and unmask it by replacing the object ID with zero values (using e.g. IRAF/imreplace).

9. *See a bright convolution/PSF region in the model?* That usually means one or more of the following: 1. the convolution box (*Item I*) is too small (enlarge it), 2. the convolution PSF image size (*Item D*) is too small (extract a larger PSF image size), or 3. the PSF has a sky pedestal that is not removed properly. Be sure not to confuse 2 with 3 – a PSF image that cuts off the wings may look like it has a sky pedestal, but confusing the difference will lead to bad consequences. To confirm visually that the sky is subtracted off completely, make sure first that the PSF image is large enough to include the wing, then multiply it by a large value; do not take for granted that "close to zero" means there is no sky (especially when the image ADU is in counts/second).

10. *Take a look at the individual subcomponents.* It is useful to scrutinize individual model subcomponents to make sure that they "make sense," instead of just looking at the parameter values. If the results do not make intuitive sense, often times looking at the images of subcomponents, individually, will reveal quite easily what GALFIT "sees" in the image. Use the following command to produce subcomponents: *galfit -o3 <filename>*. Alternatively, one can set parameter P to 3. See *Item P* in § 8.1.

11. *Why do the centroids reported by* GALFIT *not match up to object centers in an image?* If the centroid reported by GALFIT seems to be offset from the galaxy peak position, then the PSF image is not centroided correctly. The absolute position parameter depends on how well the convolution PSF is centered in the PSF image. Nominally, the PSF should be centered on pixel $N/2 + 1$ for even N number of pixels along a side, and $N/2+0.5$ for odd number of pixels.

12. *Bin down the data to save iterating time.* If a galaxy covering hundreds of pixels on a side is hard to fit, try reducing the fitting region or binning down the data image to something more manageable. Once a good fit has been obtained redo the fit on the original image/fitting region. This a handy trick for fitting many large galaxy images because it speeds up the analysis and allows one to explore a larger parameter space or combination of components.

13. *Be aware of PSF sampling and mismatch.* High contrast image analysis (e.g. quasar host decomposition) is one of the most difficult analysis to do. Often times PSF mismatch makes it difficult to reliably separate the central point source from the host galaxy. For decomposition to work well, the PSF must be over-sampled, and there has to be a good match between the PSF model and the data

PSF. There is no reliable solution if the mismatch is great. For situations where the PSF is undersampled, One should either generate an oversampled PSF somehow, or one should broaden the image out to Nyquist sampling before doing the analysis.

14. *How to tell if a source is unresolved.* If one wants to test whether a compact source is a true point source, one can convolve a narrow Gaussian with a PSF. However, make sure FWHM ≥ 0.5 pixels or else GALFIT may stop converging altogether. Usually it's a good idea to hold the Gaussian FWHM fixed to $= 0.5$ pixel. Once a solution is found, fit it again by allowing the FWHM to vary.

15. *Make sure the exposure times are what you think.* Make sure the image header has the correct exposure time parameter (EXPTIME) and value, so the initial guess for the flux is not several hundred times off.

16. *Watch out for Nuker parameters.* For the Nuker profile, make sure the parameters α, β, γ, do not have the same values – this can easily cause singular values, and force the program to quit. Initially, one's best bet for α and β parameters is somewhere between 0 and 3, and 0 to 1 for γ. To get a better feel for the behaviors of the parameters, take a look at Lauer et al. (1995).

I hope these tips will help when fitting very difficult cases. If the suggestions above do not work after all of this, one is more than welcome to email me (cyp@nrc-cnrc.gc.ca), and I'd be happy to give it a go to see what the problem might be.

REFERENCES

Elson, R. A. 1999, "Stellar Dynamics in Globular Clusters", in 10th Canary Islands Winter School of Astrophysics: Globular clusters, p. 209 - 248. Eds C. Martinez Roger, P. Fourńon, F. Sanchez. Cambridge Contemp., Cambridge University Press.

Krist, J. E., & Hook, R. N. 1997, in *HST* Calibration Workshop with a New Generation of Instruments, eds. S. Casertano, et al. (Baltimore: STScI), p. 192

Lauer, T. R., Ajhar, E. A., Byun, Y.-I., Dressler, A., Faber, S. M., Grillmair, C., Kormendy, J., Richstone, D., & Tremaine, S. 1995, AJ, 110, 2622

Pence, W. 1999, ASP Conf. Ser., Vol. 172, 487

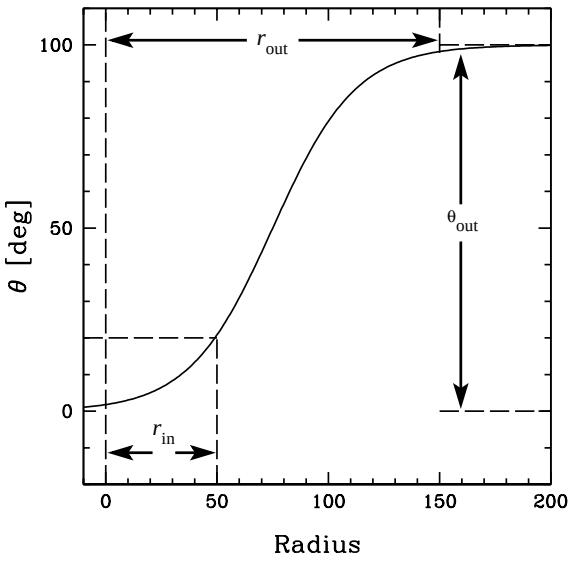
Peng, C. Y., Ho, L. C., Impey, C. D., & Rix, H.-W. 2002, AJ, 124, 266.

Peng, C. Y., Ho, L. C., Impey, C. D., & Rix, H.-W. 2010, AJ, 139, 2097.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 1997, Numerical Recipes in C (Cambridge: Cambridge Univ. Press)

```
# IMAGE PARAMETERS
A) gal.fits            # Input data image (FITS file)
B) imgblock.fits       # Output data image block
C) none                # Sigma image name (made from data if blank or "none")
D) psf.fits   #        # Input PSF image and (optional) diffusion kernel
E) 1                   # PSF fine sampling factor relative to data
F) none                # Bad pixel mask (FITS image or ASCII coord list)
G) none                # File with parameter constraints (ASCII file)
H) 1    93   1    93   # Image region to fit (xmin xmax ymin ymax)
I) 100    100          # Size of the convolution box (x y)
J) 26.563              # Magnitude photometric zeropoint
K) 0.038  0.038        # Plate scale (dx dy)   [arcsec per pixel]
O) both                # Display type (regular, curses, both)
P) 0                   # Options: 0=normal run; 1,2=make model/imgblock & quit

# INITIAL FITTING PARAMETERS
#
#   For object type, the allowed functions are:
#       nuker, sersic, expdisk, devauc, king, psf, gaussian, moffat,
#       ferrer, and sky.
#
#   Hidden parameters will only appear when they're specified:
#       C0 (diskyness/boxyness),
#       Fn (n=integer, Azimuthal Fourier Modes).
#       R0-R10 (PA rotation, for creating spiral structures).
#
# ------------------------------------------------------------------------------
#   par)    par value(s)    fit toggle(s)    # parameter description
# ------------------------------------------------------------------------------

# Object number: 1 -- A TRUE point source
 0) psf                    #    Object type
 1) 50.00  50.00  1 1      #  position x, y
 3) 18.000     1           #  total magnitude
 Z) 0                      #  Output option (0 = residual, 1 = Don't subtract)

# Object number: 2
 0) sersic                 #  object type
 1) 48.5180  51.2800  1 1  #  position x, y
 3) 20.0890     1          #  integrated magnitude
 4) 5.1160      1          #  R_e (half-light radius)   [pix]
 5) 4.2490      1          #  Sersic index n (de Vaucouleurs n=4)
 9) 0.7570      1          #  axis ratio (b/a)
10) -60.3690    1          #  position angle (PA) [deg: Up=0, Left=90]
F1) 0.0001   0.0000   1 1  #  azim. Fourier mode 1, amplitude, & phase angle
F3) 0.0001   0.0000   1 1  #  azim. Fourier mode 3, amplitude, & phase angle
 Z) 0                      #  output option (0 = resid., 1 = Don't subtract)

# Object number: 3
 0) expdisk                #  object type
 1) 48.5180  51.2800  1 1  #  position x, y
 3) 20.0890     1          #  integrated magnitude
 4) 5.1160      1          #  R_s (disk scale-length)   [pix]
 9) 0.7570      1          #  axis ratio (b/a)
10) -60.3690    1          #  position angle (PA) [deg: Up=0, Left=90]
C0) -0.05       0          #  diskyness(-)/boxyness(+)
 Z) 0                      #  output option (0 = resid., 1 = Don't subtract)

# Object number: 4
 0) nuker                  #  object type
 1) 48.5180  51.2800  1 1  #  position x, y
 3) 20.0890     1          #     mu(Rb)   [mag/arcsec^2]
 4) 5.1160      1          #      Rb      [pix]
 5) 4.2490      1          #     alpha
 6) 1.1000      1          #      beta
 7) 0.3000      1          #     gamma
 9) 0.7570      1          #  axis ratio (b/a)
10) -60.3690    1          #  position angle (PA) [deg: Up=0, Left=90]
 Z) 0                      #  output option (0 = resid., 1 = Don't subtract)

# Object number: 5
 0) sky                    #  object type
 1) 1.3920      1          #  sky background at center of fitting region [ADUs]
 2) 0.0000      0          #  dsky/dx (sky gradient in x)
 3) 0.0000      0          #  dsky/dy (sky gradient in y)
 Z) 0                      #  output option (0 = resid., 1 = Don't subtract)
```

FIG. 17.— Example of an input file. The object list is dynamic and can be extended as needed.



APPENDIX

A — HYPERBOLIC TANGENT ROTATION FUNCTION

The hyperbolic tangent ($\tanh(r_{\rm in}, r_{\rm out}, \theta_{\rm incl}, \theta_{\rm PA}^{\rm sky}; r)$) portion of the α-tanh (Eq. 27) and pow-tanh (Eq. 28) rotation function is given by Equation 5 below. The constant CDEF is defined such that the rotation angle at the mathematical "bar radius" $r_{\rm in}$, the rotation angle reaches 20 degrees. This definition is entirely empirical. The above Figure shows a pure tanh rotation function, where the rotation angle reaches a maximum $\theta_{\rm out}$ near $r = r_{\rm out}$. Beyond $r_{\rm out}$, the rotation angle peaks out at $\theta_{\rm out}$. This function is multiplied with either a logarithmic or a powerlaw function to produce the desired asymptotic rotation behavior seen in more realistic galaxies (see Section 5).

$$\mathrm{CDEF} = 0.23 \qquad (\text{constant for "bar}'' \text{ definition}) \tag{1}$$

$$A = \frac{2 \times \mathrm{CDEF}}{|\theta_{\rm out}| + \mathrm{CDEF}} - 1.00001 \tag{2}$$

$$B = \left(2 - \tanh^{-1}(A)\right) \left(\frac{r_{\rm out}}{r_{\rm out} - r_{\rm in}}\right) \tag{3}$$

$$r = \sqrt{\Delta x^2 + \Delta y^2} \qquad (\mathrm{circular - centric\ distance}) \tag{4}$$

$$\tanh(r_{\rm in}, r_{\rm out}, \theta_{\rm incl}, \theta_{\rm PA}^{\rm sky}; r) \equiv 0.5 \times \left(\tanh\ \left[B\left(\frac{r}{r_{\rm out}} - 1\right) + 2\right] + 1\right) \tag{5}$$

B — HYPERBOLIC TANGENT TRUNCATION FUNCTION

The hyperbolic tangent truncation function ($\tanh(x_c, y_c; r_{\rm break}, r_{\rm soft}, q, \theta_{PA})$) (see Section 6) is very similar to the coordinate rotation formulation in Appendix A, except for different constants that define the flux ratio at the truncation radii: at $r = r_{\rm break}$ the flux is 99% of the untruncated model profile, whereas at $r = r_{\rm soft}$ the flux is 1%. With this definition, Equation 7 is the truncation function:

$$B = 2.65 - 4.98 \left(\frac{r_{\rm break}}{r_{\rm break} - r_{\rm soft}}\right) \tag{6}$$

$$P \equiv \tanh(x_c, y_c; r_{\rm break}, r_{\rm soft}, q, \theta_{\rm PA}) \equiv 0.5 \times \left(\tanh\ \left[(2 - B)\frac{r}{r_{\rm break}} + B\right] + 1\right) \tag{7}$$

Note that the radius r is a generalized radius (as opposed to a circular-centric distance), i.e. one which is perturbed by C_0, bending modes, or Fourier modes, of the truncation function. When softening length ($\Delta r_{\rm soft}$) is used a free parameter, it is defined as $\Delta r_{\rm soft} = r_{\rm break} - r_{\rm soft}$.