Imfit: A New 2D Galaxy Decomposition Code

Peter Erwin

Max-Planck-Institute for Extraterrestrial Physics

Brief Outline

1. A new code for fitting images of galaxies
2. Biases in fitting low-S/N galaxy images

Imfit: (Yet Another) 2D Image-Fitting Program

- Inspirations: GIM2D, BUDDA, GALFIT
- Modular, object-oriented design using standard C++
- Uses well-tested open-source code for:
    - Image I/O (CFITSIO — heasarc.gsfc.nasa.gov/fitsio/)
    - FFT-based convolution (FFTW — www.fftw.org)
    - Minimization algorithms
- Special focus on making it easy to add new functions
- Multi-threaded for speed (via OpenMP and FFTW)
- Open source release (GPL)

How to Use It

```
$ imfit --config <config-file> <image-file-to-fit> [options]
```

Examples:

Simplest possible use:

```
$ imfit --config sersic+exp.dat ngc1023.fits
```

Specify an image subsection to fit (& can use compressed images):

```
$ imfit --config sersic+exp.dat ngc1023.fits.gz[150:300,406:800]
```

Specify a mask and a PSF image (to convolve with):

```
$ imfit --config sersic+exp.dat ngc1023.fits --mask=ngc1023_mask.fits
--psf=psf_for_n1023.fits
```

Specify names for output files (best-fit parameters, model image, residual image)

```
$ imfit --config sersic+exp.dat ngc1023.fits --save-params
best_fit.dat --save-model best_fit.fits --save-residual resid.fits
```

Speed

Single-threaded mode: typically ~ 50% faster than GALFIT 3 (using χ^2 minimization with L-M algorithm for both)

BUT:

- Multithreaded image computation (using OpenMP)
- Multithreaded PSF convolution (FFTW library)

Typical speedup is ~3–4 × single-thread speed for 4-core CPU

Speed

Single-threaded mode: typically ~ 50% faster than GALFIT 3 (using χ^2 minimization with L-M algorithm for both)

BUT:

Multithreaded image computation (using OpenMP)

Multithreaded PSF convolution (FFTW library)

Typical speedup is ~3–4 × single-thread speed for 4-core CPU

	GALFIT	Imfit: Single-thread	Imfit: Multi-thread
Sérisc + exp. fit (840 x 870 image)	44s	28s	9s
Same, w/ PSF convolution	125s	86s	30s

Timings on MacBook Pro (2011): 2.3 GHz Core i7 (quad-core)

Components of a Model:

Functions and Function Blocks

Model = one or more *function blocks*, each with one or more *functions*
function = generates 2D surface-brightness distribution

All functions in function block share same (x,y) location in image

Multiple function blocks — e.g. multiple galaxies, or off-center components within a single galaxy (e.g. off-center bar or stellar nucleus)

As many function blocks as you like (though the fit will take longer!)

Parameter Constraints

Individual parameters can be held *fixed*, or restricted to lie within upper and lower values

Indicated by (optional) extra notes on the same line as the parameter's initial value

Example of Configuration File

```
# A function block with one function (Sersic); all parameters
# are unconstrained
X0   129.0
Y0   129.0
FUNCTION Sersic
PA    18.0
ell    0.2
n      1.5
I_e    15
r_e    25
```

Example of Configuration File

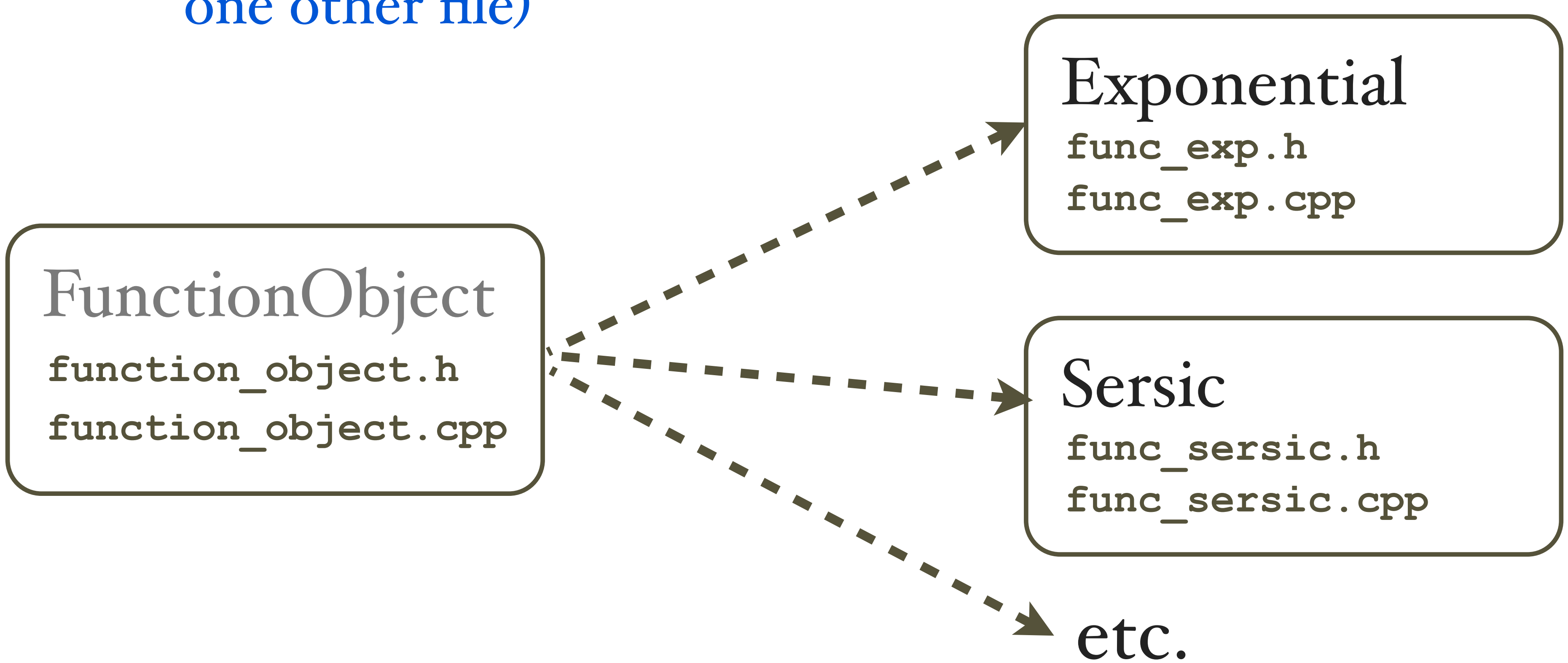
```
# A function block with one function (Sersic); all parameters
# are unconstrained
X0   129.0
Y0   129.0
FUNCTION Sersic
PA    18.0
ell    0.2
n      1.5
I_e    15
r_e    25


# Another function block with two functions (Sersic & Exp.)
# (Note that parameter values for these functions have limits)
X0   231.0    230,240     # x-value: lower limit, upper limit
Y0   307.0    300,310
FUNCTION Sersic
PA    18.0    0,90
ell    0.1    0,1
n      4.0    fixed       # this parameter's value is constant
I_e    100    0,500
r_e    30     0,100
FUNCTION Exponential
PA    18.0    0,90
ell    0.5    0.3,0.8
I_0    15     0,500
h      25                 # no limits for scale length
```

Function Objects

FunctionObject = abstract base class

All 2D image functions are derived classes of FunctionObject

- Returns pixel intensity value, given pixel coordinates and current parameters for that function
- Handles pixel subsampling, geometric transforms as needed
- Rest of program doesn't need to know details of how they work
- Easy to write new functions: only 2 files (plus minor modification of one other file)



Some of the Built-in Functions

Gaussian

Moffat

Sérsic w/ generalized ellipse (boxy/disk isophote shape)

Exponential w/ generalized ellipse (boxy/disk isophote shape)

Some of the Built-in Functions

Gaussian

Moffat

Sérsic w/ generalized ellipse (boxy/disk isophote shape)

Exponential w/ generalized ellipse (boxy/disk isophote shape)

Broken Exponential (two slopes; Erwin+2008)

— disk breaks/truncations, antitruncations

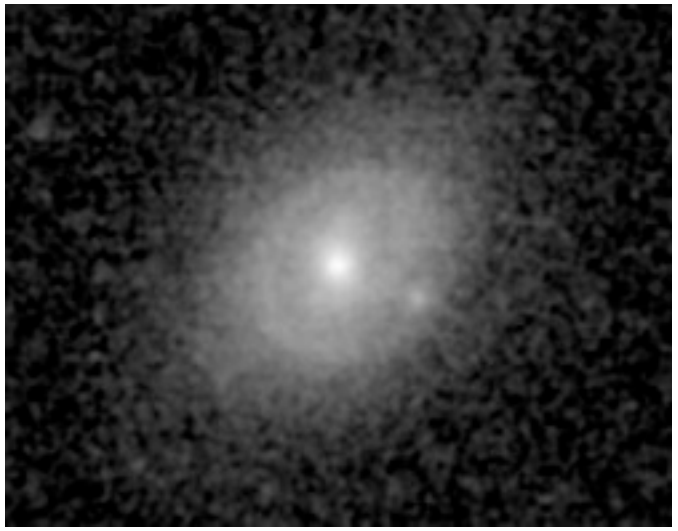
Elliptical Ring (Gaussian profile; symmetricor 2-sided)

Analytic Edge-on Disk

Edge-on Ring (symmetric & 2-sided)

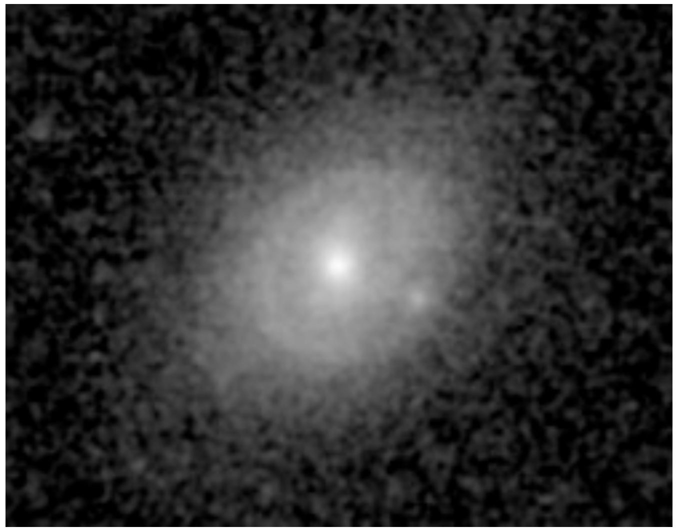
Elliptical Core-Sérsic (Graham+2003; Trujillo+2004)

Example: Barred, Ringed Galaxy at z=0.03



Narrow-band (*R*) continuum image

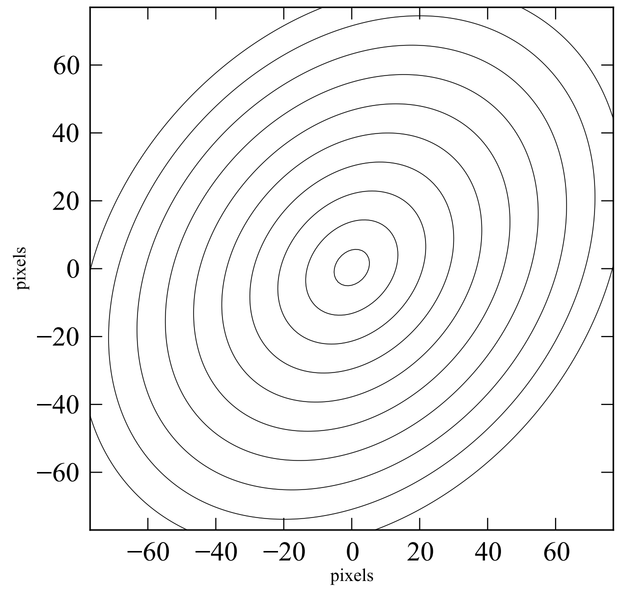
Example: Barred, Ringed Galaxy at z=0.03



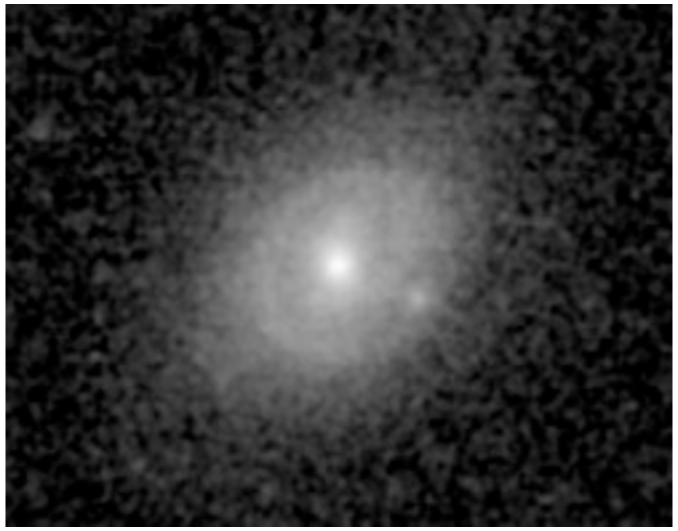
Narrow-band (*R*) continuum image

−

Sérsic fit



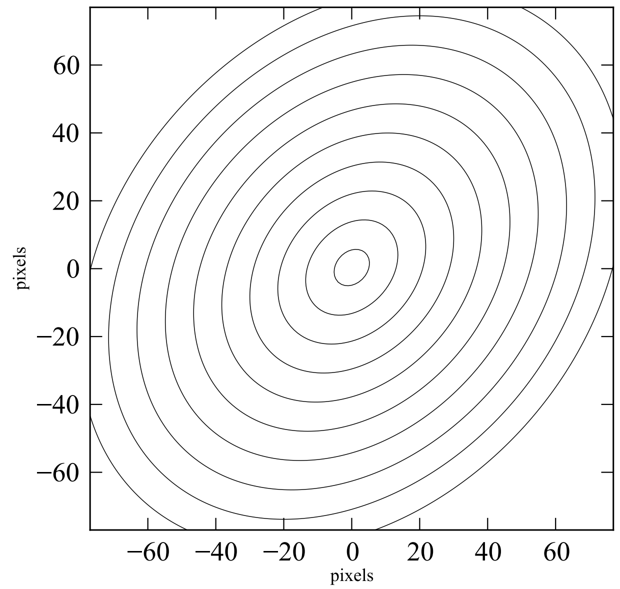
Example: Barred, Ringed Galaxy at z=0.03



Narrow-band (*R*) continuum image

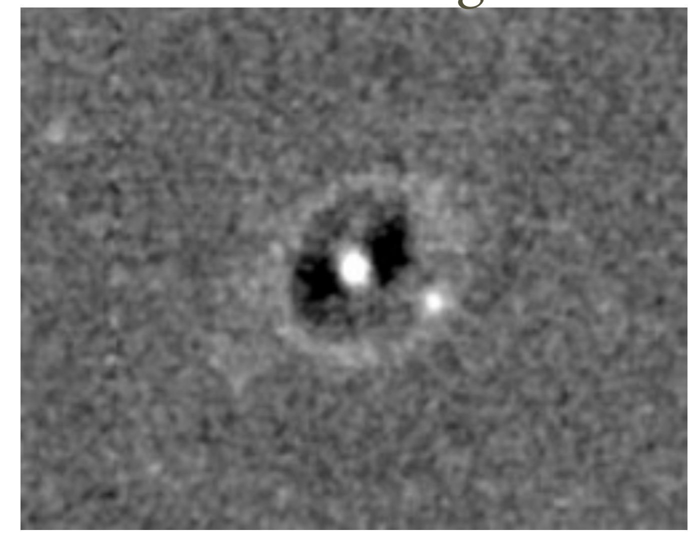
−

Sérsic fit

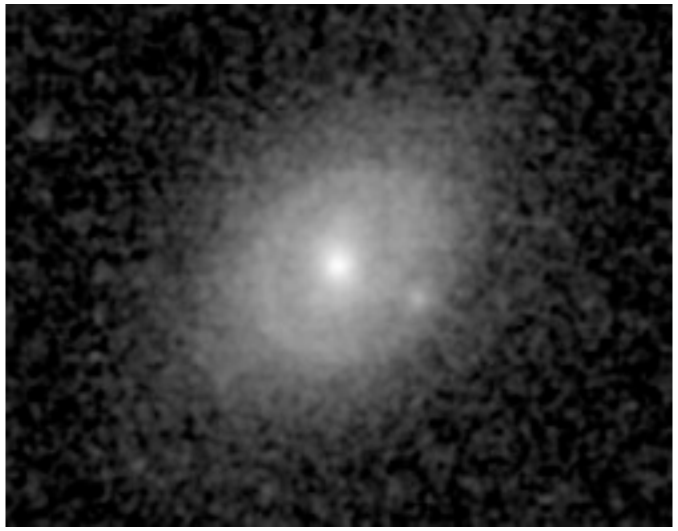


=

Residual Image



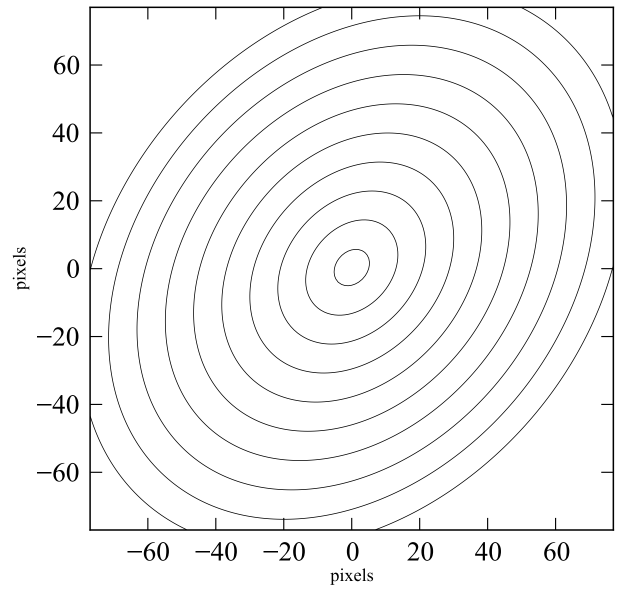
Example: Barred, Ringed Galaxy at z=0.03



Narrow-band (*R*) continuum image

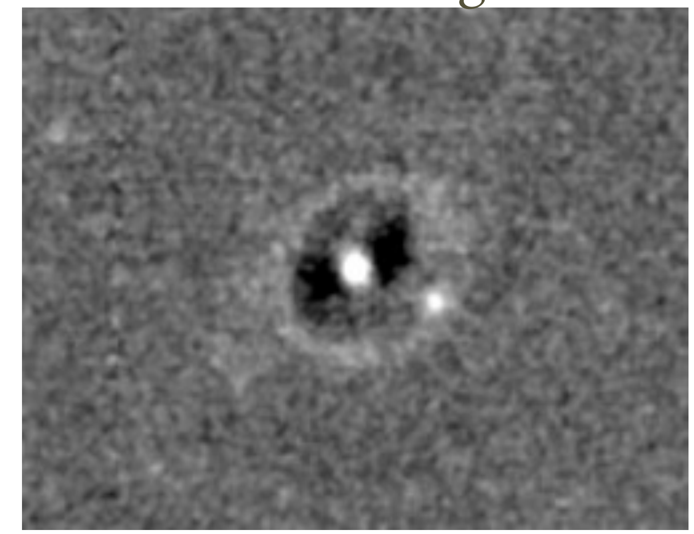
−

Sérsic fit

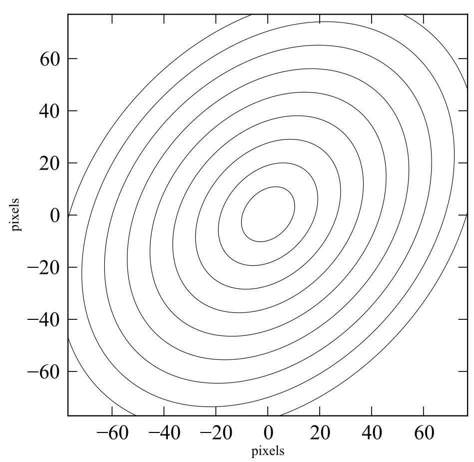


=

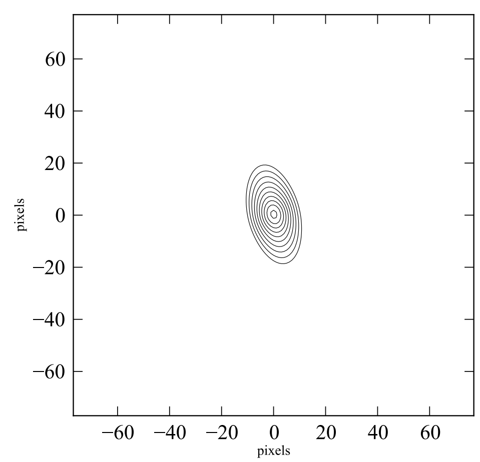
Residual Image



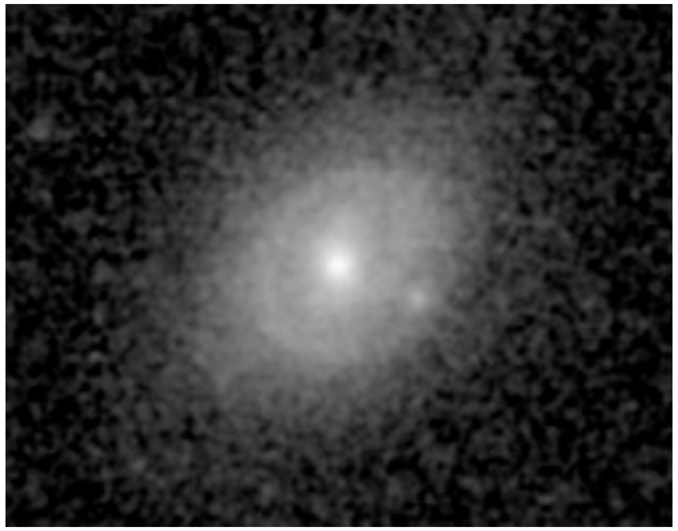
Exponential + Sérsic (bar) fit



+



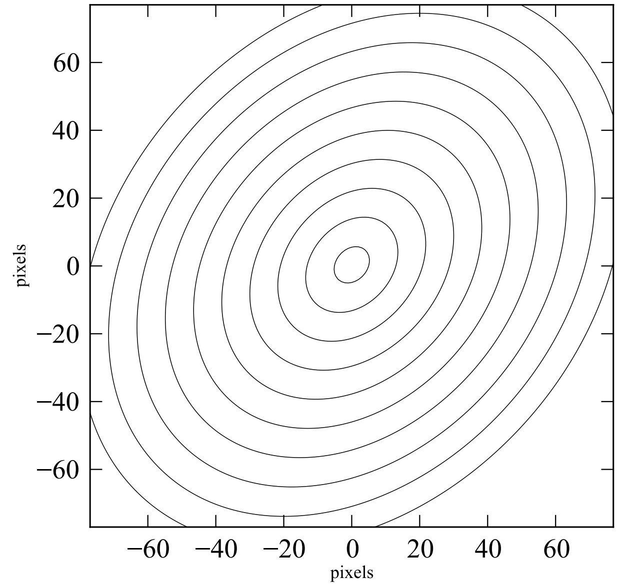
Example: Barred, Ringed Galaxy at z=0.03



Narrow-band (*R*) continuum image

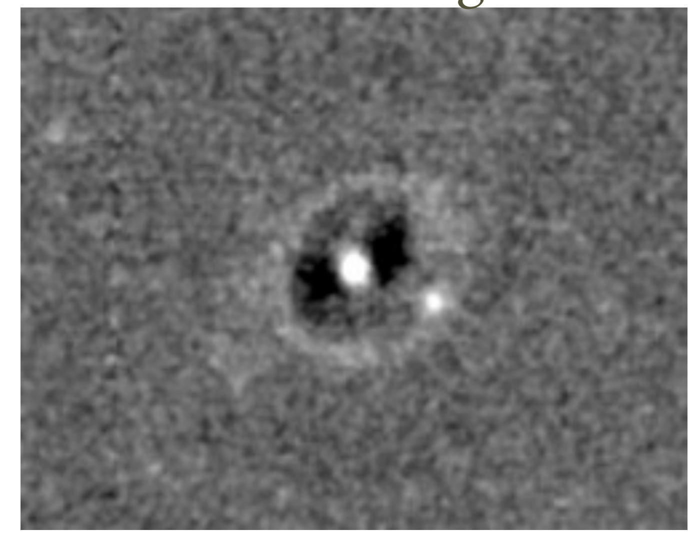
−

Sérsic fit

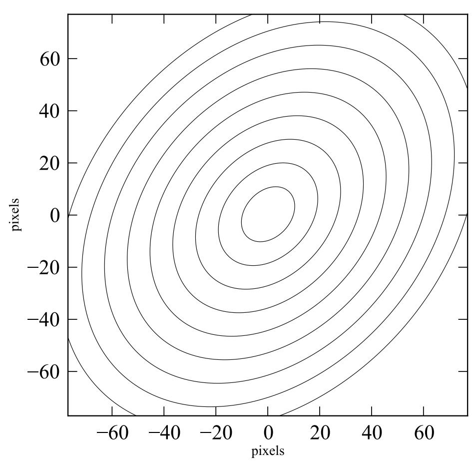


=

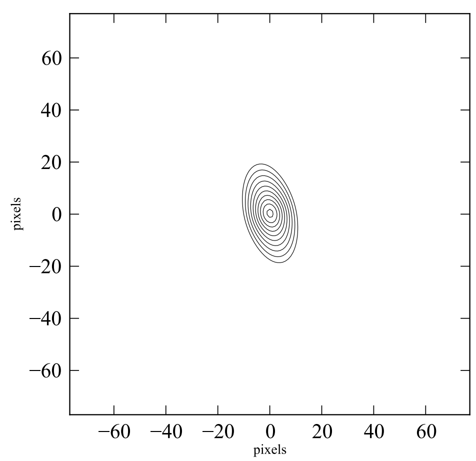
Residual Image



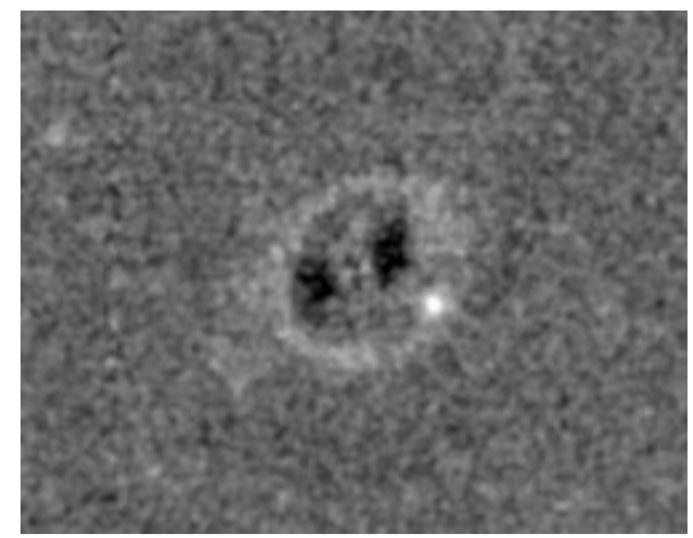
Exponential + Sérsic (bar) fit



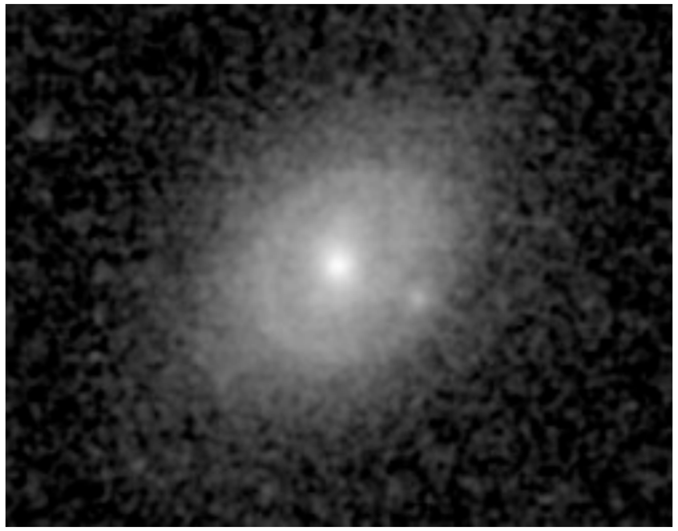
+



=



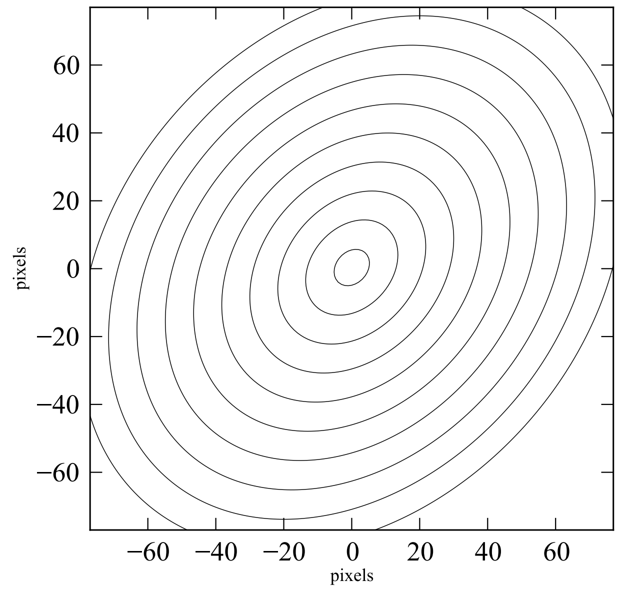
Example: Barred, Ringed Galaxy at z=0.03



Narrow-band (*R*) continuum image

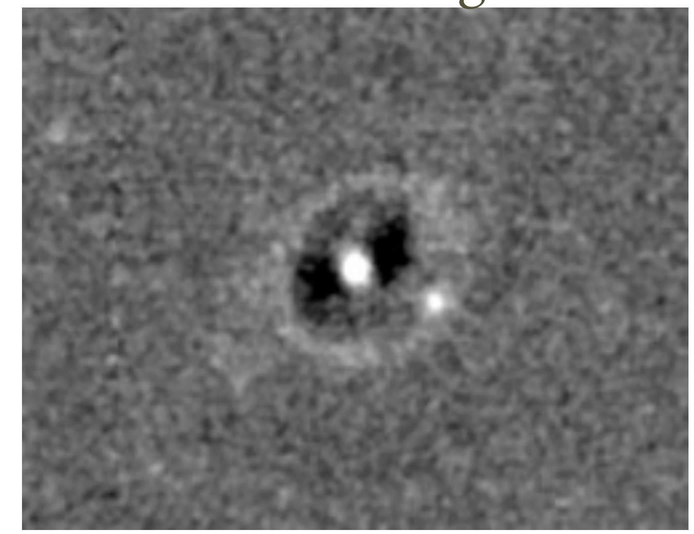
−

Sérsic fit

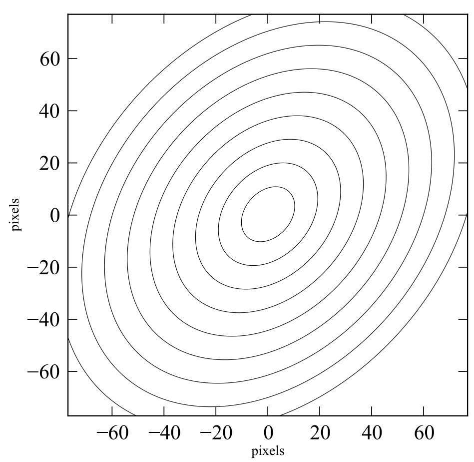


=

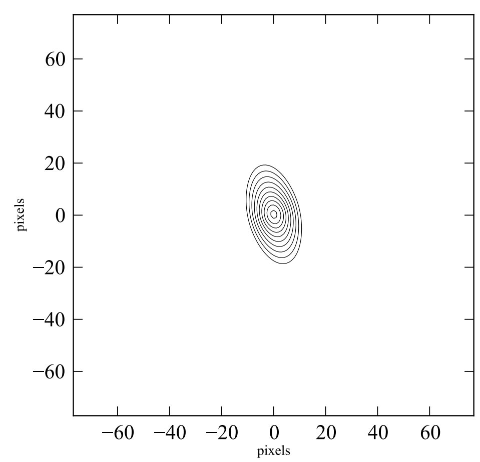
Residual Image



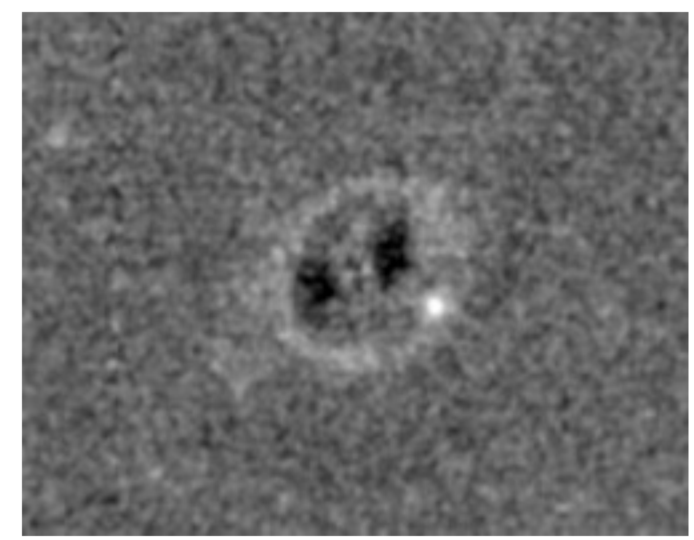
Exponential + Sérsic (bar) fit



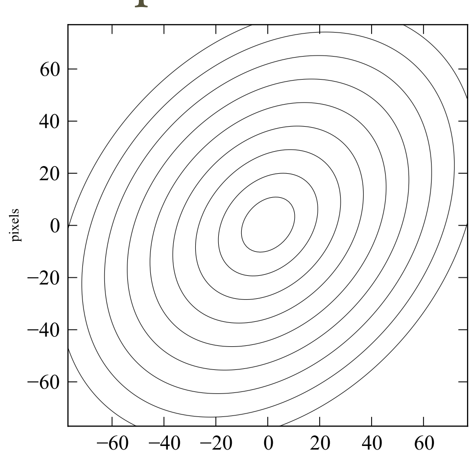
+



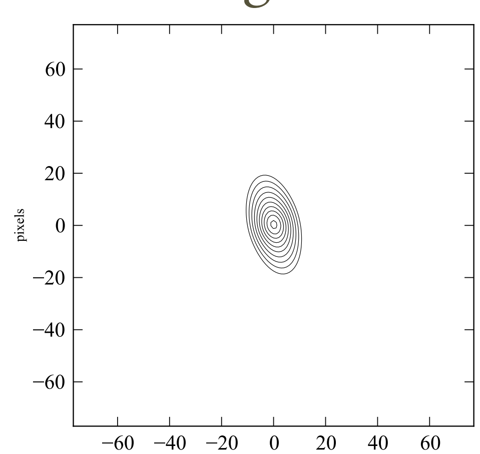
=



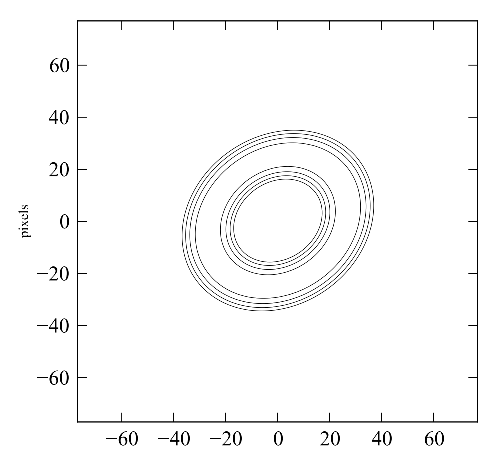
Exponential + Sérsic (bar) + Ring fit



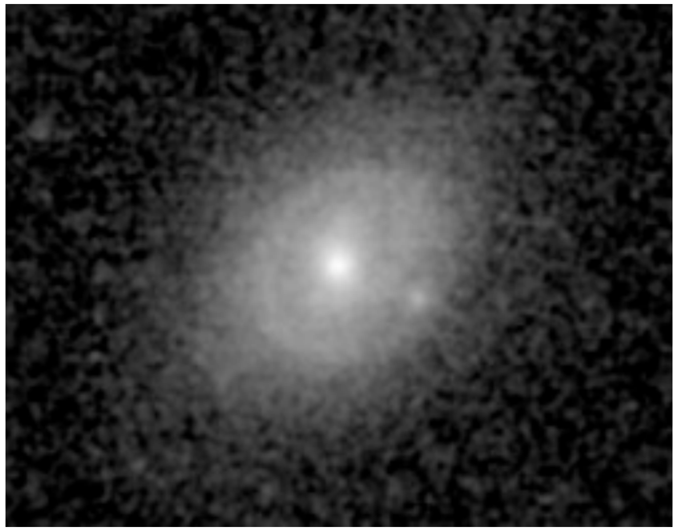
+



+



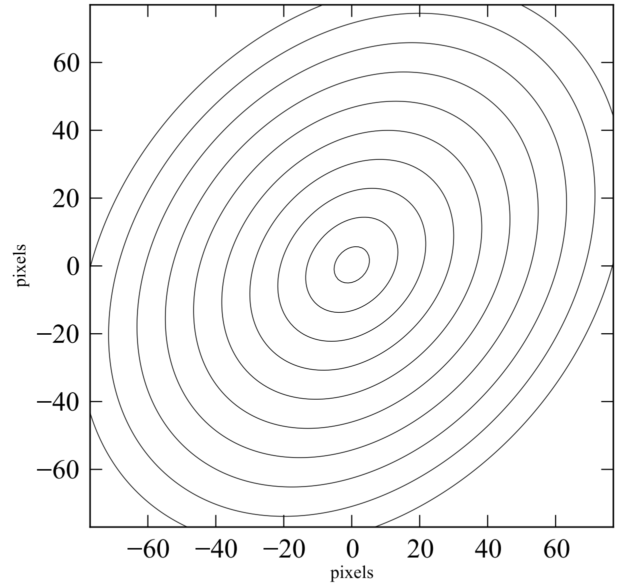
Example: Barred, Ringed Galaxy at z=0.03



Narrow-band (*R*) continuum image

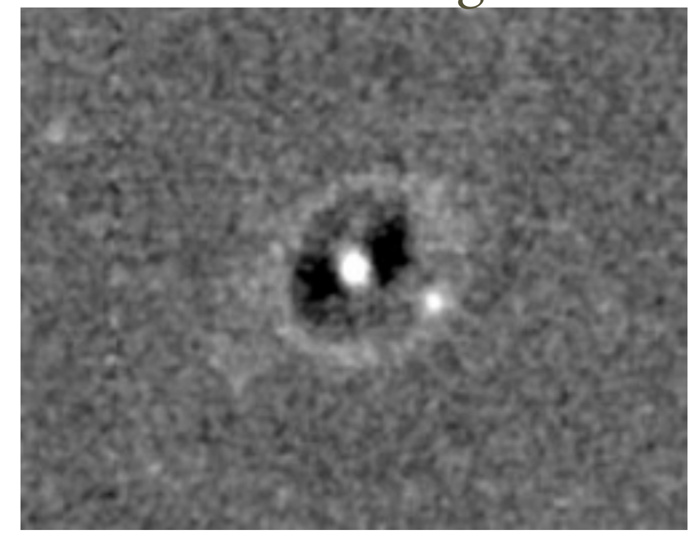
−

Sérsic fit

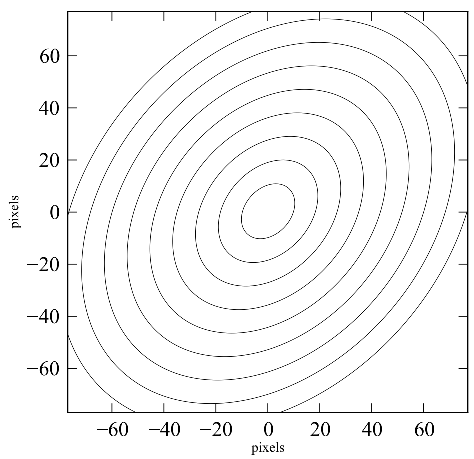


=

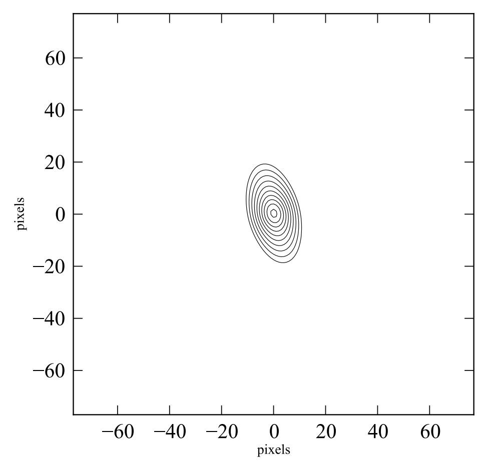
Residual Image



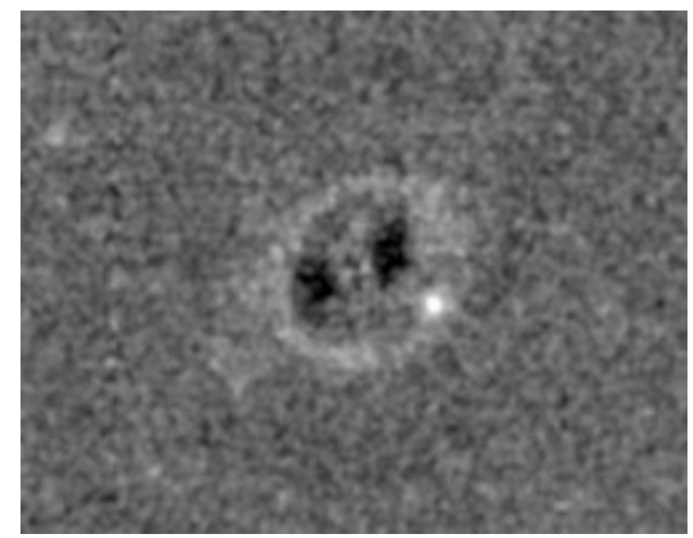
Exponential + Sérsic (bar) fit



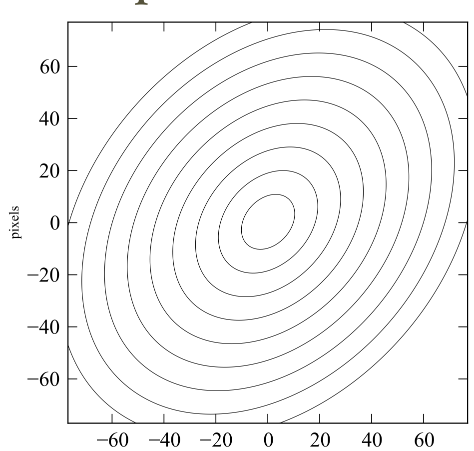
+



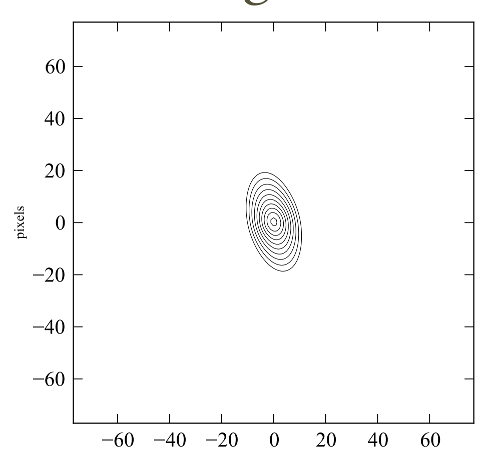
=



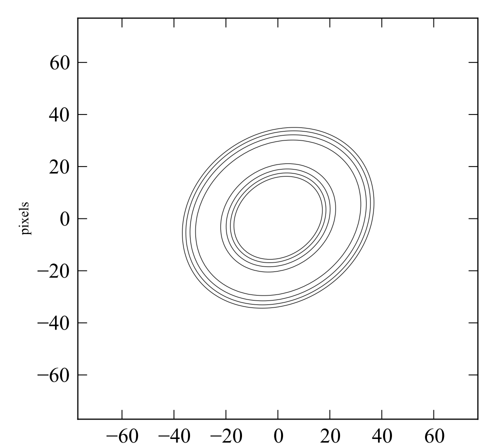
Exponential + Sérsic (bar) + Ring fit



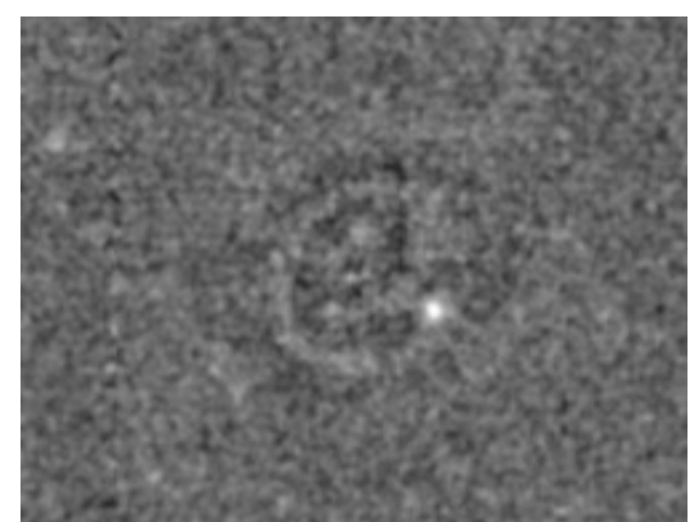
+



+



=



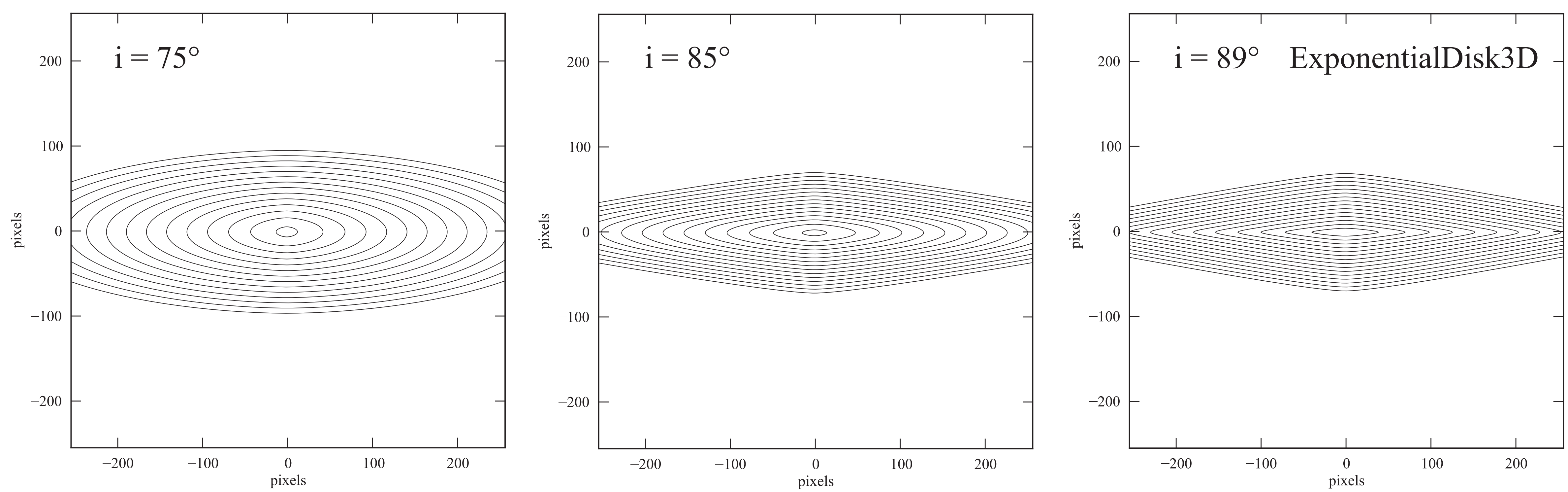
3D Integration Functions

Each pixel = Line-of-sight integration through
3D luminosity-density model

Axisymmetric disk at arbitrary inclination: radial exponential + generalized $\text{sech}^{2/n}$ vertical profile (van der Kruit 1988)

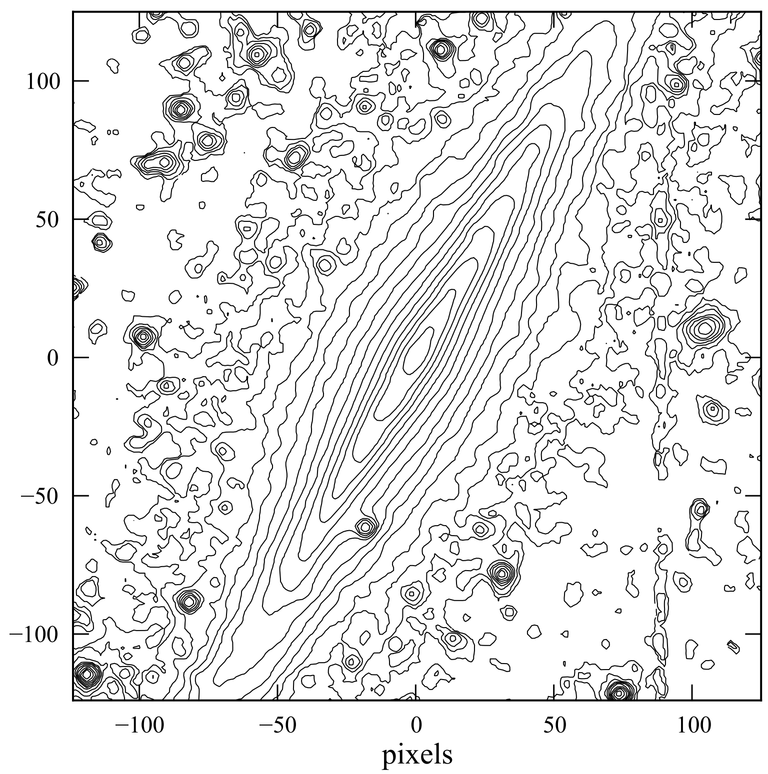
Axisymmetric broken-exponential disk

Elliptical Gaussian ring (with exponential vertical profile)



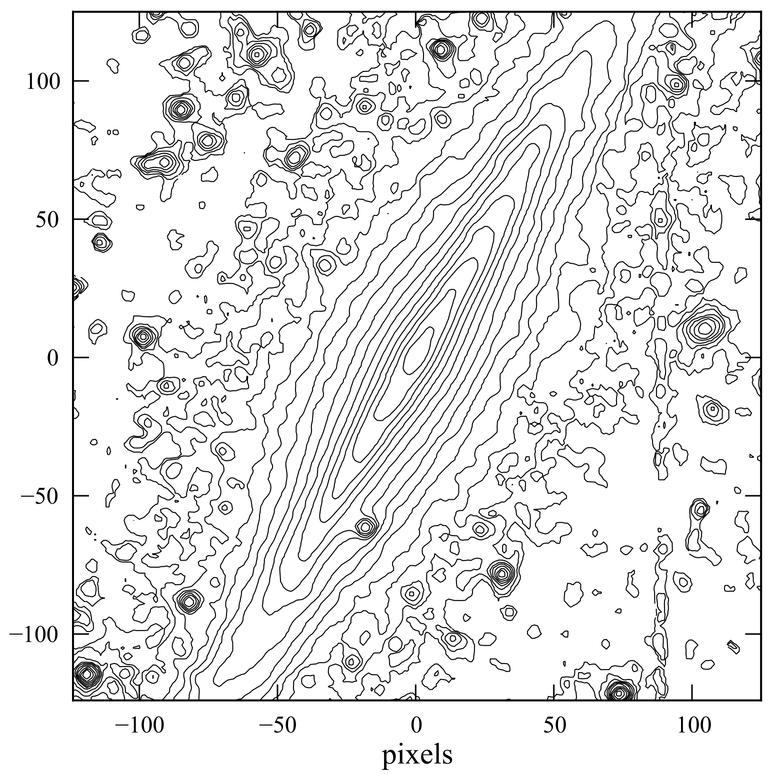
IC 5176: Edge-on Thin + Thick Disk

Spitzer IRAC1 (3.6μm)

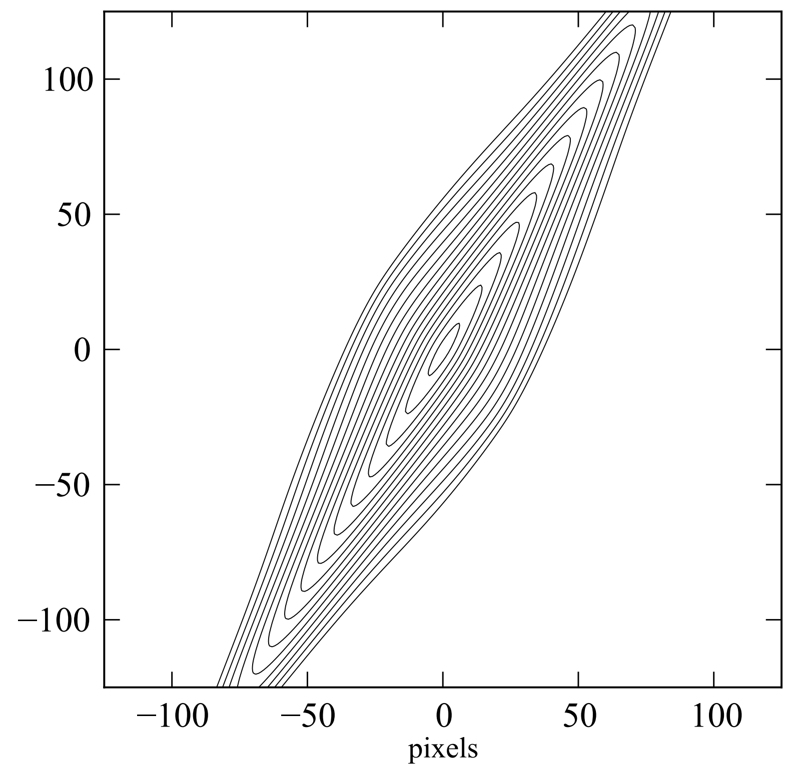


IC 5176: Edge-on Thin + Thick Disk

Spitzer IRAC1 (3.6μm)

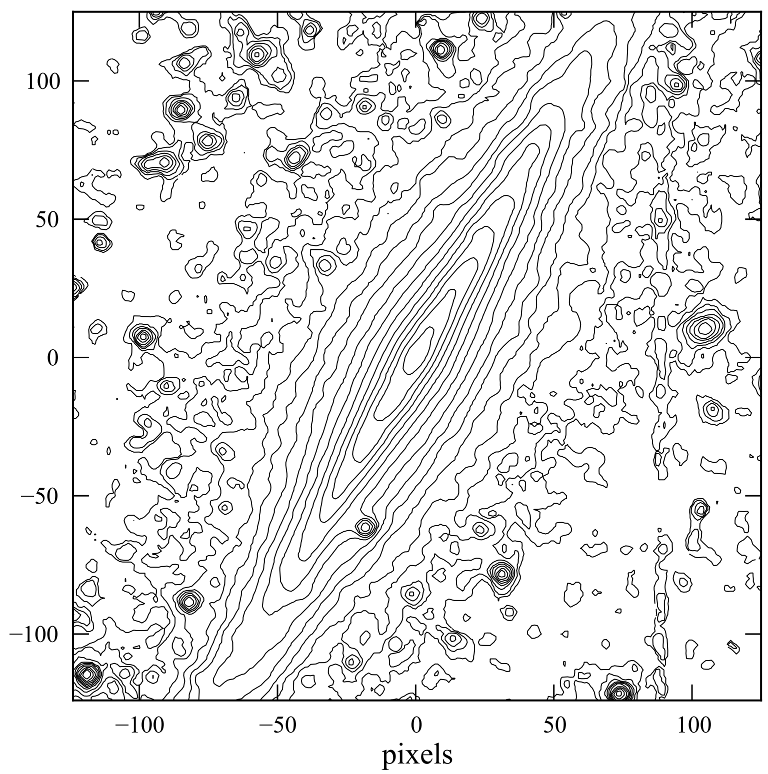


3D Thin disk

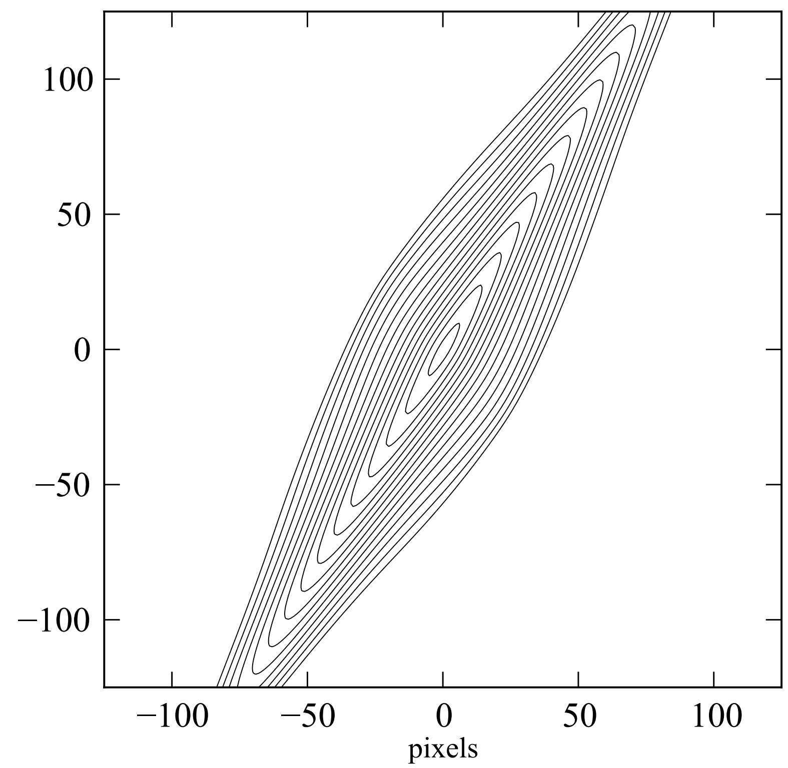


IC 5176: Edge-on Thin + Thick Disk

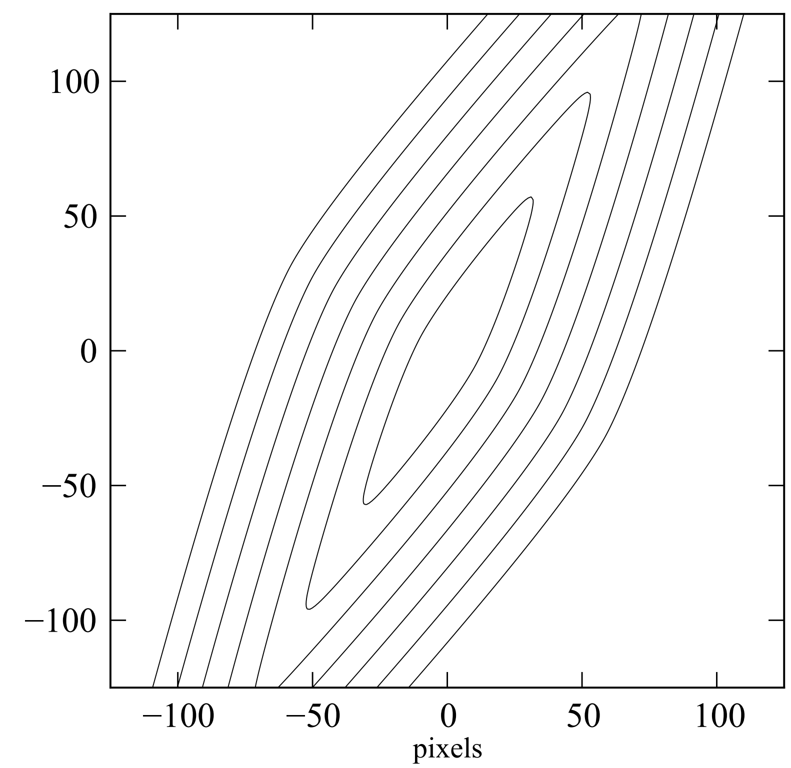
Spitzer IRAC1 (3.6μm)



3D Thin disk

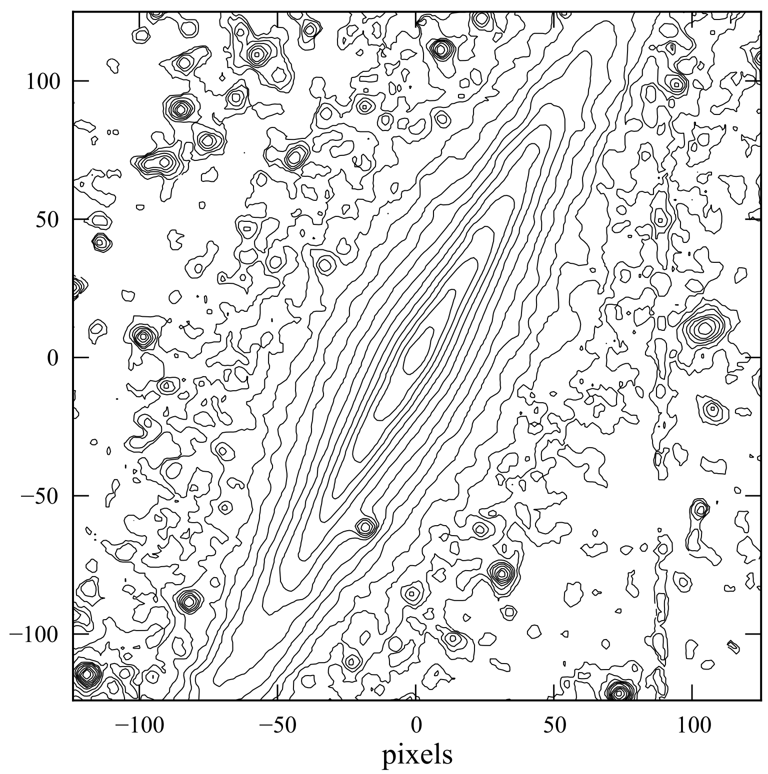


3D Thick disk

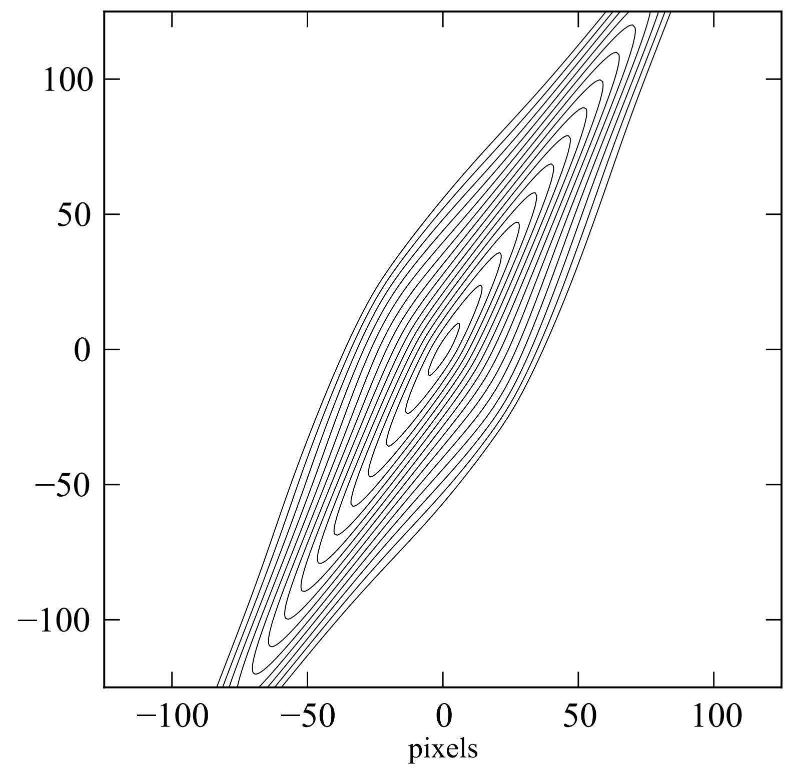


IC 5176: Edge-on Thin + Thick Disk

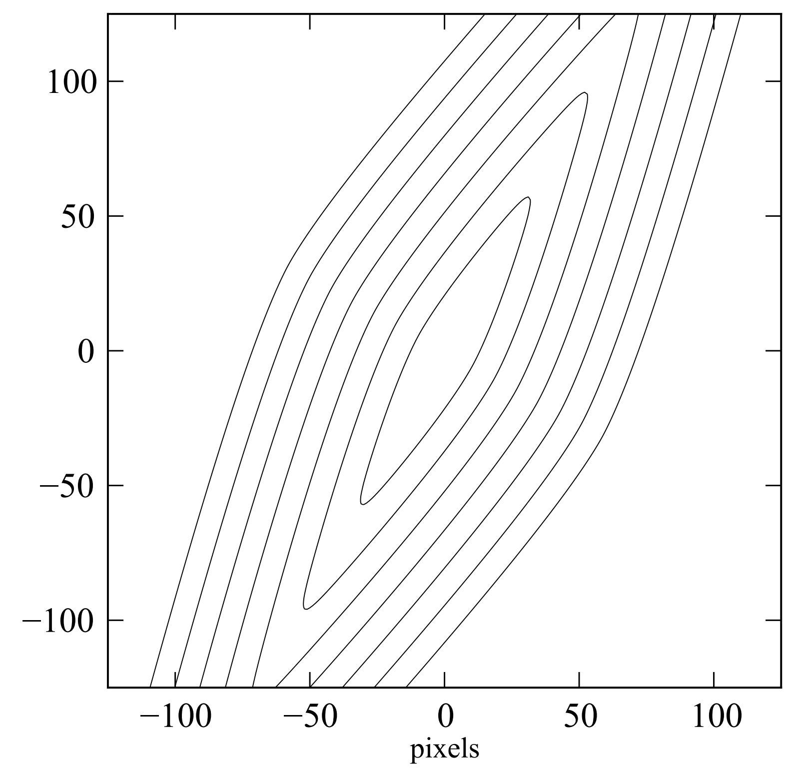
Spitzer IRAC1 (3.6μm)



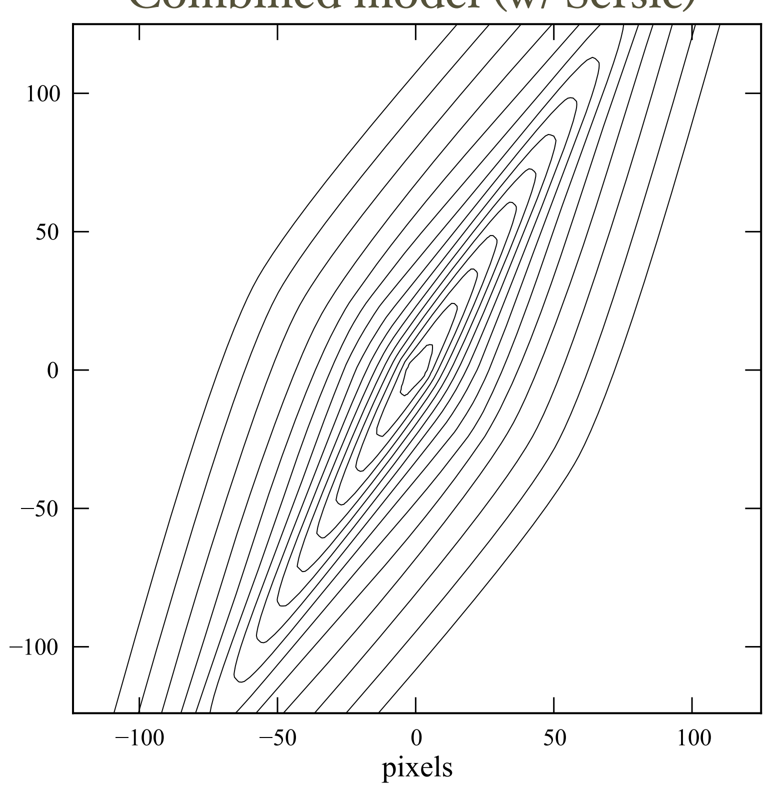
3D Thin disk



3D Thick disk

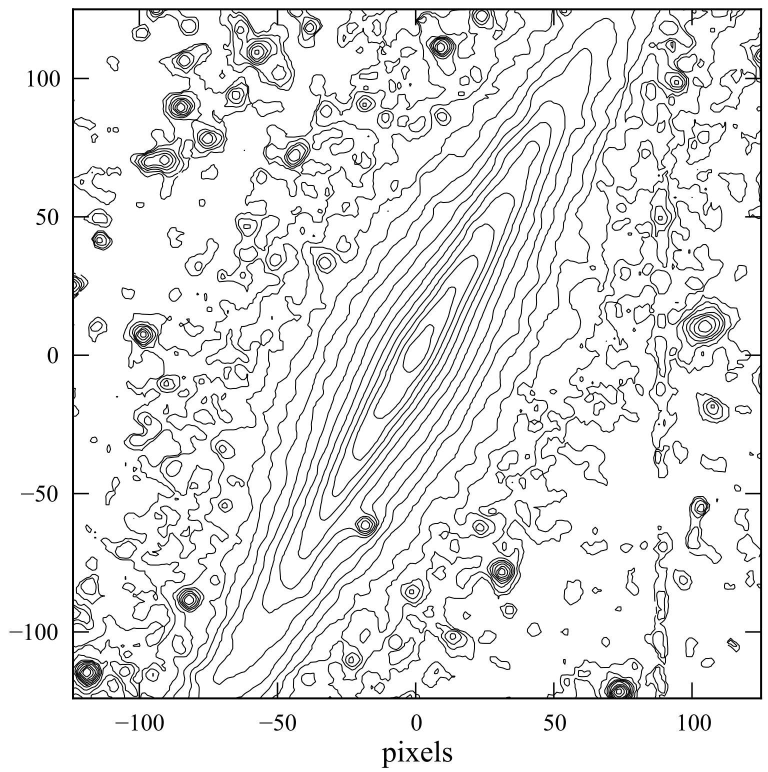


Combined model (w/ Sérsic)

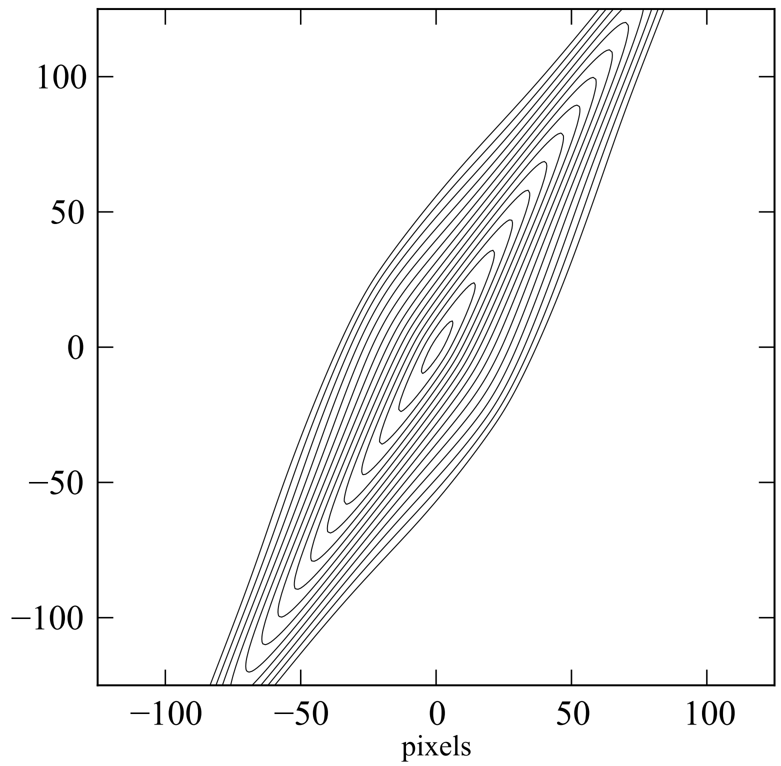


IC 5176: Edge-on Thin + Thick Disk

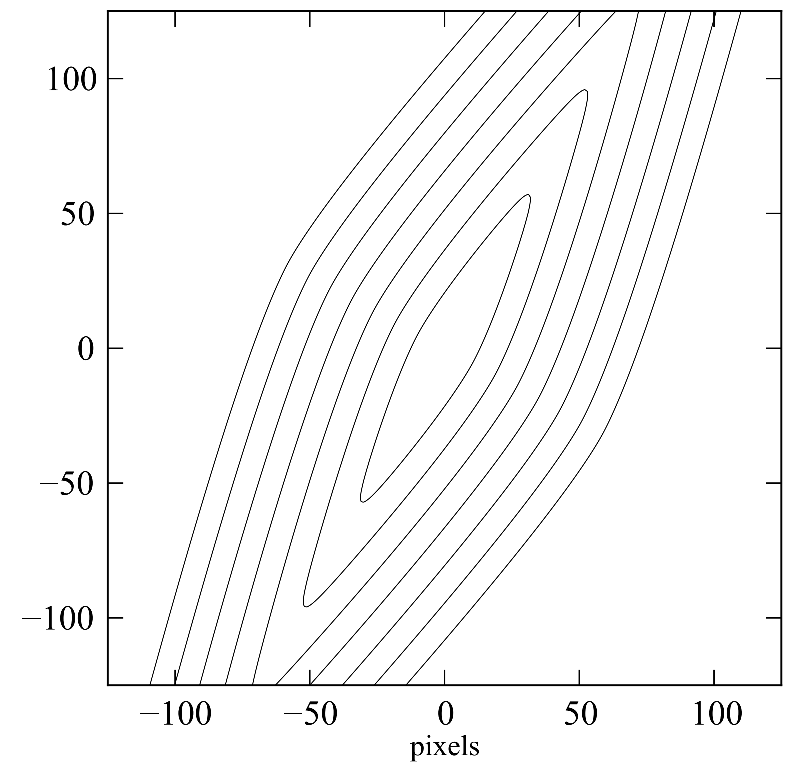
Spitzer IRAC1 (3.6μm)



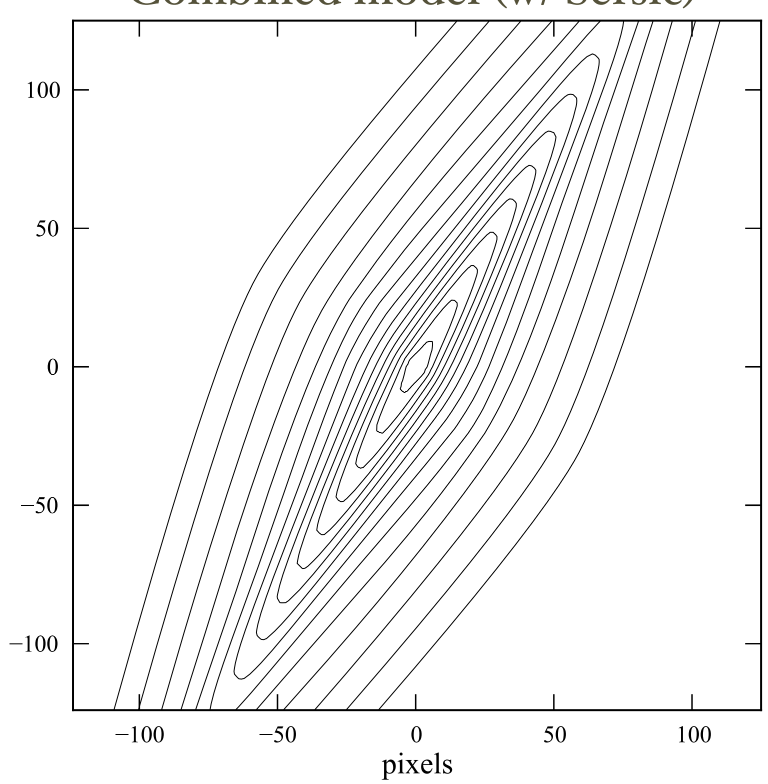
3D Thin disk



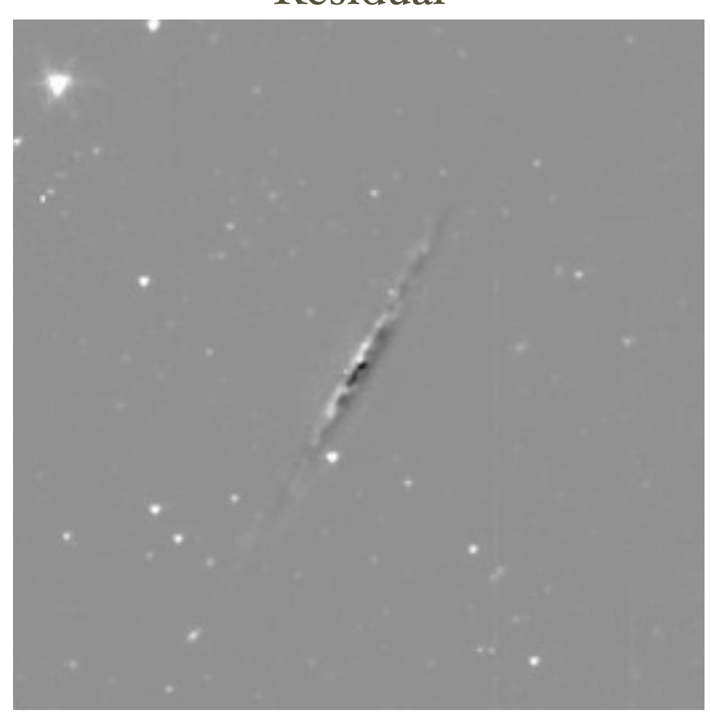
3D Thick disk



Combined model (w/ Sérsic)

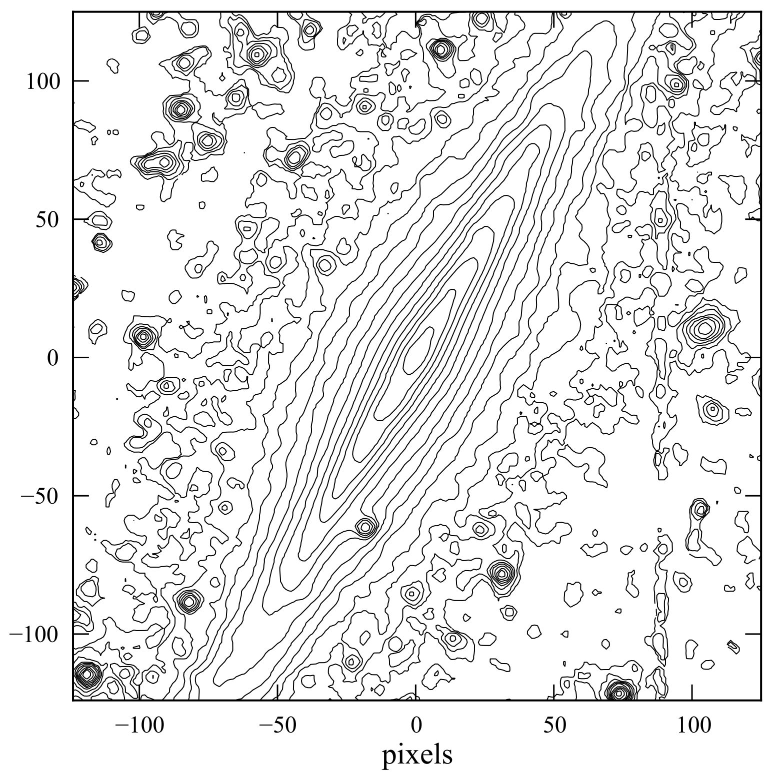


Residual

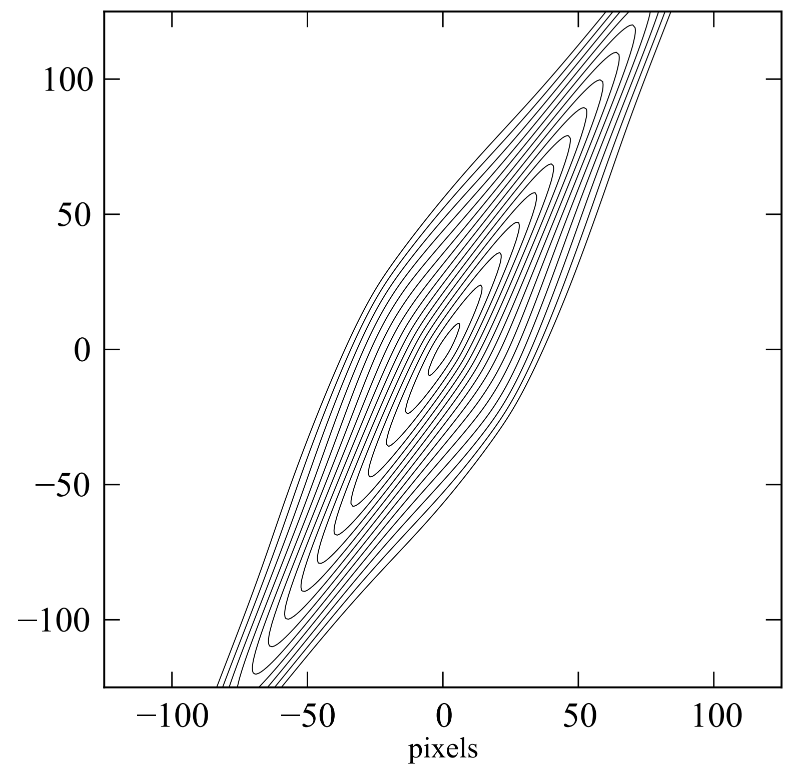


IC 5176: Edge-on Thin + Thick Disk

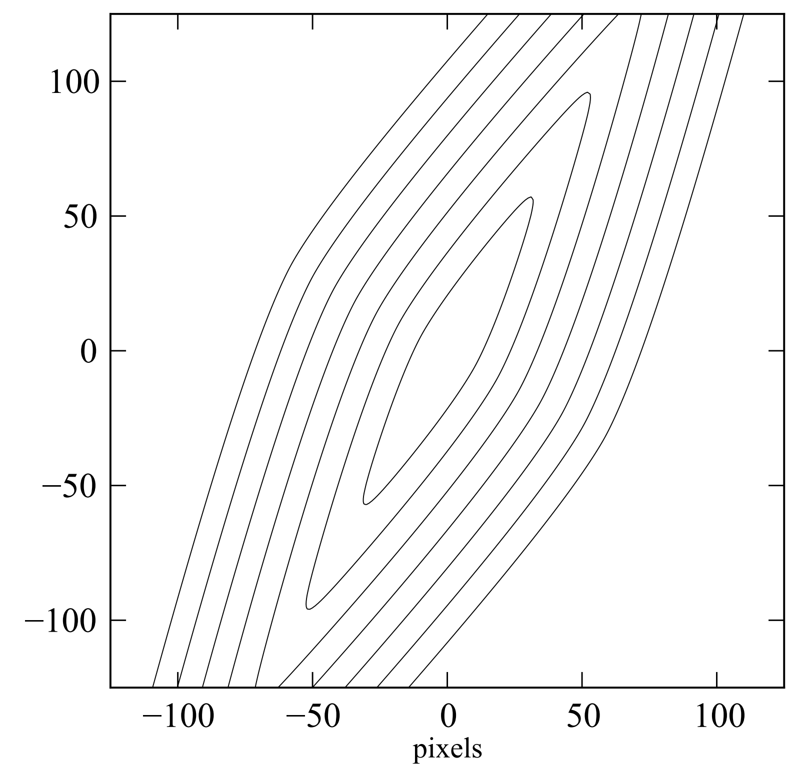
Spitzer IRAC1 (3.6μm)



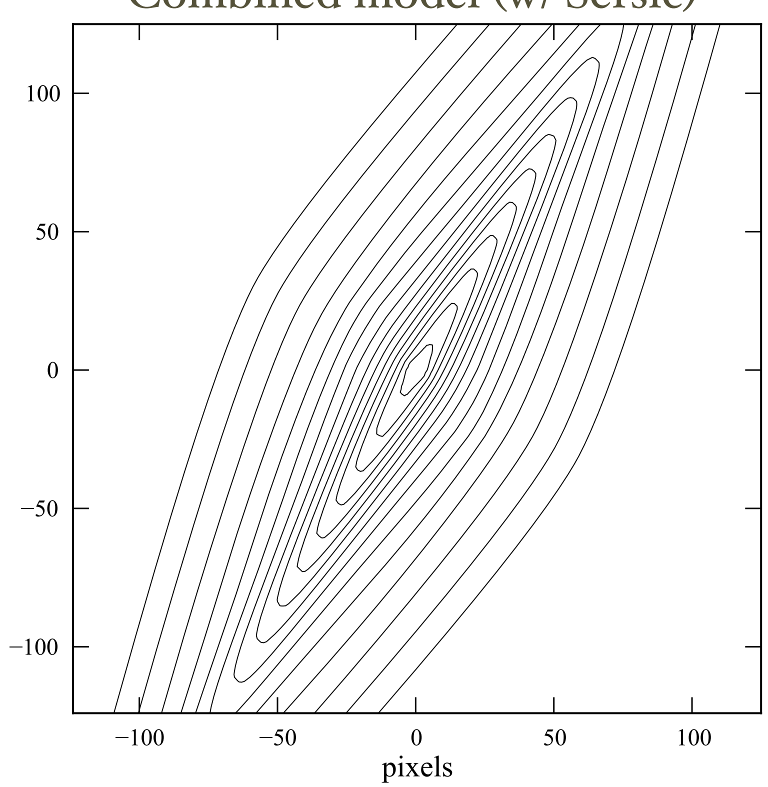
3D Thin disk



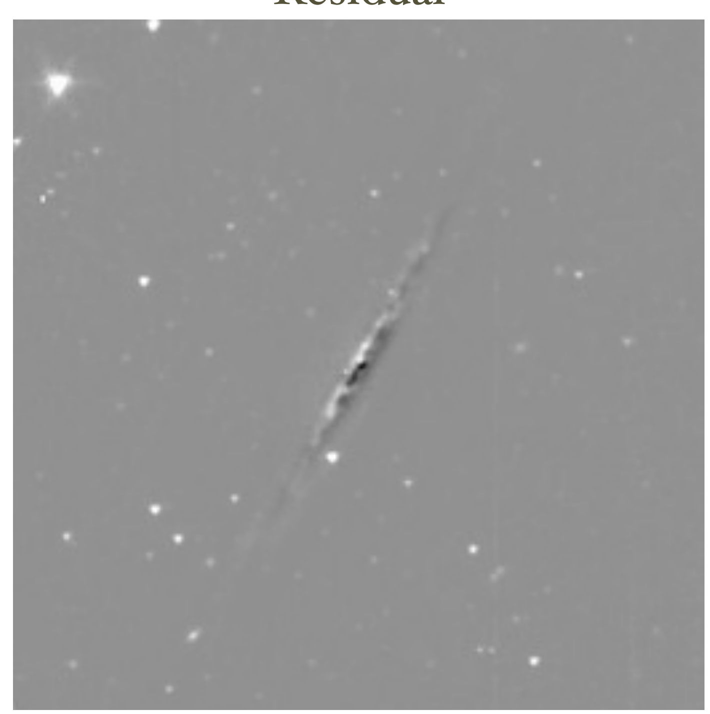
3D Thick disk



Combined model (w/ Sérsic)



Residual



1. Sérsic = 0.018
2. Thin disk = 0.72, exponential scale height = 260 pc, i = 87°
3. Thick disk = 0.27, exponential scale height = 1510 pc, i = 90°

Thick/thin height = 5.8

Thick/thin luminosity = 0.37

Minimization Algorithms

1. Levenberg-Marquardt least-squares minimization

   C++ version of "MPFIT" C module of Craig Markwardt (ultimately from Fortran MINPACK-1 code; Garbow+1980)
   - very fast
   - numerical differentiation
   - can be trapped in local minima in fit landscape

2. Nelder-Mead Simplex (NLopt library)
   - less sensitive to local minima in fit landscape
   - slower (~5–10 times)

3. Differential Evolution (Storn & Price 1997) — genetic-algorithms approach
   - even less sensitive to local minima in fit landscape
   - even slower (~50–100 times slower than L-M!)

Applications

- S0/spiral decompositions for SMBH modeling (Rusli et al. 2013; Erwin et al., in prep; Saglia et al., in prep)
- Fitting of *kinematically decomposed* bulge, disk+ring components from IFU data of NGC 7217 (Fabricius et al., in prep)
- Preliminary EUCLID photometric pipeline simulations (Kümmel et al. 2013)
- Python bindings by André Luiz de Amorim (to be contributed to AstroPy project as affiliated package):

https://github.com/streeto/python-imfit

What Should You Minimize When Fitting Images?

- What statistical model should you use for comparing data values with the model?
- Astronomical images are usually a combination of:
  - Poisson statistics (e.g., detected photoelectrons)
  - Gaussian noise contributions (e.g., read noise)

What Should You Minimize When Fitting Images?

- What statistical model should you use for comparing data values with the model?
- Astronomical images are usually a combination of:
  - Poisson statistics (e.g., detected photoelectrons)
  - Gaussian noise contributions (e.g., read noise)

Maximum likelihood approach: maximize the product of per-pixel probabilities p_i of the data value d_i given the model value m_i

==> Usually simpler to *minimize the log-likelihood*

$$\mathcal{L} = \prod_{i=1}^{N} p_i \longrightarrow -\ln \mathcal{L} = -\sum_{i=1}^{N} \ln p_i$$

What Should You Minimize When Fitting Images?

- What statistical model should you use for comparing data values with the model?

- Astronomical images are usually a combination of:
  - Poisson statistics (e.g., detected photoelectrons)
  - Gaussian noise contributions (e.g., read noise)

Maximum likelihood approach: maximize the product of per-pixel probabilities p_i of the data value d_i given the model value m_i

==> Usually simpler to *minimize the log-likelihood*

$$\mathcal{L} = \prod_{i=1}^{N} p_i \longrightarrow -\ln \mathcal{L} = -\sum_{i=1}^{N} \ln p_i$$

So... minimize log-likelihood of combined Gaussian & Poisson contributions

Sounds pretty simple, right?

Actually, Poisson + Gaussian log-likelihood is pretty scary ...

$$-\ln \mathcal{L} = \sum_{i=1}^{N} \left(m_i - \ln \left[\sum_{x_i=0}^{\infty} \frac{m_i^{x_i}}{x_i!} \exp\left(\frac{-(d_i - x_i)^2}{2\sigma^2} \right) \right] \right)$$

[e.g., Llacer & Nuñez 1992]

Actually, Poisson + Gaussian log-likelihood is pretty scary ...

$$-\ln \mathcal{L} \;=\; \sum_{i=1}^{N} \left(m_i \;-\; \ln \left[\sum_{x_i=0}^{\infty} \frac{m_i^{x_i}}{x_i!} \exp\left(\frac{-(d_i - x_i)^2}{2\sigma^2} \right) \right] \right)$$

[e.g., Llacer & Nuñez 1992]

Actually, Poisson + Gaussian log-likelihood is pretty scary ...

$$-\ln \mathcal{L} = \sum_{i=1}^{N} \left(m_i - \ln \left[\sum_{x_i=0}^{\infty} \frac{m_i^{x_i}}{x_i!} \exp\left(\frac{-(d_i - x_i)^2}{2\sigma^2} \right) \right] \right)$$

[e.g., Llacer & Nuñez 1992]

Actually, Poisson + Gaussian log-likelihood is pretty scary ...

$$-\ln \mathcal{L} = \sum_{i=1}^{N} \left(m_i - \ln \left[\sum_{x_i=0}^{\infty} \frac{m_i^{x_i}}{x_i!} \exp\left(\frac{-(d_i - x_i)^2}{2\sigma^2} \right) \right] \right)$$

[e.g., Llacer & Nuñez 1992]

“Run away! Run away!”

Simple Solution I

Use the Gaussian approximation to the Poisson distribution:

$$\sigma_s = \sqrt{s}$$

Reduces to familiar χ^2:

$$-2 \ln \mathcal{L} = \chi^2 = \sum_{i=1}^{N} \frac{(d_i - m_i)^2}{\sigma_i^2}$$

Can estimate sigma from:

- data values (+ Gaussian read noise)
- model values (+ Gaussian read noise)
- external error/variance map (e.g., from a reduction pipeline)

Simple Solution II

Ignore Gaussian contributions and use Poisson distribution

$$p_i(d_i|m_i) \;=\; \frac{m_i^{d_i} e^{-m_i}}{d_i!}$$

Reduces to "Cash statistic" C (Cash 1979):

$$-2\ln\mathcal{L} \;=\; C \;=\; 2\sum_{i=1}^{N}(m_i - d_i \ln m_i)$$

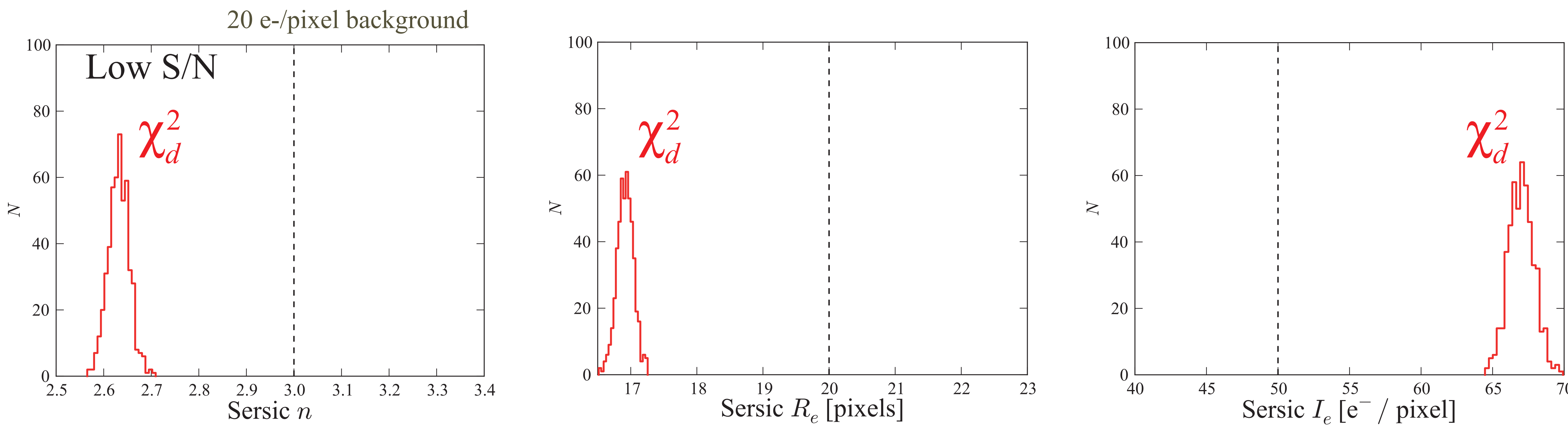
About as easy to calculate as χ^2

Especially apt for low-count regimes (well-known in X-ray fitting)

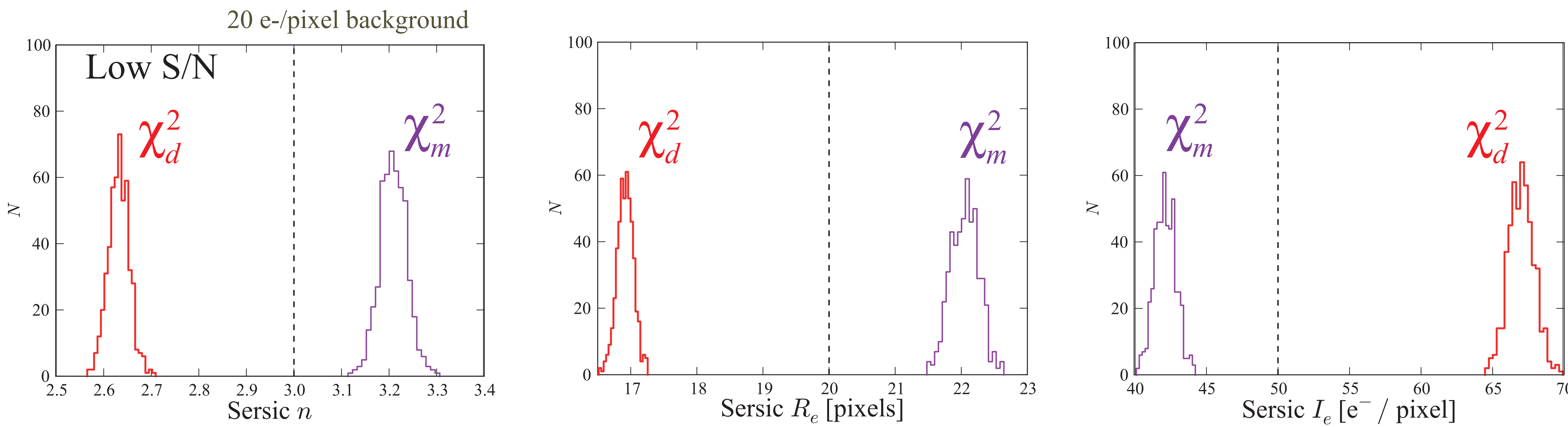
Model Images

- 500 realizations of simple elliptical $n = 3$ Sérsic model
- Three S/N regimes:
  - low = 20 electrons/pixel background
  - medium & high = 5 & 25 times
- True Poisson statistics (+ optional Gaussian read noise)
- Fit with Imfit using data-based χ^2, model-based χ^2, and Cash statistic (simplification: assume we know sky exactly)

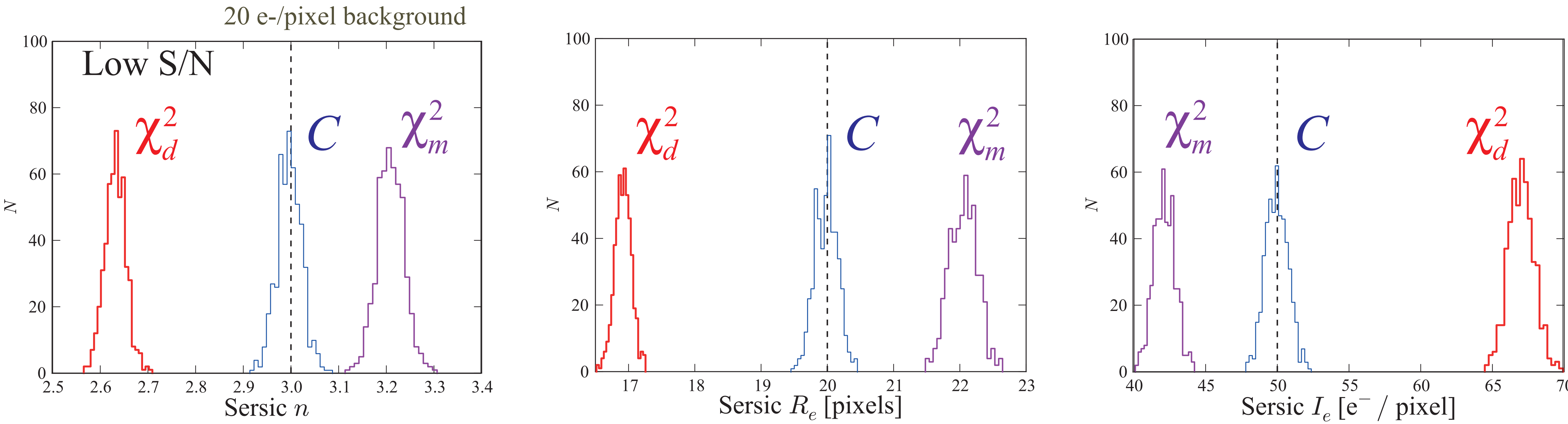
Chi-Squared Biases



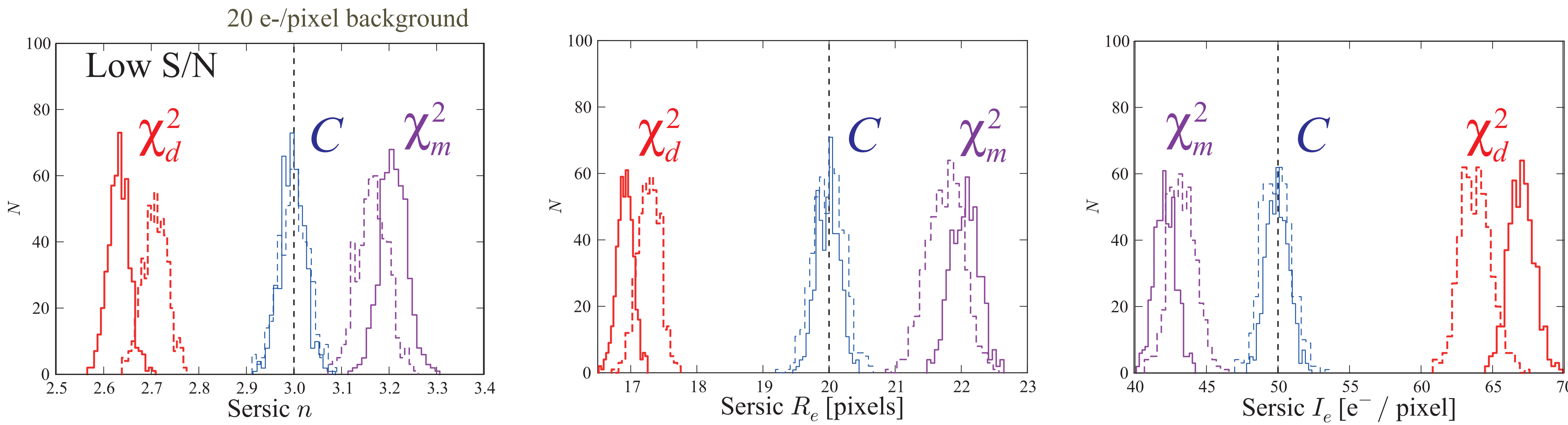
Chi-Squared Biases



Chi-Squared Biases



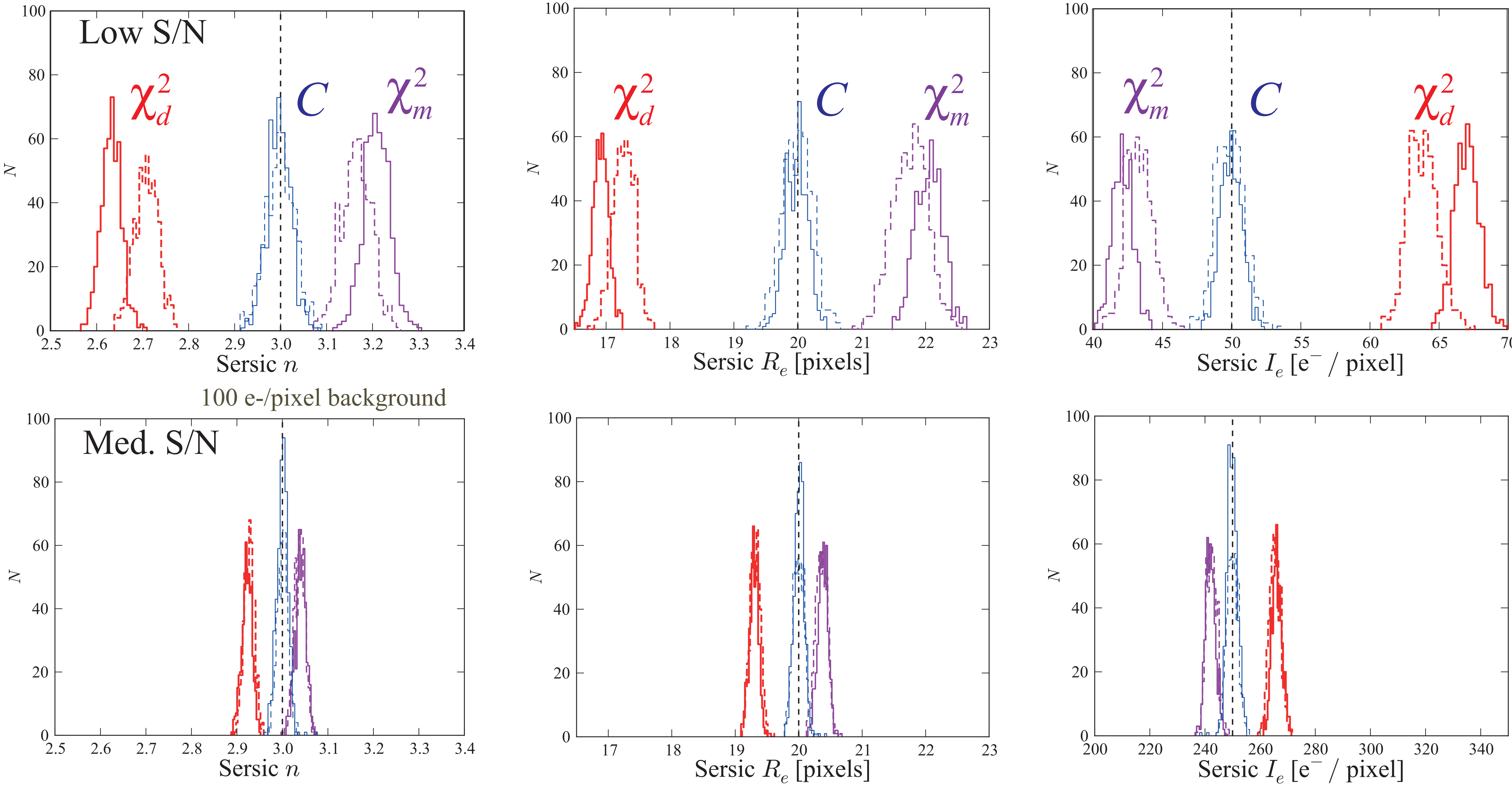
Chi-Squared Biases



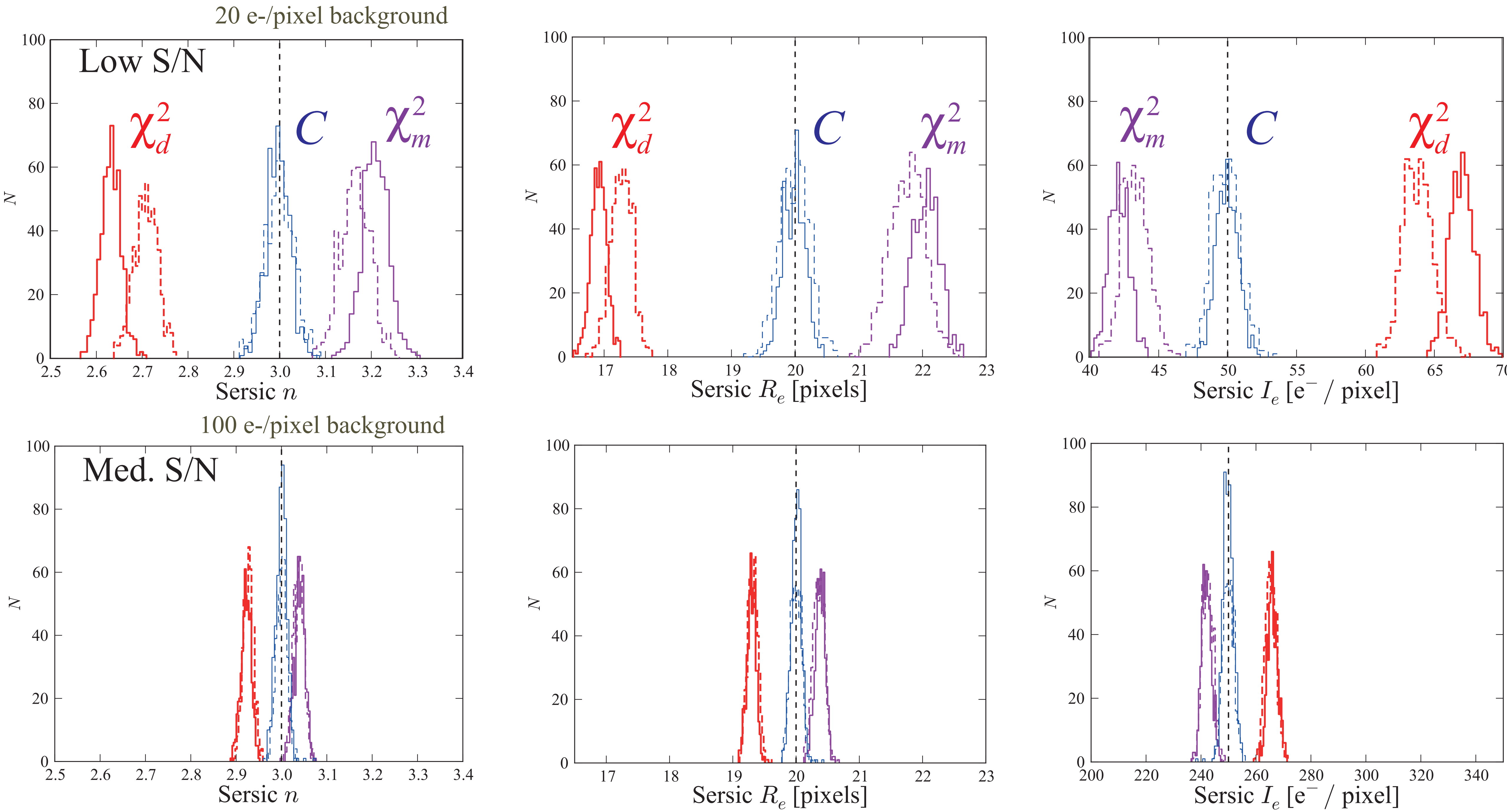
Chi-Squared Biases

20 e-/pixel background

100 e-/pixel background



Chi-Squared Biases



χ^2 fits yield *biased* model parameters; model-based bias is smaller

Bias shrinks as S/N increases

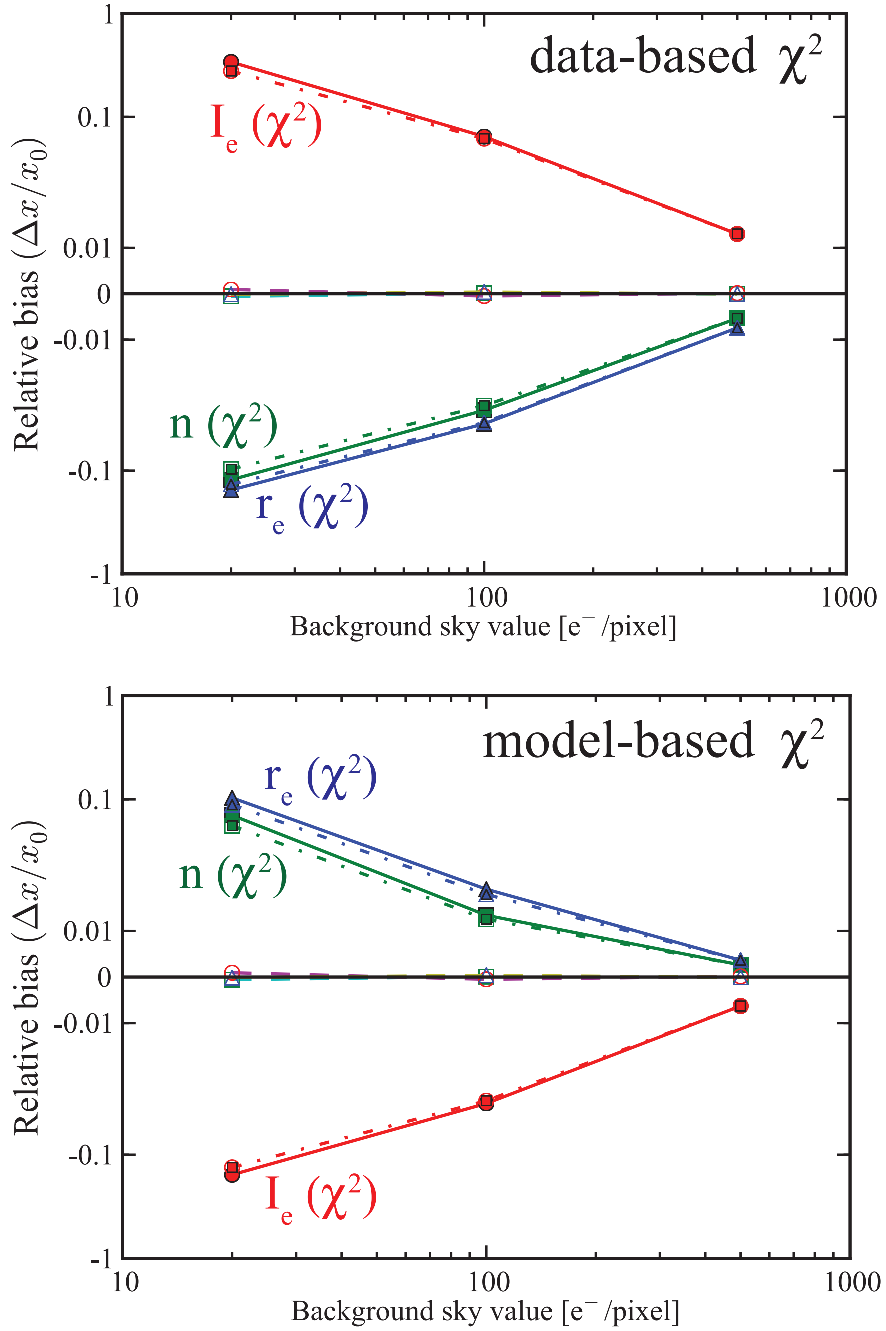
Size & direction of bias as in Humphrey+2009

Cash-statistic fits are effectively *unbiased*!

Does This Apply to Real Galaxy Images?

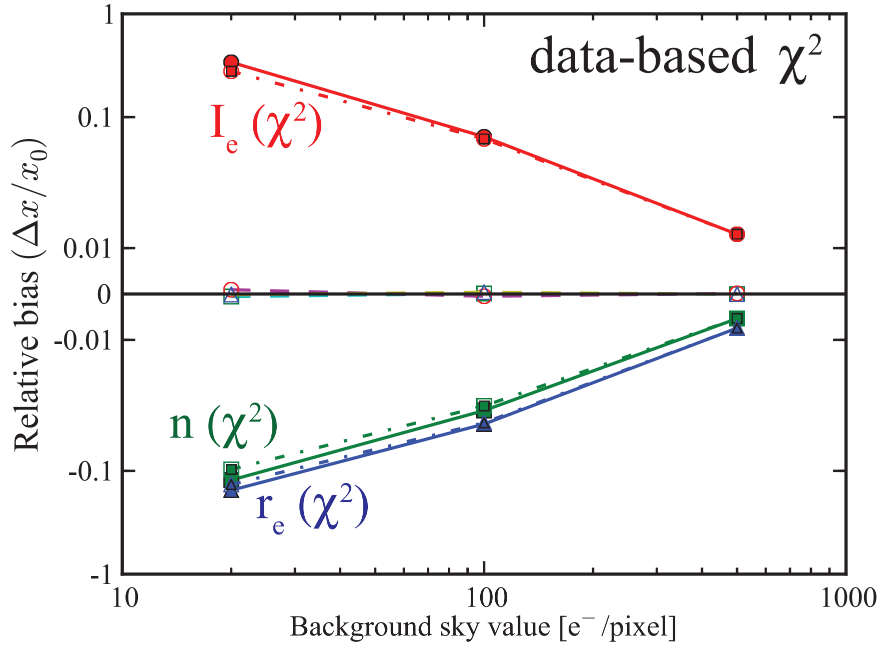
Does This Apply to Real Galaxy Images?

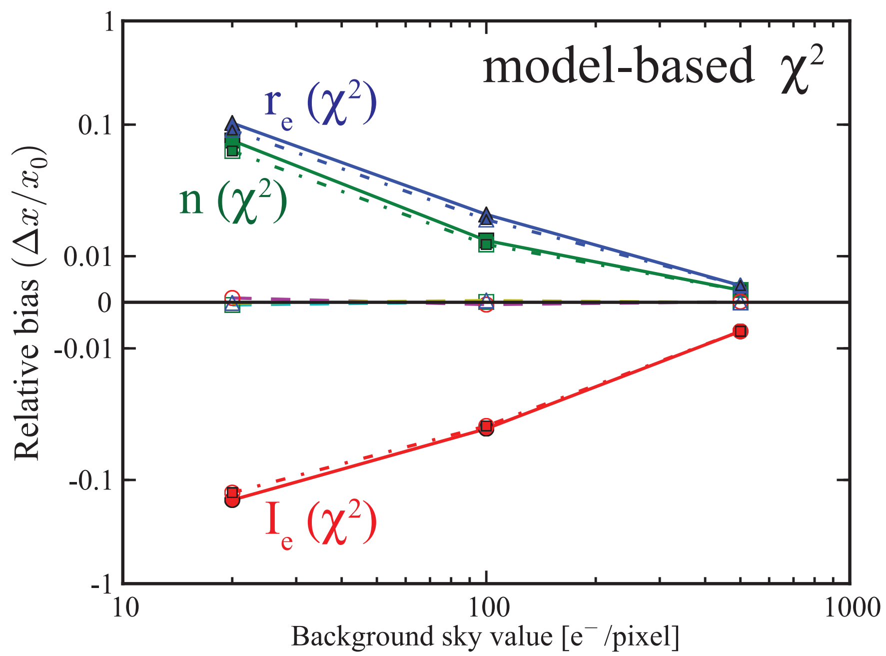
Model-Galaxy Images



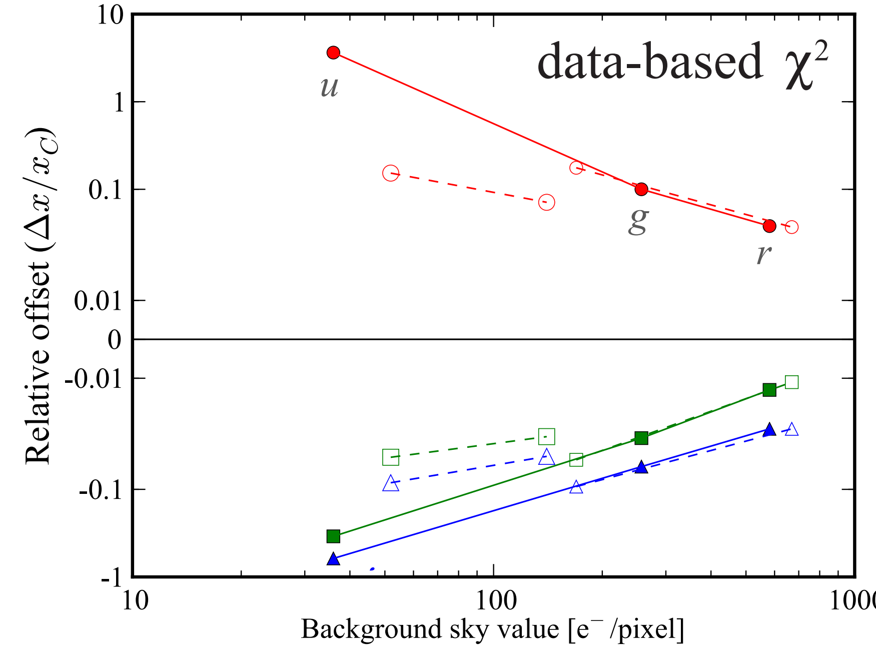
Does This Apply to Real Galaxy Images?

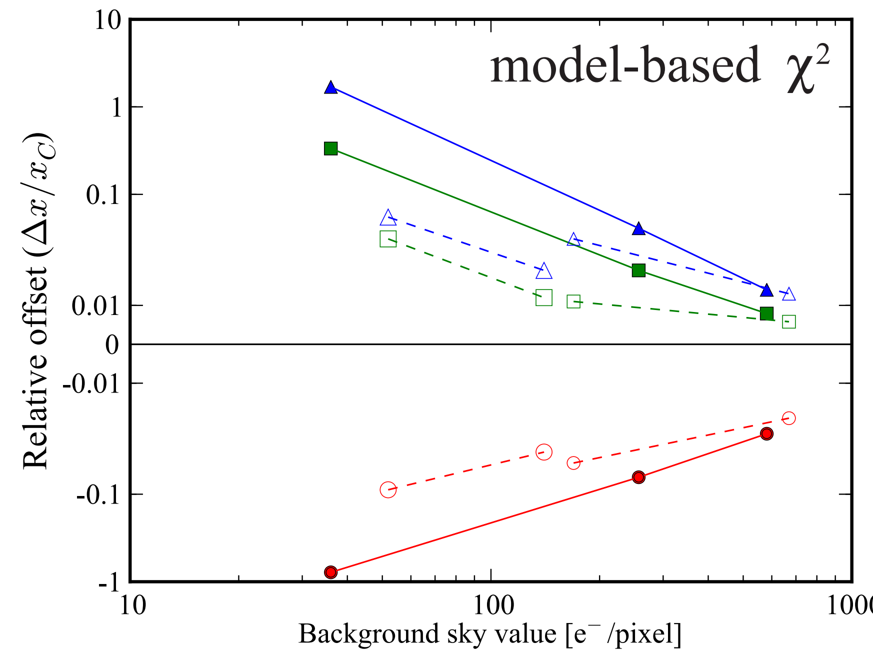
Model-Galaxy Images





Real-Galaxy Images





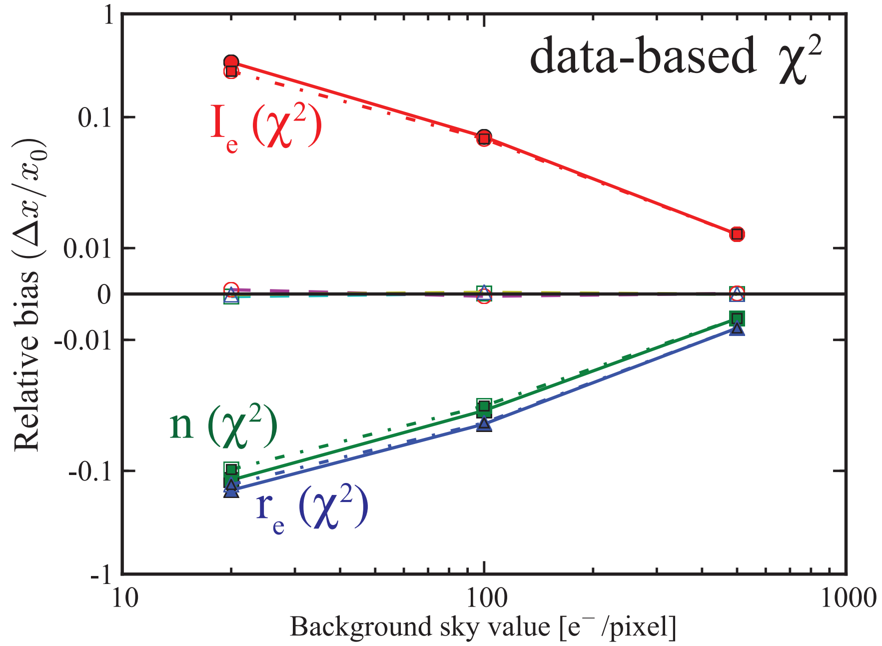
NGC 5831: SDSS *ugr*

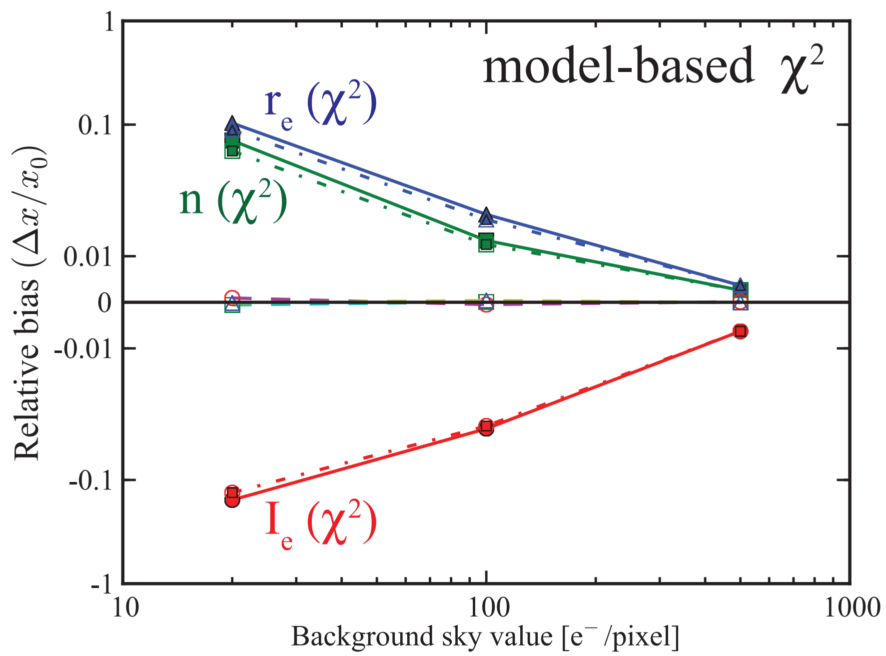
NGC 3379: *V* 15,40s

NGC 4967: *r* 15,60s

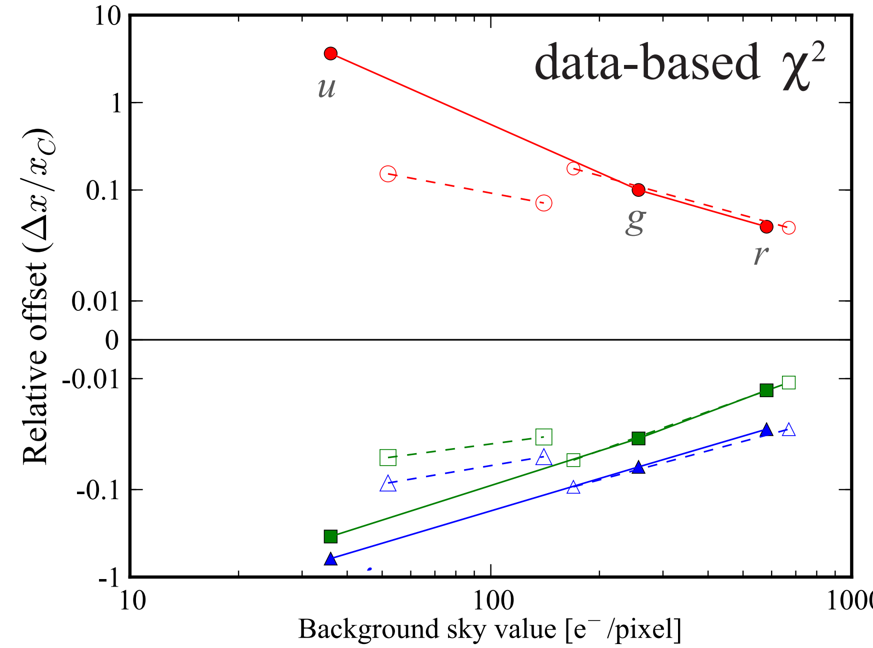
Does This Apply to Real Galaxy Images?

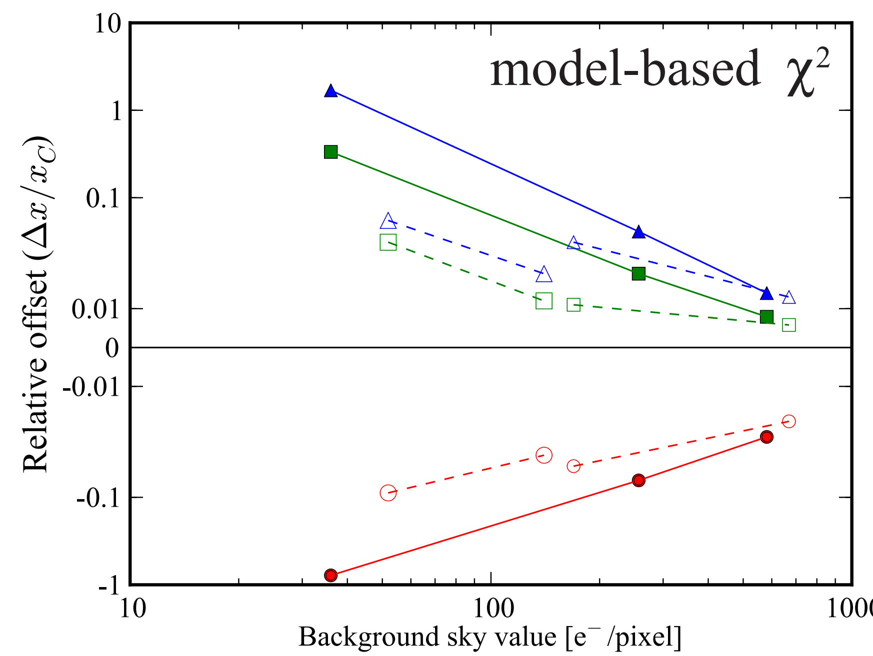
Model-Galaxy Images





Real-Galaxy Images





NGC 5831: SDSS *ugr*

NGC 3379: *V* 15,40s

NGC 4967: *r* 15,60s

YES: χ² Sérsic fits to real elliptical-galaxy images show same pattern of deviations relative to Cash-statistic fits as in model-galaxy images.

So, should we always use Cash Statistic?

- Cash statistic means lower bias, even when read noise is present
- Probably important whenever background < 100 e^{-}/pixel
- Drawback:
  - C is usually < 0, so can't use least-squares minimization algorithms (like L-M)
  - Can use N-M simplex – but that's about 5–10 times slower
  - Compromise solution when speed is critical: use *model-based* χ^2 instead of data-based χ^2

Summary

I. Imfit

- Fast, flexible galaxy image-fitting code
- Easy to add new components
- Multiple minimization options & algorithms

II. Biases in fitting galaxies:

- Fits using χ^2 lead to *biased parameters* (as large as 10%)
- Cash-statistic fits are unbiased (but slower)
- When background counts < 100 e^-/pix, should consider using Cash statistic, or at least model-based χ^2

Source code, compiled programs (MacOS X, Ubuntu Linux), documentation, & sample files available here:

www.mpe.mpg.de/~erwin/code/imfit/

(Or just Google for "Peter Erwin" and "imfit")