Pygfit Users Cookbook

Anthony Gonzalez, Leonidas Moustakas, and Conor Mancone

Last updated: October 7, 2013

The **Py**thon **G**alaxy **Fit**ter (PyGFit) is a code designed to yield matched photometry for multiresolution data sets. In an era where quantitative morphological fitting is commonplace and automated, we have designed PyGFit with the aim of providing fast, easy, and robust matched photometry for data sets where quantitative morphologies have already been derived from the highest resolution imaging. The code is described in detail in Mancone et al. (2013). This paper also discusses issues such as morphological k-corrections of which the user should be aware. Here we aim to provide a cookbook to guide users through the process from start to finish. In this example we assume that the quantitative morphologies will be derived with Galapagos/GALFIT, which is currently the most commonly used combination for automated quantitative morphologies. The PyGFit code is sufficiently general however that it can handle any analytic parameterizations of galaxy structure through the inclusion of additional modules beyond the current Sersic and point source models. We encourage users who either develop or need additional structural models to contact us.

1. Getting Started

1.1 Python Setup

Obtaining the Code

The PyGFit source code can be found at www.baryons.org/pygfit. PyGFit was developed on python 2.6.2. The README file enclosed includes detailed installation instructions.

Software Requirements

Before installing PyGFit, the following python libraries and external software should first be installed on the system.

Libraries:

scipy, numpy, matplotlib (scipy.org)

pyfits (http://www.stsci.edu/institute/software_hardware/pyfits)

External software:

SExtractor (www.astromatic.net/software/sextractor)

Users may wish to consider using the Scisoft distribution, which contains all of the required libraries plus SExtractor (www.eso.org/sci/software/scisoft). A Mac version of the Scisoft

distribution can also be found at scisoftosx.dyndns.org.

Installation

Once the code is downloaded, cd to the directory in which you wish to install pygfit and type:

> tar xzvf pygfit_1.0.tgz

Next, add this folder to your python path. For csh add the following line to your .cshrc file

setenv PYTHONPATH '/PATH_TO_PYGFIT/'

Additionally, make sure that SExtractor execitable is in your path and if not add the following line:

setenv PATH ${PATH}:/PATH_TO_SEXTRACTOR:

1.2 PyGFit Inputs

To run PyGFit, the following inputs are required:
1. Catalog with Quantitative Catalog from High Resolution Image (ASCII or FITS format)
2. Low resolution image
3. PSF for low resolution image
4. RMS map for low resolution image
5. Configuration file

In the sections below we describe how to generate or prepare each of these inputs for PyGFIT. For simplicity, we recommend setting up your files with a directory structure similar to the following:
OBJECT/
    B/ I/ z/ CH1/
where the files corresponding to each filter are contained directly within the appropriate subdirectory.

2. The High Resolution Catalog

The high resolution catalog defines the structural parameters of all objects identified in the high-resolution image that PyGFIT can attempt to match and fit in the lower resolution image. By default PyGFIT is set up to handle point sources and sersic profiles. Note that the astrometric calibration of the high-resolution catalog does not need to precisely match that of the low resolution image, as PyGFIT will calculate a global astrometric offset between the two. It is however important that there exists no net

rotation between the high-resolution catalog and the low resolution image.

There are several factors to keep in mind with the high resolution catalog:
1. Garbage In - Garbage Out. If the structural fits are poor in the high-resolution image, as evidenced by either large chi-squared values or structural parameters driven to the boundaries of the permitted range (such as n=8), then the matched photometry derived from PyGFIT will not be reliable.
2. Unresolvable blends. If two objects in the high-resolution catalog are separate by << the PSF in the low-resolution image, PyGFIT will also not be able to derive robust photometry.

2.1 Starting with a Catalog

If you are starting with a previously generated catalog, then you will need to know the orientation of the image relative to north and pixel scale, as this information is necessary to interpret position angles and effective radii. The catalog can be in either FITS or ASCII format. The following section in the configuration file directs the code to the proper columns for a FITS table, where the second column lists the appropriate keywords in the table:

```
### Column Layout of High Resolution Catalog ###
# Specify column index (zero-indexed) of each data field in the high
resolution catalog
MODEL_TYPE              type        # galaxy model type (sersic or point)
ID                      number      # galaxy id
RA                      ra          # ra
DEC                     dec         # dec
MAG                     mag         # magnitude
POINT_MAG               mag         # magnitude column for point sources
RE                      re
BA                      ba
N                       n
PA                      pa

For an ASCII file, the second column would instead contain the column number
corresponding to each quantity. Note that the errors associated with each
quantity (eg. MAG_ERR, RE_ERR, BA_ERR, N_ERR, PA_ERR) can also be included,
which is necessary if you wish to run simulations where the structural
parameters are permitted to change.
```

2.2 Starting by Running Galapagos

If you do not have a catalog for the high-resolution image in hand, then it will be necessary to generate one yourself, and then follow the instructions above for editing the configuration file. We refer the user to the Galapagos/GALFIT manuals for instructions on how to run these codes. On the web page we do include example configuration, parameter, and setup files for galapagos for an F606W image, as well as a script, build_psfs_acs.py that can be used to generate a suitable input PSF for galapagos (in the same fashion as described below for PSF generation for PyGFIT).

3. The Low Resolution Image

The low resolution image from which you aim to extract photometry should ideally be free of major artifacts that may be mistaken as objects. Masking these objects is fine as long as this masking is also reflected in the RMS map.

4. Generating PSFs

The user may use any means they prefer to generate a PSF for the low resolution image. From experience, we have found that empirical PSFs are greatly superior to synthetic PSFs. The approach that we take is to combine a number of relatively bright, unsaturated stars within the image to create a median PSF. Posted on the web page is an example code, build_psfs_pygfit.py, that can be used for this purpose. With this code the user first identifies a set of input stars in ds9, saving a region file (xy format, ra/dec in degrees) to the file "stars" in the same directory as the image. build_psfs_pygfit.py can then be run to generate a PSF. This code also generates a diagnostic plot showing the normalized radial profiles of each star so that the user can remove any deviant objects.

5. RMS Images

An RMS image associated with the low resolution image is required so that the code appropriately accounts for the pixel-to-pixel noise variations. This image should include both the sky noise and the Poisson noise associated with the objects in the image.

5.1 Starting with pre-existing RMS images

In the ideal case where you already have an existing RMS image, then there is no additional work necessary. You simply need to point to it in the configuration file (see below).

5.2 Making your own RMS image

It is sometimes the case that a proper RMS image generated during the image processing is unavailable. In these cases it is possible to reconstruct an approximate RMS image directly from the data and an exposure map. For images where there has been no sky subtraction, the RMS will simply be sqrt(data*exposure map/Gain). For images where the sky has been subtracted, one can compute the sky RMS in blank regions of the image and use this information to add back in an approximate sky as part of this procedure, such that the RMS = sqrt((sky+ (data*exposure map))/Gain).

6. The PyGFIT Configuration File

The command:

```
> pygfit.py --config
```

will generate the example configuration file shown below. The parameters can be broken down into eight subsets:

1. Detection Parameters for Source Extractor. These should be self-explanatory if you are familiar with Source Extractor. If not, the Source Extractor documentation provides a detailed explanation
2. High Resolution Catalog & Image Properties. This set of parameters defines the catalog name, and the image orientation and plate scale. The latter two are needed to correctly interpret position angles and effective radii for Sersic models.
3. High Resolution Catalog Format. These parameters specify the column numbers (for ASCII catalogs) or column names (for FITS catalogs) corresponding to each parameter. For simulations in which the model parameters are to be varied, one should also include the columns corresponding to the model parameter errors.
4. Low Resolution Image Properties. Here is where one specifies the science and rms images, the PSF, the photometric zeropoint.
5. Fitting Parameters. The parameters in this section specify the details of the object fitting. The items which the user should adjust to be appropriate for a given data set are the following:

```
MIN_MAG,MAX_MAG - These define the magnitude range of objects detected by Source
Extractor in the low resolution image that the user wishes to fit with PyGFit.

GLOBAL_MAX_SHIFT   -  This is the maximum astrometric shift allowed between the high and
low resolution data sets.
ALIGN_MIN_MAG, ALIGN_MAX_MAG - These define the magnitude range over which to calculate
the global astrometric shift.
N_ALIGN - This parameter defines the maximum number of objects to use in calculating the
global shift.

MAX_SHIFT - This defines the maximum positional shift permitted for an individual object
(after application of the global shift) during the fitting process.
```

The remaining parameters in this section are normally best left to the default values. We do note however that the code is capable of running on GPUs. If you wish to do so to shorten the run time, simply change GPU to True and set the appropriate number of threads per block in GPU_NTHREADS.

6.Output Settings. Here the user can select which columns to include in the output catalog and define details of the output catalog and images. These parameters should be relatively self-explanatory and the default values can be used with no editing if desired.
7. Diagnostic Plots. The code outputs a series of diagnostic plots The default values can be used with no editing.

```
### Source Extractor ###
EXTRACTOR_CONFIG          extractor.config    # Name of source extractor configuration
file
EXTRACTOR_PARAMS          extractor.param     # Name of source extractor parameters
```

```
file (will be generated automatically)
EXTRACTOR_CATALOG          extractor.cat       # Output catalog name for source
extractor
EXTRACTOR_CMD              sex                 # Location of source extractor executable
SKIP_EXTRACTOR             True                # Skip running source extractor if it has
already been run (True/False)

### High Resolution Catalog ###
HRES_CATALOG               hres.cat            # Filename of high resolution catalog
HRES_ROTANGLE              0.000000e+00        # Roll angle for high resolution image,
West of North in Degrees
HRES_PIXSCALE              1.388889e-05        # Pixel scale for high resolution image,
Degrees per pixel

### High Resolution Catalog Format ###
MODEL_TYPE                 0                   # Galaxy model type (either 'sersic' or
'point')
ID                         1                   # Unique id
RA                         2                   # Right Ascension
DEC                        3                   # Declination
MAG                        4                   # Magnitude for sersic models
POINT_MAG                  4                   # Magnitude for point models
RE                         5                   # Effective radius (sersic only)
N                          6                   # Sersic index (sersic only)
PA                         7                   # Position Angle (sersic only)
BA                         8                   # Axis Ratio (sersic only)

### Low Resolution Images ###
LRES_IMAGE                 lres.fits           # Filename of low resolution fits image
LRES_RMS                   lres_rms.fits       # Filename of low resolution rms image
LRES_PSF                   lres_psf.fits       # Filename of low resolution psf image
LRES_MAGZERO               22.5000             # Magnitude zeropoint for low resolution
image

### Fitting Settings ###
USE_INTEGRATION            True                # Whether or not to use integration to
properly calculate hard-to-estimate sersic models
GPU                        False               # Whether or not to attempt to speed up
calculations with a GPU
GPU_NTHREADS               512                 # The number of threads per block to
execute on the GPU
N_THREADS                  1                   # Maximum number of cpu threads to use
(will never use more than the actual number of cpus)
GLOBAL_MAX_SHIFT           2.0000              # Maximum allowed (global) positional
offset between high and low resolution catalog (arcseconds)
MAX_SHIFT                  0.2000              # Maximum allowed positional shift for a
high resolution object during fit (arcseconds)
MIN_MAG                    16.0000             # Minimum magnitude of objects to fit
MAX_MAG                    22.0000             # Maximum magnitude of objects to fit
ALIGN_MIN_MAG              18.0000             # Minimum magnitude of objects to include
```

```
in alignment
ALIGN_MAX_MAG              20.0000             # Maximum magnitude of objects to include
in alignment
N_ALIGN                    25                  # Maximum number of objects used in
alignment calculation
PAD_LENGTH                 25                  # Padding region (in pixels) around model
to allow room for interpolation/convolution

### Output Settings ###
SUBTRACT_BACKGROUND        False               # Whether or not to subtract the
background from the final residuals image
IMAGE_DIR                  cutouts/img         # Directory for outputting image cutouts
(relative to working directory)
RMS_DIR                    cutouts/rms         # Directory for outputting rms cutouts
SEGMENTATION_DIR           cutouts/seg         # Directory for outputting segemntation
cutouts
SEGMENTATION_MASK_DIR      cutouts/mask        # Directory for outputting segmentation
mask cutouts
OUTPUT_ALL_MODELS          True                # Whether or not to output an extension
for individual models in the _fit.fits files
OUTPUT_CATALOG             pygfit.cat          # Filename for output catalog
OUTPUT_FORMAT              ascii               # 'fits' or 'ascii' - output type for
final catalog
# Fields to output to final catalog
OUTPUT_COLUMNS             lres_id, hres_id, nblend, nearest, nearest_mag, model, ra,
dec, x, y, img_x, img_y, mag, mag_image, mag_initial, mag_hres, mag_brightest,
mag_warning, flux, total_flux, total_mag, blend_fraction, sky, re_hres, re_lres,
re_arcsecs, n, pa, ba, chisq_nu, chisq, nf, segmentation_mag, segmentation_residuals,
segmentation_fraction, mask_mag, mask_residuals, mask_fraction

### Check Plots ###
# Type of check plots to generate
CHECK_PLOTS                isolated, alignment, alignment_mag
# Filenames for check plots
CHECK_PLOT_FILES           isolated.fits, alignment.fits, alignment_mag.fits
```

7. Running PyGFit

The command

```
> pygfit.py --help
```

provides a list of possible command line flags:

```
pygfit.py [--help --simulate --plot_sims --skip_extractor --residuals_file=file
--output_catalog=file config_file log_file warn_file]
```

The default values for the input configuration file and output log, and warning files are pygfit.config,

pygfit.log, and pygfit.warn, respectively For regular execution of PyGFit, it is sufficient to type

```
> pygfit.py
```

or

```
> pygfit.py  config_file
```

if the configuration file name differs from the default value. The simulation flag is required on the command line to engage the simulation module.

8. Running PyGFit on Multiple Images

It is often the case that one will wish to run pygfit on data in multiple passbands starting with a single high-resolution catalog. Doing so is straightforward with PyGFit. If the same high-resolution catalog is used as input for each image, then the output catalogs will contain an identical set of objects. If the catalogs are in ASCII format, then they will be line-matched. If they are in FITS format, then the script such as merge_pygfit.py (posted on the web page) can be used to append relevant columns into a single master catalog.

9. PyGFIT Simulations to Quantify Uncertainties

PyGFit also comes with the ability to run simulations to quantify the photometric uncertainty as a function of magnitude and other properties. Use of the --simulate flag:

```
> pygfit.py --simulate
```

instructs PyGFIT to generate N mock sources, insert them into M copies of the original image, recover the best fit photometry, and generate a file results.cat containing the results. The key parameters are:

```
SIMULATED_FRACTION   2.5   # Fraction of catalog to insert in individual image
NUMBER_FRAMES       40     # Number of images into which to insert objects
```

It is advisable to keep the simulated fraction in any individual image near the default value to avoid superpositions of simulated sources upon one another, while the number of frames can then be set to yield the desired number of total simulated objects. By default the structure parameters and magnitudes of the objects will be drawn directly from the PyGFit output catalog such that the simulated sample will have similar properties to the real distribution. This simulated catalog can then be analyzed as desired. At some point we may add simulation analysis scripts to the web page as well.

Important: When running simulations, you must edit the SExtractor configuration file to make sure that the paths to the SExtractor .conv, .nnw, and .param files are all absolute. Simulations are executed in subdirectories, and the code with crash if these paths are relative.

10. Questions and Comments

If you are using this code and have questions or suggestions, feel free to contact us via email (anthonyhg@ufl.edu, leonidas@jpl.nasa.gov). Please also let us know if you make modifications or additions to the code that you wish to contribute to the main distribution.