BBarolo Documentation

Release 1.4

Enrico Di Teodoro

Dec 13, 2017

BBAROLO:

3D-Barolo (3D-Based Analysis of Rotating Objects via Line Observations) or **BBarolo** is a tool for fitting 3D tilted-ring models to emission-line datacubes. The Python wrapper is **pyBBarolo**.

For a detailed of the algorithms used in this code, please refer to BBarolo's main paper.

Table of contents:

CHAPTER

ONE

INSTALLING BBAROLO

1.1 Pre-compiled binaries

Pre-compiled executable files for the GUI are available at this page. Binaries are for Linux x86_64 and Mac OS X (> 10.6).

The command line executable (`BBarolo`) is also included in these binary packages and can be found at:

- **Linux**: same directory of BBaroloGUI
- **MacOS**: BBaroloGUI.app/Contents/MacOS/BBarolo

The pre-compiled executables should work on most systems. If they don't, please compile BBarolo from source following the instructions below.

1.2 Compiling from source

The best way to exploit BBarolo functionalities is to compile the code directly on your computer. BBarolo compiles and runs on Unix machines only.

1.2.1 Requirements

To compile the code, all you need is:

- a C++ compiler supporting C++11 standard, like the GNU compiler.
- CFITSIO library.
- FFTW library.
- WCS library.
- QT toolkit (> 4.0), only if you would like to use the GUI.
- Gnuplot and Python (> 2.6) with the Astropy package.

Most of these libraries and packages should already be installed on scientific machines. Otherwise, you can easily install them through the terminal commands of the various package managers, i.e. *apt-get* on Ubuntu-based, *pacman* on Arch-based, *yum* on RPM-based distros, *brew* or *port* on Mac OS X. Note that the QT toolkit is only needed to compile the GUI, which is not mandatory. Gnuplot and Python are not needed to successfully compile the code, but without them BBarolo will not produce any outputs.

1.2.2 Compiling

If your machine satisfies the above requirements, compiling BBarolo will hopefully be a piece of cake.

1. **Download** the latest stable release. From a terminal:

```
> wget https://github.com/editeodoro/Bbarolo/archive/X.Y.tar.gz .
```

   where *X.Y* is the release version. If you are brave, you can also try the last (non stable) source code from here.

2. **Uncompress** it and enter the BBarolo directory:

```
> tar -xvzf Bbarolo-X.Y.tar
> cd Bbarolo-X.Y
```

3. **Configure** running autoconfigure script:

```
> ./configure
```

   If the script is not executable: `> chmod +x configure`. The configure script takes a number of optional arguments. For instance, it is possible to specify installation and library directories, or the compiler to use:

```
> ./configure CXX=icpc --prefix=/dir/to/install --with-cfitsio=/dir/of/
↪cfitsio --with-wcslib=/dir/to/wcslib
```

   If the configuration fails, follow the suggestions given by the script to manually set the path of libraries.

4. **Compile** the source:

```
> make
```

   To compile in parallel: `> make -j N`, where N is the number of processors. If the compilation succeeds, the executable **BBarolo** will appear in the current directory.

Optional steps

5. Install, e.g. copy the executable in the installation path (default is /usr/local/bin):

```
> make install
```

6. Compile the GUI (QT > 4 needed):

```
> make gui
```

   This can fail for a number of reasons.

7. Compile BBarolo as a library:

```
> make lib
```

8. Clean up unnecessary files:

```
> make clean
```

CHAPTER

TWO

RUNNING BBAROLO

2.1 Graphical User Interface

BBarolo comes with a Graphical User Interface (GUI) that can help the user in setting up the input parameters. Running BBarolo through the GUI should be quite straightforward: you do not have to learn the annoying list of *available parameters*: the user needs just to fill the required fields in the GUI and this will create a text file with the correct parameters and run BBarolo.

N.B.: Although the GUI allows the user to set the main parameters, many options can only be enabled through the command line tool. Moreover, I stress that the GUI has not been updated together with the last releases of BBarolo. If you experience any issues with the GUI or want to have full control of the code, I recommend you to use the command line.

2.2 Command line

BBarolo is mainly meant to be run from the command line. For a very quick guide and to appreciate the biggest achievement of my PhD, type `BBarolo` in your terminal.

Regular execution: BBarolo takes input parameters specified through a parameter file, provided at the runtime. This is a text file containing a list of parameter names and values:

```
PARAM1      VALUE1
PARAM2      VALUE2
PARAM3      VALUE3
...         ...
```

All available parameters are described in the *task documentation*. In the input file, parameter names are not case-sensitive and lines starting with # or // are not read in. The order in which parameters are listed is unimportant, but, if a parameter is listed more than once, only the last value is considered.

Some parameters are mandatory, some others are optional and have default values which are assumed when not explicitly set. A template parameter file can be obtained with the command `> BBarolo -t`. Parameters with default values can be printed with `> BBarolo -d`. An example of parameter file can be found here, full runnable instances can be downloaded from this page.

After your parameter file is ready, BBarolo can be run with the following:

```
> BBarolo -p paramfile
```

where *paramfile* is the name of the user-defined input file.

Automated execution: BBarolo can otherwise be run in a completely automated way, i.e. providing no parameters but the input FITS datacube. In this case, the code uses the source finder to identify the galaxy in the datacube, tries to

guess initial values for the rings and fits a 3D model to the data. Although this procedure might work and return nice best-fit models if used with high resolution and high S/N data, it should be used carefully. In particular, the algorithm for guessing initial values for the fit is still quite coarse. Wrong initial guesses may lead to completely inappropriate models.

If you still want to try the automated execution:

```
> BBarolo -f fitsfile
```

where *fitsfile* is the name of the FITS file of the galaxy to analyse.

CHAPTER

THREE

LIST OF TASKS AND PARAMETERS

BBarolo's main algorithm for fitting 3D kinematic models to emission line data (*3DFIT*) makes use of a number of utilities. These tasks include, for example, the disk modeling (*GALMOD*), the source finder (*SEARCH*) and the smoothing utility (*SMOOTH*), and can be conveniently used outside the main algorithm as well.

In this page, I list the main tasks and related input parameters available in BBarolo. Parameter names are in **boldface**, default values are in brackets. The names of parameters are not case-sensitive.

3.1 General parameters

In the following, a list of general parameters (e.g., not task-specific).

3.1.1 Input/output parameters

- **FITSFILE** [none]. The name of the input FITS file. This is a mandatory parameter for **all tasks**.
- **OUTFOLDER** [./output]. The directory where the output files will be written.
- **VERBOSE** [true]. Enable all the output messages.
- **THREADS** [1]. Number of CPUs to use for parallelized tasks.
- **SHOWBAR** [true]. Whether to show progress bars.

3.1.2 Beam parameters

Following parameters can be used to specify the size and shape of the Point Spread Function (PSF or beam). These parameters are ignored if beam information is written in the header of the input FITS, either through BMAJ, BMIN and BPA keywords or in the HISTORY. The code defines the beam following the priority order: header -> bmaj,bmin,bpa params -> beamfwhm param -> default to 30 arcsec.

- **BMAJ** [none]. The FWHM of the major axis of the elliptical Gaussian beam in *arcsec*.
- **BMIN** [none]. The FWHM of the minor minor axis of the elliptical Gaussian beam in *arcsec*.
- **BPA** [none]. The position angle of the major axis of the elliptical Gaussian beam in *degrees*, counter-clock from the North direction.
- **BEAMFWHM** [none]. The FWHM of a circular Gaussian beam in *arcsec* .

3.2 3DFIT task

3DFIT is the main BBarolo's routine: it fits a 3D tilted-ring model to an emission-line data-cube. Algorithms used are described in this paper.

- **3DFIT** [false]. This flag enables the 3D fitting algorithm. Can be *true* or *false*. The old flag GALFIT is now deprecated and will be no more supported in future BBarolo's releases.

3.2.1 Rings IO

Following parameters are used to define the set of rings used for the fit. All parameters are allowed to vary ring-by-ring or to be fixed. In the first case, given values represent initial guesses for the fit.

All parameters listed below (except NRADII and RADSEP) can be given in the form of a single value valid for all rings or through a text file containing values at different radii. In this second case, the sintax to be used is *file(filename,N,M)*, where *filename* is the name of the file with values, *N* is the column number (counting from 1) and *M* is the starting row (all rows if omitted).

- **NRADII** [none]. The number of rings to be used and fitted. If not given, BBarolo tries to guess it from the size of the galaxy.
- **RADSEP** [none]. The separation between rings in *arcsec*. If N radii have been requested, the rings will be placed at N*RADSEP + RADSEP/2. If not given, BBarolo assumes the FWHM of the beam major axis as radius separation.
- **RADII** [none]. This parameter can be used as an alternative to NRADII and RADSEP. This needs to be a text file (see above).
- **XPOS** [none]. X-center of rings. Accepted format are in *pixels* (starting from 0, unlike GIPSY) or in WCS coordinates in the format +000.0000d (*degrees*) or +00:00:00.00 (*sexagesimal*). If not specified, BBarolo tries to guess it using the 3D source finder.
- **YPOS** [none]. Like XPOS, but for the y-axis.
- **VSYS** [none]. Systemic velocity in *km/s*. If not given, BBarolo tries to guess it from the global line profile.
- **VROT** [none]. Rotation velocity in *km/s*. If not given, BBarolo tries to guess it.
- **VDISP** [8]. Velocity dispersion in *km/s*.
- **VRAD** [0]. Radial velocity in *km/s*.
- **INC** [none]. Inclination in *degrees*. If not given, BBarolo tries to guess it from the column density map.
- **PA** [none]. Position angle in *degrees*, measured anti-clockwise form the North direction. If not given, BBarolo tries to guess it from the velocity field.
- **Z0** [0]. Scale-height of the disc in *arcsec*.
- **DENS** [1]. Gas surface density in units of *1E20 atoms/cm2*. Fit of this parameter is not currently implemented and its value is not relevant if a normalization is used.

3.2.2 Additional options

Additional parameters to control and refine the fit. All following parameters have default values and are therefore optional.

- **DELTAINC** [5]. This parameter fixes the boundaries of parameter space at [INC-DELTAINC, INC+DELTAINC]. It is not advisable to let the inclination varying over the whole range [0,90].

- **DELTAPA** [15]. This parameter fixes the boundaries of parameter space at [PA-DELTAINC, PA+DELTAPA]. It is not advisable to let the position angle varying over the whole range [0,360].
- **FREE** [VROT VDISP INC PA]. The list of parameters to fit.
- **FTYPE** [2]. Function to be minimized. Accepted values are: 1 = chi-squared, 2 = |mod-obs|, (default) and 3 = |mod-obs|/(mod+obs)).
- **WFUNC** [2]. Weighting function to be used in the fit. Accepted values are: 0 = uniform weight, 1 = |cos(θ)| and 2 = cos(θ)^2, default), where θ is the azimuthal angle (= 0 for galaxy major axis).
- **LTYPE** [1]. Layer type along z. Accepted values are: 1 = Gaussian (default), 2 = sech^2, 3 = exponential, 4 = Lorentzian and 5 = box.
- **CDENS** [10]. Surface density of clouds in the plane of the rings per area of a pixel in units of *1E20 atoms/cm^2* (see also GIPSY GALMOD).
- **NV** [nchan]. Number of subclouds in the velocity profile of a single cloud (see also GIPSY GALMOD). Default is the number of channels in the datacube.
- **SIDE** [B]: Side of the galaxy to be fitted. Accepted values are: A = approaching, R = receding and B = both (default)
- **MASK** [SMOOTH]. This parameter tells the code how to build a mask to identify the regions of genuine galaxy emission. Accepted values are *SMOOTH*, *SEARCH*, *THRESHOLD*, *NONE* or a FITS mask file:
  - *SMOOTH*: the input cube is smoothed according to the *smooth parameters* and the mask built from the region at S/N>BLANKCUT, where **BLANKCUT** is a parameter representing the S/N cut to apply in the smoothed datacube. Defaults are to smooth by a FACTOR = 2 and cut at BLANKCUT = 3.
  - *SEARCH*: the source finding is run and the largest detection used to determine the mask. The *source finding parameters* can be set to change the default values.
  - *THRESHOLD*: blank all pixels with flux < THRESHOLD. A **THRESHOLD** parameter must be specified in the same flux units of the input datacube.
  - *NONE*: all regions with flux > 0 are used.
  - *file(fitsname.fits)*: A mask FITS file (i.e. filled with 0,1).
- **NORM** [LOCAL]. Type of normalization of the model. Accepted values are: *LOCAL* (pixel by pixel), *AZIM* (azimuthal) or *NONE*.
- **TWOSTAGE** [true]. This flag enables the second fitting stage after parameter regularization. This is relevant just if the user wishes to fit parameters other than VROT, VDISP, VRAD and VVERT. The inclination and the position angle are regularized by polynomials of degree POLYN or a Bezier function (default), while the other parameters by constant functions.
- **POLYN** [-1]. Degree of polynomials for the regularization of inclination and position angles. -1 enables the Bezier function.
- **BWEIGHT** [2]. Exponent of weight for blank pixels. See Section 2.4 of reference paper for details.
- **FLAGERRORS** [false]. Whether the code has to estimate the errors. This usually heavily slows down the run.
- **STARTRAD** [0]. This parameter allows the user to start the fit from the given ring.
- **LINEAR** [0.85]. This parameter controls the spectral broadening of the instrument. It is in units of channel and it represents the standard deviation, not the FWHM. The default is for data that has been Hanning smoothed, so that FWHM = 2 channels and σ = FWHM/2.355.

3.2.3 Additional parameters for high-z galaxies (BBarolo > 1.2.1)

For high-z galaxies you need to set two additional parameters.

- **RESTWAVE** [none]. The rest wavelength of the line you want to fit. Units must be the same of the spectral axis of the cube. For example, if we want fit H-alpha and CUNIT3 = "angstrom", set 6563.
- **REDSHIFT** [none]. The redshift of the galaxy.

These two parameters are used to calculate the conversion from wavelengths to velocities. The velocity reference is set to 0 at RESTWAVE*(REDSHIFT+1). VSYS has to be set to 0, but can be also used to fine-tune the redshift. Finally, if these two parameters are not set, BBarolo will use the CRPIX3 as velocity reference and the proper VSYS has to be set based on that.

3.3 GALMOD task

GALMOD is the routine underlying the 3DFIT task. It builds a 3D simulated datacube of a disk galaxy starting from the a set of concentric rings with given column density and kinematics. The routine is an updated version of the namesake routine in GIPSY (see also GIPSY GALMOD).

Parameters for rings are the same of the *3DFIT task*. Options are LTYPE, CDENS and NV (see *3DFIT options*).

Additional GALMOD-specific parameters are:

- **GALMOD** [false]. This flag enables the 3D disk modelling. Can be *true* or *false*.
- **VVERT** [0]. Vertical velocity in *km/s*.
- **DVDZ** [0]. Gradient of rotation velocity as we move away from the disk plane. This is in *km/s/arcs*.
- **ZCYL** [0]. Height in *arcsec* from the disk plane where the gradient DVDZ begins.
- **SM** [true]. Whether to smooth the model to the same spatial resolution of data.

3.4 SEARCH task

BBarolo's search algorithm is derived from Duchamp, a 3D source finder for spectral-line data developed by Matthew Whiting. BBarolo adds a few functionalities and a (mild) parallelization. For a comprehensive description of the algorithm and the input parameters, see Duchamp's main paper and user guide.

Main parameters to control the source finder are as follows.

- **SEARCH** [false]. This flag enables the source finding algorithm. Can be *true* or *false*.
- **FLAGROBUSTSTATS** [true]. Whether to use to robust estimators (median and MADFM) instead of normal estimators (mean and standard deviation) when calculating cube statistics.
- **SEARCHTYPE** [spatial]. How the search is performed. Accepted values are *spatial* and *spectral*. Spatial search is done in 2D channel maps, spectral search along 1D spectra.
- **SNRCUT** [5]. The primary S/N cut (number of σ above the mean/median).
- **THRESHOLD** [none]. Alternatively to SNRCUT, the primary threshold can be given in the same flux units of the input datacube. This overrides SNRCUT.
- **FLAGGROWTH** [true]. Whether to grow detected sources to a secondary threshold.
- **GROWTHCUT** [3]. Secondary S/N cut used when growing objects (number of σ above the mean/median).

- **GROWTHTHRESHOLD** [none]. Alternatively to GROWTHCUT, the secondary threshold can be given in the same flux units of the input datacube. This overrides GROWTHCUT.
- **MINPIX** [beam area]. The minimum number of spatial pixels for a detection to be accepted. Default is the area covered by the observational beam.
- **MINCHANNELS** [2]. The minimum number of channels for a detection to be accepted.
- **MINVOXELS** [none]. The minimum number of voxels for a detection to be accepted. If not set, MINVOXELS = MINPIX*MINCHANNELS.
- **MAXCHANNELS** [none]. The maximum number of channels for a detection to be accepted. Default is no limits.
- **MAXANGSIZE** [none]. The maximum angular size of a detection to be accepted in *arcmin*. Default is no limits.
- **FLAGADJACENT** [true]. Whether to use the adjacent criterion to merge objects. If *false*, the next two parameters are used to determine whether objects are to be merged.
- **THRESHSPATIAL** [2]. The maximum minimum spatial separation in *pixels* for two objects to be merged into a single one. Ignored if FLAGADJACENT is *true*.
- **THRESHVELOCITY** [3]. The maximum minimum channel separation in *channels* for two objects to be merged into a single one. Ignored if FLAGADJACENT is *true*.
- **REJECTBEFOREMERGE** [true]. Whether to reject sources before merging them.
- **TWOSTAGEMERGING** [true]. Whether to do a partial merge during search.

3.5 SMOOTH task

This task convolves each channel map in a datacube with a given elliptical Gaussian.

- **SMOOTH** [false]. This flag enables the smooth algorithm. Can be *true* or *false*.
- **OBMAJ** [none]. Major axis of the initial beam in *arcsec*. Do not set if you want to use the beam information in the input FITS file (the parameter overrides it).
- **OBMIN** [none]. Minor axis of the initial beam in *arcsec*. Do not set if you want to use the beam information in the input FITS file (the parameter overrides it).
- **OBPA** [none]. Position angle of the major axis of the initial beam in *degrees*. Do not set if you want to use the beam information in the input FITS file (the parameter overrides it).
- **BMAJ** [none]. Major axis of the final beam in *arcsec*.
- **BMIN** [none]. Minor axis of the final beam in *arcsec*.
- **BPA** [none]. Position angle of the major axis of the final beam in *degrees*.
- **FACTOR** [2]. If set, the beam of the output cube is [FACTOR*OBMAJ,FACTOR*OBMIN,OBPA]. Ignored if BMAJ, BMIN, BPA are specified.
- **SCALEFACTOR** [none]. Scaling factor for output datacube. BBarolo will calculate an appropriate one if left unset.
- **FFT** [true]. Whether to convolve by using Fast Fourier Transform or not.
- **REDUCE** [false]. If *true*, BBarolo repixels the output datacube to preserve the number of pixels in a beam.
- **SMOOTHOUTPUT** [none]. Output smoothed FITS file. Default is input file name with a suffix indicating the new beam size.

3.6 2DFIT task

The classical 2D tilted-ring modelling of a galaxy: a model velocity field is fitted to the observed velocity field (see, e.g., Begeman 1987). This technique is fast and good for high spatial resolution data, but completely unreliable for low resolution data (no beam smearing correction).

- **2DFIT** [false]. This flag enables the 2D fitting of the velocity field.

Parameters and options that control the task are in common with *3DFIT*. In particular, 2DFIT supports the following parameters: **NRADII**, **RADSEP**, **XPOS**, **YPOS**, **VSYS**, **VROT**, **VRAD**, **PA**, **INC**, **FREE**, **SIDE**, **WFUNC**. The velocity field to fit is extracted using a mask for the input datacube defined by the **MASK** parameter. To check the velocity field out before the fit, make use of the **VELOCITYMAP** parameter (see *moment maps*).

3.7 Moment maps and position-velocity cuts

BBarolo can be used to extract global profiles, moment maps and position velocity diagrams. For moment maps and profile, the input datacube can be masked using the MASK parameter (see *3DFIT*).

- **GLOBALPROFILE** [false]. It *true*, calculate the total line profile from a datacube and write it to a text file.
- **TOTALMAP** [false]. It *true*, calculate the total intensity map from a datacube and write it to a FITS file.
- **VELOCITYMAP** [false]. It *true*, calculate the velocity field from a datacube and write it to a FITS file.
- **DISPERSIONMAP** [false]. It *true*, calculate the velocity dispersion field from a datacube and write it to a FITS file.
- **FLAGPV** [false]. If *true*, extract position-velocity image from a datacube and write it to a FITS file. The cut is defined by a point and an angle, as set with the following parameters.
- **XPOS_PV** [none]. Reference X pixel of the cut.
- **YPOS_PV** [none]. Reference Y pixel of the cut.
- **PA_PV** [none]. Position angle of the cut, defined anti-clockwise from the X direction.

CHAPTER

FOUR

INSTALLING PYBBAROLO

4.1 From pip

pyBBarolo is available as a package in the Python Package Index (PyPI). The easiest way of installing it is through `pip`:

```
> pip install pyBBarolo
```

This will download the package, compile BBarolo source code and install pyBBarolo in the python library path. Make sure you have permissions to write in the installing directory. Depending on your computer setup, you may need to run pip with superuser privileges (e.g.: `> sudo pip install pyBBarolo`).

N.B.: The above command will compile BBarolo, which means that your machine needs to have pre-installed all the libraries which the C++ code depends from (see *requirements*). If pip fails during compilation, please follow the procedure below.

4.2 Build and install

The python package can be alternatively installed from the main repository. You'll need to compile BBarolo as a library and then install pyBBarolo:

1. Follow steps 1-4 of procedure to *compile BBarolo from source*:

   ```
   > wget https://github.com/editeodoro/Bbarolo/archive/X.Y.tar.gz .
   > tar -xvzf Bbarolo-X.Y.tar && cd Bbarolo-X.Y
   > ./configure
   > make
   ```

   where *X.Y* is the software release.

2. Install the python package:

   ```
   > python setup.py install
   ```

If either compilation or installation fail, refer to BBarolo *compiling* and *troubleshooting* pages.

CHAPTER

FIVE

QUICKSTART

5.1 Running a task

In pyBBarolo, BBarolo's task are wrapped as python classes. The generic procedure to run a task is as follow:

1. Import the task from pyBBarolo module:

   ```
   from pyBBarolo import Task
   ```

   where Task is one of pyBBarolo *tasks*.

2. Create an object of the task class:

   ```
   bb = Task(fitsname)
   ```

   where *fitsname* is the input FITS file. All tasks need an initial FITS file.

3. Initialize the task:

   ```
   bb.init(args)
   ```

   where *args* are required arguments that depend on the various tasks.

4. Set options:

   ```
   bb.set_options(opts)
   ```

   where *opts* are the task-dependent available options. All options have default values, so this step is not mandatory. To see a list of available options:

   ```
   bb.show_options()
   ```

5. Run the task:

   ```
   bb.compute()
   ```

5.2 Example 1: 3D fit of a galaxy

Suppose you have an astonishing observation of your favorite galaxy and you want to fit a 3D kinematic model to your emission line datacube.

Let's fit the HI datacube of NGC 2403, which is available as a part of BBarolo's working examples. We just have to set initial conditions, options and then run the task.

First of all, we import and start FitMod3D:

```
from pyBBarolo import FitMod3D

# FITS file of the galaxy to model
filen = "./examples/ngc2403.fits"
# Initializing a 3DFIT object
f3d = FitMod3D(filen)
```

Secondly, we initialize rings with initial guesses for the fit:

```
# Initializing rings. Parameters can be values or arrays
f3d.init(radii=np.arange(15,450,30),xpos=77,ypos=77,vsys=132.8,vrot=120,vdisp=8,
↪vrad=0,z0=10,inc=60,phi=123.7)
```

Thirdly, we can change some default options for the fit. For a list of available options: `f3d.show_options()`.

For instance, we can set a mask made through the source-finding algorithm (`mask="SEARCH"`), parameters to fit (`free="VROT VDISP"`), the distance of the galaxy in Mpc (`distance=3.2`) and the directory for outputs (`outfolder='output/ngc2403`):

```
f3d.set_options(mask="SEARCH",free="VROT VDISP",wfunc=2,distance=3.2,outfolder=
↪'output/ngc2403')
```

If the beam information is not available in the FITS header, it is fundamental to set the size of the beam:

```
f3d.set_beam(bmaj=60,bmin=60,bpa=-80)
```

It is now time to run the fit:

```
bfrings, bestmod = f3d.compute()
```

This function performs the fit and writes relevant FITS files in the output directory. The function returns a n x m matrix containing the best-fit rings (`bfrings`), where n = number of rings and m = number of parameters, and a FITS astropy object (`bestmod`) containing the best-fit model. These are also written in `bfit` and `outmodel` methods of FitMod3D class.

Finally, we can use BBarolo built-in routines to write plots of data and model, like channel maps, moment maps, position-velocity diagrams and best-fit parameters:

```
f3d.plot_model()
```

5.3 Example 2: 3D model of a galaxy

It is also possible to simply build a 3D model datacube from given parameters. This is accomplished with the GalMod task. The procedure is similar to the one above:

```
from pyBBarolo import GalMod

# FITS file of the galaxy to model
filen = "./examples/ngc2403.fits"
# Initializing a GalMod object
gm = GalMod(filen)
# Initializing rings. Parameters can be values or arrays
gm.init(radii=np.arange(15,1200,30),xpos=74,ypos=74,vsys=132.8,vrot=120,vrad=10,
↪vvert=5,vdisp=8,z0=10,inc=60,phi=123.7)
```

```
# Now, let's take a look to the default options (see BB documentation)
gm.show_options()
# Changing some options
gm.set_options(ltype=1)
# Compute the model
mymodel = gm.compute()
# Smooth to the same resolution of data
mymodel = gm.smooth()
# mymodel is an astropy cube and we can do whatever we like with it.
mymodel.writeto("awesome_model.fits",overwrite=True)
```

5.4 Example 3: 3D model of an outflow

CHAPTER

SIX

AVAILABLE TASKS

CHAPTER

SEVEN

API

class pyBBarolo.pyBBarolo.**FitMod3D**(*fitsname*)
Bases: *pyBBarolo.pyBBarolo.Model3D*

Fit a galaxy model to a 3D datacube

Args: fitsname (str): FITS file of the galaxy to fit

compute()
Compute the model

This function needs to be called after *init()*.

init(**args*, ***kwargs*)
Initialize the model. Refer to derived class for *args and **kwargs.

plot_model()
Plot the model using BBarolo built-in output functions.

set_beam(*bmaj*, *bmin*, *bpa=0*)
Change the Beam parameters

Args: bmaj (float): Beam major axis in arcsec bmin (float): Beam minor axis in arcsec bpa (float): Beam position angle in degrees (default 0.0)

set_options(***kwargs*)
Set options for the task. Keywords **kwargs are given in self._opts.

show_options()
Show current options for the task

smooth(*beam=None*)
Smooth the model and return the smoothed array

This function needs to be called after *compute()*. The smoothed array is written in outmodel instance and returned.

Args:

beam (tuple,list): The beam in the form (bmaj,bmin,bpa), bmaj and bmin in arcs. If None, use the beam from the FITS header.

Returns: astropy PrimaryHDU: a datacube with the output smoothed model

class pyBBarolo.pyBBarolo.**FitsCube**(*fitsname*)
Bases: object

setBeam(*bmaj*, *bmin*, *bpa=0*)
Change the Beam parameters

Args: bmaj (float): Beam major axis in degrees bmin (float): Beam minor axis in degrees bpa (float): Beam position angle in degrees (default 0.0)

class pyBBarolo.pyBBarolo.**GalMod**(*fitsname*)
Bases: *pyBBarolo.pyBBarolo.Model3D*

Produce a simulated 3D datacube of disk galaxy

Args: fitsname (str): FITS file of the galaxy to model

compute()
Compute the model

This function needs to be called after *init()*.

init(**args*, ***kwargs*)
Initialize the model. Refer to derived class for *args and **kwargs.

set_beam(*bmaj*, *bmin*, *bpa=0*)
Change the Beam parameters

Args: bmaj (float): Beam major axis in arcsec bmin (float): Beam minor axis in arcsec bpa (float): Beam position angle in degrees (default 0.0)

set_options(***kwargs*)
Set options for the task. Keywords **kwargs are given in self._opts.

show_options()
Show current options for the task

smooth(*beam=None*)
Smooth the model and return the smoothed array

This function needs to be called after *compute()*. The smoothed array is written in outmodel instance and returned.

Args:

beam (tuple,list): The beam in the form (bmaj,bmin,bpa), bmaj and bmin in arcs. If None, use the beam from the FITS header.

Returns: astropy PrimaryHDU: a datacube with the output smoothed model

class pyBBarolo.pyBBarolo.**GalWind**(*fitsname*)
Bases: *pyBBarolo.pyBBarolo.Model3D*

Produce a simulated 3D datacube of a biconical outflow

Args: fitsname (str): FITS file of the galaxy to fit

compute()
Compute the model

This function needs to be called after *init()*.

init(**args*, ***kwargs*)
Initialize the model. Refer to derived class for *args and **kwargs.

set_beam(*bmaj*, *bmin*, *bpa=0*)
Change the Beam parameters

Args: bmaj (float): Beam major axis in arcsec bmin (float): Beam minor axis in arcsec bpa (float): Beam position angle in degrees (default 0.0)

set_options(***kwargs*)
Set options for the task. Keywords **kwargs are given in self._opts.

show_options()
Show current options for the task

smooth(*beam=None*)
Smooth the model and return the smoothed array

This function needs to be called after *compute()*. The smoothed array is written in outmodel instance and returned.

Args:

beam (tuple,list): The beam in the form (bmaj,bmin,bpa), bmaj and bmin in arcs. If None, use the beam from the FITS header.

Returns: astropy PrimaryHDU: a datacube with the output smoothed model

class pyBBarolo.pyBBarolo.**Model3D**(*fitsname*)
Bases: *pyBBarolo.pyBBarolo.Task*

Superclass for all 3D models, derived from Task

Args: fitsname (str): FITS file of the galaxy to model

compute()
Compute the model

This function needs to be called after *init()*.

init(**args*, ***kwargs*)
Initialize the model. Refer to derived class for *args and **kwargs.

set_beam(*bmaj*, *bmin*, *bpa=0*)
Change the Beam parameters

Args: bmaj (float): Beam major axis in arcsec bmin (float): Beam minor axis in arcsec bpa (float): Beam position angle in degrees (default 0.0)

set_options(***kwargs*)
Set options for the task. Keywords **kwargs are given in self._opts.

show_options()
Show current options for the task

smooth(*beam=None*)
Smooth the model and return the smoothed array

This function needs to be called after *compute()*. The smoothed array is written in outmodel instance and returned.

Args:

beam (tuple,list): The beam in the form (bmaj,bmin,bpa), bmaj and bmin in arcs. If None, use the beam from the FITS header.

Returns: astropy PrimaryHDU: a datacube with the output smoothed model

class pyBBarolo.pyBBarolo.**Rings**(*nrings*)
Bases: object

Wrapper class for C++ Rings structure (galmod.hh)

set_rings(*radii*, *xpos*, *ypos*, *vsys*, *vrot*, *vdisp*, *vrad*, *vvert*, *dvdz*, *zcyl*, *dens*, *z0*, *inc*, *phi*, *nv*)
Define rings given the input parameters

param radii: List or array param other: float or array

class pyBBarolo.pyBBarolo.**Search**(*fitsname*)
Bases: *pyBBarolo.pyBBarolo.Task*

3D Source finder

Args: fitsname (str): FITS file to search for sources

search()
Perform the source finding

set_options(***kwargs*)
Set options for the task. Keywords **kwargs are given in self._opts.

show_options()
Show current options for the task

class pyBBarolo.pyBBarolo.**Task**(*fitsname*)
Bases: object

Superclass for all BBarolo tasks

Args: fitsname (str): Input FITS file

set_options(***kwargs*)
Set options for the task. Keywords **kwargs are given in self._opts.

show_options()
Show current options for the task

pyBBarolo.pyBBarolo.**reshapePointer**(*p*, *shape*)
Take a POINTER to c_float and reshape it.

Args: p (POINTER(c_float)): The pointer to be reshaped shape (tuple, list): The new shape of the array

Returns: ndarray: The reshaped array

CHAPTER

EIGHT

LICENSE AND CITATIONS

BBarolo and **pyBBarolo** are distributed under the terms of the GNU General Public License version 3.0. The text of the license is included in the main directory of the repository as LICENSE.

If you use **BBarolo** or **pyBBarolo** in any published work, please cite the code paper: 3D BAROLO: a new 3D algorithm to derive rotation curves of galaxies, Di Teodoro & Fraternali 2015, MNRAS, 451, 3021.

CHAPTER

NINE

TROUBLESHOOTING

I will try to write a troubleshooting page while I receive the feedback from BBarolo's users.

In the meanwhile, please report any bug or problem you have with BBarolo and pyBBarolo. If you are a Github user, you can submit an issue ticket at this page. Otherwise you can mail me. Please attach any significant error messages and tell me how to reproduce the problem.

I will try to fix the issues as soon as I can. Thank you.

PYTHON MODULE INDEX

p

INDEX