A tool for fitting interferometric data

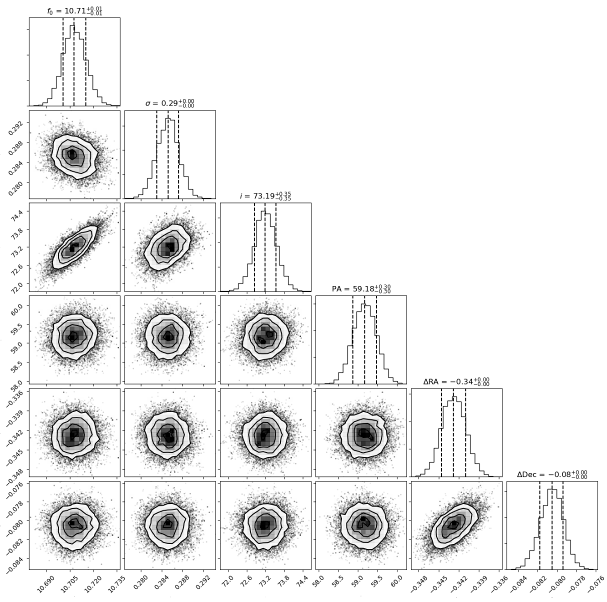
https://gitlab.mpcdf.mpg.de/szucs/SimpleDiskEnvFit (https://gitlab.mpcdf.mpg.de/szucs/SimpleDiskEnvFit)

Goals

- Assumption: a simple, physically motivated model describes the observed emission
- Optimise the model parameters to best recover the observations
- Determine the posterior distribution of model parameters

Example:

Fitting complex visibility data with parametric model.



Note that this example uses emcee and galario only.

Features and options:

- Compute and visualise model parameter probability distribution
- Envelope models: Ulrich+ (1976), Tafalla+ (2004)
- Disk model: parametric disk with hydrostatic-like vertical distribution
- Compute dust opacity using Mie theory on the fly
- Compute dust continuum emission maps and complex visibilities
- Compute χ^2 when observed complex visibilities are provided
- Parallelisation using MPI (on cluster) or Python threads (locally)

Components

RADMC-3D and radmc3dPy

Radiative transport tool for computing dust temperature and dust continuum emission (images).

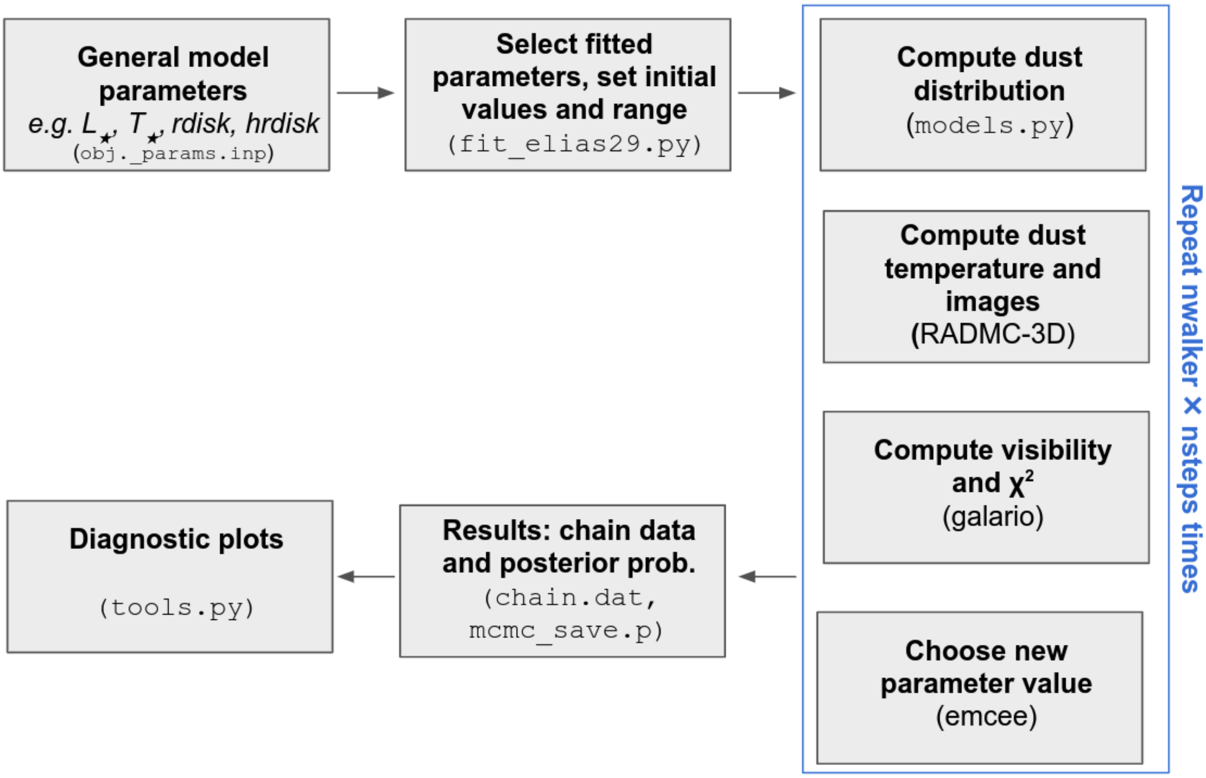
Galario

Deprojects observational data, samples model image at observation (u,v) positions and computes the χ^2 of model.

emcee

Affine invariant Markov-chain Monte Carlo sampler library: decides on next set of models based on the likelihood (χ^2) of the current set of models.

Workflow


General model parameters
e.g. L⋆, T⋆, rdisk, hrdisk
(obj._params.inp)
Select fitted parameters, set initial values and range
(fit_elias29.py)
Compute dust distribution
(models.py)
Compute dust temperature and images
(RADMC-3D)
Compute visibility and χ2
(galario)
Choose new parameter value
(emcee)
Repeat nwalker × nsteps times
Results: chain data and posterior prob.
(chain.dat, mcmc_save.p)
Diagnostic plots
(tools.py)

Model parameter file

Contains all fitted and not fitted model parameters. The not fitted parameters have the same value in all models.

Example is provided in `examples/elias29/elias29_params.inp`

```
# Read parameter file
par = SimpleDiskEnvFit.getParams(paramfile='elias29_params.inp')

# List parameter dictionary
par.ppar

# Set parameter value in python
par.setPar(["mdisk","0.01*ms"])
```

In [1]:
```
cat ../../examples/elias29/elias29_params.inp
```

```
#
# Elias 29 parameter setup
#
#
-------------------------------------------------------------------------------
-------------------------------------------
# Block: Radiation sources
#
-------------------------------------------------------------------------------
-------------------------------------------
mstar                     = [3.0*ms]         # Mass of the star(s)
pstar                     = [0.0, 0.0, 0.0]  # Position of the star(s) (cartes
ian coordinates)
rstar                     = [5.9*rs]         # Radius of the star(s)
tstar                     = [4786.0]         # Effective temperature of the st
ar(s) [K]
#
-------------------------------------------------------------------------------
-------------------------------------------
# Block: Grid parameters
#
-------------------------------------------------------------------------------
-------------------------------------------
crd_sys                   = 'sph'            # Coordinate system used (car/cy
l)
xbound                    = [1.0*au,20*au,50000.0*au]  # Boundaries for the x
grid
nx                        = [50,50]          # Number of grid points in the fi
rst dimension
ybound                    = [0.,pi/2.]       # Boundaries for the y grid
ny                        = 60               # Number of grid points in the se
cond dimension
zbound                    = [0., 2.0*pi]     # Boundaries for the z grid
nz                        = 0                # Number of grid points in the th
ird dimension
nw                        = [50, 150, 100]   # Number of points in the wavelen
gth grid
wbound                    = [0.1, 7.0, 25.0, 1e4]  # Boundaries for the wavele
ngth grid
#
-------------------------------------------------------------------------------
-------------------------------------------
# Block: Dust opacity
#
-------------------------------------------------------------------------------
-------------------------------------------
lnk_fname                 = 'porous_natta2004_rhod_1.36.lnk' # Optical constan
t (n,k) input file name
gdens                     = 1.36                             # Grain material
bulk density
ngpop                     = 2                                # Number of grain
populations
gsmin                     = [0.1 * 1.0e-4, 0.1 * 1.0e-4]     # Minimum grain s
ize [disk, envelope]
gsmax                     = [1. * 1.0e-4, 0.1 * 1.0e-4]      # Maximum grain s
ize [disk, envelope]
gsdist_powex              = -3.5                             # Power law index
of grain size distribution
ngs                       = 30                               # Number of grain
size bins between gsmin and gsmax
#
-------------------------------------------------------------------------------
-------------------------------------------
# Block: Code parameters
#
-------------------------------------------------------------------------------
-------------------------------------------
istar_sphere              = 1     # 1 - take into account the finite size of t
```

Fitting script

Example is provided in `examples/fit_elias29.py`

Make sure that:

- `visdata` dictionary is provided with keywords: u [m], v [m], Re [Jy], Im [Jy], w [1/sigma^2]
- `impar` dictionary is provided (`npix`, `wav`, `sizeau`, `incl` keywords)
- fitted parameter names listed in `parname`
- aprior parameter range is set
- initial parameter for each walker is provided
- `nstep`, `nwalker` and `use_mpi` parameters are set in `__main__`

Example:

```
# Read observational constraints
u1, v1, Re1, Im1, w1 = np.require(np.loadtxt('Elias29uvt_270.txt', unpack=Tru
e), requirements='C')
u2, v2, Re2, Im2, w2 = np.require(np.loadtxt('Elias29uvt_94.txt', unpack=Tru
e), requirements='C')

# Bundle visibility data
visdata = [{'u':u1, 'v':v1, 'Re':Re1, 'Im':Im1, 'w':w1, 'wav':1100.},
           {'u':u2, 'v':v2, 'Re':Re2, 'Im':Im2, 'w':w2, 'wav':3000.}]

# Set image parameters
impar = [{'npix':512,'wav':[1100.,3000.],'sizeau':11000,'incl':67.}]

# Projection parameters already known
parname = ['mdisk','rho0Env','gsmax_disk','gsmax_env']
p_ranges = [[-10., -2.],    # log disk mass [solar mass]
            [-23., -19.],   # log envelope density [g/cm**3]
            [-6., 0.],      # log disk grain size [cm]
            [-6., 0.]]      # log envelope grain size [cm]

# Initial guess for the parameters
p0 = [-5, -20, -4., -4.]
pos = [p0 + 1.0e-2*np.random.randn(ndim) for i in range(nwalkers)]
```

Batch script

This is used for requesting resources and queuing the pyhton script on the cluster.

In [2]:
```
cat ../../examples/elias29_slurm.sh
```

```
#!/bin/bash -l

#
## SimpleDiskEnvFit example to submit fitting job on SLURM
##
## Run fitting on 80 cores of 2 nodes.
##
## Tested on ccas cluster at MPCDF

#SBATCH --job-name=faust-elias29
#SBATCH --time=24:00:00
#SBATCH --partition=ccas256
#SBATCH --ntasks-per-node=40
#SBATCH --nodes=2
#SBATCH --mem=5gb

date

## Load required software
module load intel
module load mkl
module load fftw
module load impi
module load anaconda

## Make sure that site-packages are available
export PYTHONPATH=$PYTHONPATH:~/.local/lib/python2.7/site-packages
## Make sure that radmc3d binary is avaialble
export PATH=$PATH:~/bin

# Change to model directory, this is used as resource_dir in SimpleDiskEnvFit
cd ~/elias29
pwd

echo "Starting thread:" $SLURM_ARRAY_TASK_ID

srun -n 80 python fit_elias29.py
```

Reading and plotting data

The following commands give examples for reading ASCII and binary format chain files to `emcee_chain` type objects.

The minimum data stored in a `emcee_chain` object are the following: the chain (parameter values explored by the walkers, with dimension [nwalkers, nsteps, ndim]), logarithm of posterior probability (dimension [nwalkers, nsteps]), number of walkers (nwalkers), steps (nsteps) and fitted parameters (ndim) and the name of the fitted parameters (dimension [ndim]).

```
from SimpleDiskEnvFit import tools

# Read ASCII data
results = tools.read_chain_ascii('chain.dat')

# Read binary (pickle) data
results = tools.read_chain_pickle('chain.p')
```

See also [wiki (https://gitlab.mpcdf.mpg.de/szucs/SimpleDiskEnvFit/wikis/Working-with-emcee-chains)](https://gitlab.mpcdf.mpg.de/szucs/SimpleDiskEnvFit/wikis/Working-with-emcee-chains).

Diagnostic plots

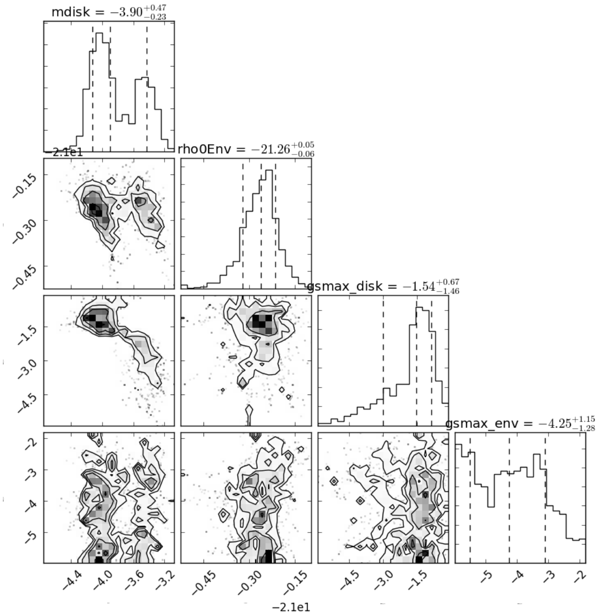
The `emcee_chain` class provides methods for plotting the distribution of explored parameters (using corner plots), the progression of parameter combinations and the posterior probability value in the chain.

```
# Show posterior parameter distribution
results.plot_corner(show=True, save=False)

# Show walker progression
results.plot_chain(show=True, save=False)

# Show posterior progression
results.plot_lnprob(show=True, save=False)
```

See also [wiki (https://gitlab.mpcdf.mpg.de/szucs/SimpleDiskEnvFit/wikis/Working-with-emcee-chains)](https://gitlab.mpcdf.mpg.de/szucs/SimpleDiskEnvFit/wikis/Working-with-emcee-chains).

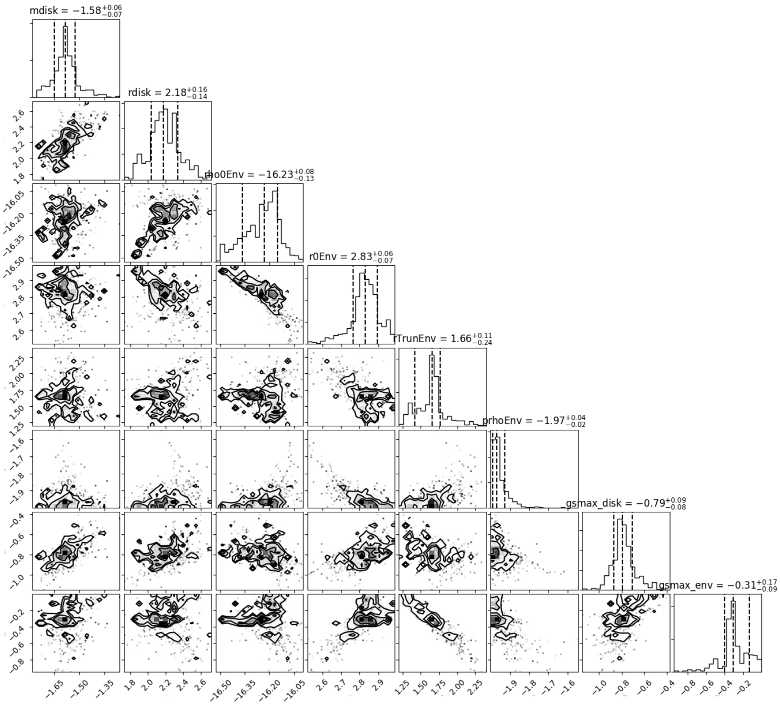
Elias 29

- Fitting pre-ALMA data at 1.2 and 3 mm wavelenght.
- Envelope density constrained
- Disk mass degenerate with grain size

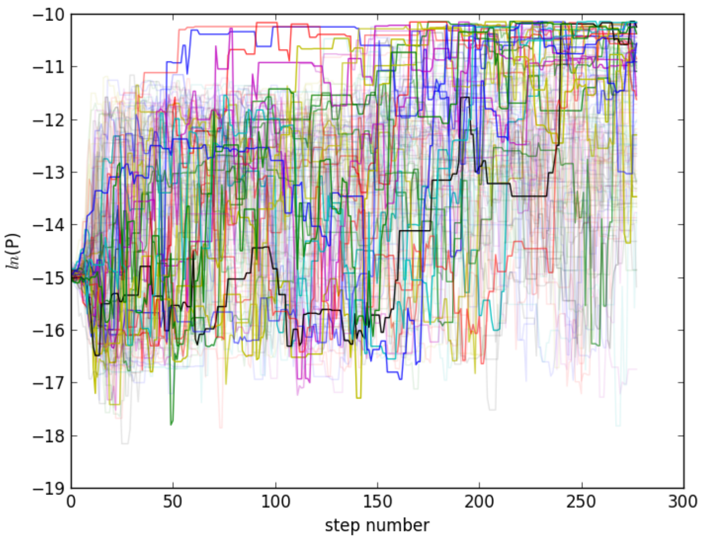


B11

Embedded binary system, ALMA data.



Chain example



What to expect and MCMC advices

- Do not expect such well sampled posterior distributions as with analytic model
- Do not fit too many parameters at once
- Use as many runners as it is practical
- Check acceptence rates (should be around 0.5)
- Check posterior probability progression of chain
- Do not necessary trust the corner plot: it show frequence rather than likelyhood

Caviats

- Long per model runtime with ALMA data ($\geq$3 min per model)
- Difficult sample posterior distribution as well as with analytic model
- Inconvenient visual inspection of model / input visibilities
- Still in developement (expect bugs)

Fine-tuning

- Adding and changing model components: edit `SimpleDiskEnvFit/models.py`
- Adding new dust optical constants: either to model folder or to `SimpleDiskEnvFit/lnk_files`

Requirements:

- python (2.7 or 3.5+)
- matplotlib
- numpy
- radmc3dPy (https://www.ast.cam.ac.uk/~juhasz/radmc3dPyDoc)
- galario (https://github.com/mtazzari/galario)
- uvplot (https://github.com/mtazzari/uvplot)
- emcee (http://dfm.io/emcee) (used for MCMC fitting)
- corner (https://corner.readthedocs.io)
- mpi4py
- RADMC3D (http://www.ita.uni-heidelberg.de/~dullemond/software/radmc-3d)

See readme (https://gitlab.mpcdf.mpg.de/szucs/SimpleDiskEnvFit/blob/master/README.md#requirements) file.

Installation:

Download the repository in your browser or using the git clone utility.

Use Python's distutil utility and the provided setup.py script.

```
$ python setup.py install --user
```

On linux this installes the module to ~/.local/lib/python{2.7/3.6}/site-packages directory, which is usually included in the python search path.

Alternatively, you may directly add the repository location to your PYTHONPATH:

```
$ export PYTHONPATH=$PYTHONPATH:/path/to/your/SimpleDiskEnvFit/directory
```

You can make this addition permanent by saving the export command to the ~/.bashrc or ~/.profile files.

After completing the installation step, the module should be available in python:

```
import SimpleDiskEnvFit
```

See readme (https://gitlab.mpcdf.mpg.de/szucs/SimpleDiskEnvFit/blob/master/README.md#installation) file.