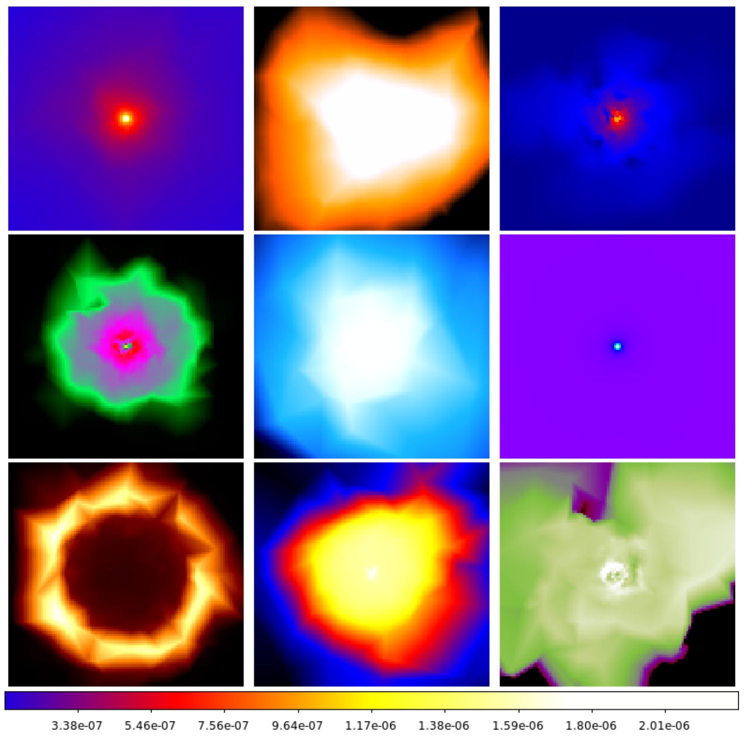


The Adaptable Radiative Transfer Innovations for Submillimeter Telescopes (ARTIST) interface with the Common Astronomy Software Applications (CASA)

European ALMA Regional Center ALLEGRO

Leiden University
Leiden, the Netherlands

October 2017

I General description

Radiation transfer tools are indispensable to plan or analyze observations with large facilities such as ALMA. Numerous advanced codes exist to calculate the structure of astronomical objects (density, temperature, velocity field, abundances of atomic or molecular species) and to determine the emerging emission of selected tracers (dust thermal continuum, emission or absorption lines of atoms or molecules). While many of these codes allow great realism in the treatment of object's astrophysics, they also require significant time investment to master.

The *ARTIST* project[1] (ARTIST) was conceived to lower the threshold in using popular star formation models and produce reliable quantitative information from the expected observables. The *ARTIST* package contains 9 models of protostars and planet-forming disks, with the associated parameters necessary to setup a full source model. Users no longer need to supply additional piece(s) of code in the construction of the source's astrophysics. Once a model is constructed, *ARTIST* uses the *LIME*[2] molecular excitation and radiation transfer code to calculate the resulting emission of selected molecular transitions and/or dust continuum emission, for a given source distance and orientation. This can be done in full statistical equilibrium of the molecular excitation, or by an assumption that the excitation follows Local Thermal Equilibrium (LTE) with the gas temperature.

In its original setup, *ARTIST* used a GUI to build the object's astrophysical model, and produce FITS image cubes. To further process these cubes and produce observables with, e.g., ALMA, the user needed to switch to CASA (the ALMA software) and carry out the desired processing steps.

To further lower the threshold for non-expert users, we have developed a CASA interface to *ARTIST*. The *ARTIST* models can now be set up through a keyword-driven interface within CASA, according to the principles that CASA users are fully familiar with. Once the astrophysical model is selected, the parameters defined, and the imaging keywords are set, the *LIME* radiation transfer calculation can be launched from the CASA prompt, and the requested FITS cubes are produced.

This document gives a succinct description of a typical CASA interface to *ARTIST* usage. For details of the astrophysical models included in *ARTIST*, users can refer to the original documentation of the *ARTIST* package (see below). For details on the inner workings of LIME and a full description of the image generation options, users may refer to the documentation of the LIME package. This document only gives top level descriptions on how to install the CASA ARTIST package including LIME, and illustrates a few typical use cases. This should allow the user to start in fairly short time and further explores *ARTIST-to-CASA* independently.

II Useful links

The links to the various documentations of the packages used in the interface is listed below :

[1]`http://youngstars.nbi.dk/artist/Welcome.html`
[2]http://www.nbi.dk/ brinch/index.php?page=lime

- CASA (`https://casa.nrao.edu/`): the main package for data reduction and imaging of interferometric data.
- ARTIST (`http://youngstars.nbi.dk/artist/Welcome.html`): a radiative transfer package for modelling submillimeter observations (see also ARTIST).
- LIME (`http://www.nbi.dk/~brinch/index.php?page=lime`): the versatile molecular line excitation and radiative transfer tool (see also LIME).
- Model Library (see § VI): a list of the analytical/semi-analytical models provided within the *ARTIST* framework.
- Molecular file (`http://home.strw.leidenuniv.nl/~moldata/`): the molecular data file that can be read by the ARTIST package.
- Dust opacity file: see *ARTIST* or *LIME* documentations for the appropriate format.

III Tasks within the interface

ARTIST solves the astrophysical problem in two steps. In the first step (through the task `limesolver`) the user selects the model of choice, provides the necessary parameters, selects the molecule of interest, and provides the information that the LIME excitation solver needs to determine the excitation of the molecule across the source.

In the second step (through the task `raytrace`), the user provides a distance, orientation of the source and selects the molecular transition to image. The user further provides the necessary imaging input parameters such as angular and spectral resolution. Subsequently, the *LIME* raytracing tool produces the FITS cube containing the specified observable. This cube can be further processed (e.g., convolved with a Gaussian beam or used as an input for the full ALMA visibility modelling). The FITS cube is fully compatible with CASA.

IV Installation

The interface is distributed with the entire *ARTIST* package. The user must make sure that both *ARTIST* and CASA are correctly installed. A complete installation of ARTIST and CASA requires the following steps:

- CASA (version > 4.6) as obtained from `https://casa.nrao.edu/`.
- The following libraries and programs are needed:
  - `gcc-c++`
  - `cfitsio` and `ciftsio-devel`
  - `GSL` and `GSL-devel`
  - `ncurses-devel`
  - `qhull` and `qhull-dev`

    - `python2` and `python-devel`
    - `swig`

  See the README file in the folder ARTISTROOT[3] for more information about necessary programs and libraries.

- ARTIST installation instructions can be found file in the file ARTISTROOT/README. The installation consists of the execution of the configuration step and the compilation step. A successful installation of artist will give the users access to `lime` and `modellib` packages.

- The python script ARTISTROOT/test/testModels.py performs test runs on all models implemented in the `modellib` package. The successful execution of this script ensures that both *ARTIST* and *LIME* are installed correctly.

- The CASA interface is located in ARTISTROOT/casa. The file ARTISTROOT/doc/README_CASA.txt informs the users on the available tasks and details on the compilation. The interface needs to be compiled within the interface directory with `buildmytasks` that is distributed with CASA. Finally, the incorporation of the generated files have to be included in the CASA initialization file.

V Example sessions

In the following, we provide two typical example models with ARTIST-for-CASA. Through these two cases, we give the user a global overview of the different options to setup *ARTIST* models, define *LIME* excitation calculations, and *LIME* image generation parameters. This is not an exhaustive description of all options (please refer to the separate documentation of the *ARTIST* models and the *LIME* code), but does allow the user for a quick start and independent exploration of the interface's functionality. **Please note that the interfaces consider only a single molecular data file with a single collisional partner.**

V.I Spherical envelope

The first example adopts the Bonnor Ebert (http://adsabs.harvard.edu/abs/1955ZA.....37..217E and http://adsabs.harvard.edu/abs/1956MNRAS.116..351B) sphere is used to generate an image cube of the HCO^+ 1–0 line from an infalling sphere. The first step is to use the task *limesover* generate the physical model. The input parameters are shown below. The task performs the following

- Passes the parameters to the *BonnorEbert56* function within the *modellib* package of *ARTIST*.

- The model is passed to *LIME* where the grids are generated and the molecular excitation is calculated.

[3]The location of the unpacked folder

- A model grid is produced at the end of the task where it contains the *LIME* grid points, density, temperature, and level populations for the specified molecule.

The Bonnor Ebert sphere model has a self-consistent temperature and velocity structures. There is no need to insert temperature and velocity functions unless the user wants to modify them. The task will overwrite the self-consistent values with the given parameters. The figures below show the execution of the `limesolver` within CASA.

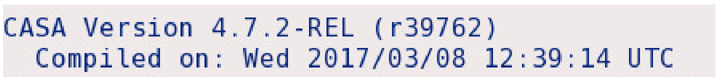
```
limesolver:  This task runs the LIne Modelling Engine
radius            = 3.0857e16
minScale          = 1.496e13
sinkPoints        = 2000
pIntensity        = 5000
nSolveIters       = 12
moldatfile        = ['hco+.dat']
dust              = 'jena_thin_e6.tab'
gridOutFile       = "BE56_test.fits"
modelID           = 'BonnorEbert56'
      T       = 50
      rhoc    = 5e6
bmag               = "vectorConstR"
bmag_args         = [0.0]
abundance          = "scalarConst"
abundance_args    = [1e-9]
doppler             = "scalarConst"
doppler_args      = [300.]
```

Minimum scale resolved by the grid

Points for excitation and intensity calculations

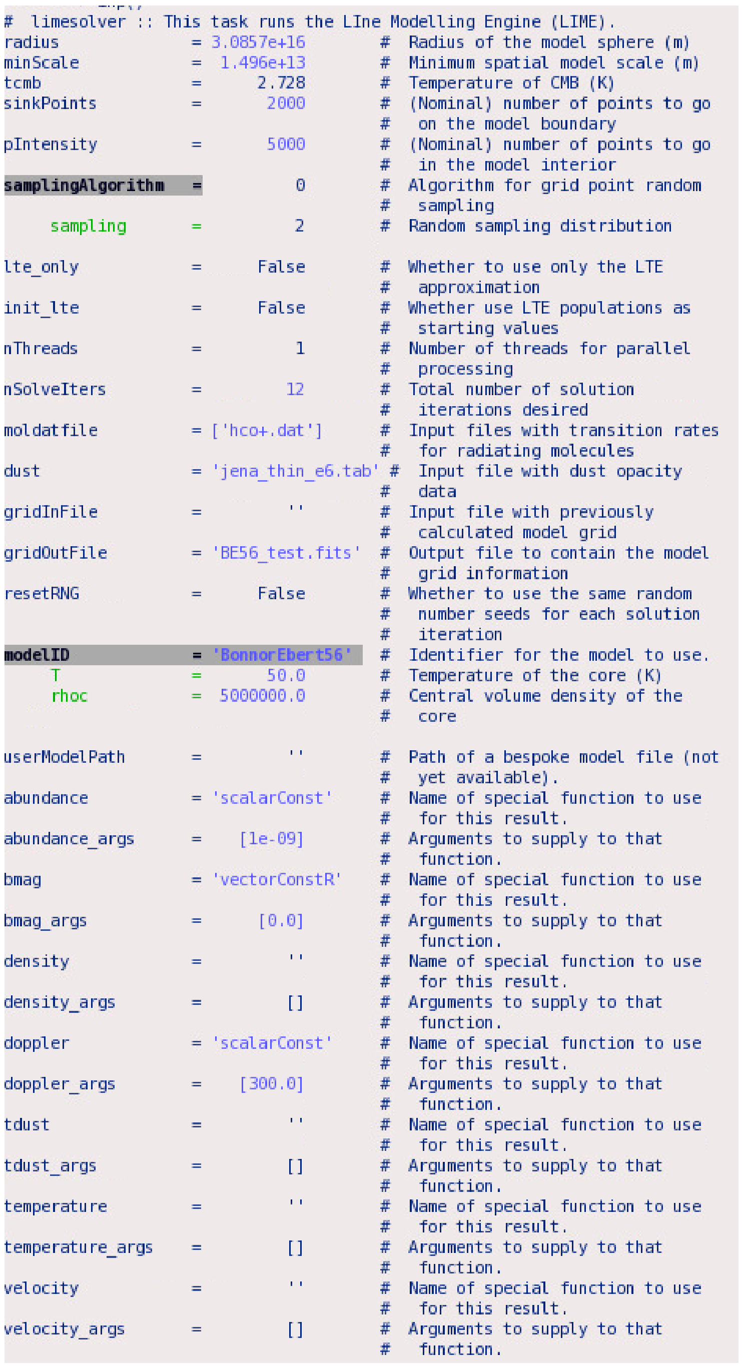
Molecular data file in the Leiden Atomic and Molecular DAtabase format see above

IMPORTANT: This needs to be defined and set as 0



```
CASA Version 4.7.2-REL (r39762)
  Compiled on: Wed 2017/03/08 12:39:14 UTC
```

Figure 1: CASA version used. (CASA version must be greater than 4.6)

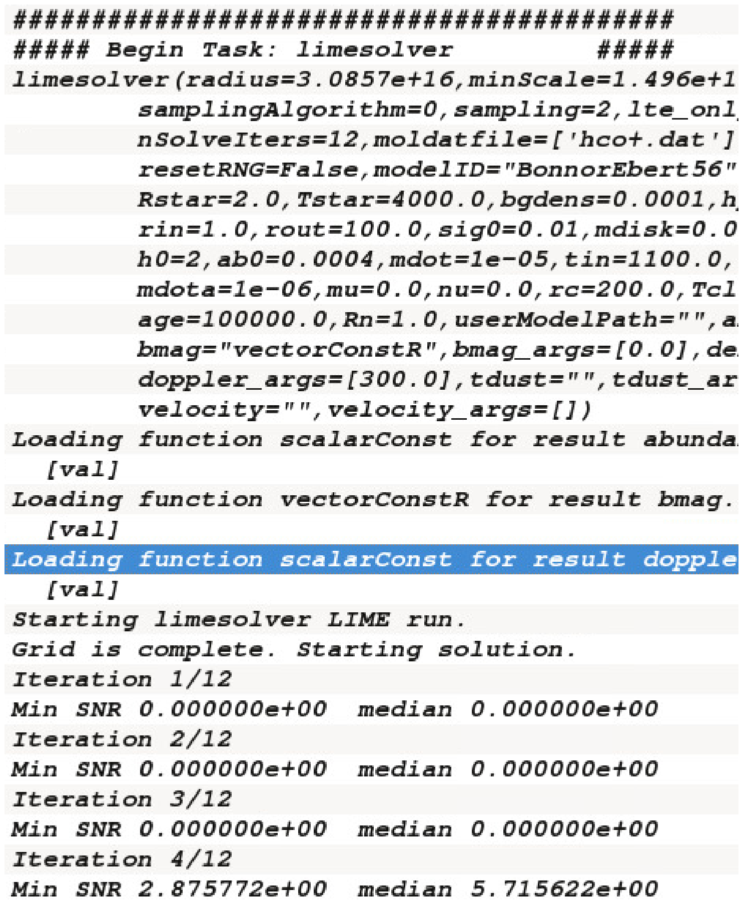


```
#  limesolver :: This task runs the LIne Modelling Engine (LIME).
radius              = 3.0857e+16        #  Radius of the model sphere (m)
minScale            =  1.496e+13        #  Minimum spatial model scale (m)
tcmb                =      2.728        #  Temperature of CMB (K)
sinkPoints          =       2000        #  (Nominal) number of points to go
                                        #   on the model boundary
pIntensity          =       5000        #  (Nominal) number of points to go
                                        #   in the model interior
samplingAlgorithm   =          0        #  Algorithm for grid point random
                                        #   sampling
     sampling       =          2        #  Random sampling distribution

lte_only            =      False        #  Whether to use only the LTE
                                        #   approximation
init_lte            =      False        #  Whether use LTE populations as
                                        #   starting values
nThreads            =          1        #  Number of threads for parallel
                                        #   processing
nSolveIters         =         12        #  Total number of solution
                                        #   iterations desired
moldatfile          = ['hco+.dat']      #  Input files with transition rates
                                        #   for radiating molecules
dust                = 'jena_thin_e6.tab' #  Input file with dust opacity
                                        #   data
gridInFile          =         ''        #  Input file with previously
                                        #   calculated model grid
gridOutFile         = 'BE56_test.fits'  #  Output file to contain the model
                                        #   grid information
resetRNG            =      False        #  Whether to use the same random
                                        #   number seeds for each solution
                                        #   iteration
modelID             = 'BonnorEbert56'   #  Identifier for the model to use.
     T              =       50.0        #  Temperature of the core (K)
     rhoc           =  5000000.0        #  Central volume density of the
                                        #   core

userModelPath       =         ''        #  Path of a bespoke model file (not
                                        #   yet available).
abundance           = 'scalarConst'     #  Name of special function to use
                                        #   for this result.
abundance_args      =    [1e-09]        #  Arguments to supply to that
                                        #   function.
bmag                = 'vectorConstR'    #  Name of special function to use
                                        #   for this result.
bmag_args           =      [0.0]        #  Arguments to supply to that
                                        #   function.
density             =         ''        #  Name of special function to use
                                        #   for this result.
density_args        =         []        #  Arguments to supply to that
                                        #   function.
doppler             = 'scalarConst'     #  Name of special function to use
                                        #   for this result.
doppler_args        =    [300.0]        #  Arguments to supply to that
                                        #   function.
tdust               =         ''        #  Name of special function to use
                                        #   for this result.
tdust_args          =         []        #  Arguments to supply to that
                                        #   function.
temperature         =         ''        #  Name of special function to use
                                        #   for this result.
temperature_args    =         []        #  Arguments to supply to that
                                        #   function.
velocity            =         ''        #  Name of special function to use
                                        #   for this result.
velocity_args       =         []        #  Arguments to supply to that
                                        #   function.
```

Figure 2: The limesolver task within CASA with the given parameters.



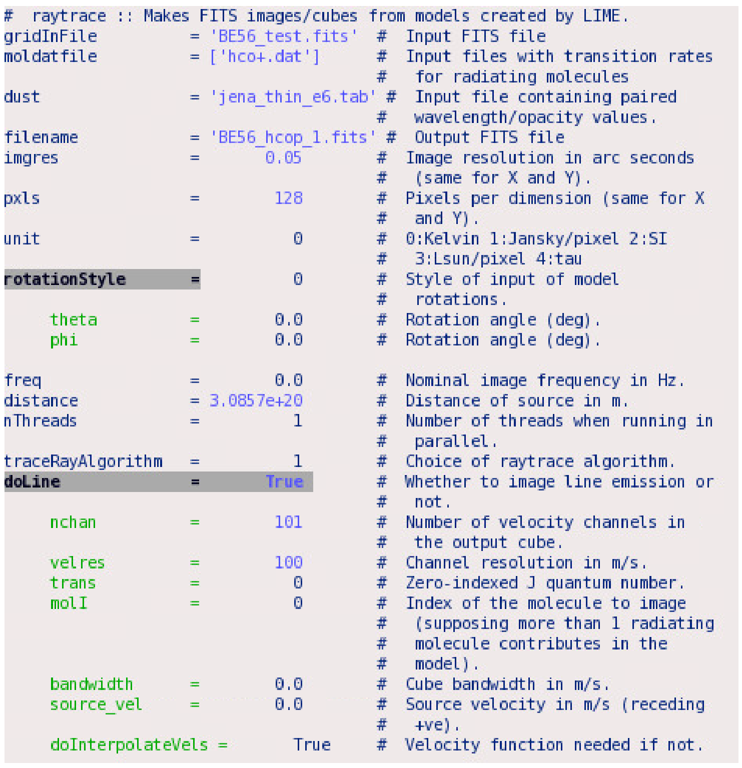
```
###########################################
##### Begin Task: limesolver         #####
limesolver(radius=3.0857e+16,minScale=1.496e+1
        samplingAlgorithm=0,sampling=2,lte_onl
        nSolveIters=12,moldatfile=['hco+.dat']
        resetRNG=False,modelID="BonnorEbert56"
        Rstar=2.0,Tstar=4000.0,bgdens=0.0001,h
        rin=1.0,rout=100.0,sig0=0.01,mdisk=0.0
        h0=2,ab0=0.0004,mdot=1e-05,tin=1100.0,
        mdota=1e-06,mu=0.0,nu=0.0,rc=200.0,Tcl
        age=100000.0,Rn=1.0,userModelPath="",a
        bmag="vectorConstR",bmag_args=[0.0],de
        doppler_args=[300.0],tdust="",tdust_ar
        velocity="",velocity_args=[])
Loading function scalarConst for result abunda
  [val]
Loading function vectorConstR for result bmag.
  [val]
Loading function scalarConst for result dopple
  [val]
Starting limesolver LIME run.
Grid is complete. Starting solution.
Iteration 1/12
Min SNR 0.000000e+00  median 0.000000e+00
Iteration 2/12
Min SNR 0.000000e+00  median 0.000000e+00
Iteration 3/12
Min SNR 0.000000e+00  median 0.000000e+00
Iteration 4/12
Min SNR 2.875772e+00  median 5.715622e+00
```

Figure 3: The execution of the task can be monitored through the CASA log.

The FITS image cube is produced by the task *raytrace*. The following parameters generate the first transition (J = 1–0) line with the given population levels calculated in the previous step (`BE56_test.fits`). The output file generated by `limesolver` is in FITS format.

```
raytrace:  Makes FITS images/cubes from models created by
LIME
gridInFile          = 'BE56_test.fits'
moldatfile          = ['hco+.dat']
dust                = 'jena_thin_e6.tab'
filename            = "BE56_hcop_1.fits'
imgres              = 0.05
pxls                = 128
distance            = 3.08572e20
doLine              = True
     nchan       = 50
     velres      = 500.0
     trans       = 0
     source_vel    = 3500.0
```

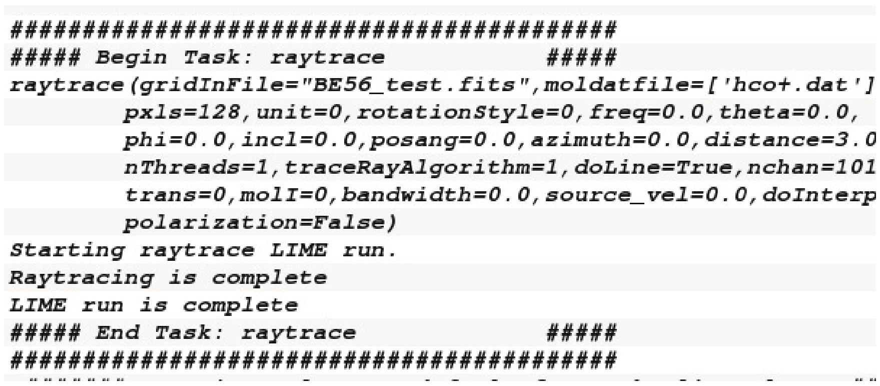
Please refer to the LAMDA database to set this integer



```
#  raytrace :: Makes FITS images/cubes from models created by LIME.
gridInFile          = 'BE56_test.fits'  #  Input FITS file
moldatfile          = ['hco+.dat']      #  Input files with transition rates
                                        #   for radiating molecules
dust                = 'jena_thin_e6.tab' #  Input file containing paired
                                        #   wavelength/opacity values.
filename            = 'BE56_hcop_1.fits' #  Output FITS file
imgres              =       0.05        #  Image resolution in arc seconds
                                        #   (same for X and Y).
pxls                =        128        #  Pixels per dimension (same for X
                                        #   and Y).
unit                =          0        #  0:Kelvin 1:Jansky/pixel 2:SI
                                        #   3:Lsun/pixel 4:tau
rotationStyle       =          0        #  Style of input of model
                                        #   rotations.
     theta          =        0.0        #  Rotation angle (deg).
     phi            =        0.0        #  Rotation angle (deg).

freq                =        0.0        #  Nominal image frequency in Hz.
distance            = 3.0857e+20        #  Distance of source in m.
nThreads            =          1        #  Number of threads when running in
                                        #   parallel.
traceRayAlgorithm   =          1        #  Choice of raytrace algorithm.
doLine              =       True        #  Whether to image line emission or
                                        #   not.
     nchan          =        101        #  Number of velocity channels in
                                        #   the output cube.
     velres         =        100        #  Channel resolution in m/s.
     trans          =          0        #  Zero-indexed J quantum number.
     molI           =          0        #  Index of the molecule to image
                                        #   (supposing more than 1 radiating
                                        #   molecule contributes in the
                                        #   model).
     bandwidth      =        0.0        #  Cube bandwidth in m/s.
     source_vel     =        0.0        #  Source velocity in m/s (receding
                                        #   +ve).
     doInterpolateVels =       True     #  Velocity function needed if not.
```

Figure 4: The raytrace task within CASA with the given parameters.



```
###########################################
##### Begin Task: raytrace            #####
raytrace(gridInFile="BE56_test.fits",moldatfile=['hco+.dat']
        pxls=128,unit=0,rotationStyle=0,freq=0.0,theta=0.0,
        phi=0.0,incl=0.0,posang=0.0,azimuth=0.0,distance=3.0
        nThreads=1,traceRayAlgorithm=1,doLine=True,nchan=101
        trans=0,molI=0,bandwidth=0.0,source_vel=0.0,doInterp
        polarization=False)
Starting raytrace LIME run.
Raytracing is complete
LIME run is complete
##### End Task: raytrace              #####
###########################################
```

Figure 5: Similarly, the execution of the task can be monitored through the CASA log.

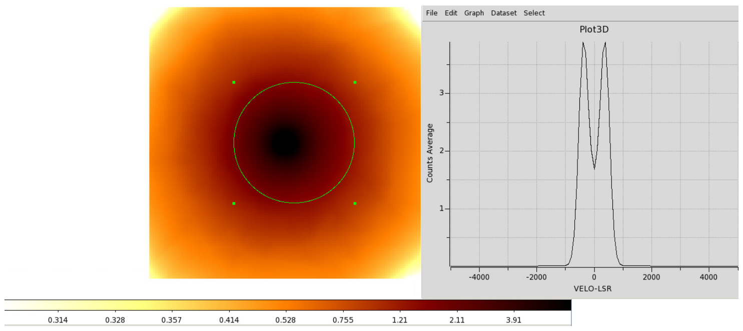
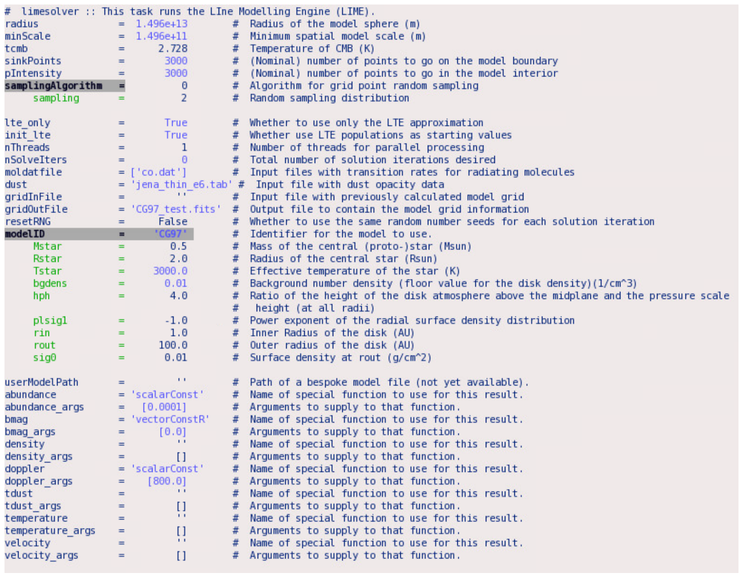


Figure 6: The FITS file generated by raytrace can be viewed with the user's favorite viewer.

V.II Chiang and Goldreich disk

A generic semi-analytical disk model is the Chiang & Goldreich type disk (http://adsabs.harvard.edu/abs/1997ApJ...490..368C). It is a self-consistent analytical model. This time, we will generate the CO 3-2 line from a protoplanetary disk.

```
limesolver:  This task runs the LIne Modelling Engine
Radius             = 1.496e13
minScale           = 1.496e11
sinkPoints         = 3000
pIntensity         = 3000
samplingAlgorithm = 0
     sampling    = 2
init_lte           = True
nSolveIters        = 5
moldatfile         = [‘co.dat’]
dust               = ‘jena_thin_e6.tab’
gridOutFile        = "CG97_test.fits"
modelID            = ‘CG97’
     Mstar     = 0.5
     Rstar     = 2.
     Tstar     = 3e3
     bgdens    = 1e-2
     hph       = 3.3
     plsig1    = -1.0
     rin       = 1.0
     rout      = 1e2
     sig0      = bgdens
abundance           = "scalarConst"
abundance_args     = [1e-4]
dopper              = "scalarConst"
doppler_args       = [0.8e3]
```

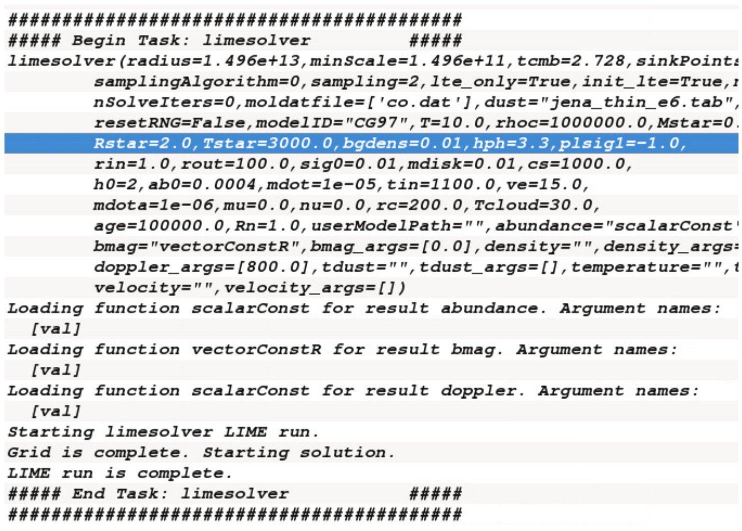


```
#  limesolver :: This task runs the LIne Modelling Engine (LIME).
radius              =  1.496e+13        #  Radius of the model sphere (m)
minScale            =  1.496e+11        #  Minimum spatial model scale (m)
tcmb                =      2.728        #  Temperature of CMB (K)
sinkPoints          =       3000        #  (Nominal) number of points to go on the model boundary
pIntensity          =       3000        #  (Nominal) number of points to go in the model interior
samplingAlgorithm   =          0        #  Algorithm for grid point random sampling
     sampling       =          2        #  Random sampling distribution

lte_only            =       True        #  Whether to use only the LTE approximation
init_lte            =       True        #  Whether use LTE populations as starting values
nThreads            =          1        #  Number of threads for parallel processing
nSolveIters         =          0        #  Total number of solution iterations desired
moldatfile          = ['co.dat']        #  Input files with transition rates for radiating molecules
dust                = 'jena_thin_e6.tab' #  Input file with dust opacity data
gridInFile          =         ''        #  Input file with previously calculated model grid
gridOutFile         = 'CG97_test.fits'  #  Output file to contain the model grid information
resetRNG            =      False        #  Whether to use the same random number seeds for each solution iteration
modelID             =     'CG97'        #  Identifier for the model to use.
     Mstar          =        0.5        #  Mass of the central (proto-)star (Msun)
     Rstar          =        2.0        #  Radius of the central star (Rsun)
     Tstar          =     3000.0        #  Effective temperature of the star (K)
     bgdens         =       0.01        #  Background number density (floor value for the disk density)(1/cm^3)
     hph            =        4.0        #  Ratio of the height of the disk atmosphere above the midplane and the pressure scale
                                        #   height (at all radii)
     plsig1         =       -1.0        #  Power exponent of the radial surface density distribution
     rin            =        1.0        #  Inner Radius of the disk (AU)
     rout           =      100.0        #  Outer radius of the disk (AU)
     sig0           =       0.01        #  Surface density at rout (g/cm^2)

userModelPath       =         ''        #  Path of a bespoke model file (not yet available).
abundance           = 'scalarConst'     #  Name of special function to use for this result.
abundance_args      =   [0.0001]        #  Arguments to supply to that function.
bmag                = 'vectorConstR'    #  Name of special function to use for this result.
bmag_args           =      [0.0]        #  Arguments to supply to that function.
density             =         ''        #  Name of special function to use for this result.
density_args        =         []        #  Arguments to supply to that function.
doppler             = 'scalarConst'     #  Name of special function to use for this result.
doppler_args        =    [800.0]        #  Arguments to supply to that function.
tdust               =         ''        #  Name of special function to use for this result.
tdust_args          =         []        #  Arguments to supply to that function.
temperature         =         ''        #  Name of special function to use for this result.
temperature_args    =         []        #  Arguments to supply to that function.
velocity            =         ''        #  Name of special function to use for this result.
velocity_args       =         []        #  Arguments to supply to that function.
```

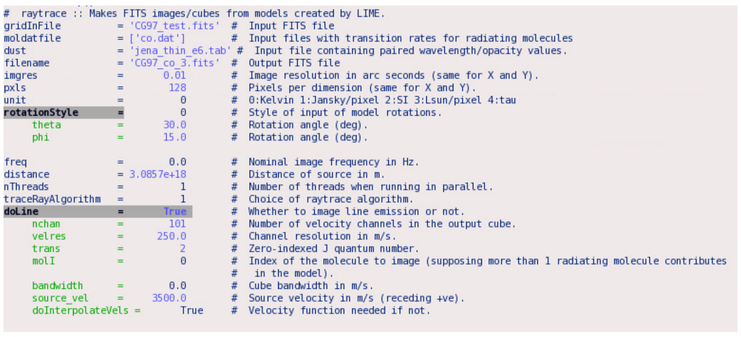
Figure 7: The `limesolver` task within CASA with the given parameters.



```
##########################################
##### Begin Task: limesolver         #####
limesolver(radius=1.496e+13,minScale=1.496e+11,tcmb=2.728,sinkPoint
        samplingAlgorithm=0,sampling=2,lte_only=True,init_lte=True,
        nSolveIters=0,moldatfile=['co.dat'],dust="jena_thin_e6.tab",
        resetRNG=False,modelID="CG97",T=10.0,rhoc=1000000.0,Mstar=0.
        Rstar=2.0,Tstar=3000.0,bgdens=0.01,hph=3.3,plsig1=-1.0,
        rin=1.0,rout=100.0,sig0=0.01,mdisk=0.01,cs=1000.0,
        h0=2,ab0=0.0004,mdot=1e-05,tin=1100.0,ve=15.0,
        mdota=1e-06,mu=0.0,nu=0.0,rc=200.0,Tcloud=30.0,
        age=100000.0,Rn=1.0,userModelPath="",abundance="scalarConst'
        bmag="vectorConstR",bmag_args=[0.0],density="",density_args
        doppler_args=[800.0],tdust="",tdust_args=[],temperature="",t
        velocity="",velocity_args=[])
Loading function scalarConst for result abundance. Argument names:
  [val]
Loading function vectorConstR for result bmag. Argument names:
  [val]
Loading function scalarConst for result doppler. Argument names:
  [val]
Starting limesolver LIME run.
Grid is complete. Starting solution.
LIME run is complete.
##### End Task: limesolver           #####
##########################################
```

Figure 8: The execution of the task for the LTE only run.

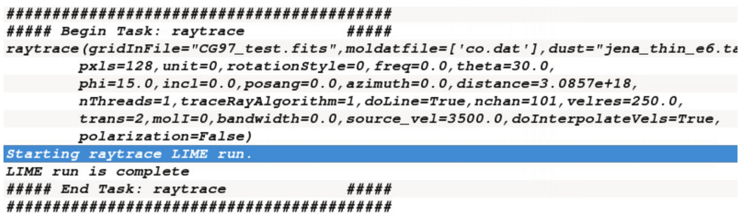
```
raytrace:  Makes FITS images/cubes from models created by
LIME
gridInFile          = 'CG97_test.vtk'
moldatfile          = ['co.dat']
dust                = 'jena_thin_e6.tab'
filename            = "CG97_co_3.fits'
imgres              = 128
theta               = 30.
phi                 = 15.
distance            = 3.08572e18
doLine              = True
     nchan      = 101
     velres     = 250.0
     trans      = 2
     source_vel   = 3500.0
```



```
#  raytrace :: Makes FITS images/cubes from models created by LIME.
gridInFile          = 'CG97_test.fits'  #  Input FITS file
moldatfile          = ['co.dat']        #  Input files with transition rates for radiating molecules
dust                = 'jena_thin_e6.tab' #  Input file containing paired wavelength/opacity values.
filename            = 'CG97_co_3.fits'  #  Output FITS file
imgres              =       0.01        #  Image resolution in arc seconds (same for X and Y).
pxls                =        128        #  Pixels per dimension (same for X and Y).
unit                =          0        #  0:Kelvin 1:Jansky/pixel 2:SI 3:Lsun/pixel 4:tau
rotationStyle       =          0        #  Style of input of model rotations.
     theta          =       30.0        #  Rotation angle (deg).
     phi            =       15.0        #  Rotation angle (deg).

freq                =        0.0        #  Nominal image frequency in Hz.
distance            = 3.0857e+18        #  Distance of source in m.
nThreads            =          1        #  Number of threads when running in parallel.
traceRayAlgorithm   =          1        #  Choice of raytrace algorithm.
doLine              =       True        #  Whether to image line emission or not.
     nchan          =        101        #  Number of velocity channels in the output cube.
     velres         =      250.0        #  Channel resolution in m/s.
     trans          =          2        #  Zero-indexed J quantum number.
     molI           =          0        #  Index of the molecule to image (supposing more than 1 radiating molecule contributes
                                        #   in the model).
     bandwidth      =        0.0        #  Cube bandwidth in m/s.
     source_vel     =     3500.0        #  Source velocity in m/s (receding +ve).
     doInterpolateVels =       True     #  Velocity function needed if not.
```

Figure 9: The raytrace task within CASA with the given parameters.



```
##########################################
##### Begin Task: raytrace           #####
raytrace(gridInFile="CG97_test.fits",moldatfile=['co.dat'],dust="jena_thin_e6.ta
        pxls=128,unit=0,rotationStyle=0,freq=0.0,theta=30.0,
        phi=15.0,incl=0.0,posang=0.0,azimuth=0.0,distance=3.0857e+18,
        nThreads=1,traceRayAlgorithm=1,doLine=True,nchan=101,velres=250.0,
        trans=2,molI=0,bandwidth=0.0,source_vel=3500.0,doInterpolateVels=True,
        polarization=False)
Starting raytrace LIME run.
LIME run is complete
##### End Task: raytrace             #####
##########################################
```

Figure 10: Similarly, the execution of the task can be monitored through the CASA log.

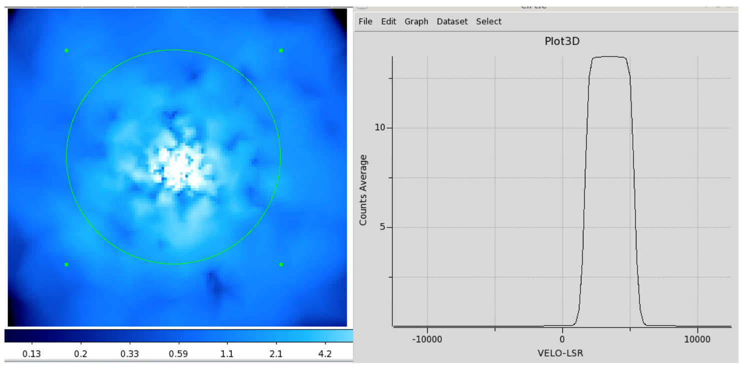


Figure 11: The FITS file generated by raytrace can be viewed with the user's favorite viewer.

VI Models library

Table 1: The model library contains 9 analytical and semi-analytical models. The table below lists the available models and their self-consistent physical parameters. It also lists the parameters that users must provide.

ModelID	Model Name		
		Provided parameters	Required input parameters
Mamon88	circumstellar envelope model of evolved stars	density, temperature, abundance, velocity	magnetic field, doppler
BonnorEbert56	Bonnor Ebert sphere	density, velocity temperature	abundance, magnetic field, doppler
Shu77	Shu 1977 spherical isothermal sphere model	density, velocity, temperature	abundance, magnetic field, doppler
Ulrich76	infall model with rotation	density, velocity	abundance, magnetic field, temperature, doppler
Mendoza09	analytical model of collapsing rotating finite cloud	density, velocity	abundance, doppler, magnetic field, temperature
LiShu96	magnetized isothermal toroid model	density, velocity magnetic field, temperature	abundance, doppler
Allen03a	Collapse model of magnetized isothermal toroids	density, velocity magnetic field, temperature	abundance, doppler
CG97	Chiang Goldreich 1997 disk model	density, velocity, temperature	abundance, doppler, magnetic field
DDN01	Dullemond, Dominik, and Natta 2001 disk model	density, velocity temperature	abundance, doppler, magnetic field

Built-in functions are used to define keyword parameters such as `abundance`, `bmag`, and similar physical parameters. Specifically, the physical parameters that require an input from the user. These functions are listed below[4].

`scalarConst`:	Insert a constant a $[a]$
`scalarPowerR`:	Power-law function in r direction `val` $= ar^p + b$ if $r > r_{\text{min}}$ else $ar^p_{\text{min}} + b$ $[p,\ a,\ R_{\text{min}},\ \text{b}]$
`scalarPowerRExpZ`	Power-law function in r with exponential function in z `valr` $= ar^p + b$ if $r > r_{\text{min}}$ else $ar^p_{\text{min}} + b$ `valz` $= c \exp \frac{-\lvert z\rvert}{d} + e$ `val` $=$ `valr` $\times$ `valz` $[p,\ a,\ c,\ R_{\text{min}},\ b,\ e,\ d]$
`scalarPowerRTheta`	Power-law function in r and $\theta = \cos^{-1}(\lvert z\rvert/r)$ `valr` $= ar^p + b$ if $r > r_{\text{min}}$ else $ar^p_{\text{min}} + b$ `val`$\theta = c\theta^q + d$ if $r > r_{\text{min}}$ else $c\theta^q_{\text{min}} + d$ `val` $=$ `valr` $\times$ `val`θ $[p,\ q,\ a,\ c,\ R_{\text{min}},\ \theta_{\text{min}},\ b,\ d]$
`scalarPowerRz`	Power-law function in r and z see `scalarPowerRTheta` for explanation and input format
`vectorConstR`	Constant value of a in the r direction see `scalarConstR` for explanation and input format
`vectorConstXYZ`	Constant values in the x, y, z directions `valX` $= a$, `valY` $= b$, `valZ` $= c$ $[a,\ b,\ c]$
`vectorDipole`	Dipole vector function for a given dipole moment μ the vector values are $\propto r^{-3}$ if $r > r_{\text{min}}$ else $\propto r^{-3}_{\text{min}}$ $[r_{\text{min}},\ \mu]$
`vectorRadialPowerR`	Power-law function in radial r see `scalarRadialPowerR` for explanation and input format
`vectorRadialPowerRTheta`	Power-law function in radial r and θ see `scalarRadialPowerRTheta` for explanation and input format
`vectorToroidalPowerR`	Power-law function in radial r and θ `valPhi` $= aR^p + b$ if $R > R_{\text{min}}$ else $aR^p_{\text{min}} + b$ see `scalarPowerR` for input format

[4] r is used to define spherical radius while R defines the cylindrical radius

VII Commonly encountered Errors

The following lists the possible causes of errors that user may encounter.

- **LIME and CASA exit when `limesolver` is executed.** This occurs if one or more data file is not in the correct format.
- **Not all results have been linked to models or functions.** This occurs when one of the parameters is not set correctly. Typically, most models require the magnetic field `bmag` to be set to a `VectorScalarR` with zero value.
- **`raytrace` exits and crashes CASA.** This occurs when the limesolver output is not correct usually driven by invalid values in the population levels. Execute the `limesolver` task with higher iteration value.