Hands-on session synthetic observations

NBI Summer school Protoplanetary disks
dr. Nienke van der Marel
NRC Herzberg, Victoria BC
http://www.nienkevandermarel.com
August 8th 2019

Goals of session

- Learn to generate a radiative transfer dust continuum image at 870 micrometer (ALMA Band 7) of a transition disk with a large cavity, using RADMC3D
- Learn basics of ALMA interferometry
- Learn how to simulate an ALMA observation in CASA using the continuum image and choices of parameters:
    - Configuration
    - Integration time
    - Declination
    - Disk mass
    - Stellar luminosity
    - Telescope

RADMC-3D

- Radiative transfer tool, to compute temperatures and resulting emission (line and continuum) based on a given disk structure and star (important: no chemistry or gas heating processes! Careful with line predictions)
- http://www.ita.uni-heidelberg.de/~dullemond/software/radmc-3d/
- Developed by Kees Dullemond (Heidelberg), follow-up of RADMC
- Code is written in Fortran, but interaction through IDL or python functions.
- We will use the python wrapper radmc3dPy, developed by Attila Juhasz: https://www.ast.cam.ac.uk/~juhasz/radmc3dPyDoc/

RADMC-3D

- A typical run consists of two parts, plus diagnostics:

  1. Set up model parameters and calculate the temperature structure
  2. Compute an output image (at any wavelength and inclination) or SED
  3. Create plots of the model structure, e.g. density, temperature, optical depth, etc.

RADMC-3D

- Important: create a new folder for each model you want to run: files will be copied and written in that folder
- Steps first part (following continuum model tutorial):
  - Copy ../python_examples/datafiles/dustkappa_silicate.inp into directory
  - Start python
  - > from radmc3dPy import *
  - > analyze.writeDefaultParfile('ppdisk')
  - > par = analyze.readParams()
  - > par.printPar()
  - > setup.problemSetupDust('ppdisk', mdisk='1e-2*ms', gap_rin='[1.0*au]', gap_rout='[40.*au]', gap_drfact='[1e-5]', nz='0')
  - > import os
  - > os.system('radmc3d mctherm')

```
>>> par.printPar()
# ------------------------------------------------------------------------------------------------
# Block: Radiation sources
# ------------------------------------------------------------------------------------------------
incl_cont_stellarsrc      = False  # # Switches on (True) or off (False) continuous stellar sou
incl_disc_stellarsrc      = True  # # Switches on (True) or off (False) discrete stellar source
mstar                     = [1.0*ms]  # # Mass of the star(s)
pstar                     = [0.0, 0.0, 0.0]  # # Position of the star(s) (cartesian coordinates
rstar                     = [2.0*rs]  # # Radius of the star(s)
staremis_type             = ["blackbody"]  # # Stellar emission type ("blackbody", "kurucz", "n
tstar                     = [4000.0]  # # Effective temperature of the star(s) [K]
# ------------------------------------------------------------------------------------------------
# Block: Grid parameters
# ------------------------------------------------------------------------------------------------
crd_sys                   = 'sph'  # Coordinate system used (car/cyl)
nw                        = [19, 50, 30]  # Number of points in the wavelength grid
nx                        = [30,50]  # Number of grid points in the first dimension
ny                        = [10,30,30,10]  # Number of grid points in the second dimension
nz                        = 0  # Number of grid points in the third dimension
wbound                    = [0.1, 7.0, 25., 1e4]  # Boundraries for the wavelength grid
xbound                    = [1.0*au,1.05*au, 100.0*au]  # Boundraries for the x grid
xres_nlev                 = 3  # Number of refinement levels (spherical coordinates only
xres_nspan                = 3  # Number of the original grid cells to refine (spherical coordin
xres_nstep                = 3  # Number of grid cells to create in a refinement level (spherica
ybound                    = [0., pi/3., pi/2., 2.*pi/3., pi]  # Boundraries for the y grid
zbound                    = [0., 2.0*pi]  # Boundraries for the z grid
# ------------------------------------------------------------------------------------------------
# Block: Dust opacity
# ------------------------------------------------------------------------------------------------
dustkappa_ext             = ['silicate']  #
gdens                     = [3.6, 1.8]  # Bulk density of the materials in g/cm^3
gsdist_powex              = -3.5  # Grain size distribution power exponent
gsmax                     = 10.0  # Maximum grain size
gsmin                     = 0.1  # Minimum grain size
lnk_fname                 = ['/disk2/juhasz/Data/JPDOC/astrosil/astrosil_WD2001_new.lnk', '/dis
mixabun                   = [0.75, 0.25]  # Mass fractions of the dust componetns to be mixed
ngs                       = 1  # Number of grain sizes
# ------------------------------------------------------------------------------------------------
# Block: Gas line RT
# ------------------------------------------------------------------------------------------------
gasspec_colpart_abun      = [1e0]  # Abundance of the molecule
gasspec_colpart_name      = ['h2']  # Name of the gas species - the extension of the molecule_E
gasspec_mol_abun          = [1e-4]  # Abundance of the molecule
gasspec_mol_dbase_type    = ['leiden']  # leiden or linelist
gasspec_mol_name          = ['co']  # Name of the gas species - the extension of the molecule_E
```

```
>>> par.printPar()
# ------------------------------------------------------------------------------------------------
# Block: Radiation sources
# ------------------------------------------------------------------------------------------------
incl_cont_stellarsrc      = False  # # Switches on (True) or off (False) continuous stellar sou
incl_disc_stellarsrc      = True  # # Switches on (True) or off (False) discrete stellar source
mstar                     = [1.0*ms]  # # Mass of the star(s)
pstar                     = [0.0, 0.0, 0.0]  # # Position of the star(s) (cartesian coordinates
rstar                     = [2.0*rs]  # # Radius of the star(s)
staremis_type             = ["blackbody"]  # # Stellar emission type ("blackbody", "kurucz", "n
tstar                     = [4000.0]  # # Effective temperature of the star(s) [K]
# ------------------------------------------------------------------------------------------------
# Block: Grid parameters
# ------------------------------------------------------------------------------------------------
crd_sys                   = 'sph'  # Coordinate system used (car/cyl)
nw                        = [19, 50, 30]  # Number of points in the wavelength grid
nx                        = [30,50]  # Number of grid points in the first dimension
ny                        = [10,30,30,10]  # Number of grid points in the second dimension
nz                        = 0  # Number of grid points in the third dimension
wbound                    = [0.1, 7.0, 25., 1e4]  # Boundraries for the wavelength grid
xbound                    = [1.0*au,1.05*au, 100.0*au]  # Boundraries for the x grid
xres_nlev                 = 3  # Number of refinement levels (spherical coordinates only
xres_nspan                = 3  # Number of the original grid cells to refine (spherical coordin
xres_nstep                = 3  # Number of grid cells to create in a refinement level (spherica
ybound                    = [0., pi/3., pi/2., 2.*pi/3., pi]  # Boundaries for the y grid
zbound                    = [0., 2.0*pi]  # Boundraries for the z grid
# ------------------------------------------------------------------------------------------------
# Block: Dust opacity
# ------------------------------------------------------------------------------------------------
dustkappa_ext             = ['silicate']  #
gdens                     = [3.6, 1.8]  # Bulk density of the materials in g/cm^3
gsdist_powex              = -3.5  # Grain size distribution power exponent
gsmax                     = 10.0  # Maximum grain size
gsmin                     = 0.1  # Minimum grain size
lnk_fname                 = ['/disk2/juhasz/Data/JPDOC/astrosil/astrosil_WD2001_new.lnk', '/dis
mixabun                   = [0.75, 0.25]  # Mass fractions of the dust componetns to be mixed
ngs                       = 1  # Number of grain sizes
# ------------------------------------------------------------------------------------------------
# Block: Gas line RT
# ------------------------------------------------------------------------------------------------
gasspec_colpart_abun      = [1e0]  # Abundance of the molecule
gasspec_colpart_name      = ['h2']  # Name of the gas species - the extension of the molecule_E
gasspec_mol_abun          = [1e-4]  # Abundance of the molecule
gasspec_mol_dbase_type    = ['leiden']  # leiden or linelist
gasspec_mol_name          = ['co']  # Name of the gas species - the extension of the molecule_E
```

Stellar parameters

$$L_* = 4\pi R_*^2 \sigma_{SB} T_*^4$$

```
# -------------------------------------------------------------------------------------------------------------
# Block: Code parameters
# -------------------------------------------------------------------------------------------------------------
istar_sphere              = 0  # 1 - take into account the finite size of the star, 0 - take th
itempdecoup               = 1  # Enable for different dust components to have different tempera
lines_mode                = -1  # Line raytracing mode
modified_random_walk      = 0  # Switched on (1) and off (0) modified random walk
nphot                     = 1000000  # Nr of photons for the thermal Monte Carlo
nphot_scat                = long(3e4)  # Nr of photons for the scattering Monte Carlo (for imag
nphot_spec                = long(1e5)  # Nr of photons for the scattering Monte Carlo (for spec
rto_style                 = 3  # Format of outpuf files (1-ascii, 2-unformatted f77, 3-binary
scattering_mode_max       = 1  # 0 - no scattering, 1 - isotropic scattering, 2 - anizotropic s
tgas_eq_tdust             = 1  # Take the dust temperature to identical to the gas temperature
# -------------------------------------------------------------------------------------------------------------
# Block: Model ppdisk   Disk parameters
# -------------------------------------------------------------------------------------------------------------
bgdens                    = 1e-30  # Background density (g/cm^3)
dusttogas                 = 0.01  # Dust-to-gas mass ratio
gap_drfact                = [1e-5]  # Density reduction factor in the gap
gap_rin                   = [10.0*au]  # Inner radius of the gap
gap_rout                  = [40.*au]  # Outer radius of the gap
gasspec_mol_dissoc_taulim = [1.0]  # Continuum optical depth limit below which all molecules di
gasspec_mol_freezeout_dfact = [1e-3]  # Factor by which the molecular abundance should be decre
gasspec_mol_freezeout_temp = [19.0]  # Freeze-out temperature of the molecules in Kelvin
gasspec_vturb             = 0.2e5  # Microturbulent line width
hpr_prim_rout             = 0.0  # Pressure scale height at rin
hrdisk                    = 0.1  # Ratio of the pressure scale height over radius at hrpivot
hrpivot                   = 100.0*au  # Reference radius at which Hp/R is taken
mdisk                     = 9.950000e+29  # Mass of the disk (either sig0 or mdisk should be s
plh                       = 1./7.  # Flaring index
plsig1                    = -1.0  # Power exponent of the surface density distribution as a fun
prim_rout                 = 0.0  # Outer boundary of the puffed-up inner rim in terms of rin
rdisk                     = 100.0*au  # Outer radius of the disk
rin                       = 1.0*au  # Inner radius of the disk
sig0                      = 0.0  # Surface density at rdisk
srim_plsig                = 0.0  # Power exponent of the density reduction inside of srim_rout*
srim_rout                 = 0.0  # Outer boundary of the smoothing in the inner rim in terms of
```

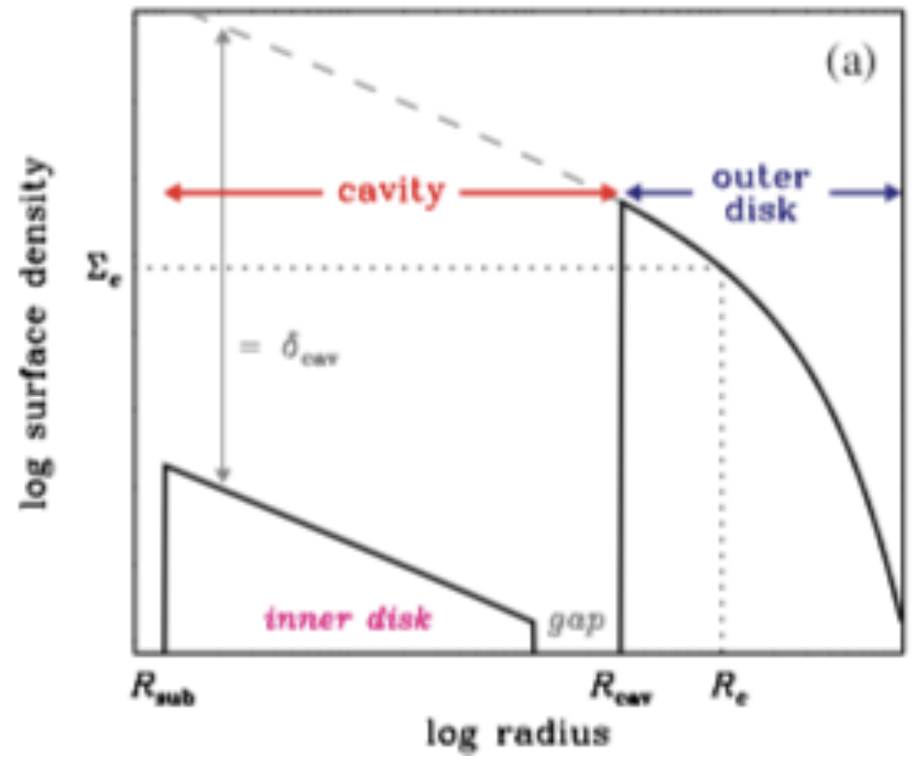
Disk parameters

Surface density profile: expontential power-law with a gap

Andrews et al. 2011

$$\Sigma_g = \Sigma_c \left(\frac{R}{R_c}\right)^{-\gamma} \exp\left[-\left(\frac{R}{R_c}\right)^{2-\gamma}\right]$$

(Note: set either M_{disk} or Σ_0)



Parameter names

dusttogas = inverse gas-to-dust ratio
gap_drfact = set to 0
gap_rin = set to 10 au
gap_rout = R_{cav}
hrdisk = h_c
hrpivot = R_c
plh = ψ
plsig1 = γ
rdisk = not set
rin = R_{sub}
sig0 = Σ_c

Vertical scale height: power-law

$$h = h_c (R/R_c)^{\psi}$$

Many additional parameters (and if you go into code you can change anything you want) but we keep it simple now

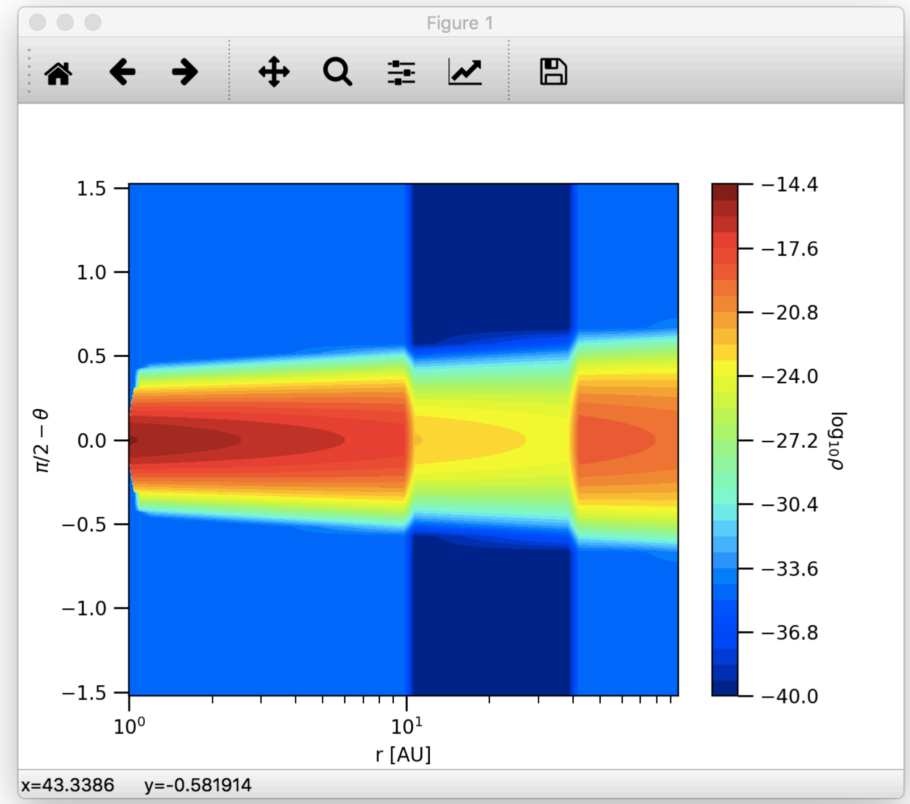
Generate image

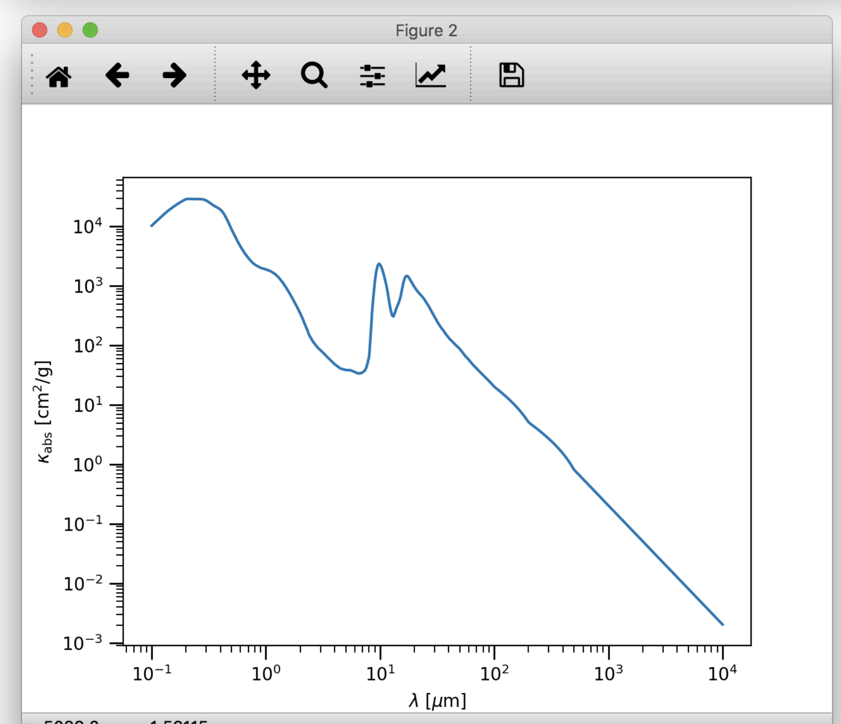
- > image.makeImage(npix=600., wav=870., incl=30., phi=20., sizeau=300.)
- Plot the image within current python session (with and without convolution):
  - > import matplotlib.pylab as plb
  - > im = image.readImage()
  - > image.plotImage(im, au=True, log=True, maxlog=10, saturate=1e-5, cmap=plb.cm.gist_heat)
  - > cim = im.imConv(fwhm=[0.06, 0.06], pa=0., dpc=140.)
  - > image.plotImage(cim, arcsec=True, dpc=140., log=True, maxlog=10, bunit='snu', cmap=plb.cm.gist_heat)

- Or save the image into a fitsfile:
  - > im = image.readImage()
  - > im.writeFits('myimage.fits', dpc=140., coord='03h10m05s -10d05m30s')

Generate plots model structure

- > import matplotlib.pylab as plb
- > import numpy as np
- 2D density contour plot:
  - >>> c = plb.contourf(data.grid.x/natconst.au, np.pi/2.-data.grid.y, np.log10(data.rhodust[:,:,0,0].T), 30)
  - >>> plb.xlabel('r [AU]')
  - >>> plb.ylabel(r'$\pi/2-\theta$')
  - >>> plb.xscale('log')
  - >>> cb = plb.colorbar(c)
  - >>> cb.set_label(r'$\log_{10}{\rho}$', rotation=270.)
- Opacity plot:
  - >>> opac = analyze.readOpac(ext=['silicate'])
  - >>> plb.loglog(opac.wav[0], opac.kabs[0])
  - >>> plb.xlabel(r'λ [μm]')
  - >>> plb.ylabel(r'$\kappa_{\rm abs}$ [cm^2/g]')
- Optical depth plot:
  - data.getTau(wav=0.5)
  - >>> c = plb.contour(data.grid.x/natconst.au, np.pi/2.-data.grid.y, data.taux[:,:,0].T, [1.0], colors='w', linestyles='solid')
  - >>> plb.clabel(c, inline=1, fontsize=10)





Now create some models!

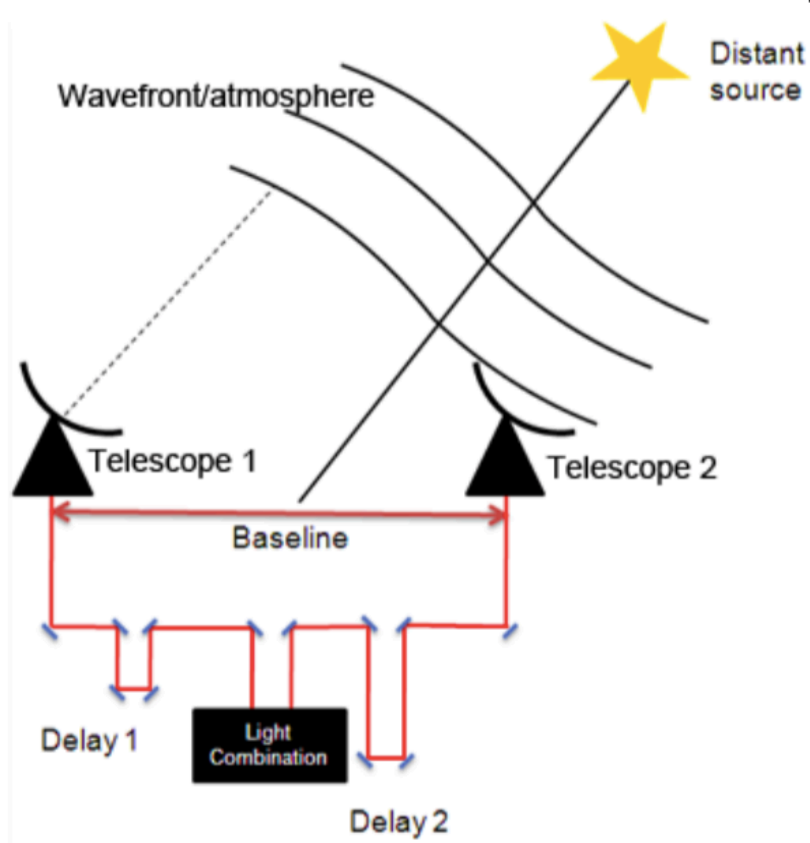
- Now create a few models with varying parameters, e.g. cavity size, disk mass, stellar luminosity (radius) that we are going to use in the ALMA simulation. Choose wavelength=870 μm (Band 7), and distance 100 pc

- Inspect your images with the plotter tool and check the output fits file with a fits viewer, e.g. SAO DS9 (http://ds9.si.edu/site/Download.html).

- DO NOT USE SAO DS9 IF YOU UPDATED TO MOJAVE10.14.6! Use other viewer, e.g. online js9 (https://js9.si.edu/) instead.

- Check the fits header to see if you understand all parameters.

ALMA interferometry



- Every antenna is connected with every antenna (baseline)
- So 3 antennas: 6 baselines, 4 antennas: 24 baselines, etc.

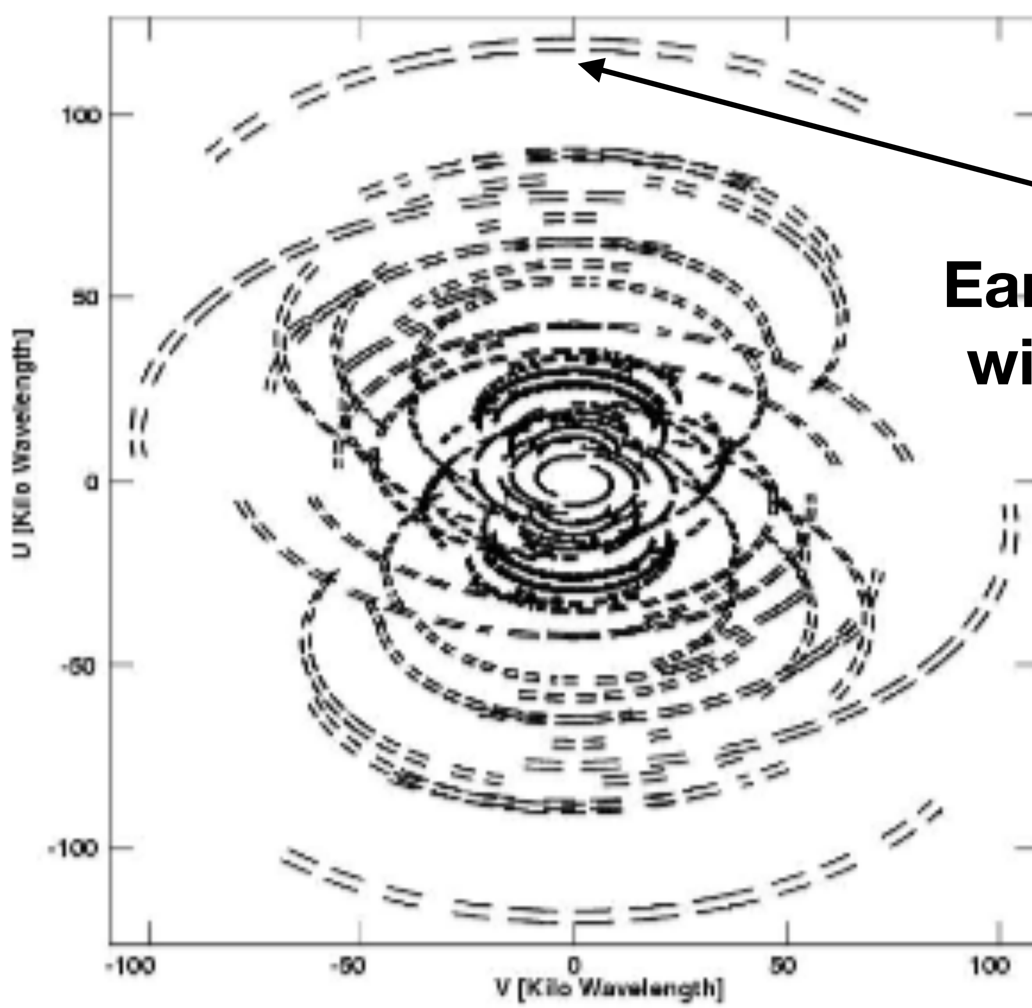
- Each baseline (interference pattern) provides a resolution element or spatial scale λ/B (B = baseline length), which can be described as point in the u,v-plane

- For observing, you want to fill the u,v-plane as much as possible to cover all possible spatial scales and get a nice clean beam

- ALMA:
  - 12m-array: 50 antennas, 10 possible configurations with baselines <16 km
  - 7m array (ACA): 12 antennas
  - Total power array: 4 antennas

ALMA interferometry

- The u,v-plane is the collection of all u,v points and can be filled up by adding more baselines and by using the Earth rotation, as the baseline orientation and position changes from the *point of view of the source* (hour angle and elevation)



Earth rotation and track source with intermissions (calibrator)

ALMA interferometry

- The u,v-plane can be considered the Fourier transform of the x,y-plane: imaging interferometry data means Fourier transforming it
- The u,v-coverage :
  - The sampling of the u,v-plane: the better it is filled, the more accurate your image will be
  - The Fourier transform of the u,v-coverage gives you your beam or PSF: perfect coverage results in a perfect Gaussian beam, imperfect coverage results in a ‘dirty beam’ with sidelobes
  - Coverage at long baselines means you are sensitive to small spatial scales: coverage at short baselines means you are sensitive to large spatial scales => you usually cannot have both at the same time!

ALMA interferometry

Table A-1: Angular Resolutions (AR) and Maximum Recoverable Scales (MRS) for the Cycle 7 Array configurations

Config	Lmax		Band 3	Band 4	Band 5	Band 6	Band 7	Band 8	Band 9	Band 10
	Lmin		100 GHz	150 GHz	183 GHz	230 GHz	345 GHz	460 GHz	650 GHz	870 GHz
C43-1	161 m	AR	3.4″	2.3″	1.8″	1.5″	1.0″	0.74″	0.52″	0.39″
	15 m	MRS	28.5″	19.0″	15.4″	12.4″	8.3″	6.2″	4.4″	3.3″
C43-2	314 m	AR	2.3″	1.5″	1.2″	1.0″	0.67″	0.50″	0.35″	0.26″
	15 m	MRS	22.6″	15.0″	12.2″	9.8″	6.5″	4.9″	3.5″	2.6″
C43-3	500 m	AR	1.4″	0.94″	0.77″	0.62″	0.41″	0.31″	0.22″	0.16″
	15 m	MRS	16.2″	10.8″	8.7″	7.0″	4.7″	3.5″	2.5″	1.9″
C43-4	784 m	AR	0.92″	0.61″	0.50″	0.40″	0.27″	0.20″	0.14″	0.11″
	15 m	MRS	11.2″	7.5″	6.1″	4.9″	3.3″	2.4″	1.7″	1.3″
C43-5	1.4 km	AR	0.54″	0.36″	0.30″	0.24″	0.16″	0.12″	0.084″	0.063″
	15 m	MRS	6.7″	4.5″	3.6″	2.9″	1.9″	1.5″	1.0″	0.77″
C43-6	2.5 km	AR	0.31″	0.20″	0.16″	0.13″	0.089″	0.067″	0.047″	0.035″
	15 m	MRS	4.1″	2.7″	2.2″	1.8″	1.2″	0.89″	0.63″	0.47″

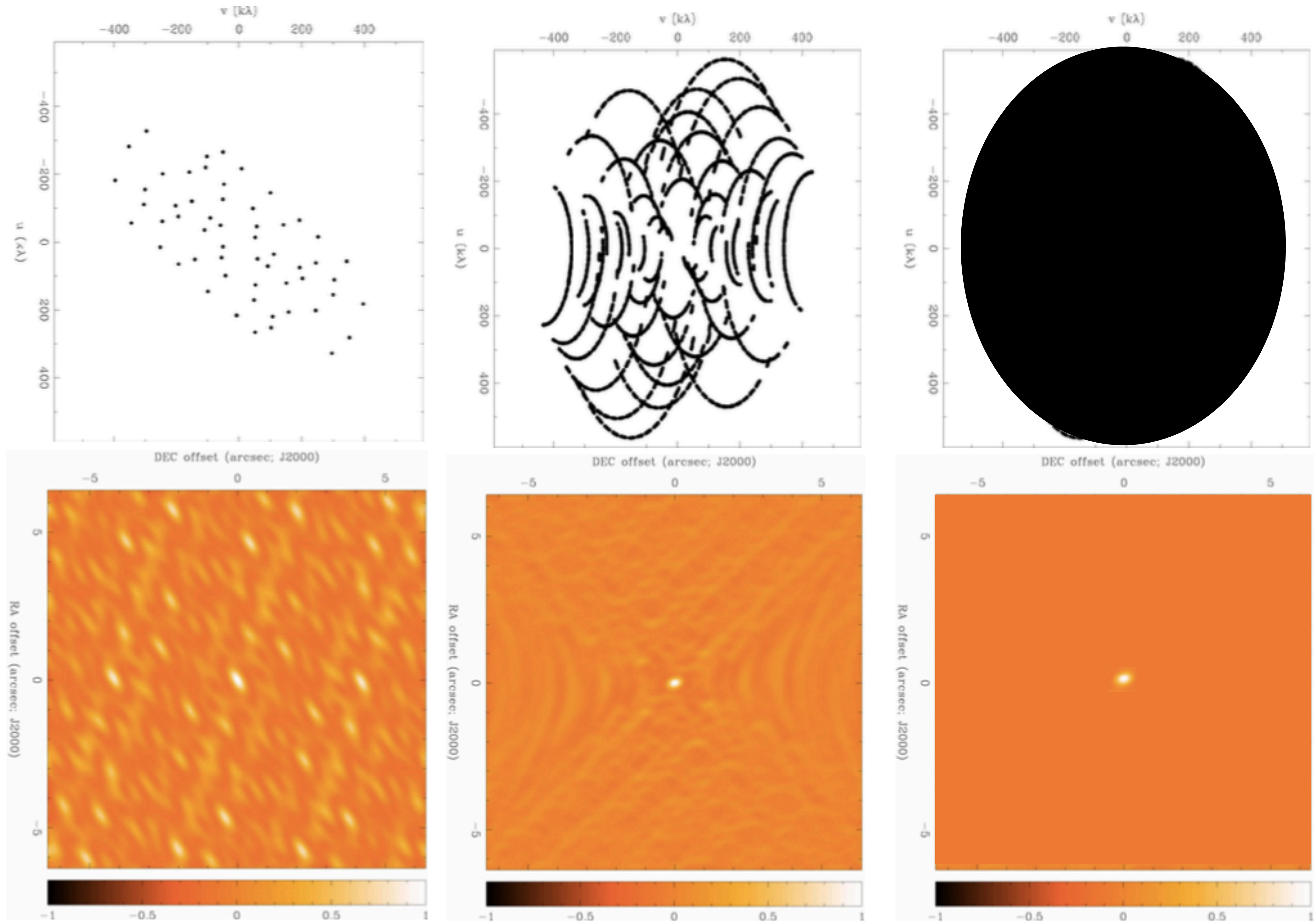
Each configuration has a limited number of baseline lengths, or a limited number of spatial scales you are sensitive to

MRS: maximum recoverable scale
AR: angular resolution

Dirty beam

u,v-coverage (example: SMA with 8 antennas)

Beam or PSF



Very dirty beam **Dirty beam** **Perfect beam**

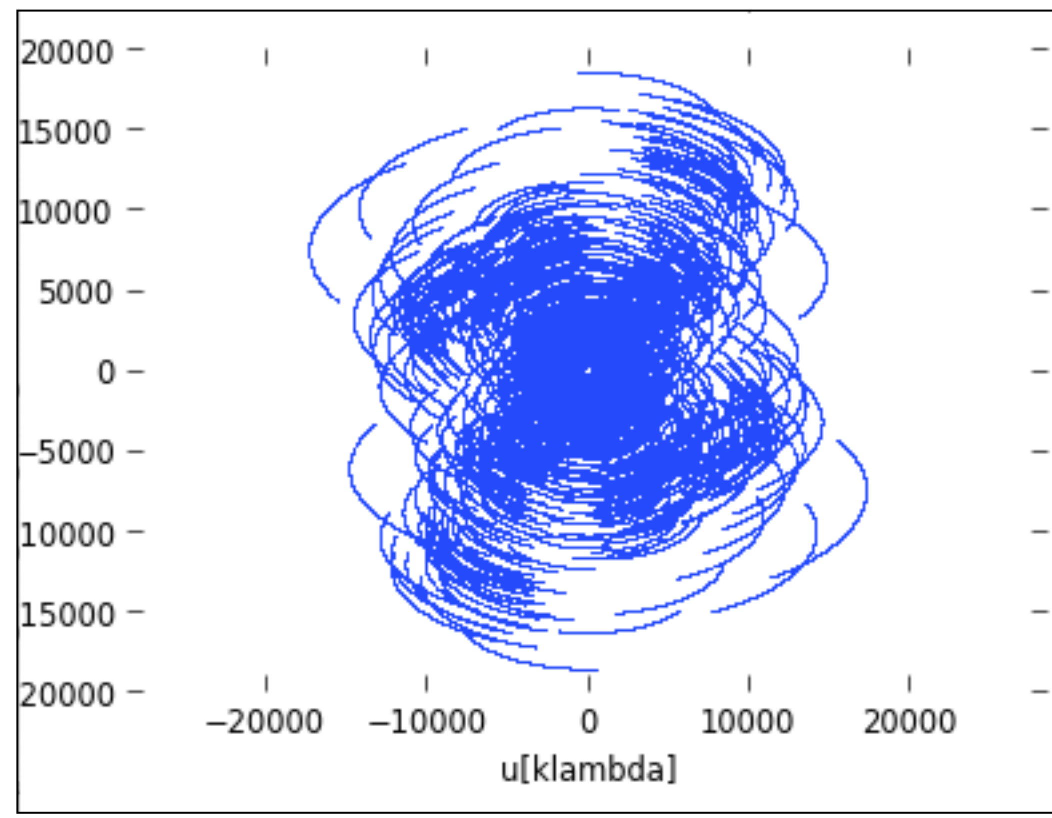
ALMA simulating

- ALMA simulation in CASA: provide an image and the details of the requested observations (configuration, integration time, declination of source, frequency) and compute how your image *would look like when observed*

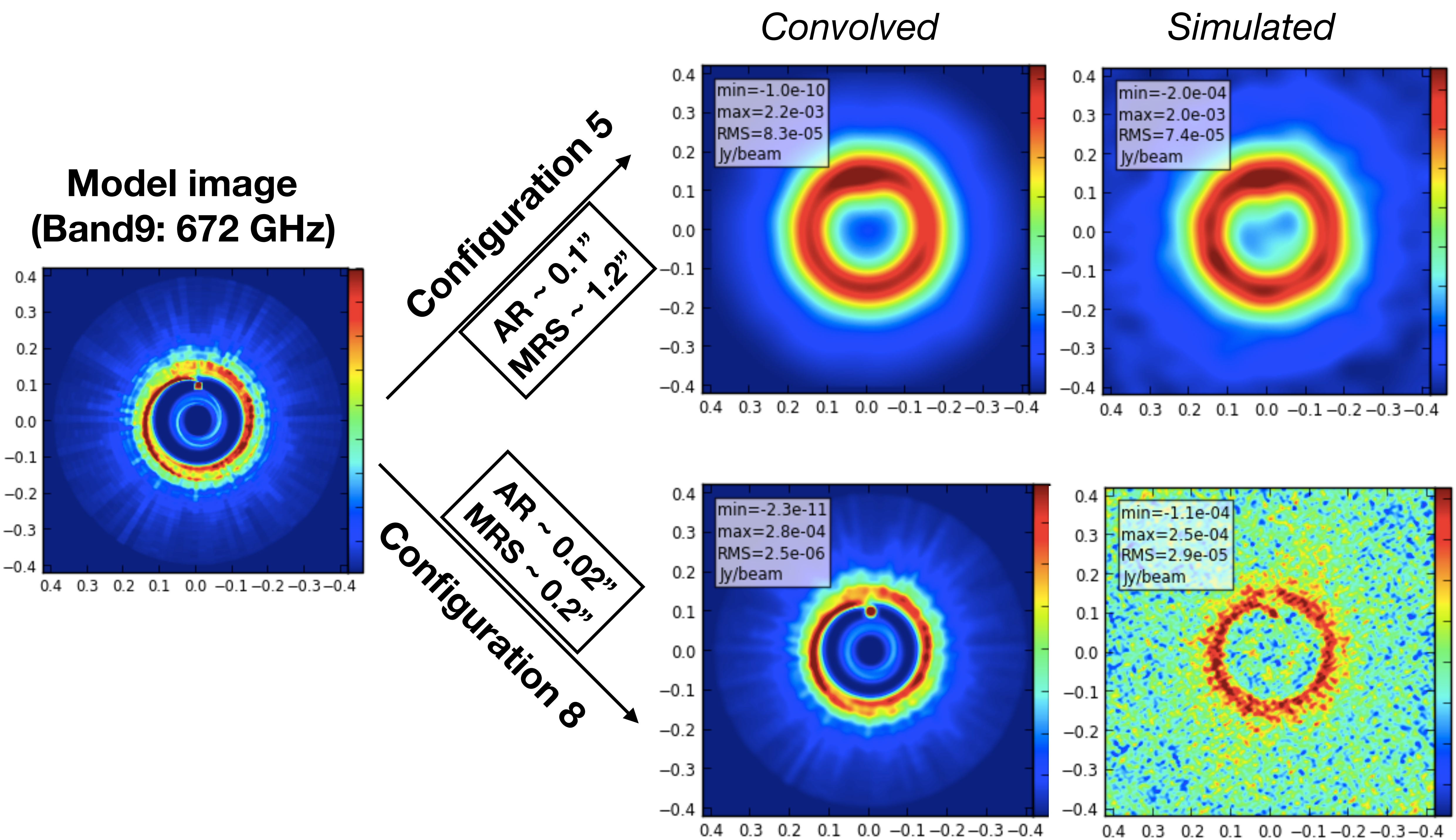
- Two main differences between convolution and ALMA simulation: spatial scales + side lobes

- Difference generally gets worse in more extended configurations: less u,v-coverage at long baselines

u,v-coverage
5h ALMA observation



ALMA simulating



Note that only 2/3 of total flux is recovered in C8

ALMA simulating

- Start CASA
- Simulator: simalma() (in the past: two separate tasks, simobserve() and simanalyze())
- How to run a task:
    - > inp(simalma) => all parameters appear in screen
    - Set parameters (e.g. skymodel=‘myimage.fits’)
    - > inp => all set parameters appear in screen (when problem: red)
    - > go
- Output images will appear in separate window (automatically saved) and a logger
- CASA will create a folder with name of the project parameter in the current directory and put all files in there

ALMA simulating

Name project folder (new one every run)

Important!

Model fits file

Location source

[(these parameters are for re-scaling if you don't like values in fits header]

Total integration time

```
#  simalma :: Simulation task fc
project             =  'longsim'      #  root prefix for output file names
dryrun              =     False       #  dryrun=True will only produce the
                                      #   informative report, not run
                                      #   simobserve/analyze
skymodel            = 'ppdisk672_GHz_50pc.fits' #  model image to observe
     inbright       =         ''      #  scale surface brightness of brightest
                                      #   pixel e.g. "1.2Jy/pixel"
     indirection    = 'J2000 16h30m30 -24d30m30' #  set new direction e.g.
                                      #   "J2000 19h00m00 -40d00m00"
     incell         =         ''      #  set new cell/pixel size e.g.
                                      #   "0.1arcsec"
     incenter       =         ''      #  set new frequency of center channel
                                      #   e.g. "89GHz" (required even for 2D
                                      #   model)
     inwidth        =         ''      #  set new channel width e.g. "10MHz"
                                      #   (required even for 2D model)
complist            =         ''      #  componentlist to observe
setpointings        =       True
     integration    =      '10s'      #  integration (sampling) time
     direction      =         ''      #  "J2000 19h00m00 -40d00m00" or "" to
                                      #   center on model
     mapsize        =   ['', '']      #  angular size of map or "" to cover
                                      #   model
antennalist         = 'alma.cycle6.8.cfg' #  antenna position files of ALMA 12m
                                      #   and 7m arrays
hourangle           =  'transit'      #  hour angle of observation center e.g.
                                      #   -3:00:00, or "transit"
totaltime           =       '5h'      #  total time of observation; vector
                                      #   corresponding to antennalist
```

ALMA Band is chosen based on frequency as set in the FITS header!

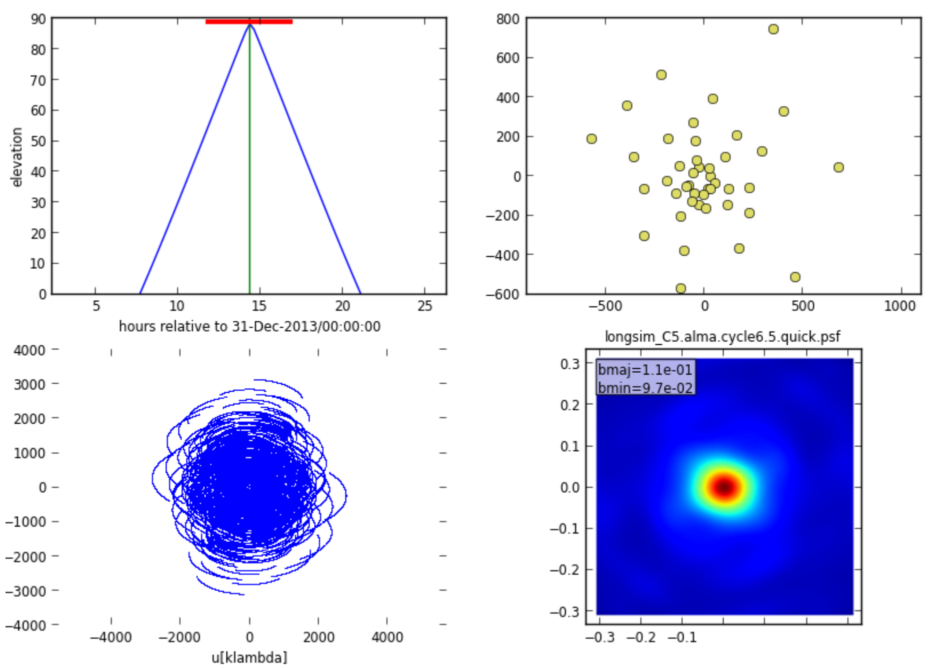
Possible configurations (more online, also other telescopes)

alma.cycle6.1.cfg
alma.cycle6.2.cfg
alma.cycle6.3.cfg
alma.cycle6.4.cfg
alma.cycle6.5.cfg
alma.cycle6.6.cfg
alma.cycle6.7.cfg
alma.cycle6.8.cfg
alma.cycle6.9.cfg
alma.cycle6.10.cfg

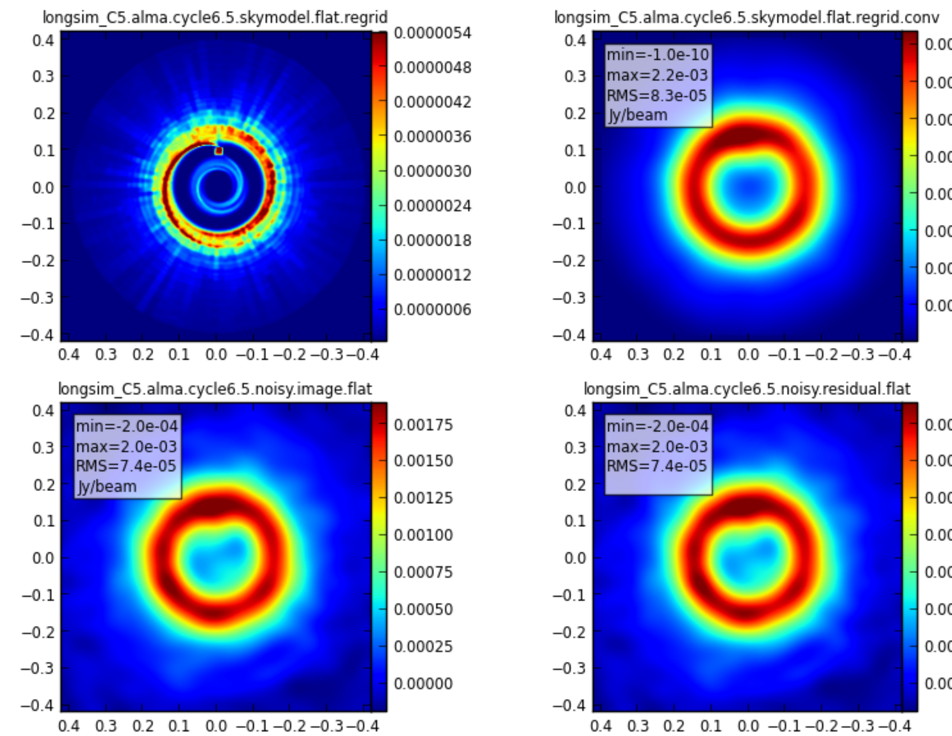
Additional parameters can be set, but generally not necessary

ALMA simulating

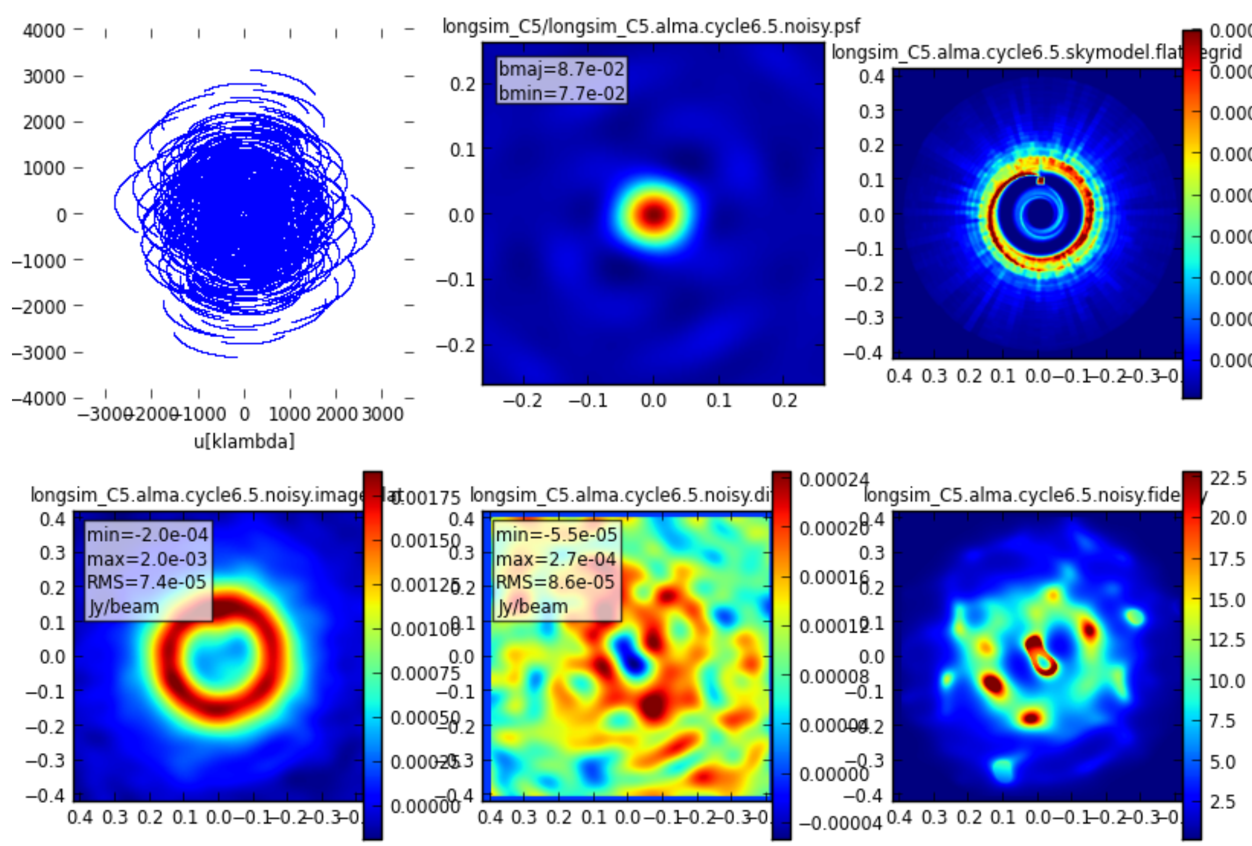
Output PNG files (in folder)



XX.observe.png



XX.image.png

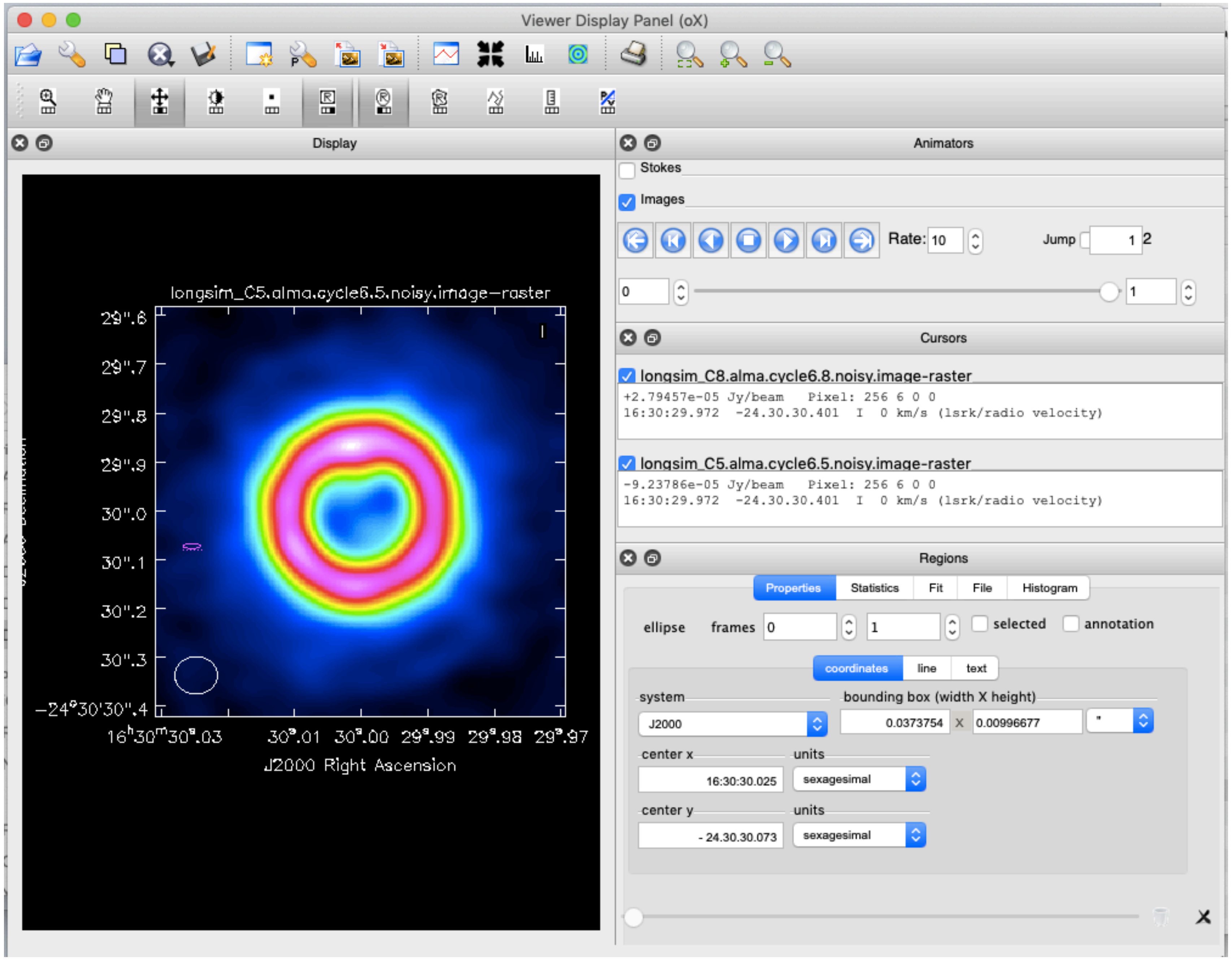


XX.analysis.png

ALMA simulating

Also possible:
Inspect, analyse and overlay generated images in the **CASA viewer**
> viewer
Useful: draw a region and measure total flux ('Statistics')

Export to fits file:
> inp(exportfits)
> imagename='XXX.image'
> fitsimage='XXX.fits'
> go



Now simulate some models!

- With the models that you generated before, do an ALMA simulation and inspect the outcome images. Play around with different configurations, integration times, source position and check the outcome for beam size, recovered flux and overall structure/noise
- Configuration table: https://almascience.nrao.edu/proposing/proposers-guide#section-37 (Note that configurations are the same for Cycle 6 and 7)
- Additional configuration files (other telescopes): https://casaguides.nrao.edu/index.php/Antenna_Configurations_Models_in_CASA_Cycle6
- You can even change the telescope with the antenna list parameter! In that case, run simobserve() and simanalyze(), input parameters are the same as simalma()
- More information: https://casaguides.nrao.edu/index.php/Simulating_Observations_in_CASA_5.1
(note that this hasn't changed for CASA 5.5)