Simulation for ALMA-ACA data combination

Toshinobu Takagi*

2020/06/01

1 Introduction

Simulation data are necessary to evaluate the effectiveness of any data combination method. The objective of this work is to provide useful simulation datasets (ms) for such evaluation, using UV–optical FITS images available in various astronomical archives such as SDSS. Here we report the initial attempt of the ALMA simulation using UV–optical images.

2 Simulation settings

Starting point of the simulation conducted here is CASA guide: ACA simulation (5.4) v2.[1] Following scripts were prepared to control and run CASA simulation tasks, i.e. `simobserve()` and `simanalyze()`.

- `ACASimulation.py`: Main script to run `simobserve()`. Parameters are stored in separate python script `param.py`. This script calls `ACAanalyze.py` at the end. This script uses `analysisUtils.py` to estimate sensitivity and beam size.
- `ACAanalyze.py`: A script to run a series of imaging and analysis tasks using `simanalyze()`. Firstly, ALMA 12m and ACA 7m measurement sets are concatenated with adjusted weighting for 7m. SD image is created by `simanalyze()`, and then both imaging of concatenated interferometer data and feathering with SD image are conducted also by `simanalyze()`. This can be replaced by other imaging scripts.
- `param.py`: A parameter setting file for the simulation.

To run the simulation, edit the parameter file `param.py` and execute `ACASimulation.py` with `execfile()` command in CASA. See appendix for the entire scripts.

2.1 Input FITS file

Basically, any 2D image with FITS format can be used for the simulation. However, large FITS files (> 1 GB) are not recommended, since it takes too much time to read image data with `imval()` task. Parameters specific to the object in FITS file, such as sky coordinate, scaling of pixel size and pixel values, can be changed by simulation parameters (parameter `indirection, incell` and `inbright`, respectively), in order to fit ALMA observation.

In `ACASimulation.py`, the maximum signal-to-noise ratio (SNR) of the input image is calculated, for which the sky noise level is estimated by simple sigma-clipping. This is to realize a similar SNR in the resulting ALMA image, by setting appropriate brightness (`inbright` value) for ALMA sensitivity for a given integration time. However, this auto-setting of `inbright` is not well tested yet. For the time being, the specific peak brightness of 0.004 Jy/pixel, as adopted in the CASA guide, has been used throughout.

*Japan Space Forum

[1] `https://casaguides.nrao.edu/index.php?title=ACA_Simulation_(CASA_5.4)_v2`

2.2 Major parameters

- `anaScript` – script name to conduct `simanalyze()` or other imaging scripts by using simulated measurement sets. This is `ACAanalyze.py` in this work.
- `fitsFile` – file name (or path) for input FITS file.
- `project` – project name for a specific simulation run. All of the simulation results are stored in the project directory. Note that `ACASimulation.py` will **delete** project directory if exists.
- `indirection` – sky coordinate of image center for simulation.
- `incell` – cell size of the image to be used for simulation, which can control the angular size of image.
- `incenter` – central frequency of the simulation. Since this changes the primary beam size, the mosaic pattern (the number of FoV) determined by `simobserve()` depends on this frequency.
- `config` – list of 12m antenna configuration (without .cfg suffix). There is no limit for the number of 12m configuration.
- `acaconfig` – configuration of ACA 7m antenna. Single configuration only.
- `hourangle` – hour angle for observation. This can be changed to check the effect of target elevation.
- `mapsize` – map size to be observed. This should be adjusted to fit in the input image by checking the message from `ACASimulation.py` shown before the run of `simobserve()`. The number of pointing in the message is the estimated value assuming the pointing spacing of 0.5 primary beam. The map size of TP is set to 1.3 times the interferometer map.
- `integration` – integration time. This is the integration time per pointing in the simulation. The number of pointing per field is controlled by `totaltime` parameter. If the total time is $120s$ for 6 pointing mosaic and the integration of $10s$, each field is observed twice.
- `timeRatio` – 12m-7m-TP time ratio of integration time per field. The 12m portion should be one. This time ratio will be adopted if the `totaltime` parameter below is set to blank.
- `totaltime12` – total observation time of each 12m array configuration. Multiple 12m configurations have the same observing time in this script. If it is blank, `totaltime12` is calculated so that each field has one scan. However, the number of pointing (mosaic field) is based on simple calculation, and sometimes wrong. It is necessary to check if `totaltime` is long enough to cover entire mosaic.
- `totaltime7/TP` – total observation time of 7m/TP array. If it is blank, `timeRatio` will be used to calculate it.

There parameters can be specified in `param.py`. For each run of `ACASimulation.py`, this parameter file will be copied to`(project).param.py`.

3 Results

3.1 M51

The CASA guide referenced here uses a FITS file of M51, which is available via the web page. Firstly, this FITS image was used to reproduce the CASA guide. The contents of the parameter file is as follows:

Listing 1: contens of `param.py`

```
## Simanalyze script
anaScript = 'ACAanalyze2.py'

fitsFile = 'M51ha.fits'

### Simulation settings
project = "m51c_ana2"

indirection = "J2000 23h59m59.96s -34d59m59.50s"
incell = "0.1arcsec"
incenter = "330.076 GHz"
inwidth = "50MHz"
inbright = '0.004' # blank for auto scaling using SN in model image
config = ["alma.cycle6.3"]
acaconfig = "aca.cycle6"
hourangle = '' # blank for 'transit'

# Mosaic setting
mapsize = "60 arcsec"
mapsizeTP = "1.3arcmin"

# Sensitivity setting
integration = '10s'
pwv = 0.9
timeRatio = [1, 3, 4] # [12m : 7m : TP] (used if totaltime* is blank)
totaltime12 = '30min'
totaltime7 = '72min'
totaltimeTP = '123min'
```

Since the imaging strategy is different from the standard procedure introduced by EA-ARC, a specific analysis script was made, named as `ACAanalyze2.py`. In this script, TP data and ACA 7m visibility are jointly input to `simanalyze()`. This is to image total power and ACA with total power as a model, according to the CASA guide.[2] The resulting 7m image is used as a model for 12m imaging.

Simulated measurement sets (MS) are produced by `simobserve()` for each 12m, 7m array and TP. MSs with simulated noise are also available. However, we used noise free version of MSs, in order to investigate idealized cases. The effect of noise will be investigated in studies elsewhere.

The input image of M51 was rescaled, regridded, and then convolved to match the restoring beam. Simulation results are summarized in output figure from `simanalyze()` as shown in Figure 1, in which UV plane, input image, output images are depicted. The input image which can directly be compared with output has a file name such as `(project).(config).skymodel.flat.regrid.conv`. This is regridded and convolved skymodel image, having the same beam as the output image, and therefore it is useful for pixel-by-pixel comparison, i.e. calculation of image difference and fidelity.

[2] However, I could not verify the fact that the SD image is used as a model for 7m imaging in the log file. According to the log file, the SD image is used for feathering after the ACA 7m imaging.

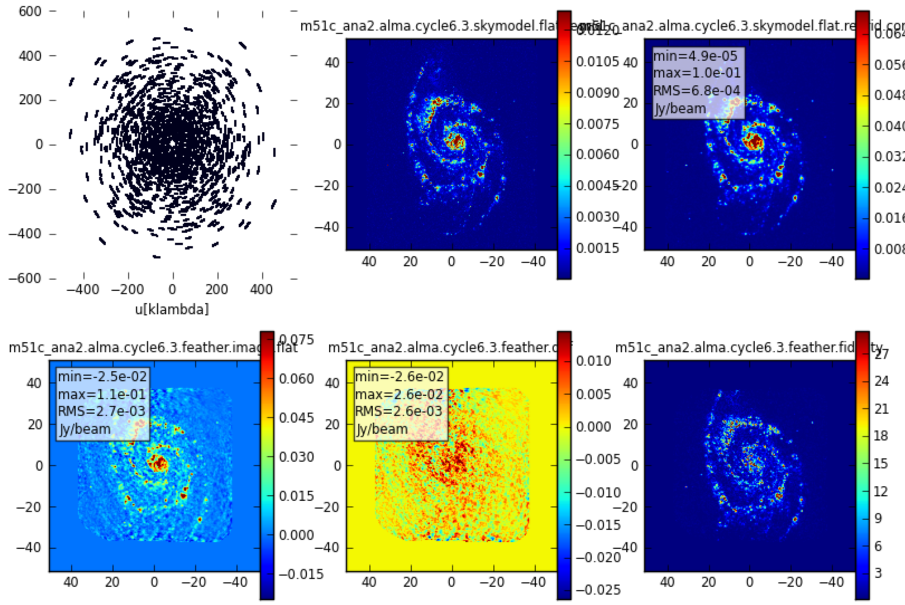

Figure 1. Results of simulation for M51 as in the CASA guide

Fidelity is defined as the inverse of relative error as follows:

$$\text{Fidelity}(i,j) = \frac{abs(\text{model}(i,j))}{abs(\text{model}(i,j) - \text{simulated}(i,j))}. \tag{1}$$

In practice, the denominator will be replaced to 0.7×rms(Difference), if it is greater than the absolute value of simple difference. This sets the upper limit on the fidelity, which is $S/(0.7\Delta)$, where S and Δ are model flux and rms, respectively. See ALMA memo No.398 for details.

Here we use skymodel and fidelity image to check the fidelity values as a function of pixel values, expecting that the high SNR regions should have high fidelity. In Figure 2, only pixels with $> 3\sigma$ values are plotted. The upper limit of fidelity is clearly visible, **but many pixels show low fidelity even if is very bright.** Specifically, it is expected that the bright central part of M51 should have high fidelity, but it not the case for most of pixels as shown in Figure 1. This plot may indicate that the observation setting is not good enough to realize good UV coverage for M51. It is interesting to see if other imaging methods can improve the image fidelity of this image or not.

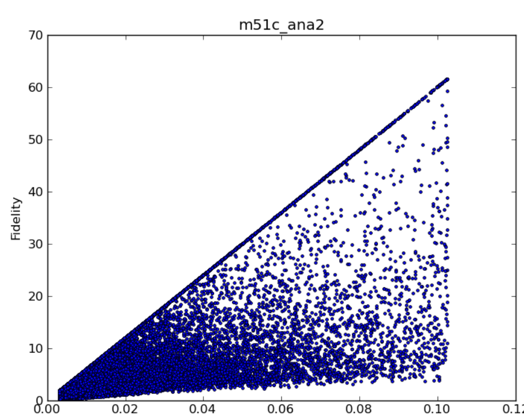

Figure 2. Fidelity vs flux values for M51

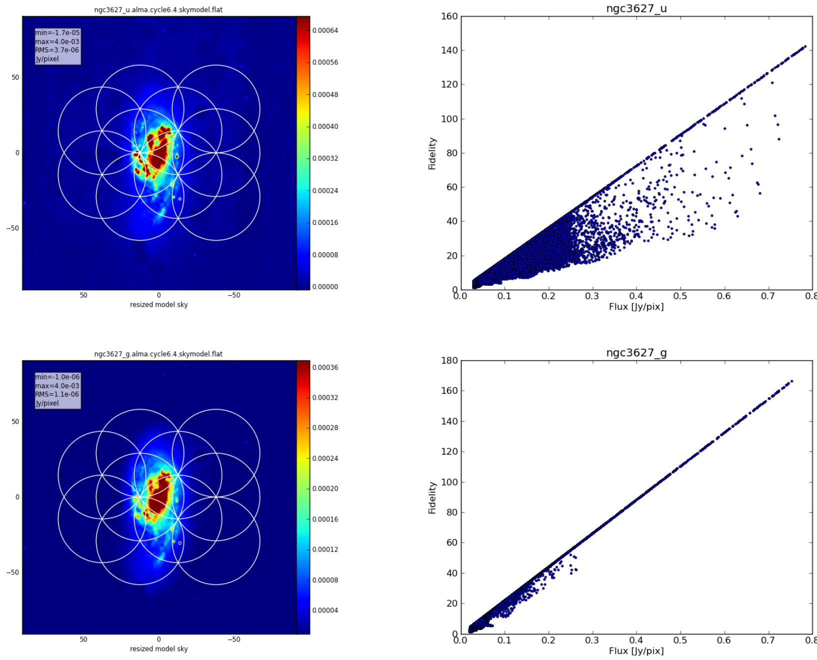

Figure 3. Sky model and fidelity plot for NGC3627 in u band (upper) and g band (lower).

3.2 NGC3627

We use SDSS image of NGC3627 in u and g bands as another example of the simulation. Foreground stars were masked out. The pixel size, map size and the central frequency were adjusted to result in reasonable mosaic pattern. The parameters different from M51 simulation is listed below.

Listing 2: contens of `param.py` for NGC3627

```
anaScript = 'ACAanalyze.py'
incenter = "100 GHz"
config = ["alma.cycle6.4","alma.cycle6.1"]
mapsize = "80 arcsec"
totaltime12 = '' # blank for 1 integ. per field
totaltime7 = '3'
totaltimeTP = '4'
```

Two configurations were used for 12m array. The integration time per ACA 7m/TP field is 3 times/4 times that for the 12m array. The skymodel image and fidelity plot in u and g bands are shown in Figure 3. This time the imaging method follows the one recommended at EA-ARC, as implemented in `ACAanalyze.py`.

The morphology of NGC3627 is more centrally concentrated in g band than in u band. This somehow results in very good fidelity in g band. The fidelity of u band image may be improved by adjusting observing parameters or imaging method.

4 Summary

In order to evaluate the data combination method, the fidelity of the simulated image should be good at least in one method. It would be difficult to discriminate imaging methods with $> 20\%$ error from others with similar error. It would be necessary to prepare images of various astronomical objects to find which data combination method produces the best fidelity. Wide field imaging surveys in UV–optical bands can be good resource for such simulation. If none of methods produces good fidelity, the observing parameters should be reconsidered.

A Simulation scripts

Contens of simulation scripts are listed below.

Listing 3: contens of `ACASimulation.py`

```
import os
import glob
import analysisUtils as aU

############
# Simulation parameters
############

## Read parameter file
execfile('param.py')


############

print "####################################################"
print "### Project directory: "+project
print "####################################################"

isThere = glob.glob(project)
if isThere:
    print " "
    print "### Deleting existing project directory: "+project
    print " "
    os.system('rm -rf %s' %project)

print "# Saving parameter files to %s.param.py" %project
os.system('rm -rf %s.param.py'%project)
os.system('cp param.py %s.param.py'%project)

print "# Using image file "+fitsFile
skymodel = fitsFile

## Use following command to get FITS file of M51.
isThere = glob.glob('M51ha.fits')
if not isThere and fitsFile == 'M51ha.fits':
    os.system('curl https://casaguides.nrao.edu/images/3/3f/M51ha.fits.txt -f -o M51ha.fits')

hd = imhead(imagename= fitsFile)
pixSize = abs(hd['incr'][0]*180/3.141592*3600) # arcsec

# RMS estimate with simple sigma clipping
imsize = hd['shape']
box = '0,0,%s,%s'%(str(imsize[0]-1),str(imsize[1]-1))
imgData = imval( fitsFile, box=box)['data']
niter = 5
data = imgData
rms = 0
data[np.isnan(data)] = -1
med = np.max(data)
for i in range(niter):
```

```
    nonzero = data > 1e-33
    noise = data < (med+3*rms)
    med = np.median( data[nonzero*noise] )
    rms = np.std( data[nonzero*noise])

snr = np.max(imgData)/rms

print " "
print "### Original FITS specification : ", fitsFile
print "# pixel scale [arcsec] = ", pixSize
print "# image shape = ", hd['shape']
print "# image size [arcsec] = ", pixSize*hd['shape'][0]
print "# SNR of image max = ", snr
print "############################################"

input_line = raw_input("Check the image spec [RETURN/n] :")
if input_line:
    sys.exit()



# [12m 7m] beam size
diam = np.array([12., 7.])
wave = 2.99792e8/ (float(incenter.split("GHz")[0])*1.e9)
hpbw = 1.02 * wave / diam *180/3.141592 *3600 # arcsec

# number of pointing with 0.5PB spacing
npoint12 = int((float(mapsize.split("arcsec")[0]) / hpbw[0] *2))**2
npoint7 = int((float(mapsize.split("arcsec")[0]) / hpbw[1] *2))**2
if not totaltime12:
    totaltime12 = str(npoint12 * float(integration.split("s")[0]))+'s'
if not totaltime7:
    totaltime7 = str(npoint7 * float(integration.split("s")[0])*timeRatio[1])
        +'s'
if not totaltimeTP:
    totaltimeTP = str(npoint12 * float(integration.split("s")[0])*timeRatio[2])
        +'s'

## Sensitivity setting
#

beam = []
mrs = []
for i in range(len(config)):
    beam.append(aU.estimateSynthesizedBeamForConfig(config[i], incenter))
    mrs.append(aU.estimateMRSForConfig(config[i], incenter))

autoSens = False
if not inbright:
    print " "
    print "# Sensitivity calculation .... "
    print " "
    rmsVis = aU.sensitivity(incenter, inwidth, integration, pwv=pwv,
        antennalist = config[0])
    maxval = rmsVis*snr # Maximum pixel value in skymodel
    inbright = str(maxval)+" Jy/pixel" # This is rough estimate (i.e. beamsize
        is not considered)
    autoSens=True

```

```
print " "
print "### Sky Model specification : "
print "# Project name = ", project
print "# pixel scale [arcsec] = ", incell
print "# Peak intensity = ", inbright
print "# image shape = ", hd['shape']
print "# image size [arcsec] = ", hd['shape'][0]*float(incell.split("arcsec")[0])
print " "
print "### Mapping specification (estimated): "
print "# ALAM configuration = ", config
print "# Estimated beamsize = ", beam
print "# Maximum recoverable scale = ", mrs
print "# ACA configuration = ", acaconfig
print "# Map size (INT) = ", mapsize
print "# Map size (TP) = ", mapsizeTP
print "# Integration time = ", integration
print "# 12m HPBW [arcsec] = ", hpbw[0]
print "# 12m Number of pointing = ", npoint12 # simple guess
print "# 12m Total time = ", totaltime12
if autoSens:
  print "# 12m Array sensitivity [Jy] = ", rmsVis
print "# 7m HPBW [arcsec] = ", hpbw[1]
print "# 7m Number of pointing = ", npoint7 # simple guess
print "# 7m Total time = ", totaltime7
print "# TP Total time = ", totaltimeTP
print "###########################################"

input_line = raw_input("Check the simulation params [RETURN/n] :")
if input_line:
    sys.exit()



# 12m Observation
cfglist = [item + ".cfg" for item in config]
for cfg in cfglist:
  simobserve(
    project = project,
    skymodel = skymodel,
    indirection = indirection,
    incell = incell,
    inbright = inbright,
    incenter = incenter,
    inwidth = inwidth,
    setpointings = True,
    integration = integration,
    mapsize = mapsize,
    maptype = "ALMA-OT",
    pointingspacing = "0.5PB",
    obsmode = "int",
    antennalist = cfg,
    refdate = "2012/12/01",
    hourangle = hourangle,
    user_pwv = pwv,
    totaltime = totaltime12,
    graphics = "both"
  )

```

```
# TP Observation
simobserve(
    project = project,
    skymodel = skymodel,
    indirection = indirection,
    incell = incell,
    inbright = inbright,
    incenter = incenter,
    inwidth = inwidth,
    integration = integration,
    mapsize = mapsizeTP,
    maptype = "square",
    obsmode = "sd",
    antennalist = "aca.tp.cfg",
    sdant = 0,
    refdate = "2012/12/02",
    hourangle = hourangle,
    user_pwv = pwv,
    totaltime = totaltimeTP ,
    graphics = "both"
)


# 7m Observation
simobserve(
    project = project,
    skymodel = skymodel,
    indirection = indirection,
    incell = incell,
    inbright = inbright,
    incenter = incenter,
    inwidth = inwidth,
    setpointings = True,
    integration = integration,
    mapsize = mapsize,
    maptype = "ALMA-OT",
    pointingspacing = "0.5PB",
    obsmode = "int",
    antennalist = acaconfig+'.cfg',
    refdate = "2012/12/02" ,
    hourangle = hourangle,
    user_pwv = pwv,
    totaltime = totaltime7,
    graphics = "both"
)


################################################


print "################################################"
print "# Start simanalyze script : "+anaScript
print "################################################"

execfile(anaScript)
```

Listing 4: contens of `ACAanalyze.py`

```

#acaconfig = "aca.cycle6"

########################
### Simulation settings
########################

cell = "0.2arcsec"
imsize = [512,512]


## Visibility setting
addNoise = False

if addNoise == False:
    visTP = project+".aca.tp.sd.ms"
    vis7 = project+"."+acaconfig+".ms"

    visINT = [project+"."+item+".ms" for item in config]
    visINT.append(vis7)
else:
    visTP = project+".aca.tp.noisy.sd.ms"
    vis7 = project+"."+acaconfig+".noisy.ms"

    visINT = [project+"."+item+".noisy.ms" for item in config]
    visINT.append(vis7)


# Concat and scale weights
wtINT = [1 for i in range(len(config))]
wtINT.append(0.193) # for ACA 7m

os.chdir(project)
concatvis = project+'.concat'
os.system('rm -rf '+concatvis+'.ms')
concat(vis= visINT,
       concatvis=concatvis+'.ms',
       visweightscale= wtINT)
os.chdir('../')

os.system('rm -rf '+project+'/'+project+'*image*')

############################################

simanalyze(
    project = project,
    vis = visTP,
    imsize = imsize,
    cell = cell,
    threshold = '30mJy',
    niter = 10000,
    pbcor = True,
    analyze = True,
    showuv = True,
    showpsf = False,
    showmodel = True,
    showconvolved = True,
    showclean = True,
    showresidual = False,
```

```
    showdifference = True,
    verbose = True,
    showfidelity = True
)

simanalyze(
    project = project,
    vis = concatvis+'.ms',
    imsize = imsize,
    cell = cell,
    featherimage = project+'/'+project+".sd.image",
    weighting = 'briggs',
    threshold = '30mJy',
    niter = 10000,
    pbcor = True,
    analyze = True,
    showuv = True,
    showpsf = False,
    showmodel = True,
    showconvolved = True,
    showclean = True,
    showresidual = False,
    showdifference = True,
    verbose = True,
    showfidelity = True
)

###########################################

#
# Feathering
#

os.chdir(project)

## Taken from ALMA imaging school script by Miyamoto-san
# ----------------------------------------------------------------------
# step 7-1: Prepare Images for Feathering
# ----------------------------------------------------------------------
#check the tp image
imhead(imagename= project+".sd.image",mode='list')

# regrid the single dish image to match 7m+12m image
os.system('rm -rf %s.sd.image.regrid'%project)
imregrid(imagename='%s.sd.image'%project,
         template= '%s.image'%concatvis,
         output='%s.sd.image.regrid'%project)

# multiply 7m+12m and the TP image to have common response on the sky
os.system('rm -rf %s.sd.image.depb' % project)
immath(imagename=['%s.sd.image.regrid'%project,'%s.flux'%concatvis],
       expr='IM0*IM1',
       outfile='%s.sd.regrid.depb'%project)

# ----------------------------------------------------------------------
# step 7-2: Feather TP Cube with 7m+12m Cube and check the result
# ----------------------------------------------------------------------

feather(imagename='%s.Feather.image'%project,
```

```
        highres='%s.image'%concatvis,
        lowres='%s.sd.regrid.depb'%project)

# ---------------------------------------------------------------------------
# step 7-3: Correct the Primary Beam Response
# ---------------------------------------------------------------------------
# apply the primary beam response to the feathered image

os.system('rm -rf %s.Feather.image.pbcor'%project)
```