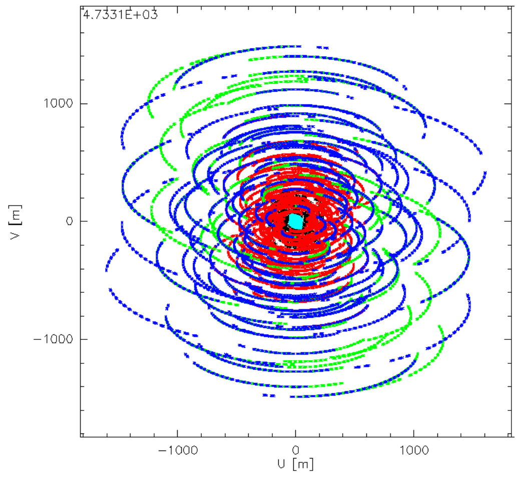
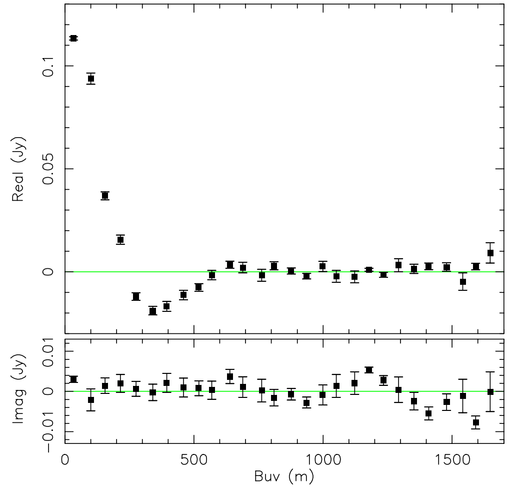
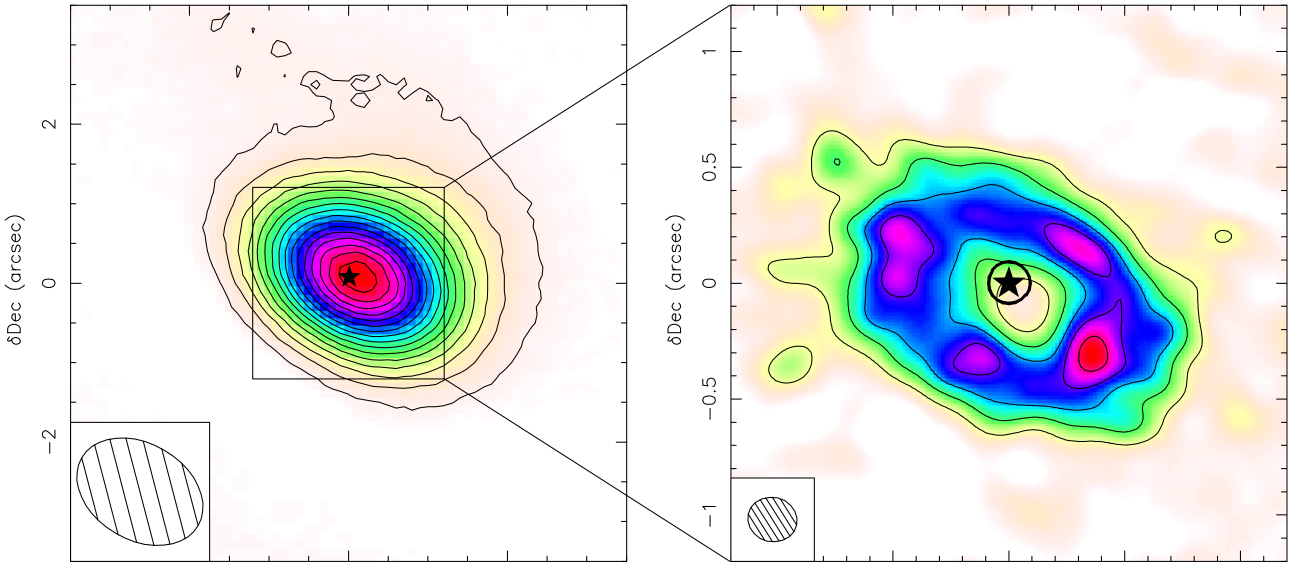
Radio Imaging and CLEAN





NATURAL WEIGHTING



UNIFROM WEIGHTING

FWHM beam size = 1.7'' x 1.2''   0.21'' x 0.19''

Credits

- Slides borrowed from :
  - https://science.nrao.edu/facilities/alma/naasc-workshops/almadata/indebetouw.pdf
    - "Imaging ALMA Data" by Remy Indebetouw
    - In turn borrowed from David Wilner, Steve Myers, and Scott Schnee
  - http://www.almatelescope.ca/workshop/Talks/CASAImaging_Hamilton.pdf
    - "Imaging with CASA" by Crystal Brogan
  - http://www.aoc.nrao.edu/events/synthesis/2010/lectures/wilner_synthesis10.pdf
    - "Imaging and Deconvolution" by David Wilner @ VLA Summer School
  - http://www.das.inpe.br/school/2011/lectures/RickPerley_Lecture4.pdf
    - "Imaging and Deconvolution" by Rick Perley
- General references
  - Thompson, A.R., Moran, J.M., & Swensen, G.W., "Interferometry and Synthesis in Radio Astronomy"
  - VLA Summer School Lectures (available online)

Outline

- Brief review of synthesis imaging
- Demonstration of deconvolution (i.e, “clean”)
- CASA clean - basics
  - images
  - imsize, cell size, mode, mask
  - when to stop cleaning
  - weighting
- CASA clean - advanced topics
  - multiscale clean
  - combining interferometry data with single dish
  - self-calibration
  - mosaicking

From Sky Brightness to Visibility

1. An interferometer measures the interference pattern produced by two apertures.
2. The interference pattern is directly related to the source brightness. For small fields of view the complex visibility, *V(u,v)*, is the 2D Fourier transform of the brightness on the sky, *T(l,m)*

van Cittert-Zernike theorem

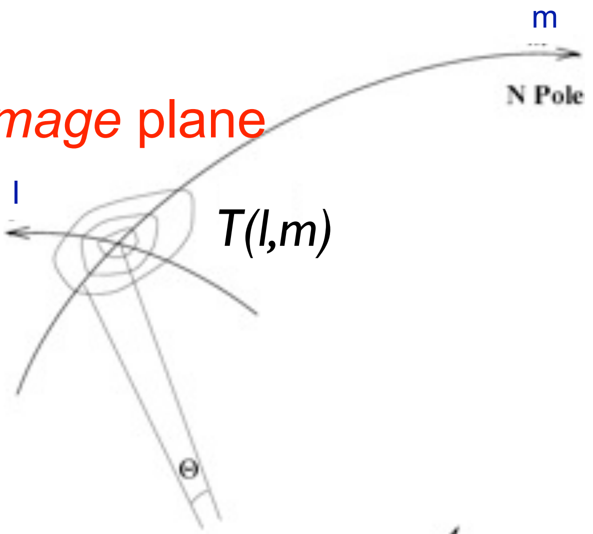
Fourier space/domain

$$V(u,v) = \iint T(l,m) e^{-i2\pi(ul+vm)}\, dl\, dm$$

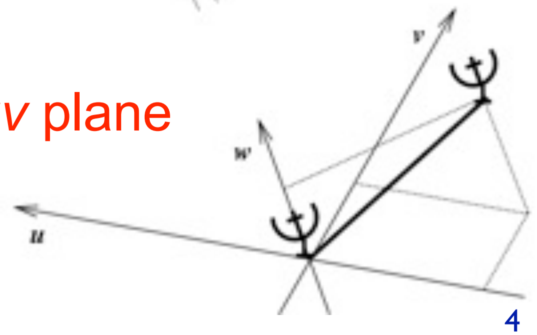
Image space/domain

$$T(l,m) = \iint V(u,v) e^{i2\pi(ul+vm)}\, du\, dv$$

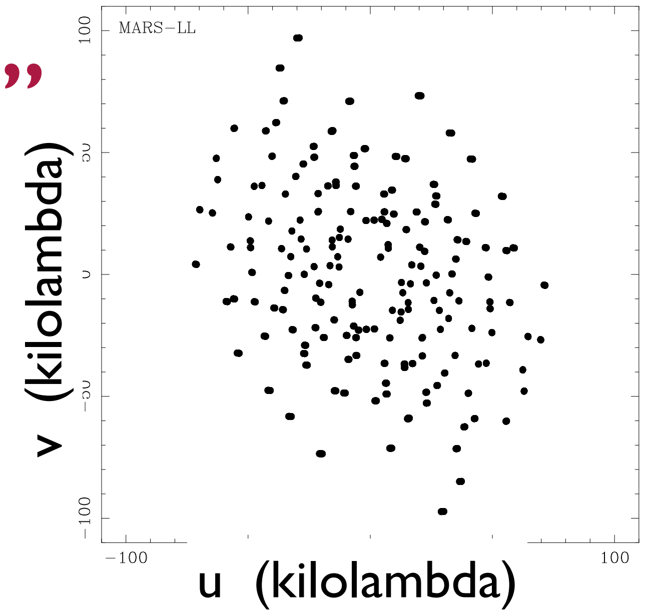
image plane



uv plane



“Dirty image” and “dirty beam”



Finite sampling of the *uv* plane

$$S(u,v) = \sum_{k=1}^{M} \delta(u-u_k, v-v_k)$$

“Dirty beam”

$$S(l,m) = \mathcal{F}^{-1}\{S(u,v)\}$$

Inverse Fourier transform

“Dirty image”

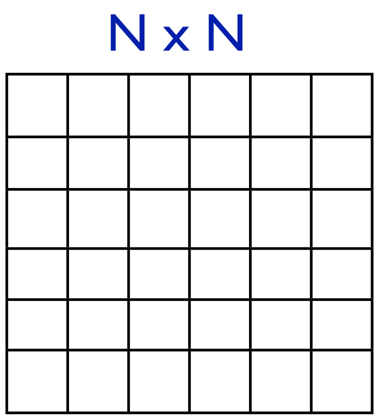
$$T^D(l,m) = \iint S(u,v)V(u,v)e^{i2\pi(ul+vm)}\,dl\;dm$$

$$\equiv \mathcal{F}^{-1}\{\,S(u,v)\times V(u,v)\}$$

$$= S(l,m)\otimes T(l,m)$$

Convolution theorem

The Discrete Fourier Transform



- Create square grid N x N cells in angle space *(l,m)*
- For each cell, compute the sum:

$$T^D(l,m) = \mathcal{F}^{-1}\{\, S(u,v) \times V(u,v)\}$$
$$= \frac{1}{M}\sum_{k=1}^{M} V(u_k,v_k)e^{i2\pi(u_k l + v_k m)}$$

- Need approximately MxN^2 multiplications
- not practical for modern synthesis arrays

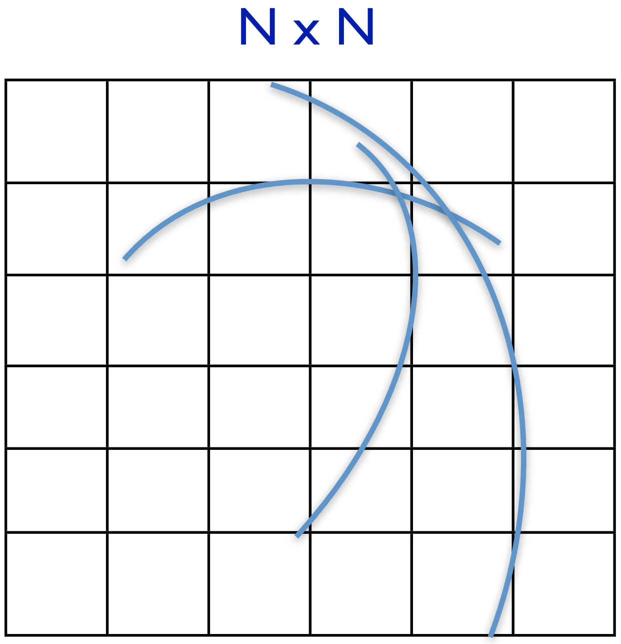
The Fast Fourier Transform (FFT)

- Grid the *u,v* data onto uniformly sampled grid

- Compute the FFT

$$T^D(l,m) = \mathcal{F}^{-1}\{\ S(u,v) \times V(u,v)\}$$

- Computation scales as $N^2 \log_2(N)$



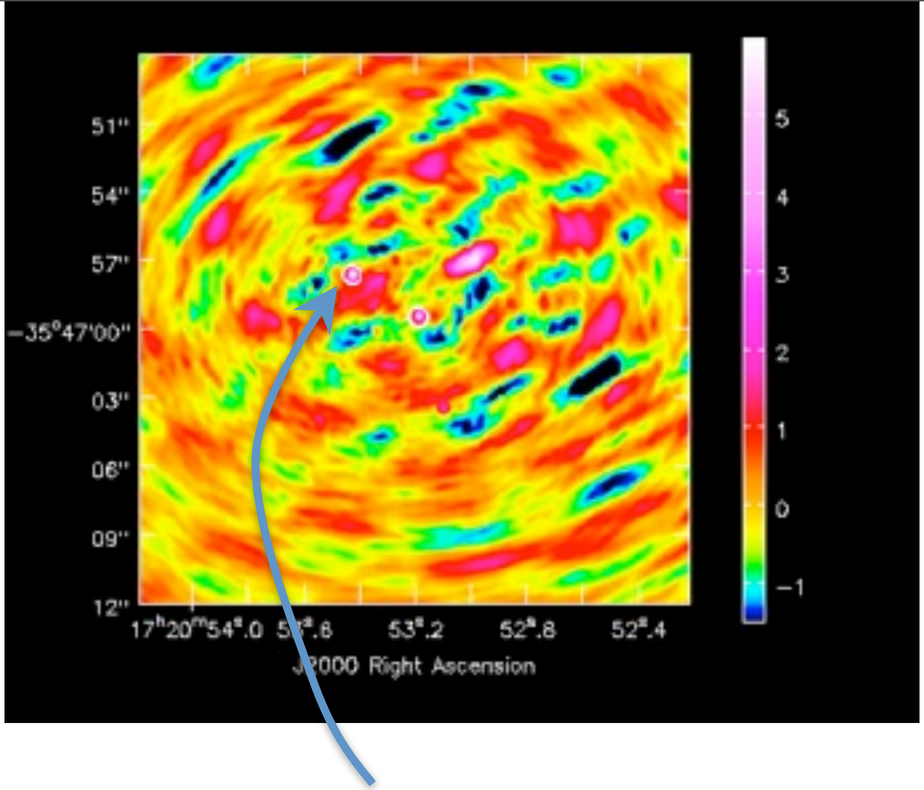
Imaging and Deconvolution

- Consequences of finite *uv* coverage
  - no data beyond u_{max}, v_{max} $\rightarrow$ unresolved structure
  - no data within u_{min}, v_{min} $\rightarrow$ limit on largest size scale
  - holes between u_{min},v_{min} and u_{max}, v_{max} $\rightarrow$ sidelobes
- Empty (u,v) cells can have ANY value
- Noise $\rightarrow$ undetected/corrupted structure in *T(l,m)*
- Interpolate/extrapolate observed *V(u,v)* values to fill in empty *uv* cells

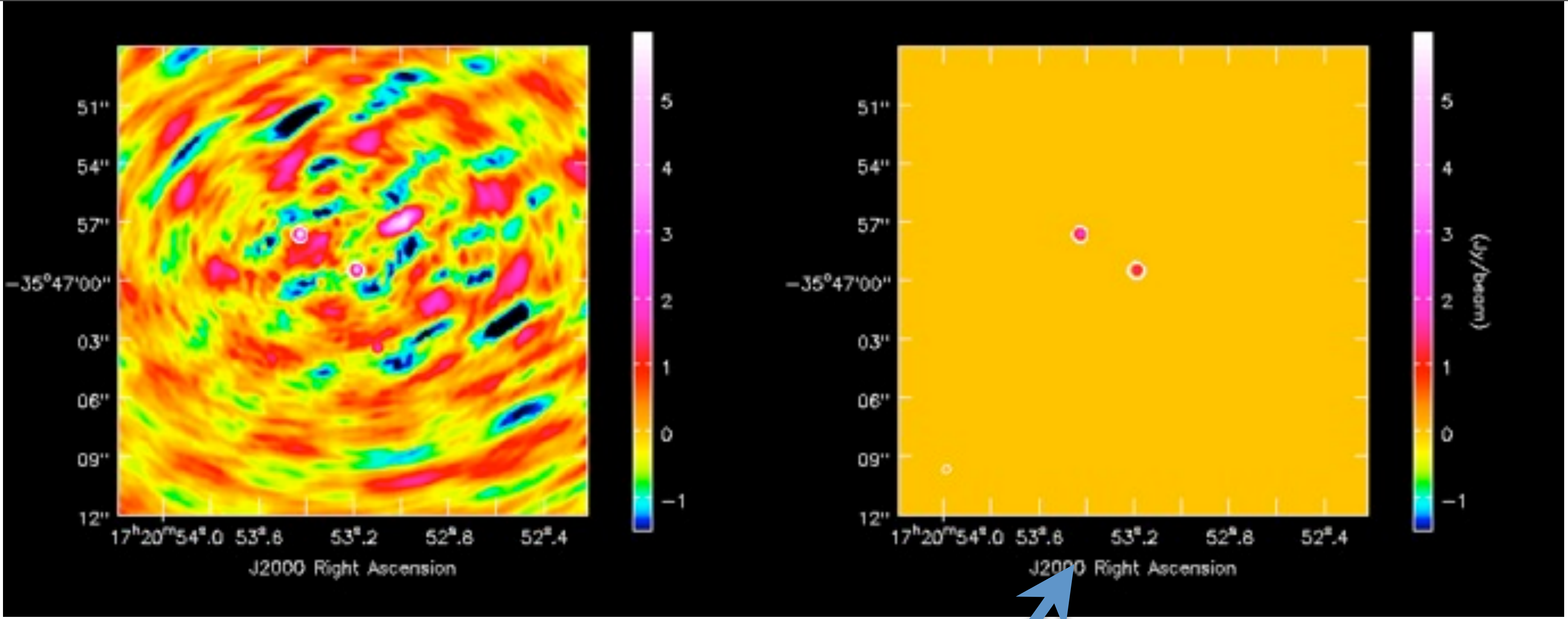
An infinite number of *T(l,m)* compatible with observed *V(u,v)*

Deconvolution Algorithms

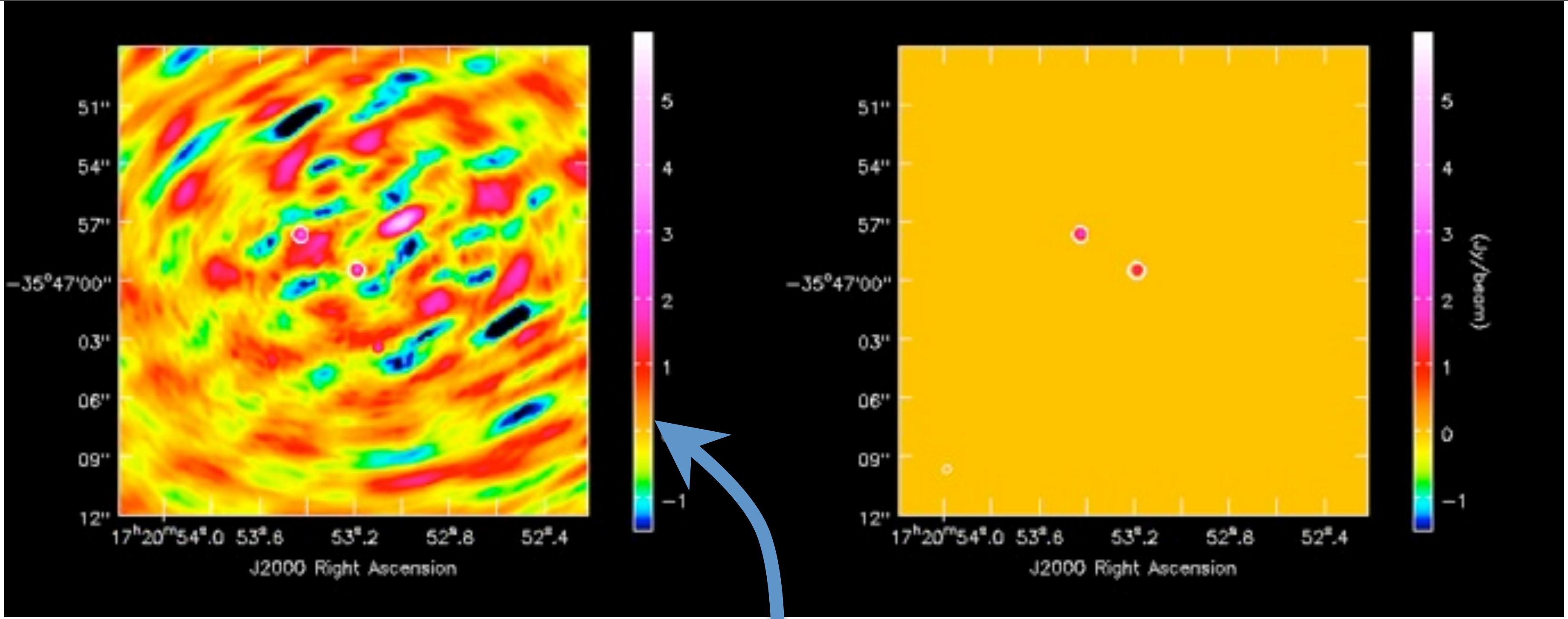
- CLEAN (Högbom 1974)
  - assumes *T(l,m)* is a collection of point sources
  - multi-scale CLEAN relaxes this assumption
- Maximum Entropy (Gull & Skilling 1983)
  - assumes *T(l,m)* is smooth and positive
- Beam shape
  - needed to deconvolve image
  - atmospheric seeing can modify effective beam shape



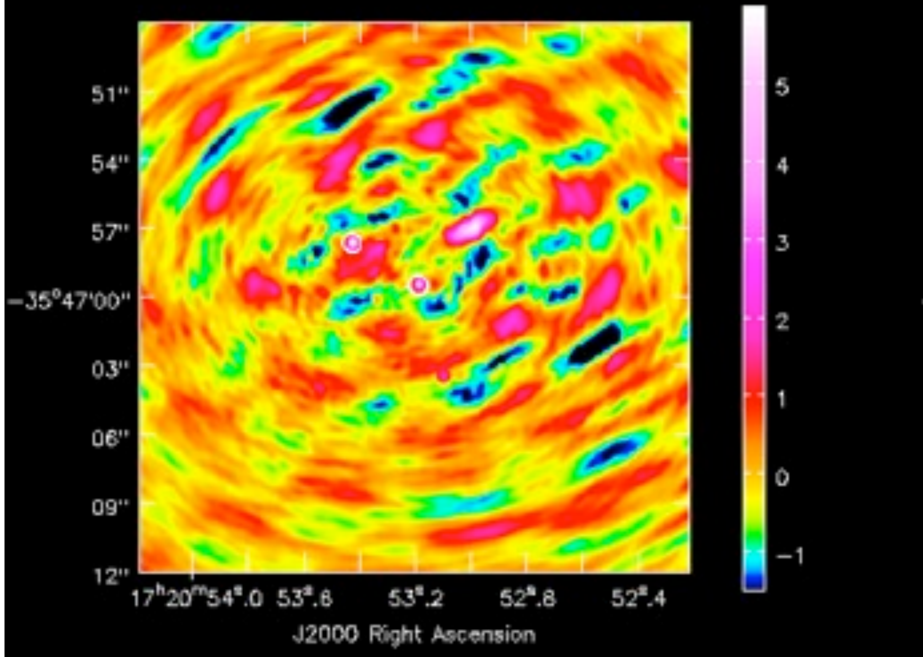
- Find brightest points in dirty image

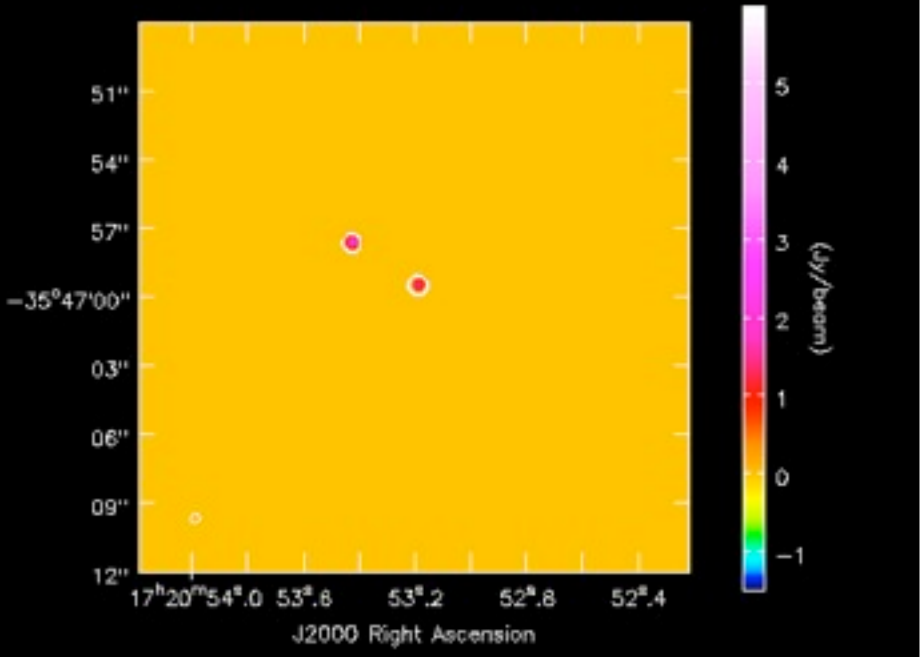


- Find brightest points in dirty image
- Create model image containing a fraction of those flux points

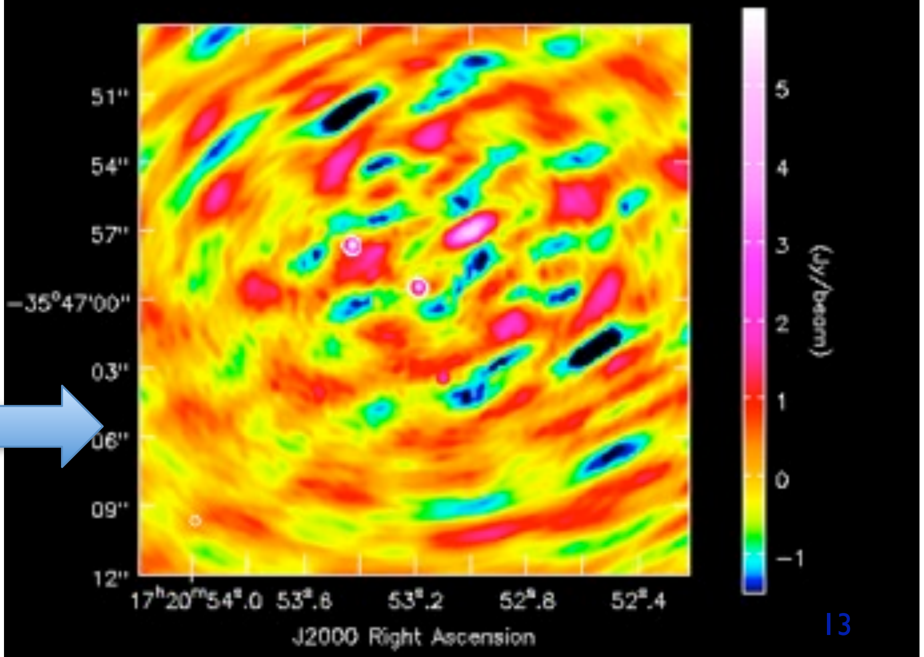


- Find brightest points in dirty image
- Create model image containing a fraction of those flux points
- Subtract model from data, leaving a residual

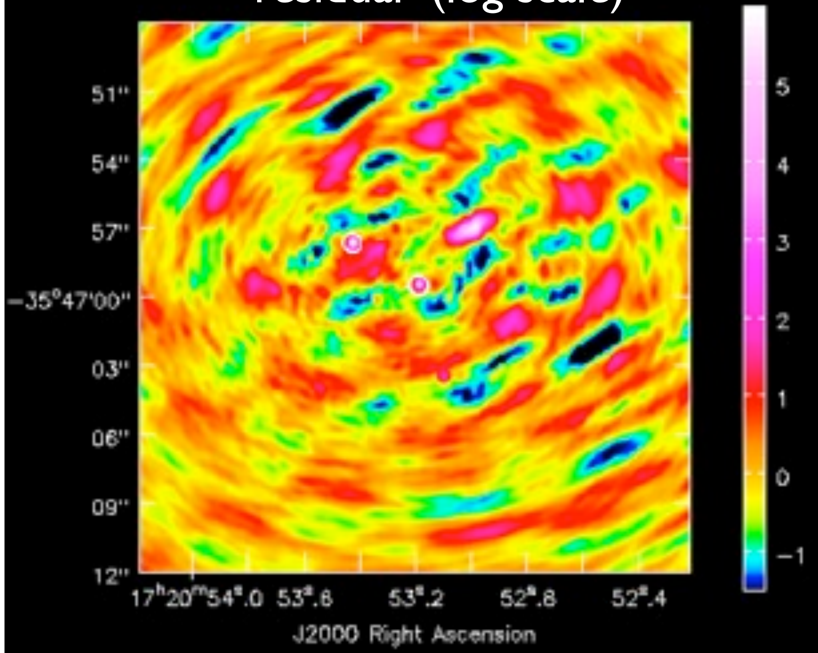




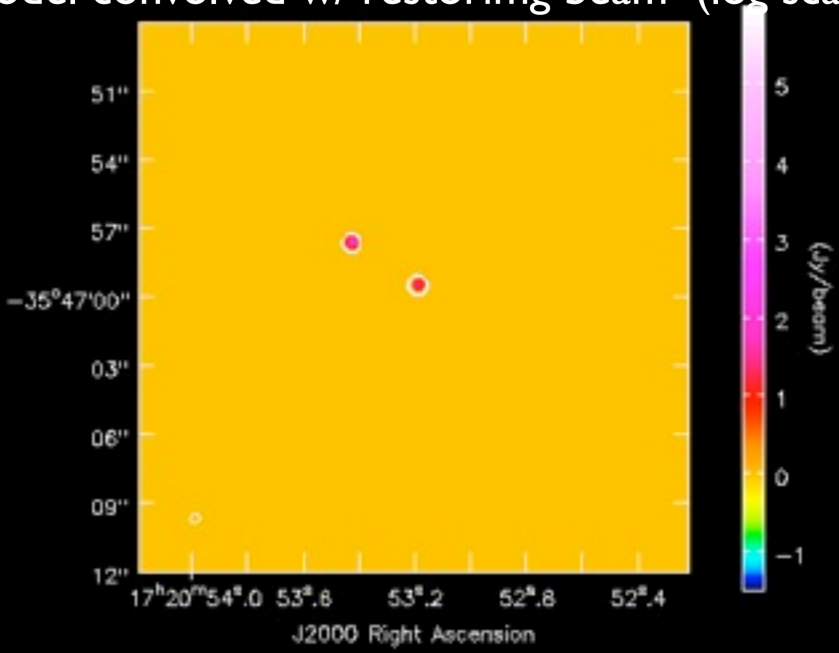
- Find brightest points in dirty image
- Create model image containing a fraction of those flux points
- Subtract model from data, leaving a residual
- Final product = residual + model (convolved with restoring Gaussian beam)



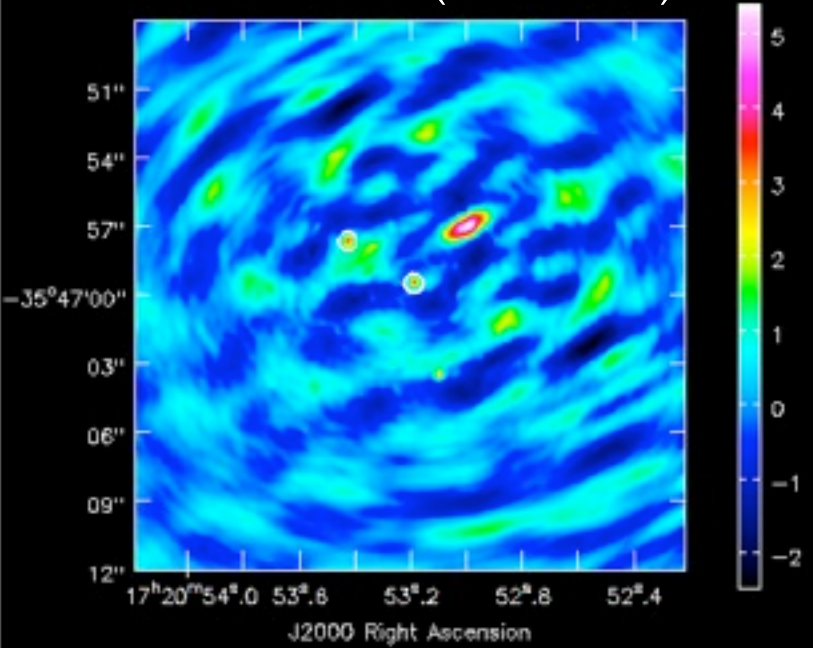
residual (log scale)



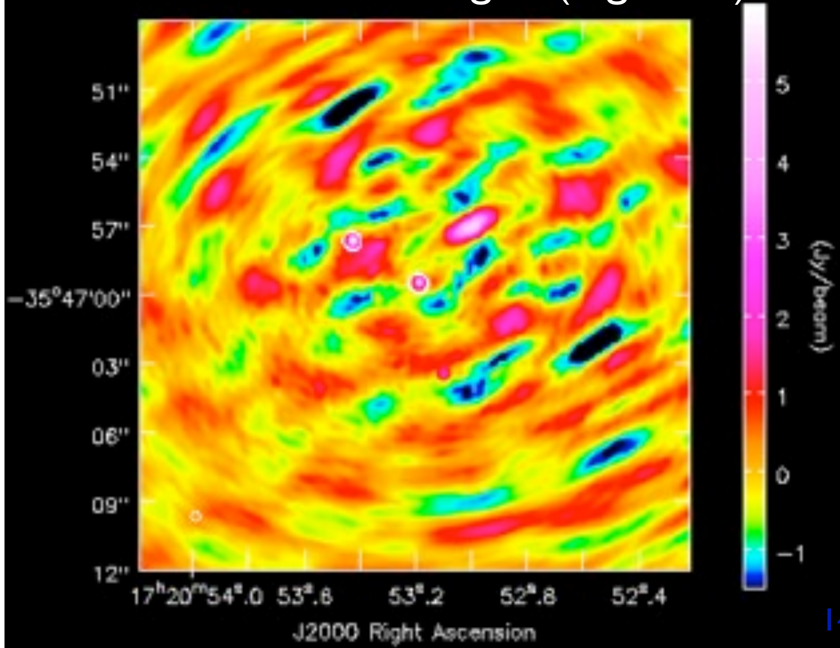
model convolved w/ restorimg beam (log scale)



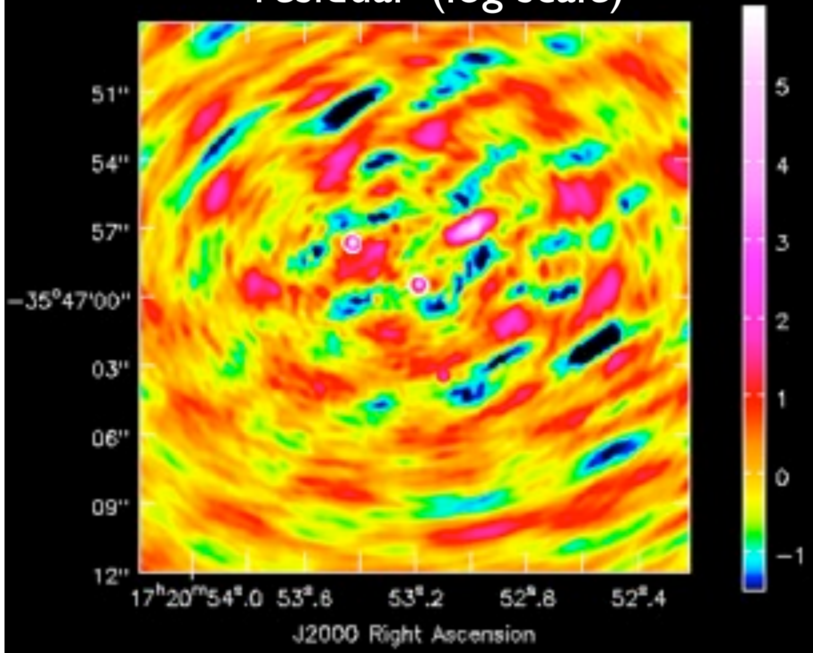
residual (linear scale)



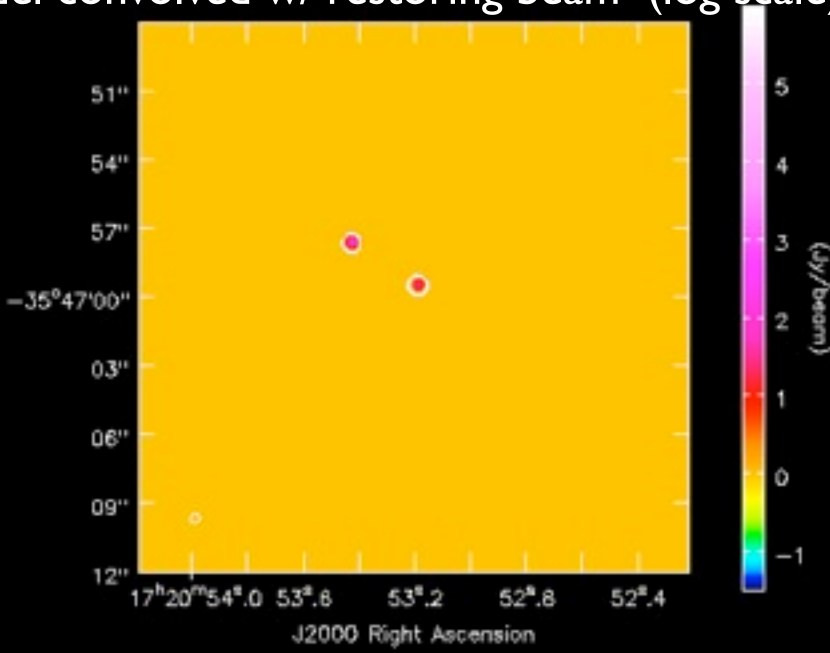
cleaned image (log scale)



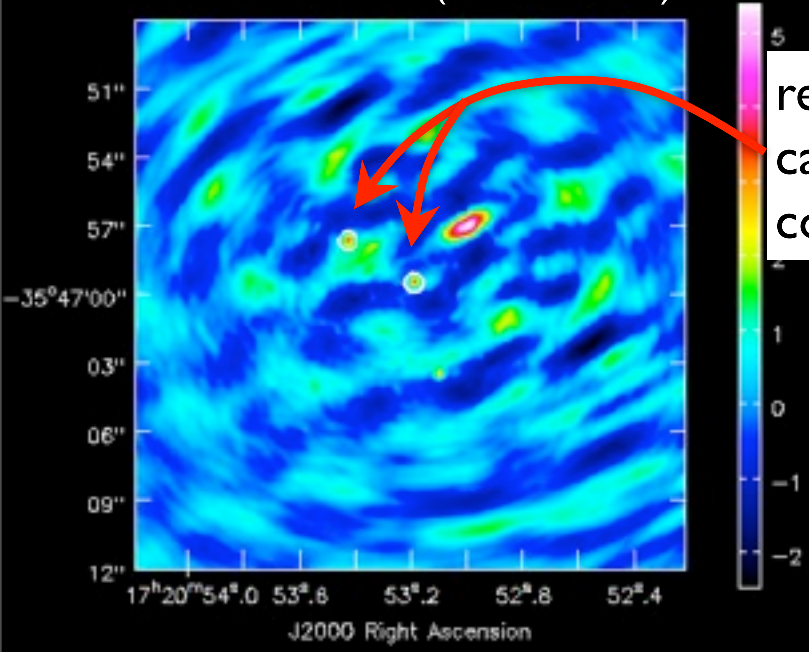
residual (log scale)



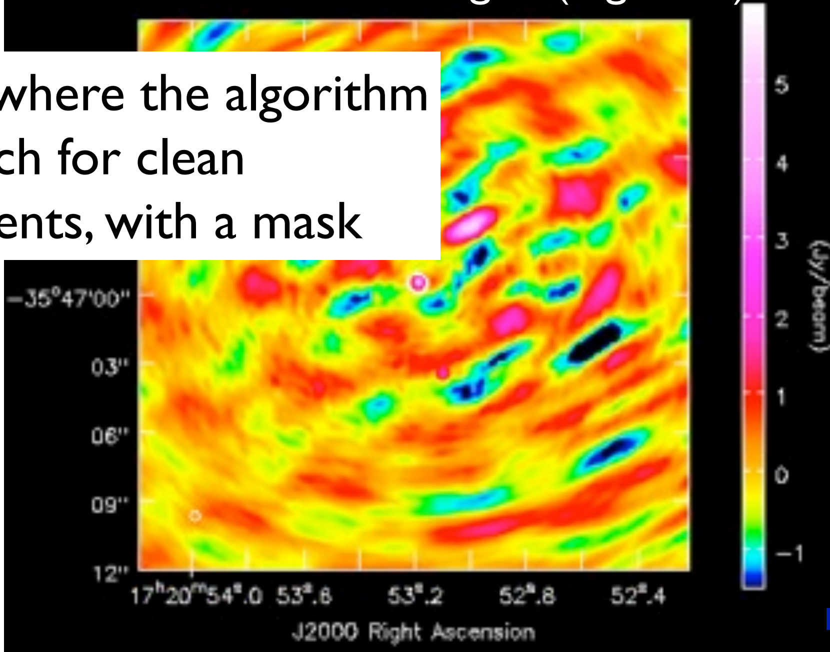
model convolved w/ restoring beam (log scale)



residual (linear scale)

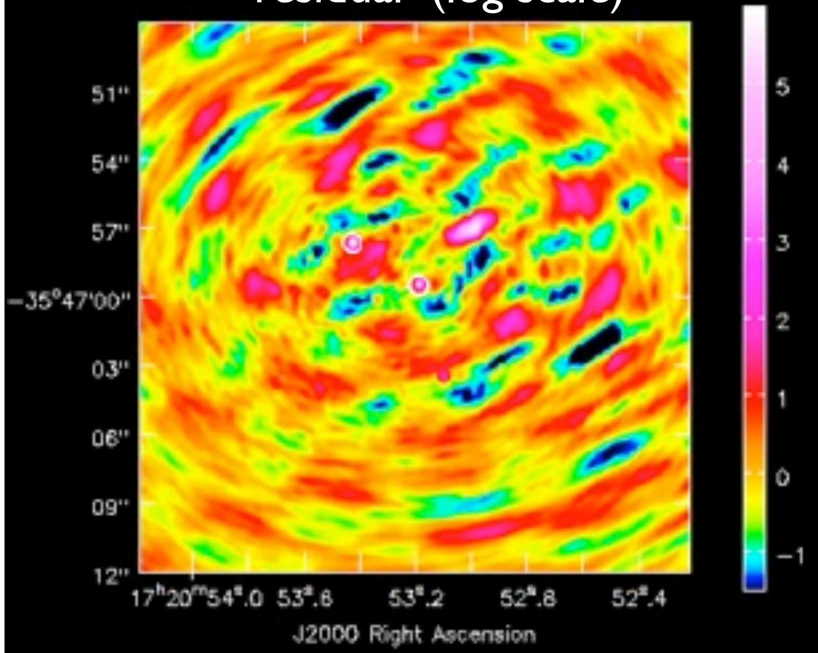


cleaned image (log scale)

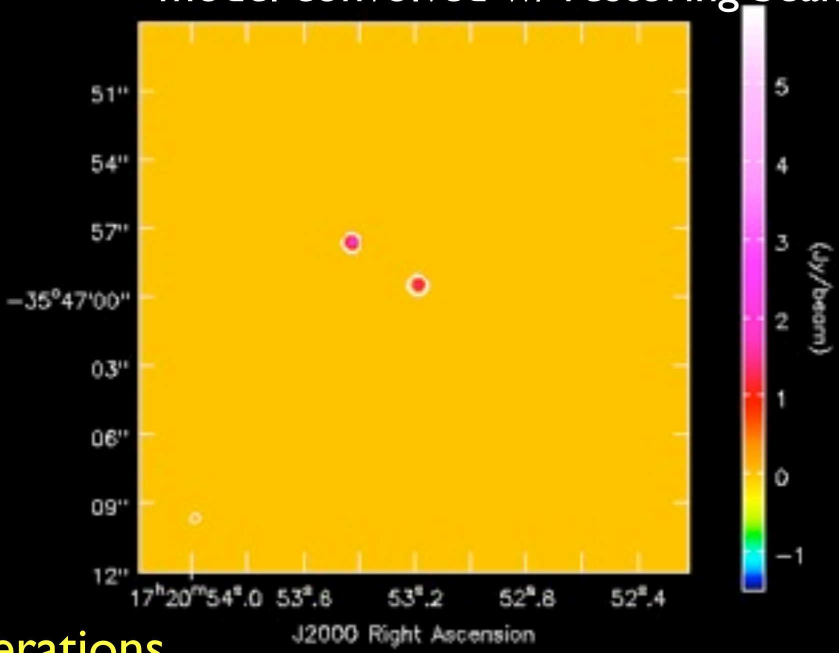


restrict where the algorithm can search for clean components, with a mask

residual (log scale)

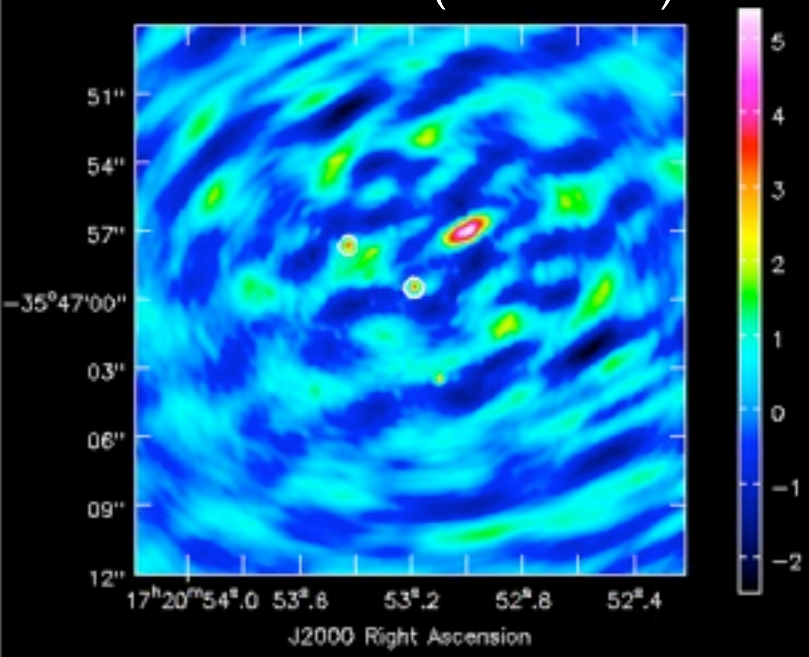


model convolved w/ restoring beam

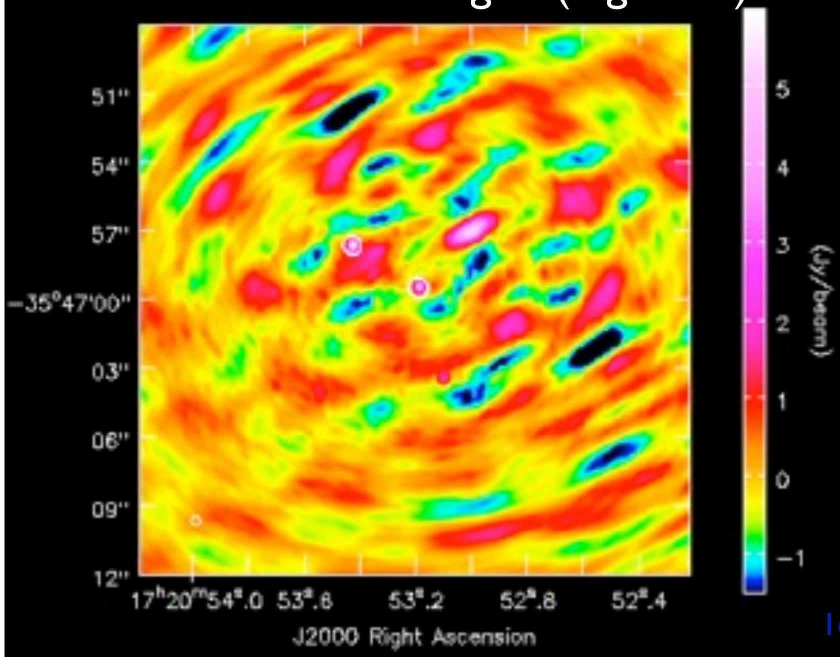


10 iterations

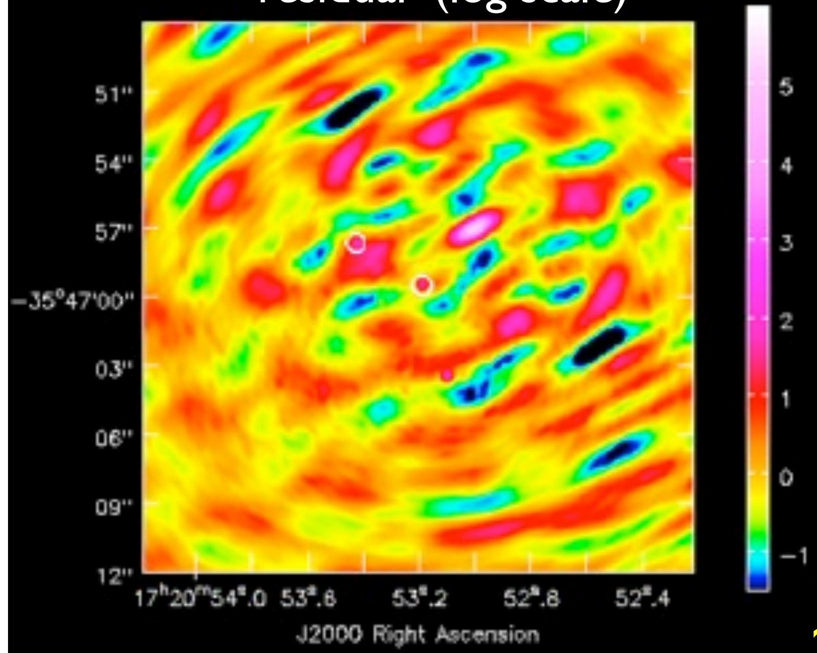
residual (linear scale)



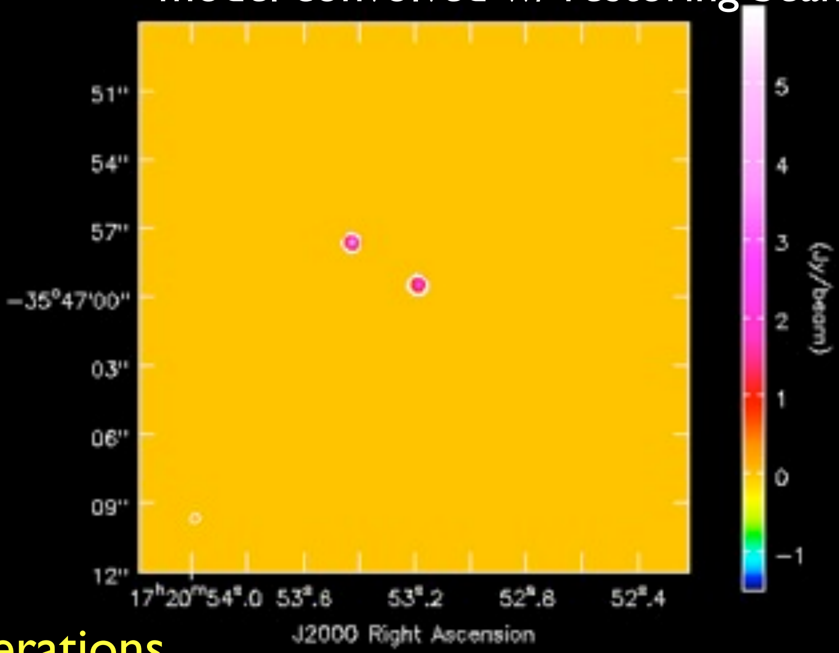
cleaned image (log scale)



residual (log scale)

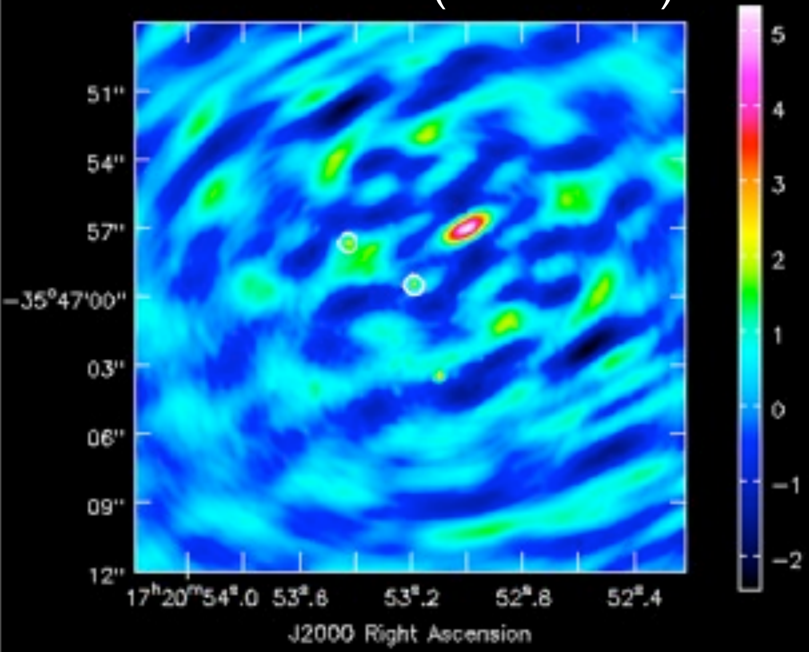


model convolved w/ restoring beam

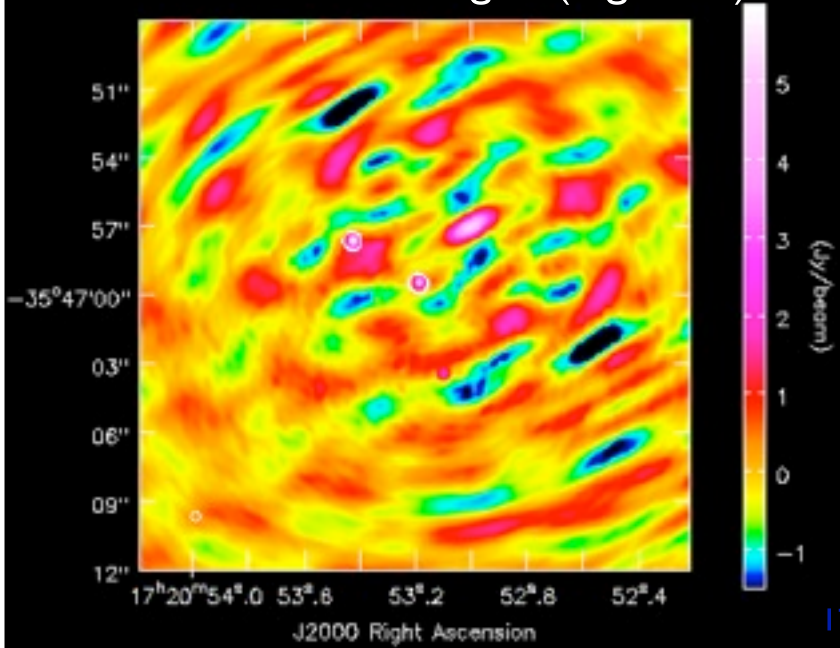


20 iterations

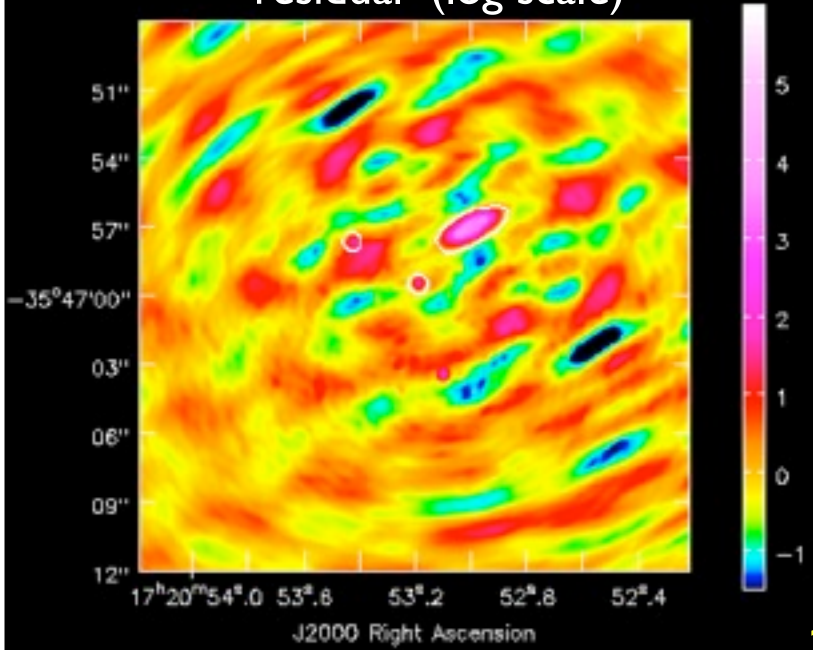
residual (linear scale)



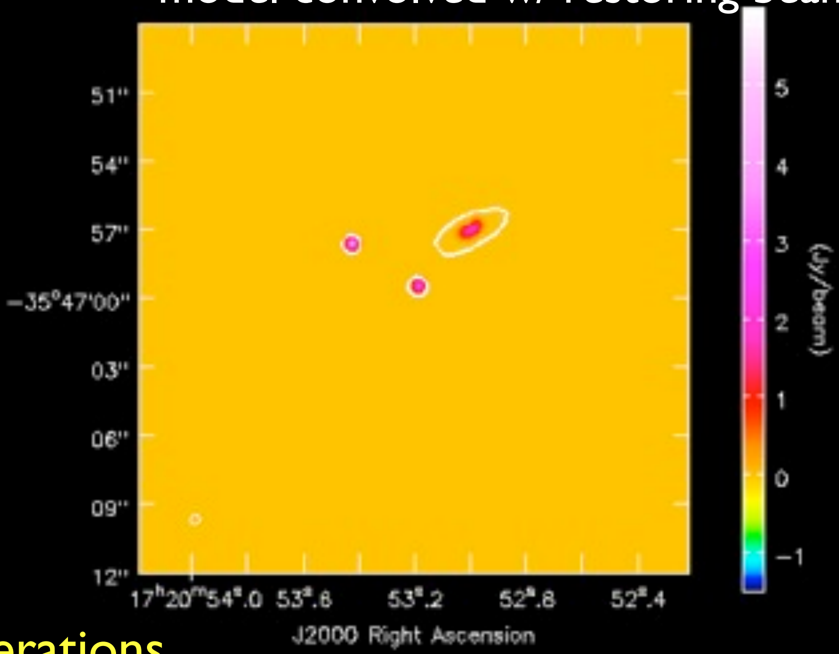
cleaned image (log scale)



residual (log scale)

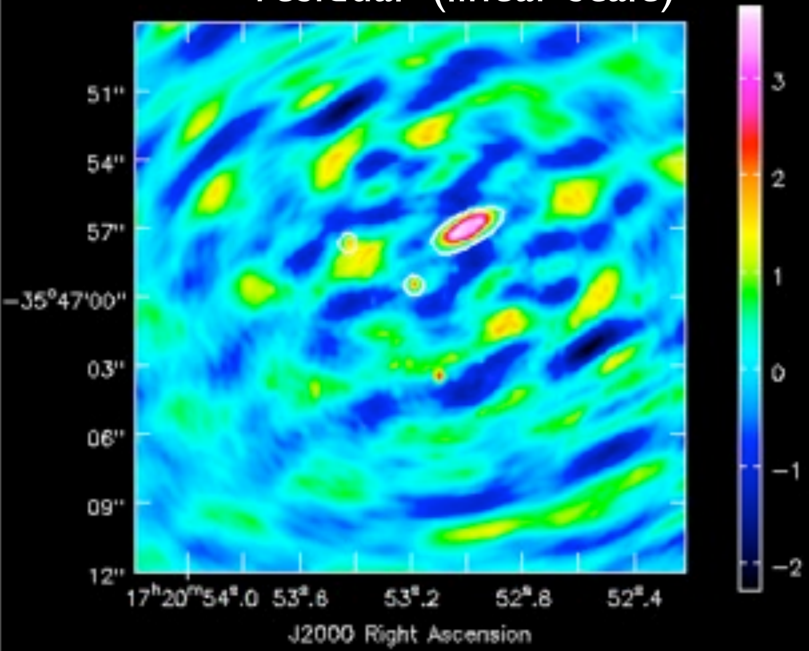


model convolved w/ restoring beam

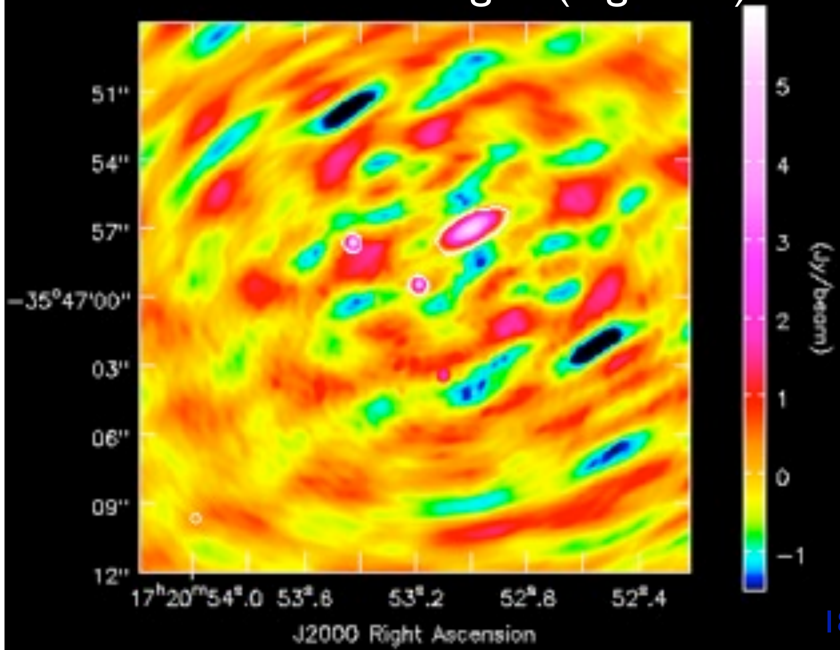


30 iterations

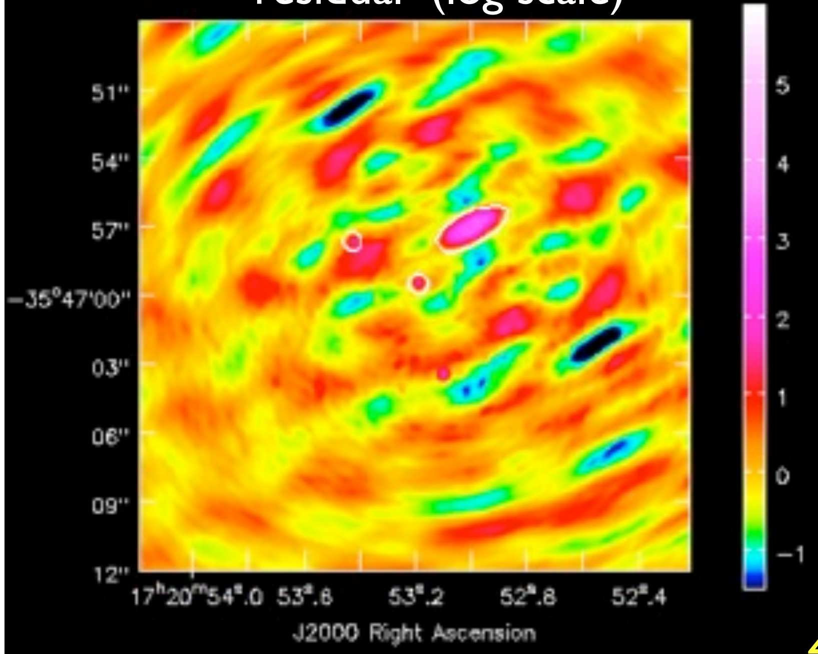
residual (linear scale)



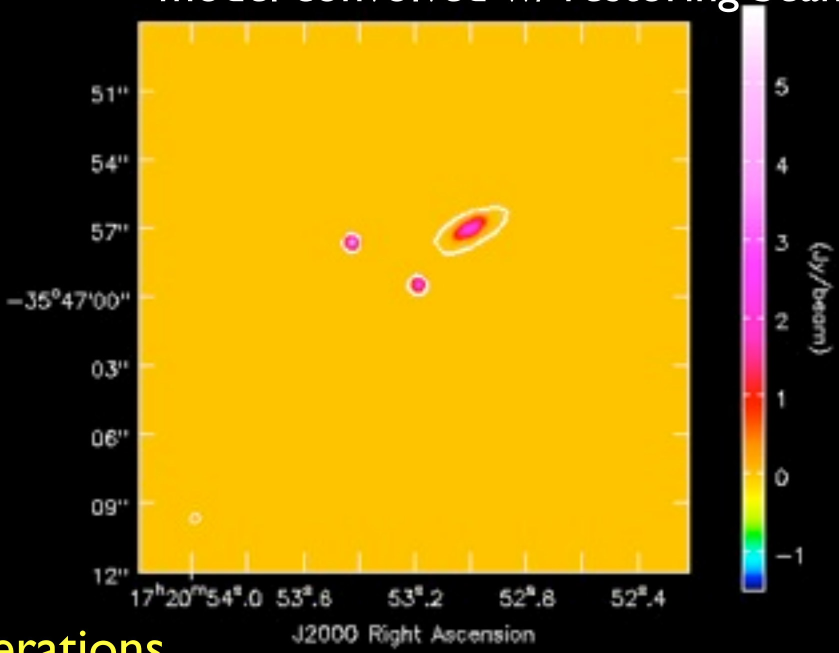
cleaned image (log scale)



residual (log scale)

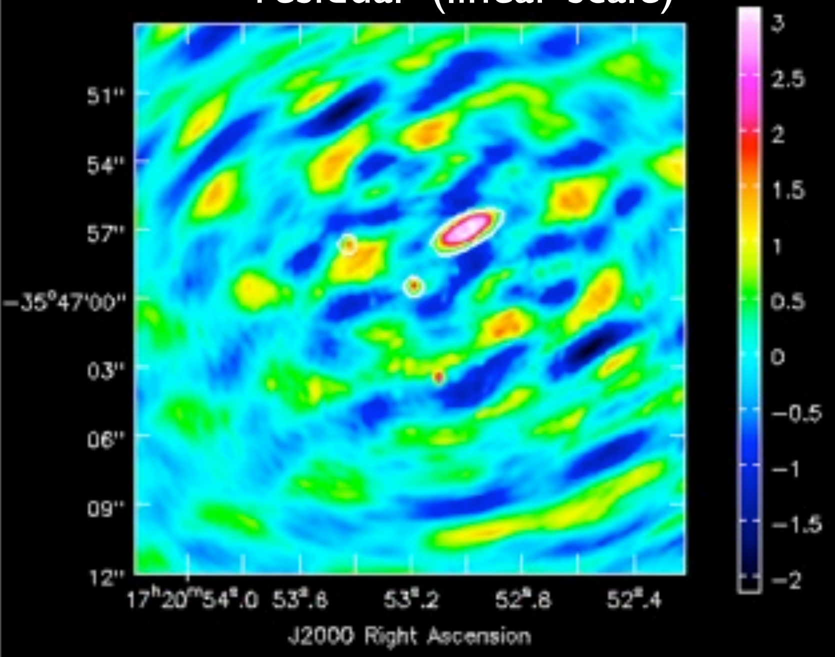


model convolved w/ restoring beam

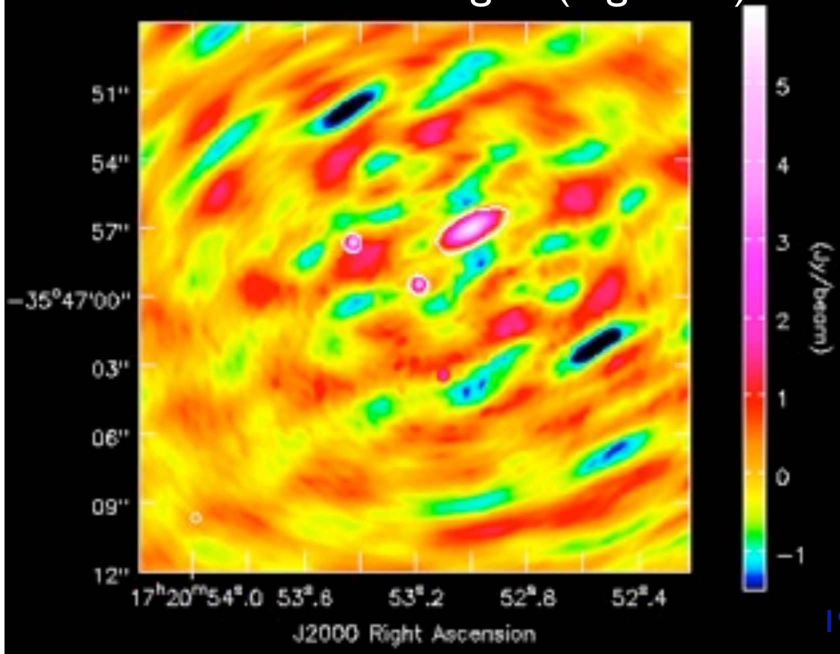


40 iterations

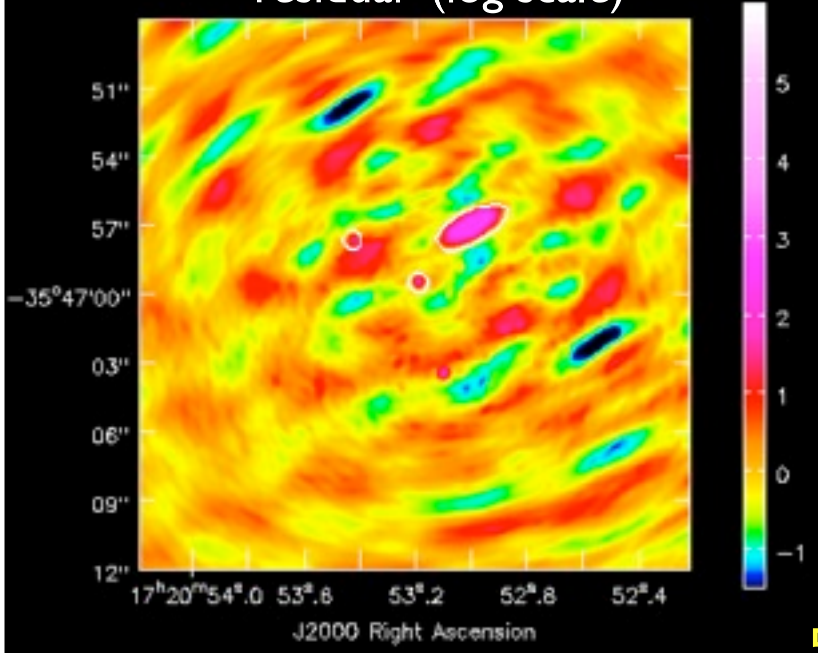
residual (linear scale)



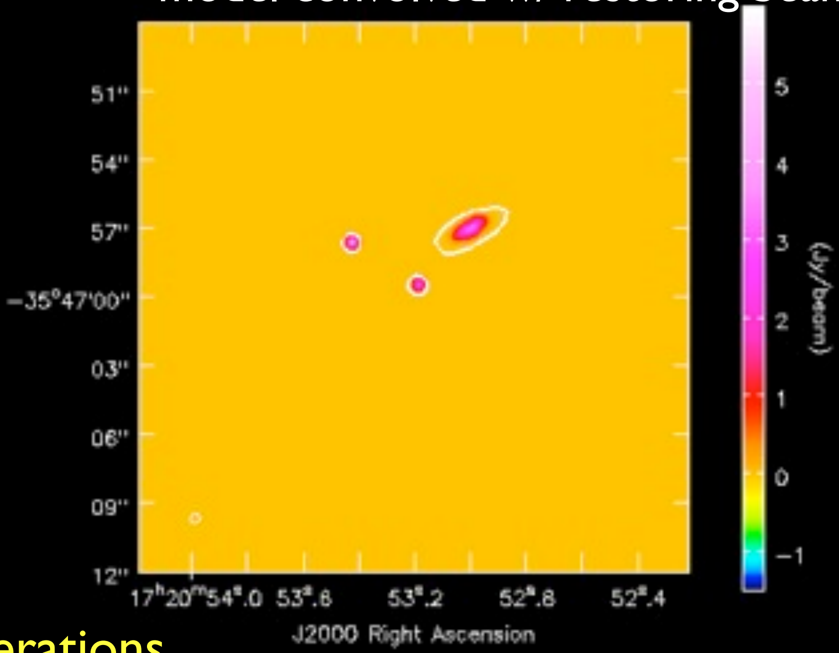
cleaned image (log scale)



residual (log scale)

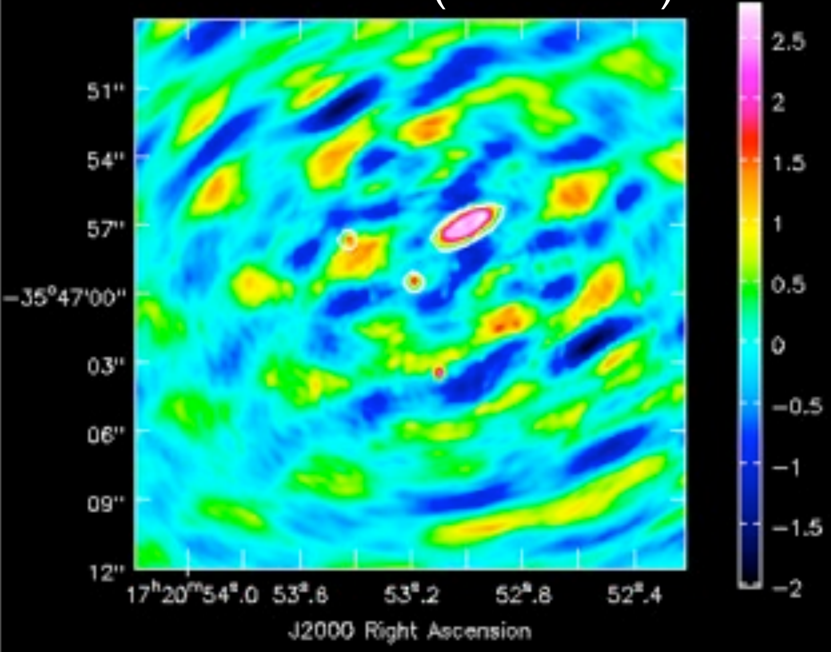


model convolved w/ restoring beam

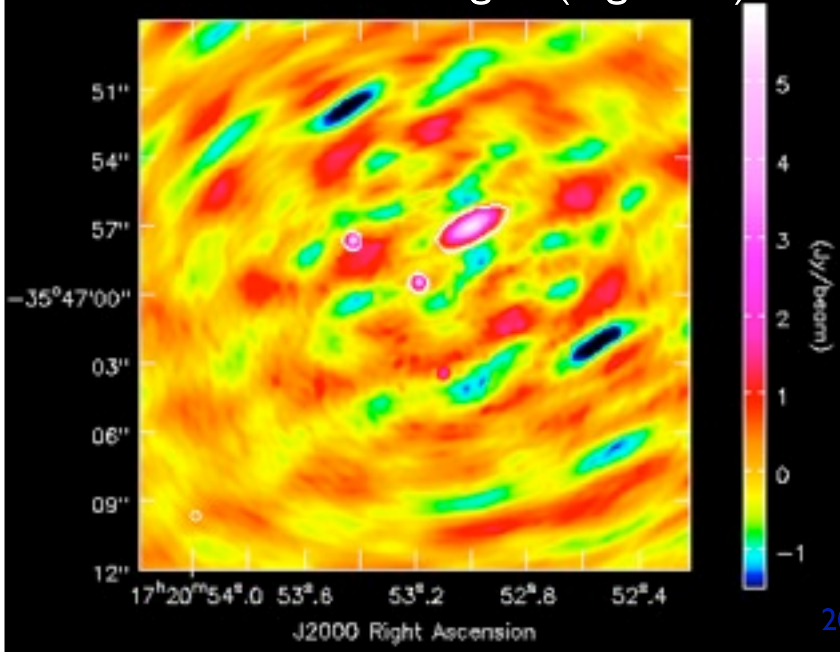


50 iterations

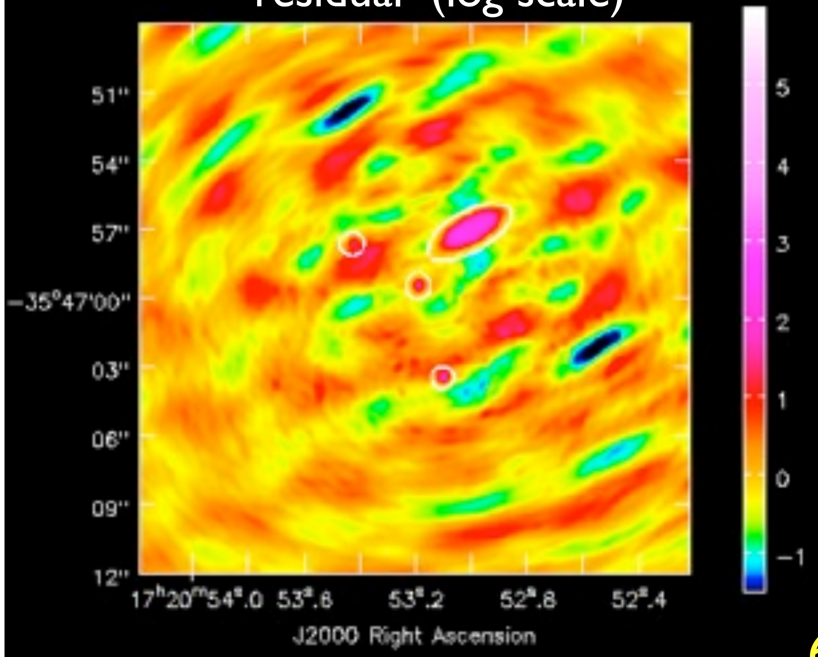
residual (linear scale)



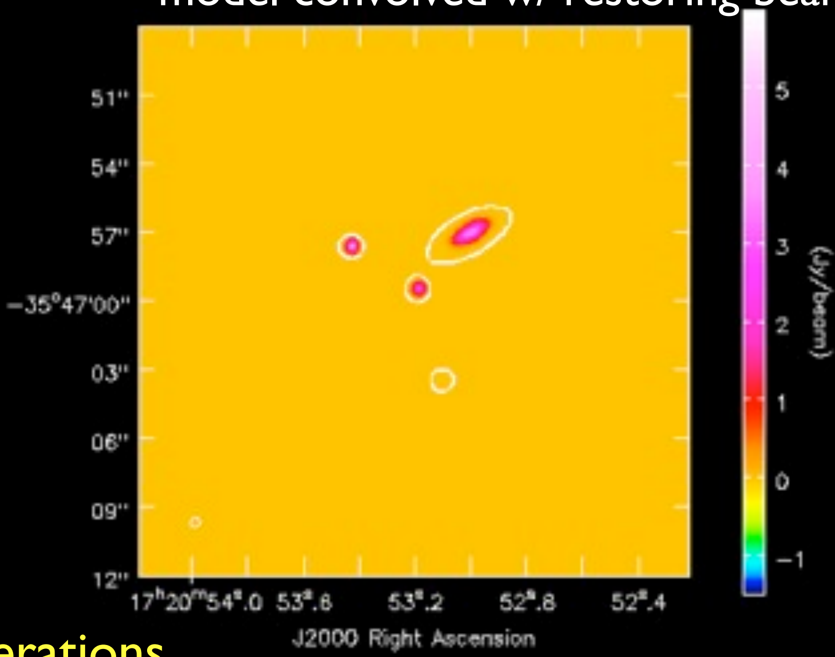
cleaned image (log scale)



residual (log scale)

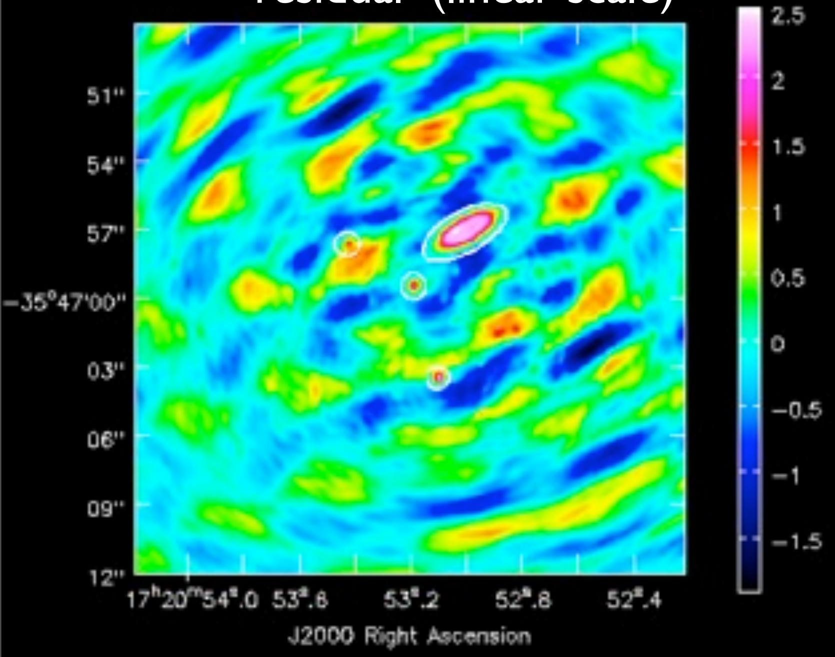


model convolved w/ restoring beam

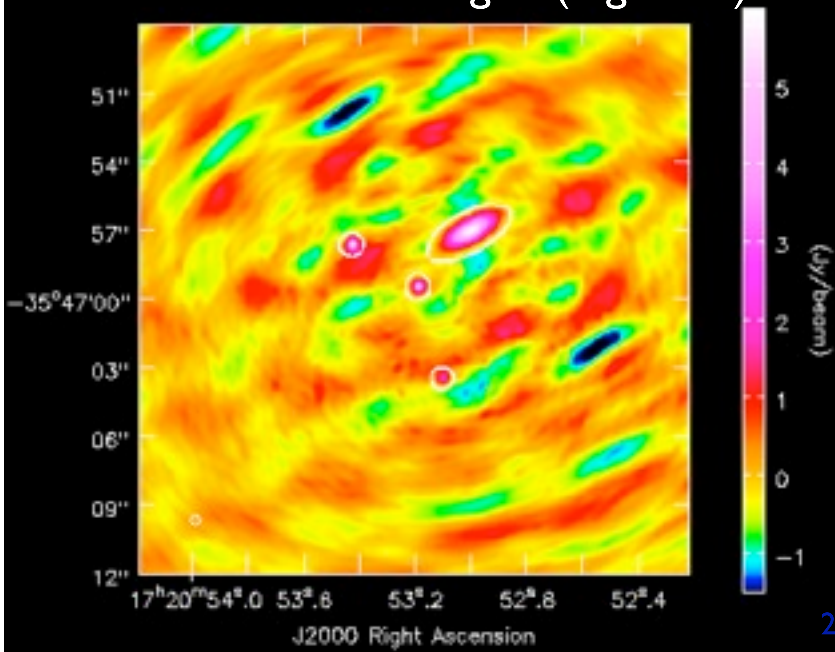


60 iterations

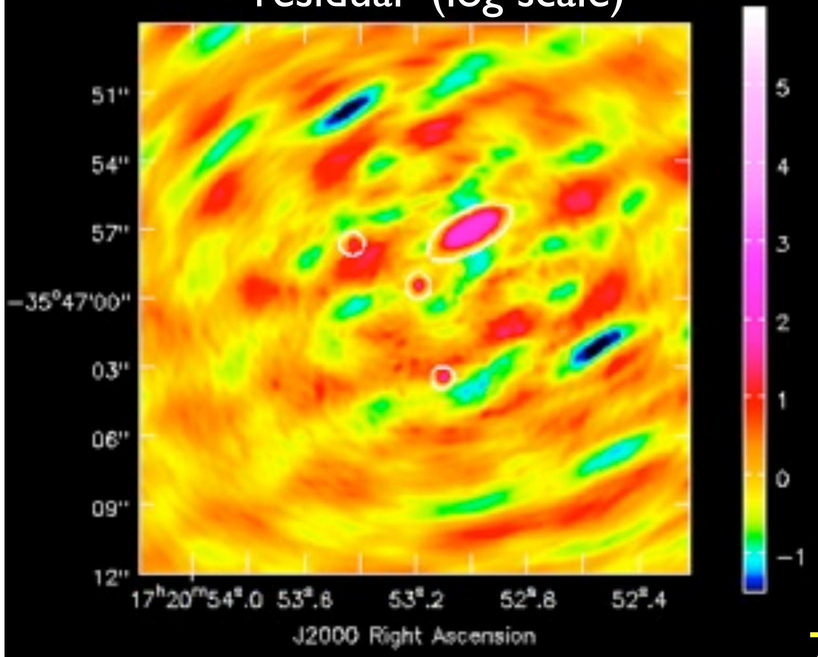
residual (linear scale)



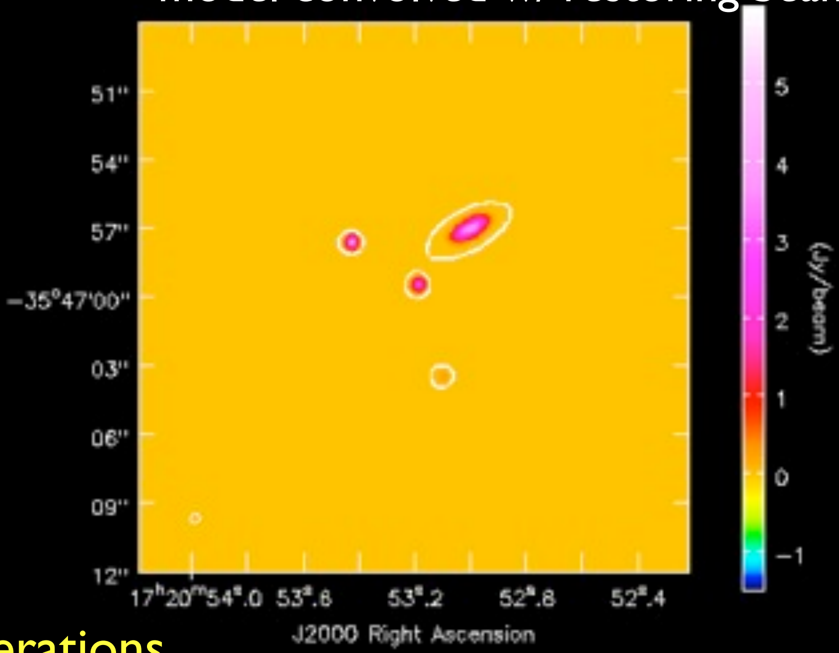
cleaned image (log scale)



residual (log scale)

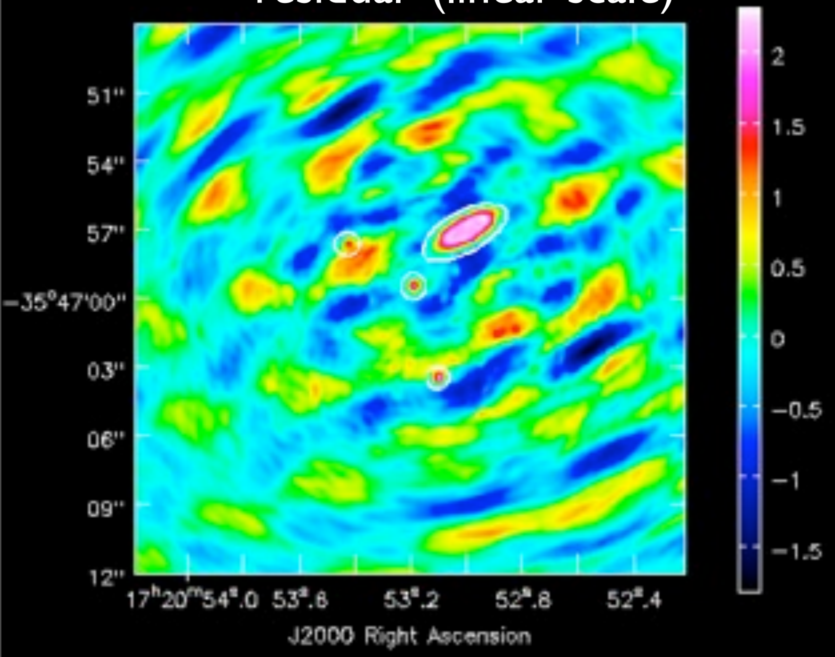


model convolved w/ restoring beam

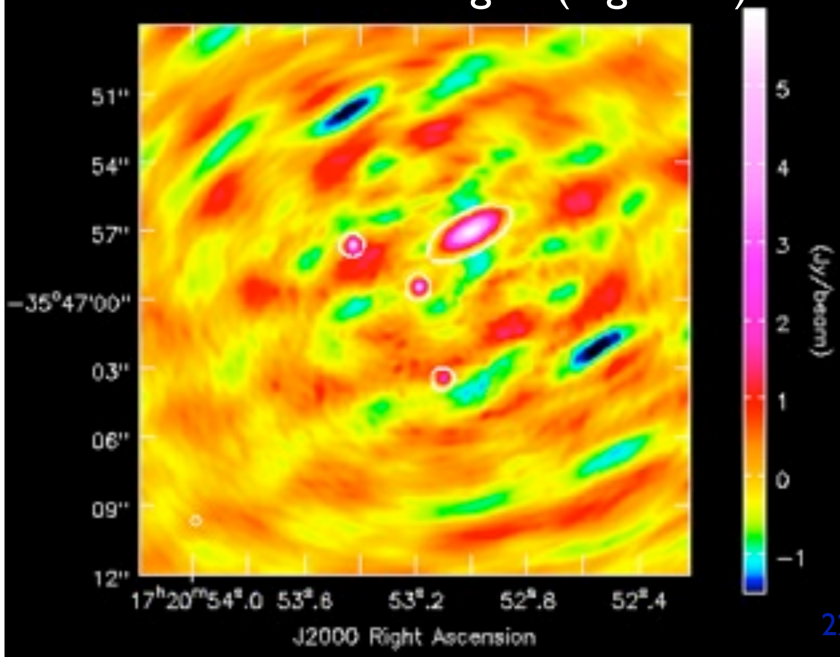


70 iterations

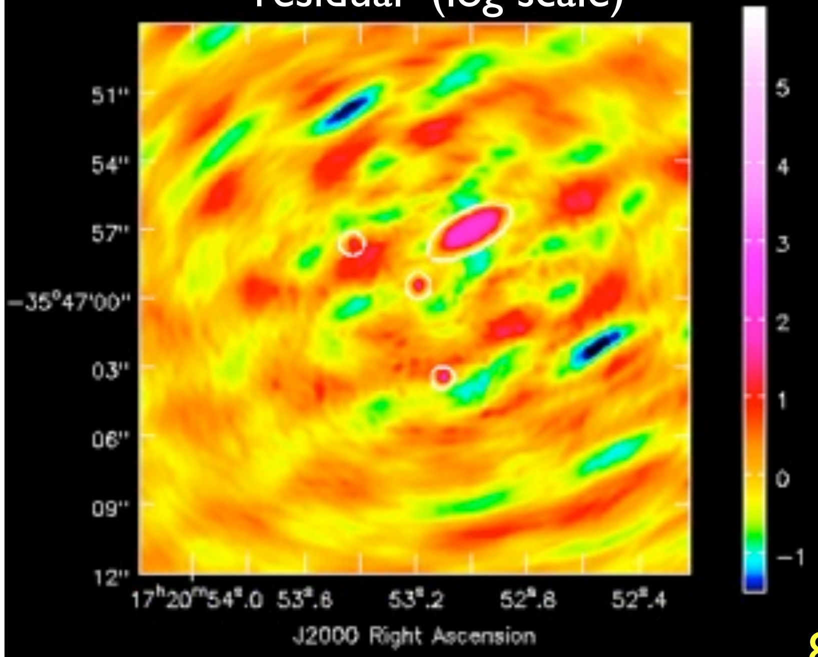
residual (linear scale)



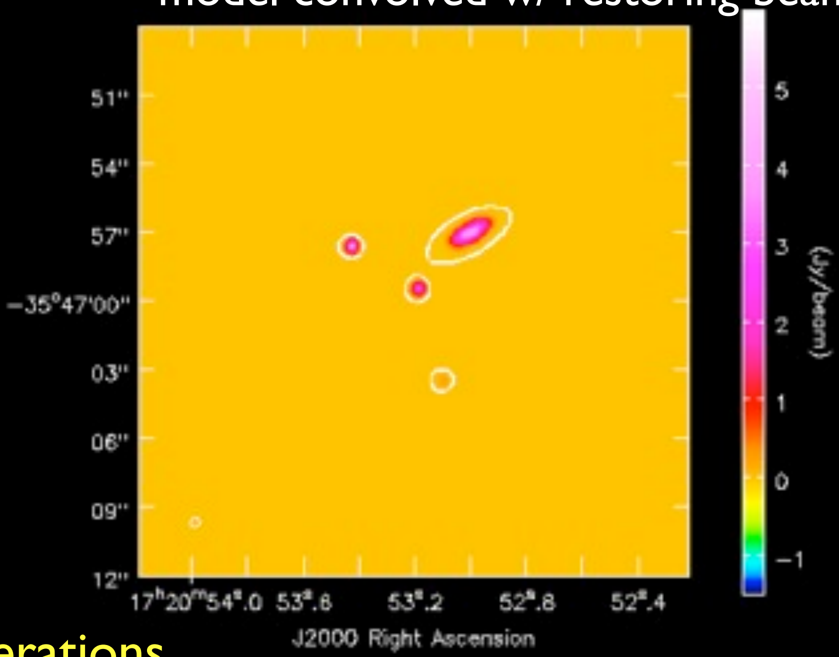
cleaned image (log scale)



residual (log scale)

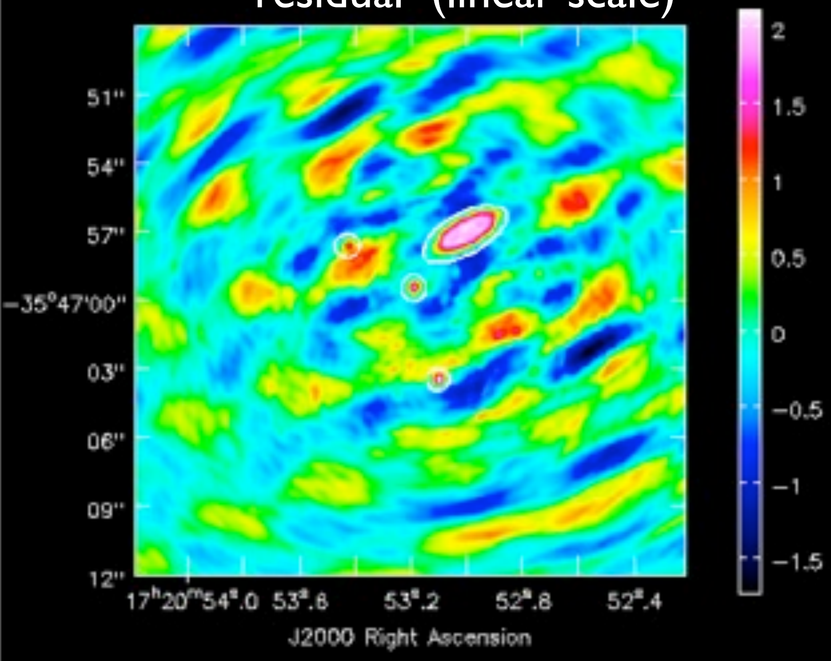


model convolved w/ restoring beam

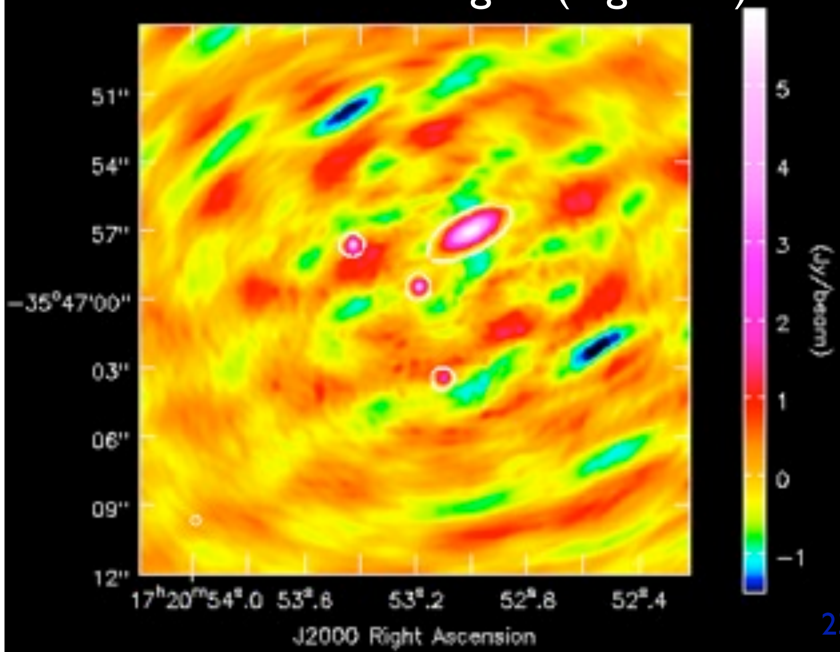


80 iterations

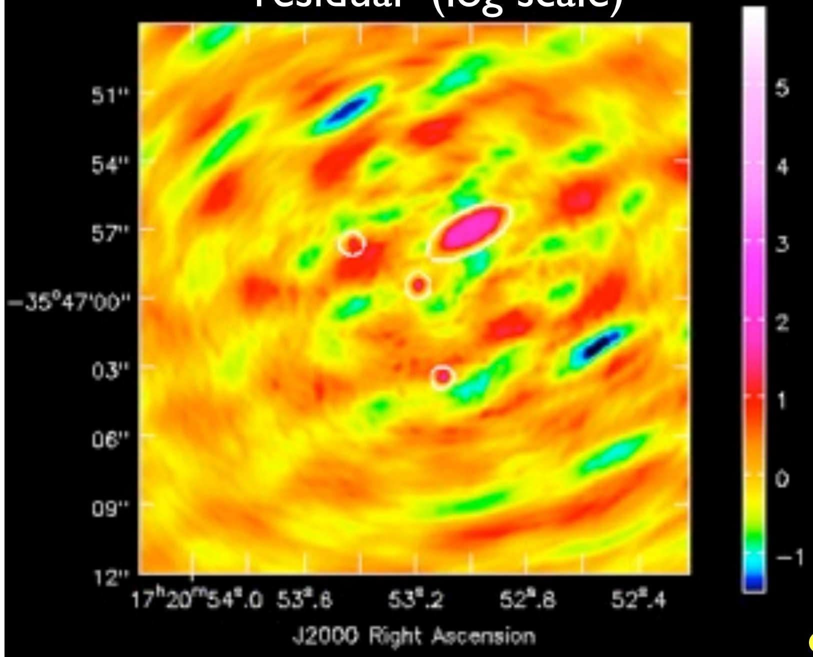
residual (linear scale)



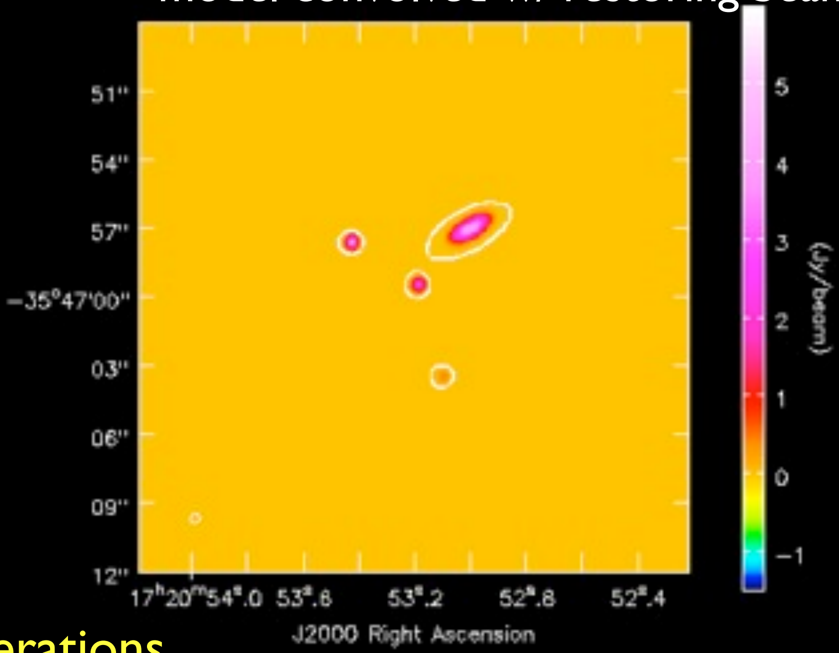
cleaned image (log scale)



residual (log scale)

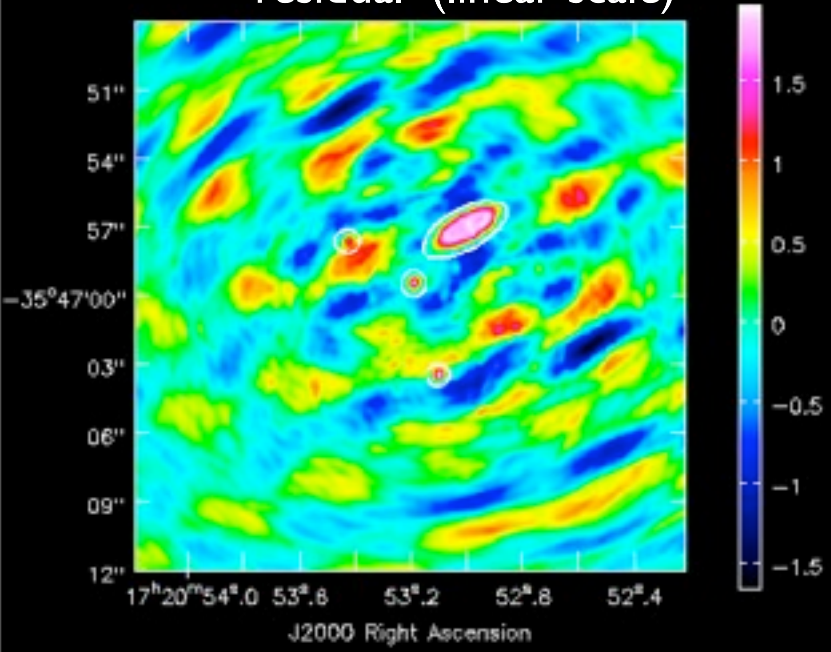


model convolved w/ restoring beam

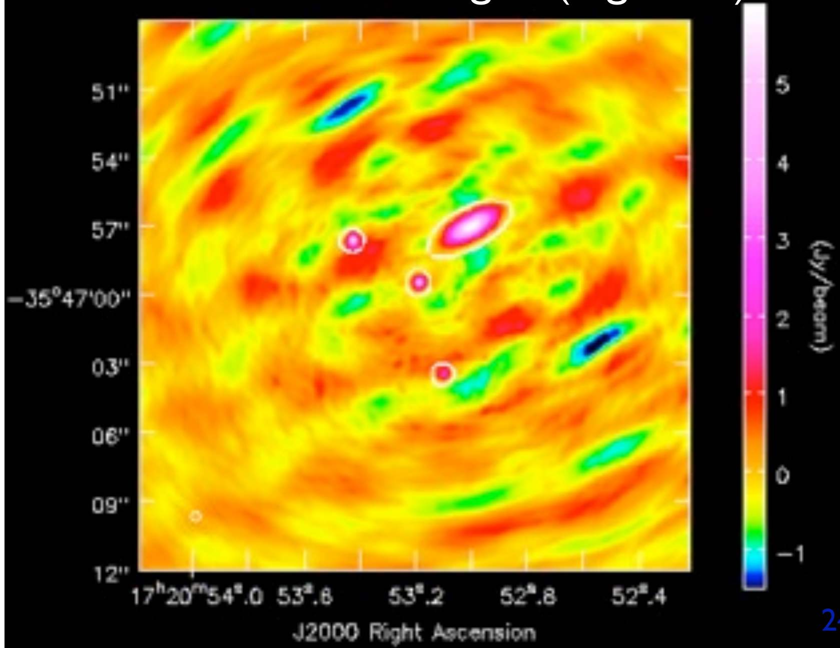


90 iterations

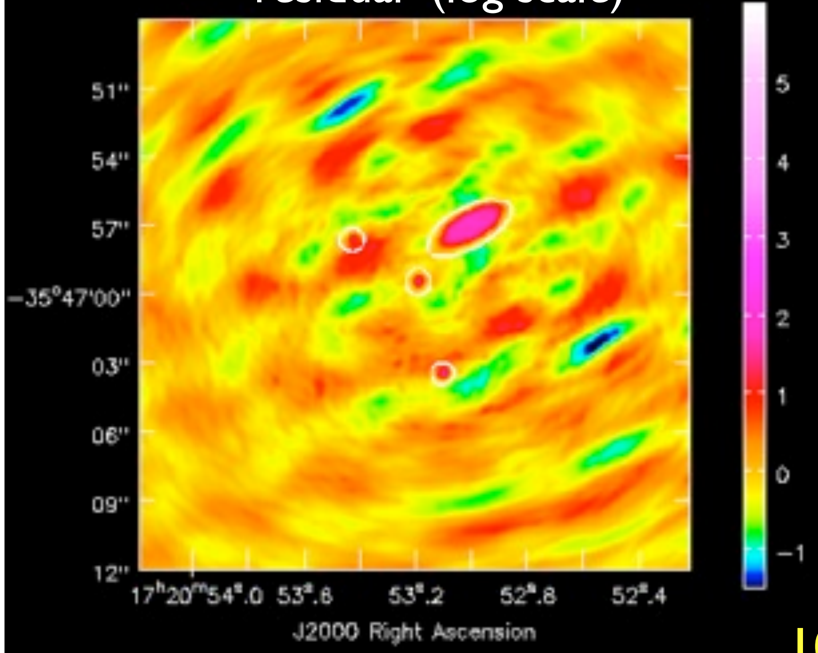
residual (linear scale)



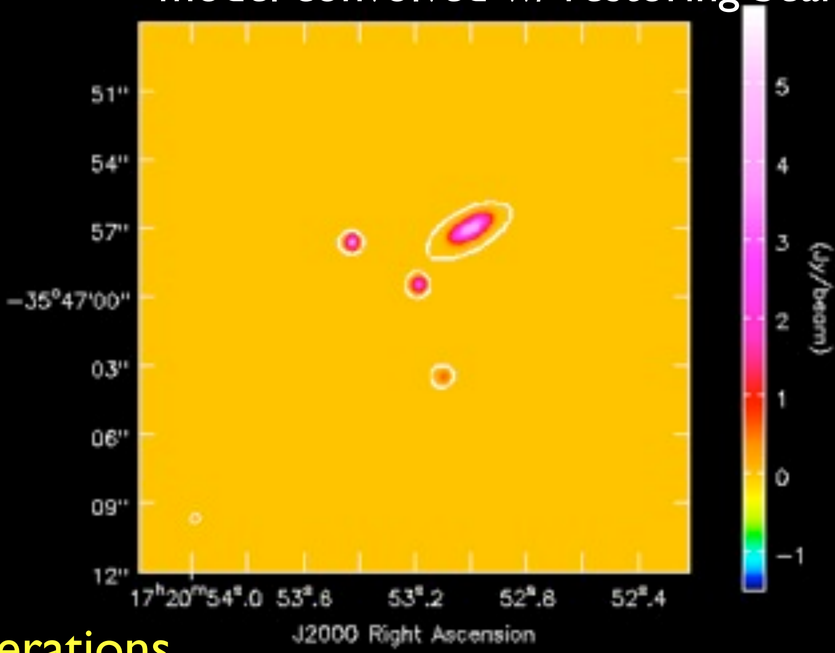
cleaned image (log scale)



residual (log scale)

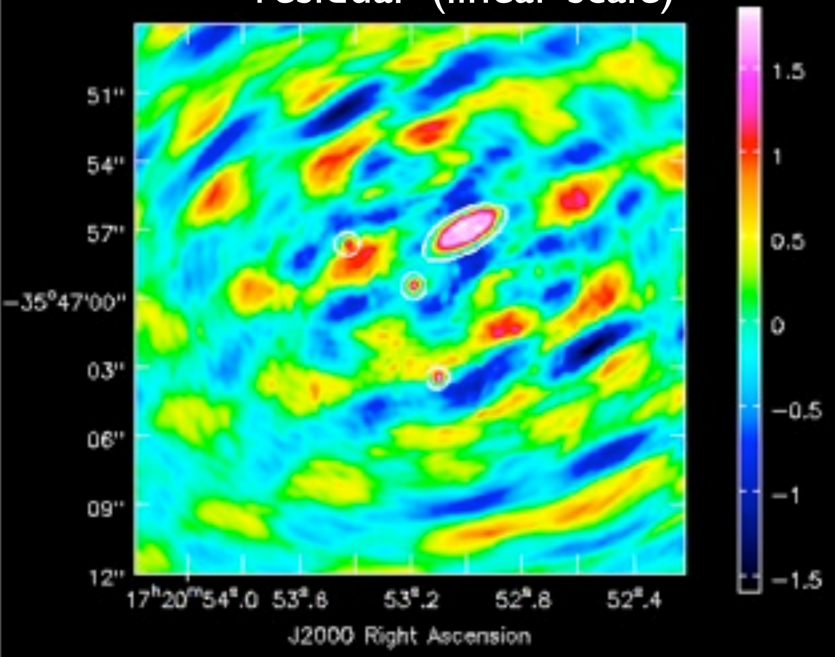


model convolved w/ restoring beam

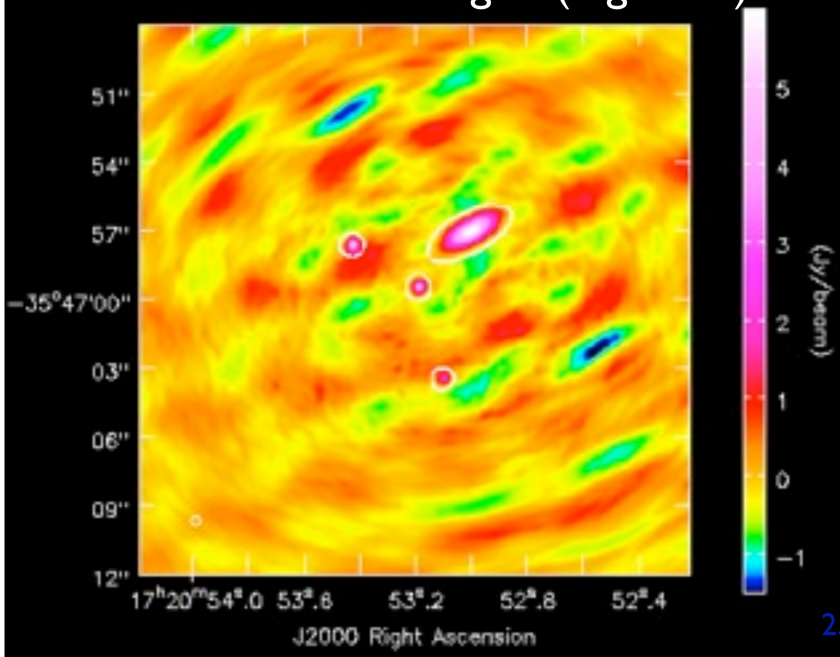


100 iterations

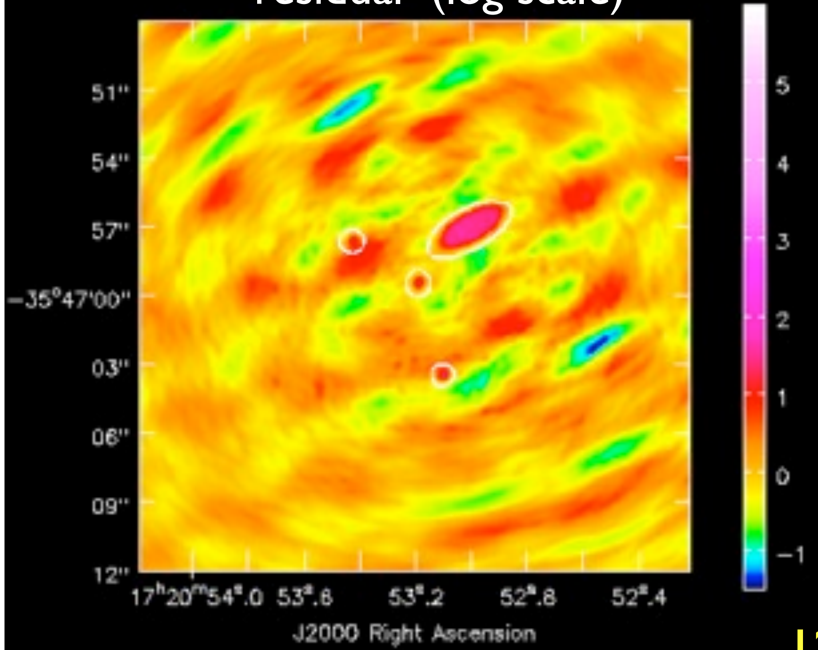
residual (linear scale)



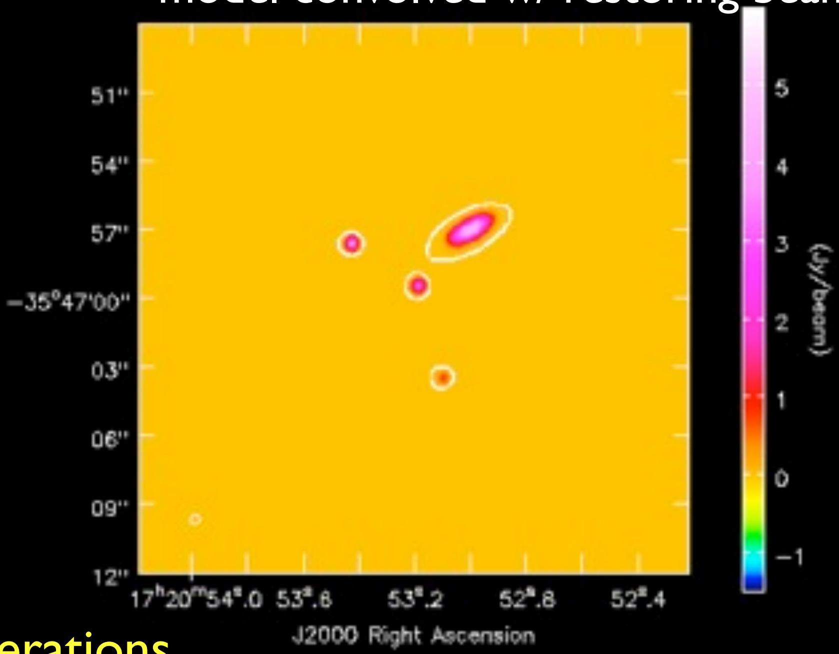
cleaned image (log scale)



residual (log scale)

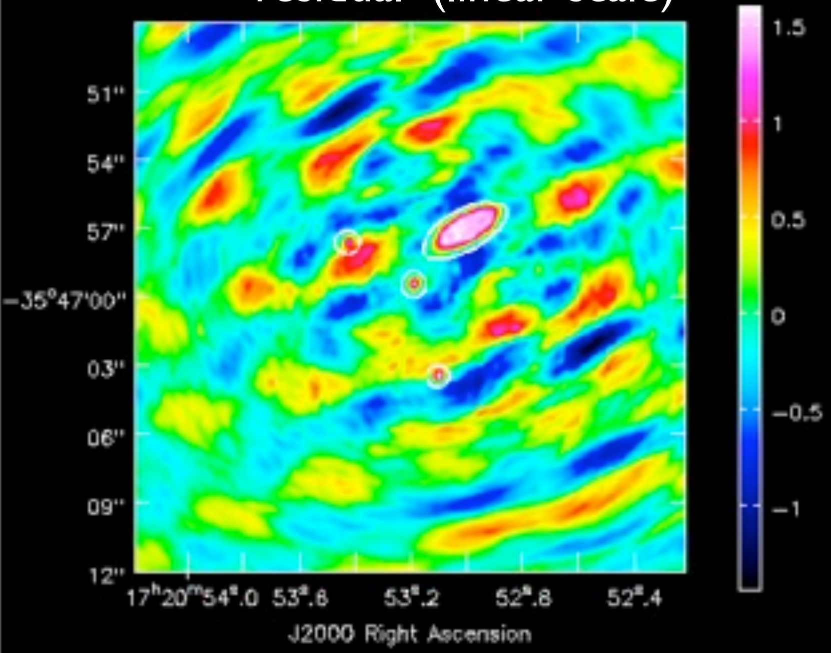


model convolved w/ restoring beam

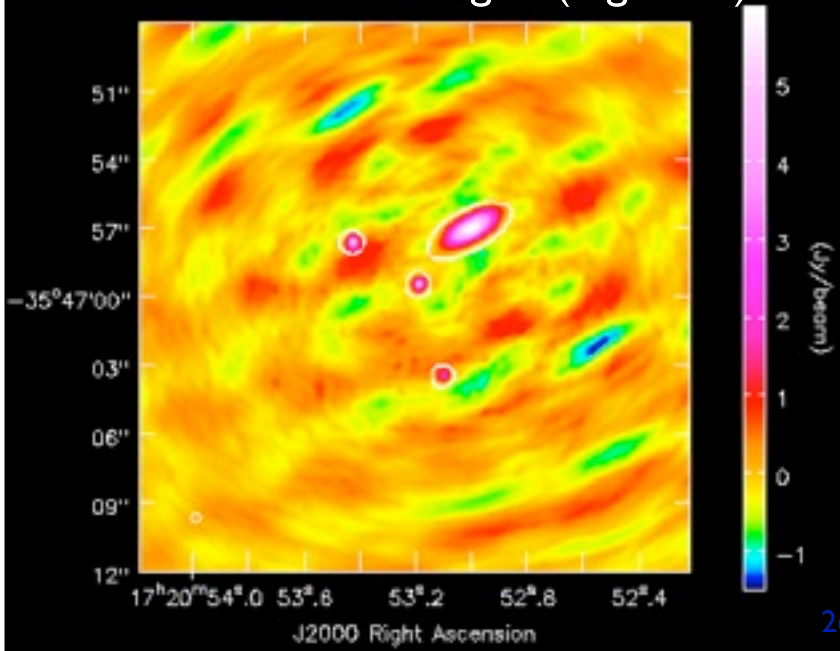


125 iterations

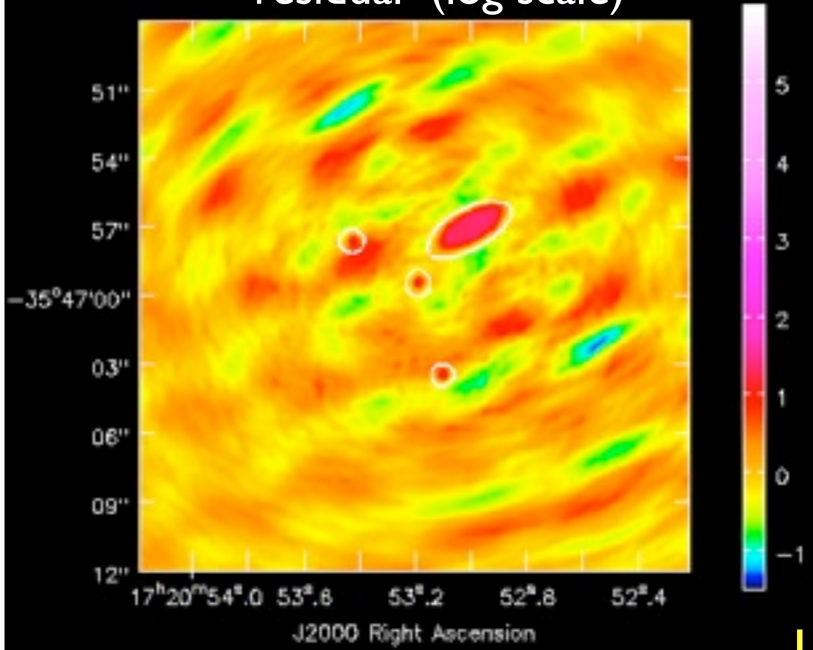
residual (linear scale)



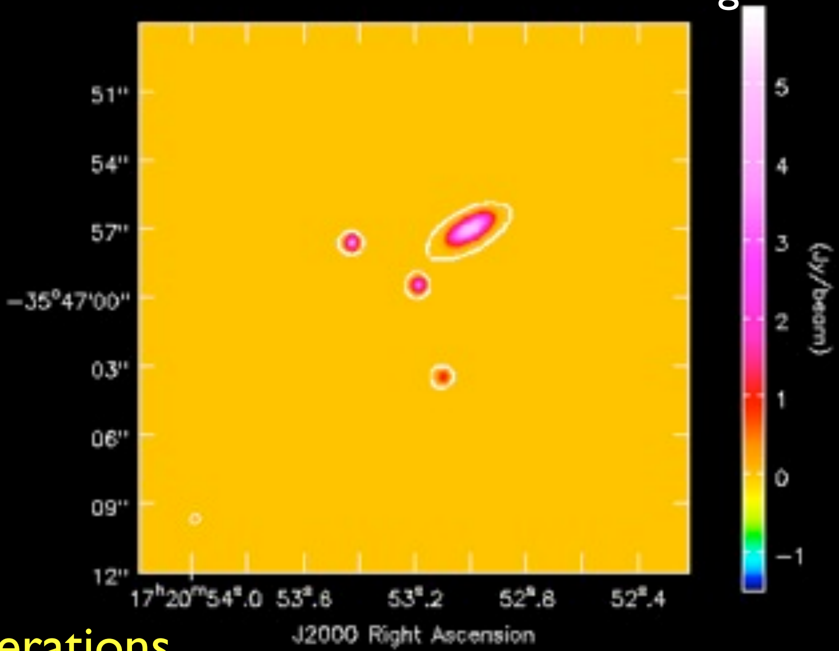
cleaned image (log scale)



residual (log scale)

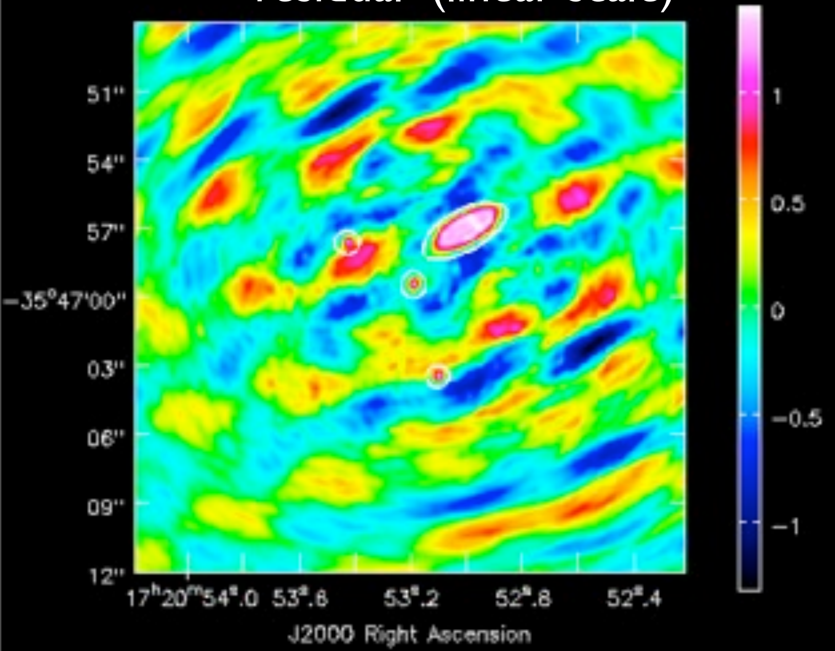


model convolved w/ restoring beam

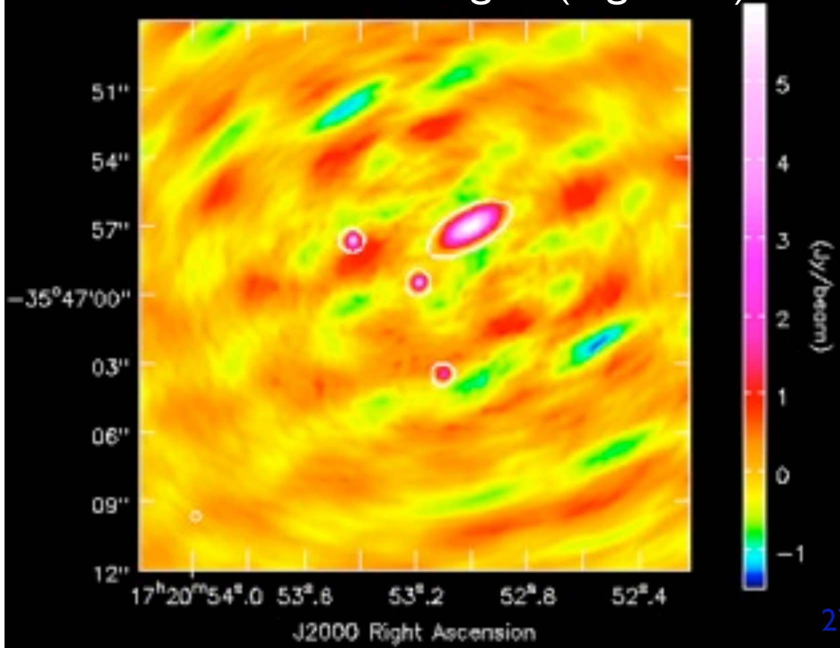


150 iterations

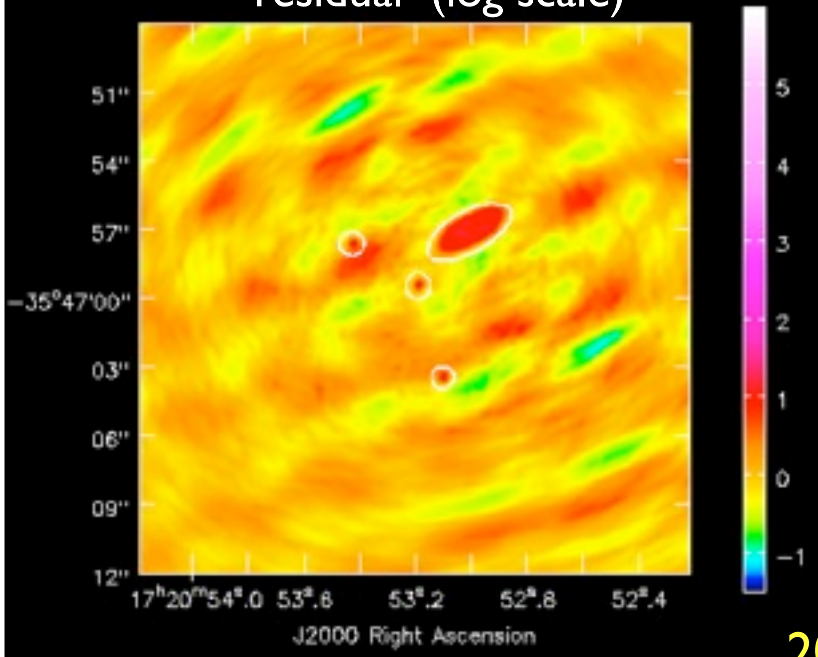
residual (linear scale)



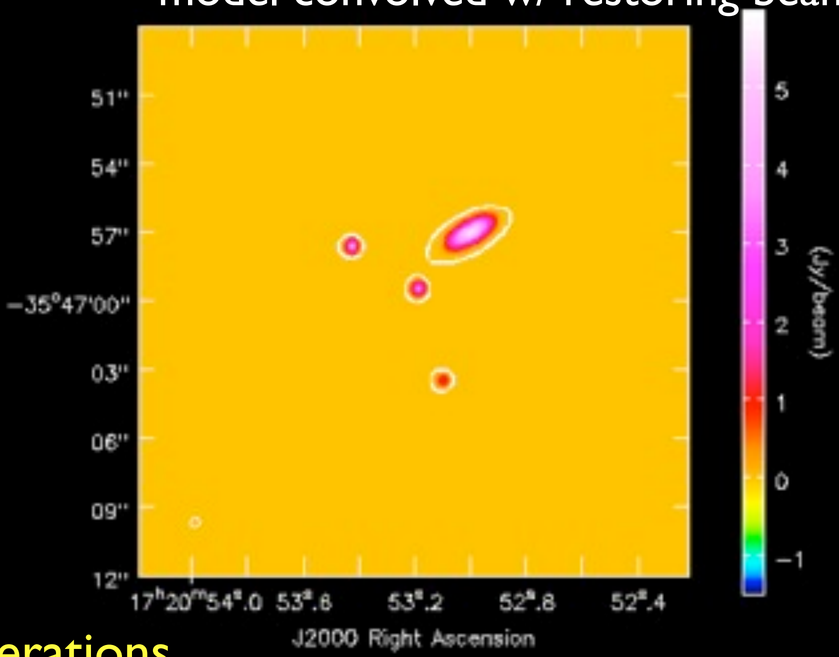
cleaned image (log scale)



residual (log scale)

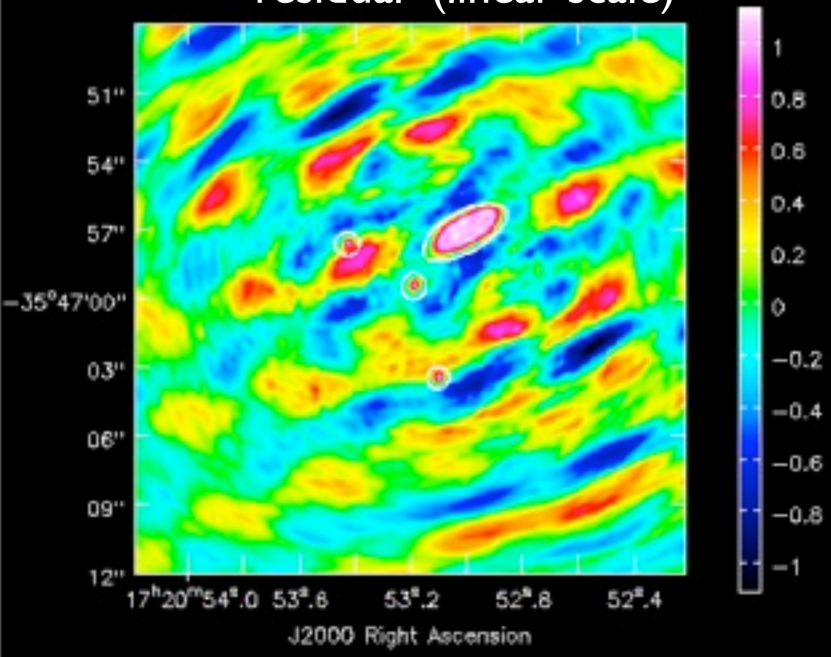


model convolved w/ restoring beam

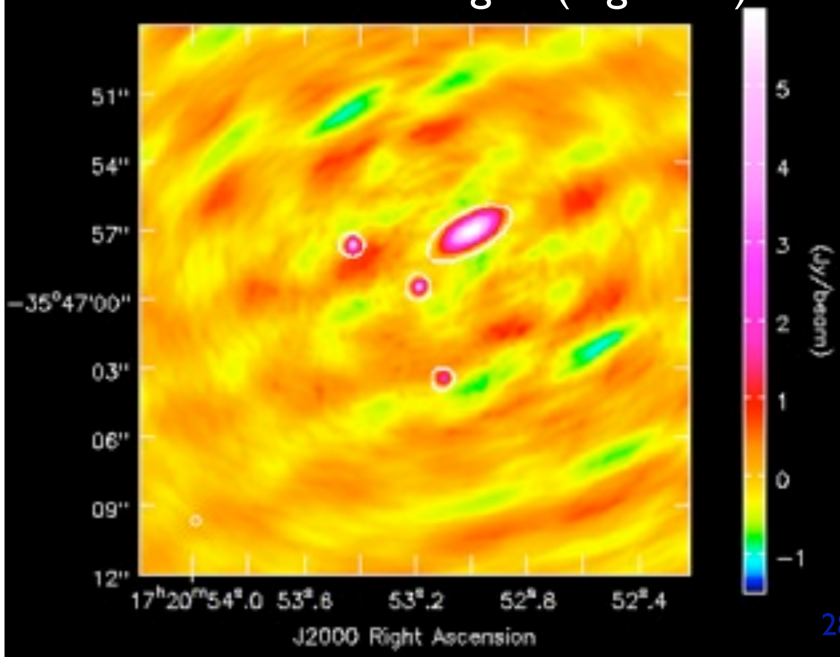


200 iterations

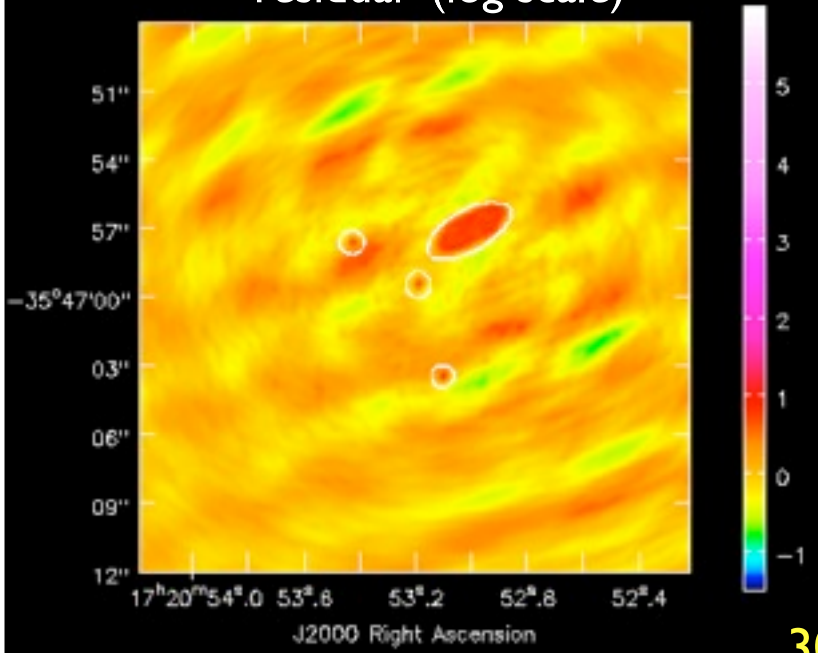
residual (linear scale)



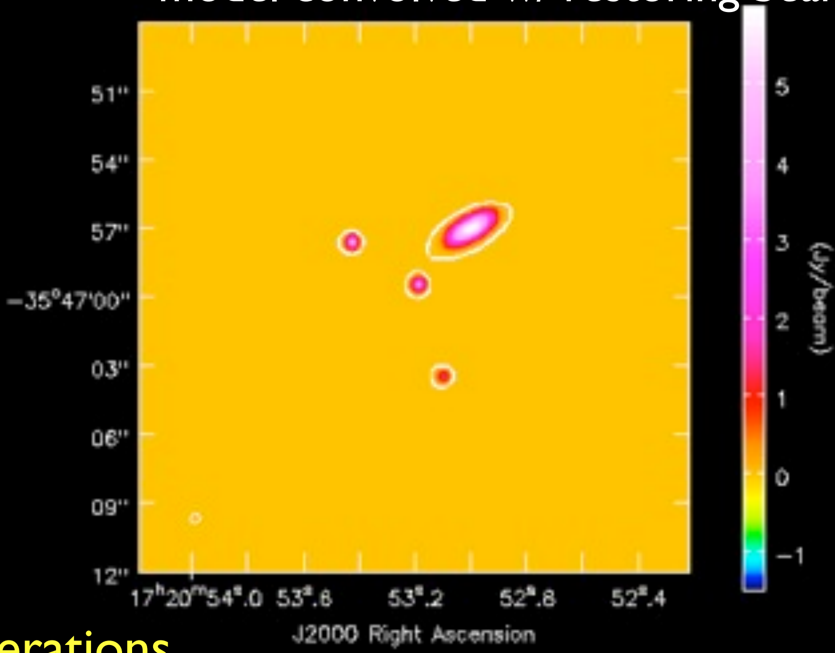
cleaned image (log scale)



residual (log scale)

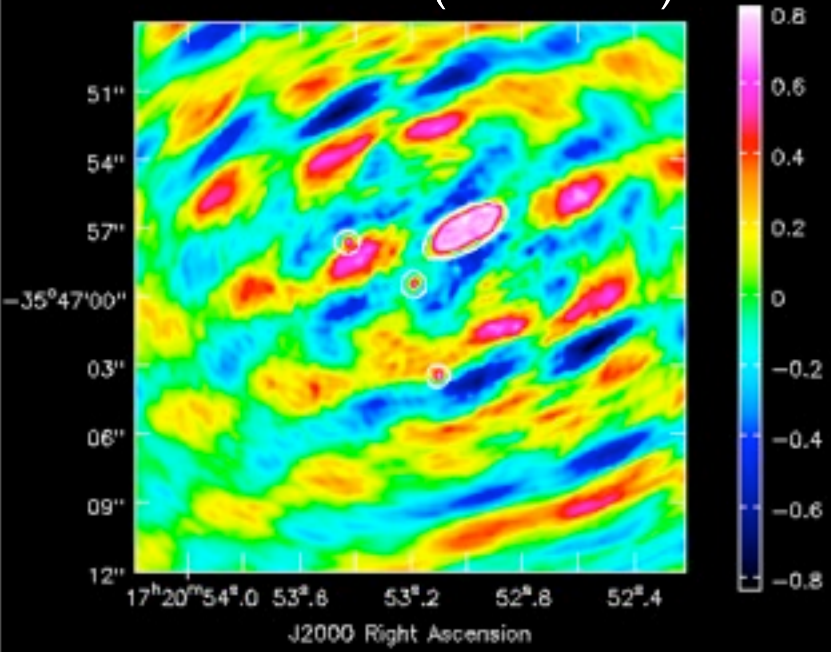


model convolved w/ restoring beam

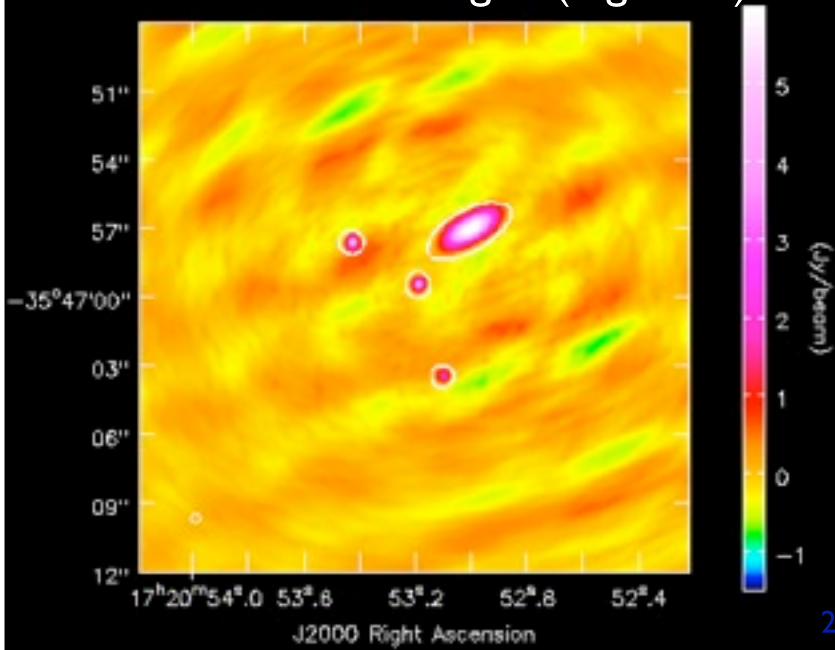


300 iterations

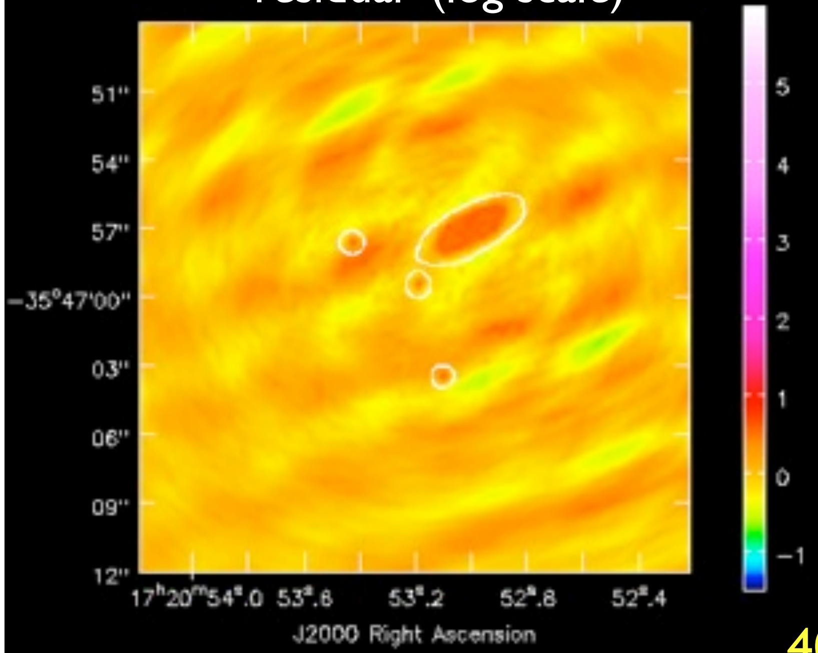
residual (linear scale)



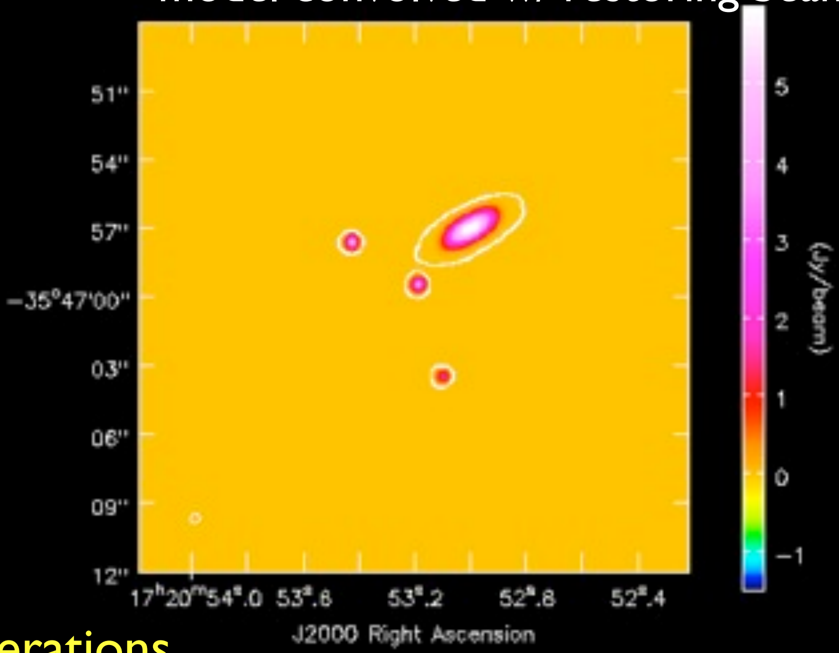
cleaned image (log scale)



residual (log scale)

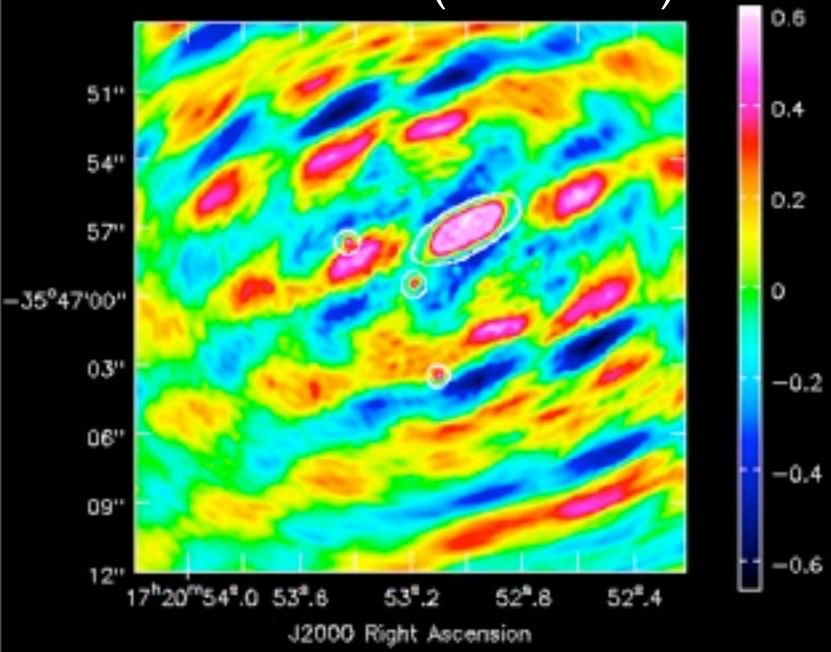


model convolved w/ restoring beam

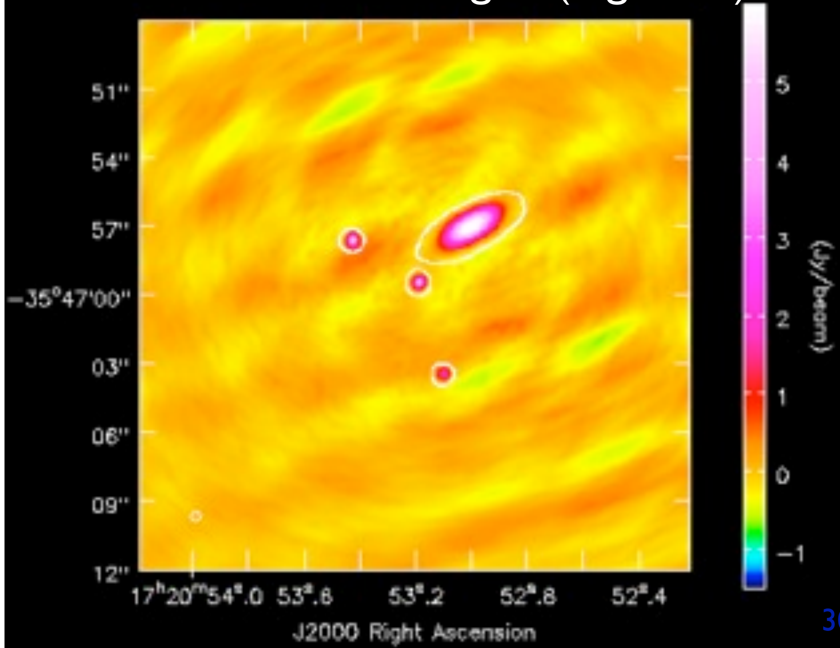


400 iterations

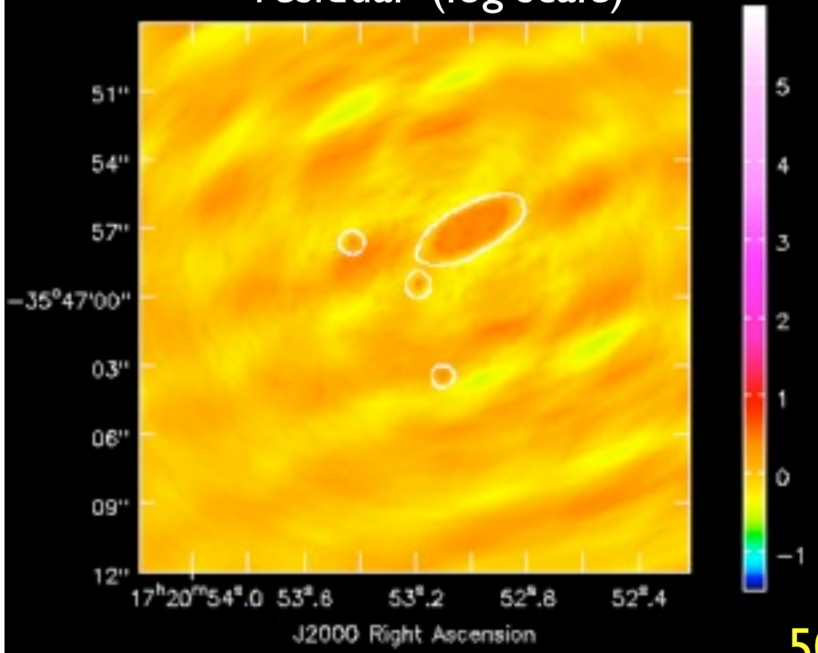
residual (linear scale)



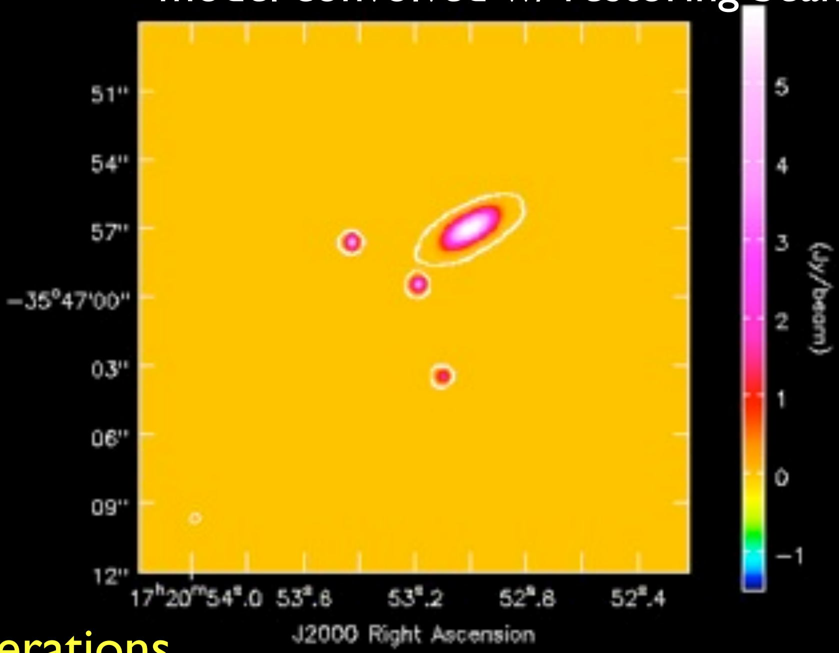
cleaned image (log scale)



residual (log scale)

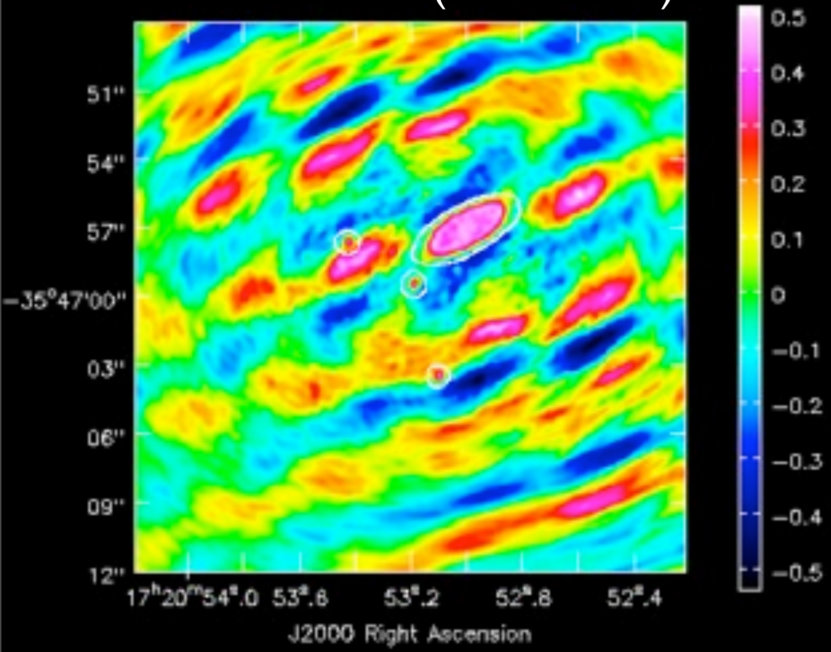


model convolved w/ restoring beam

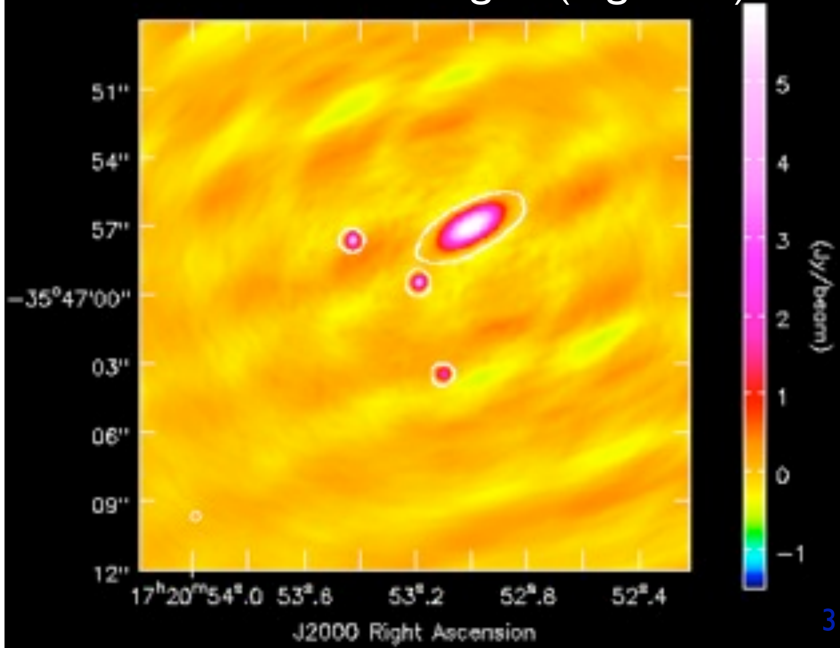


500 iterations

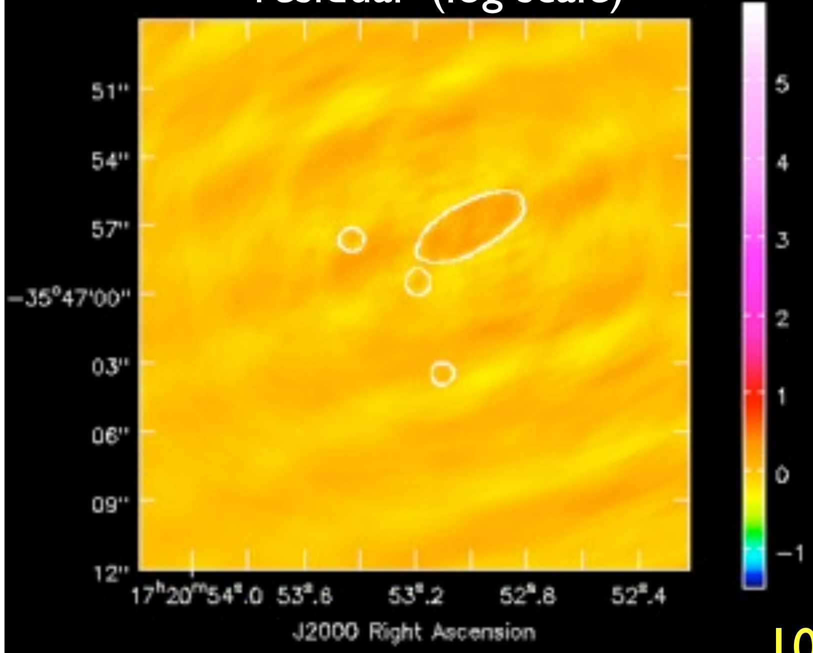
residual (linear scale)



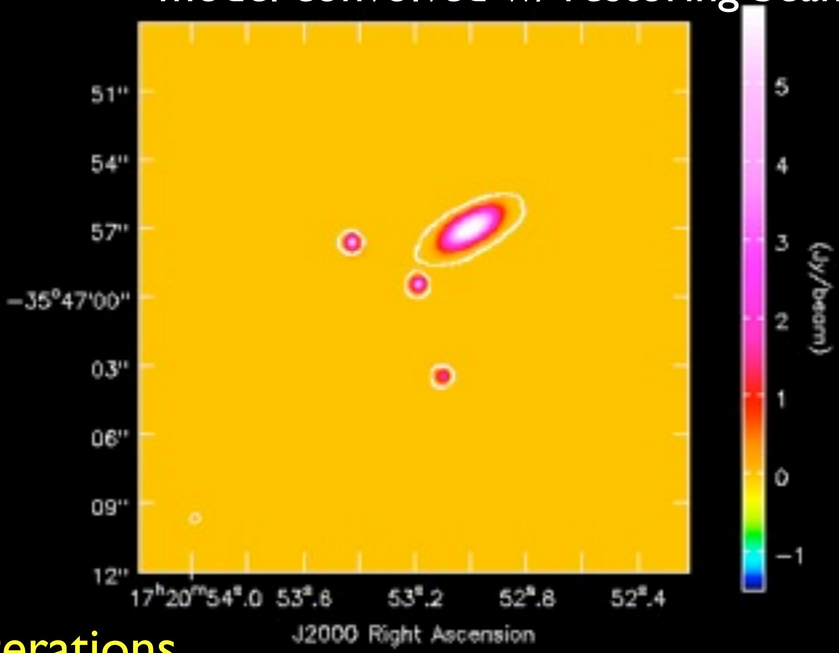
cleaned image (log scale)



residual (log scale)

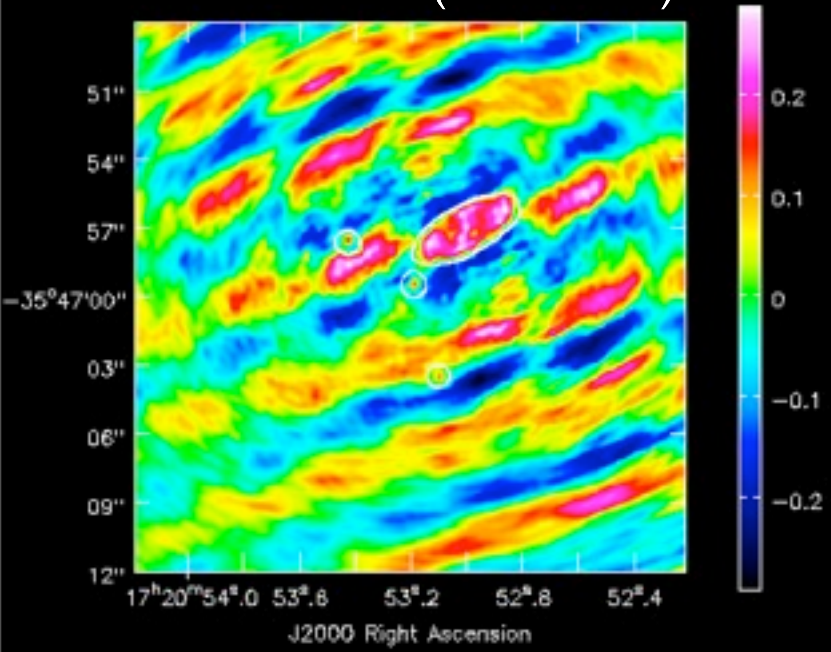


model convolved w/ restoring beam

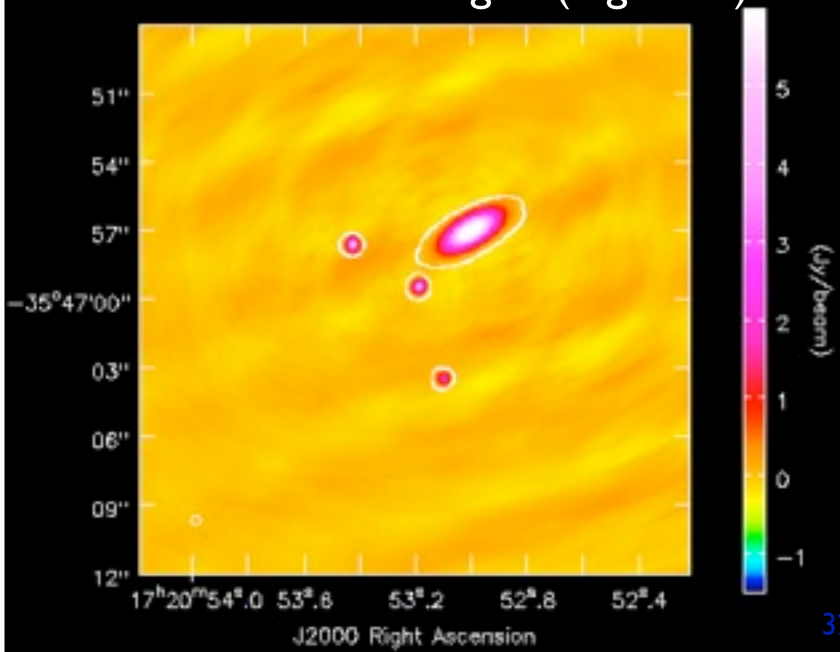


1000 iterations

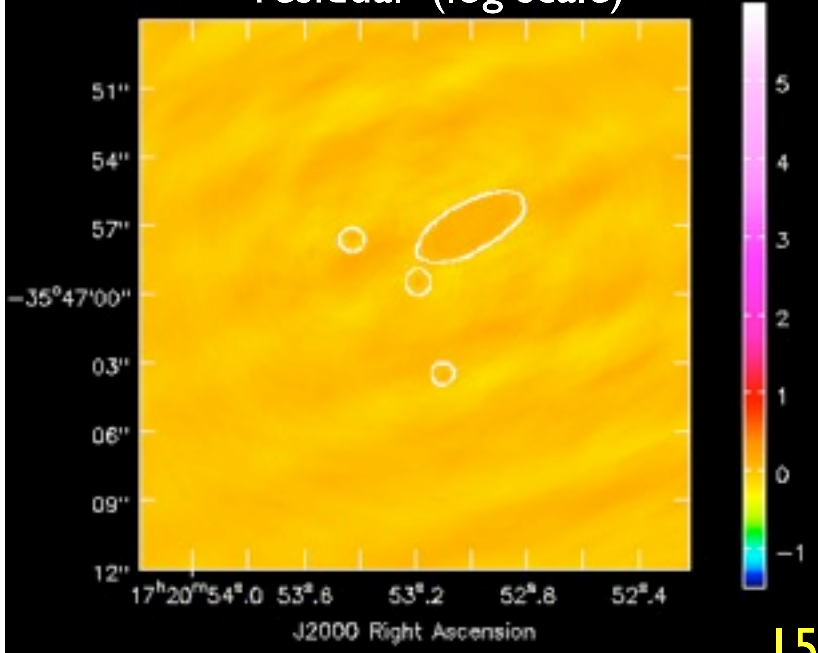
residual (linear scale)



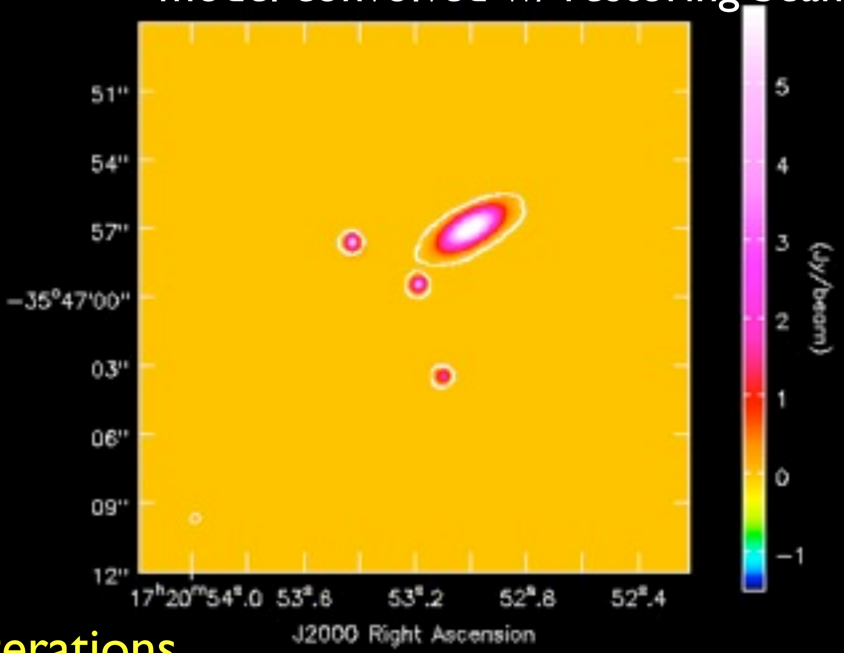
cleaned image (log scale)



residual (log scale)

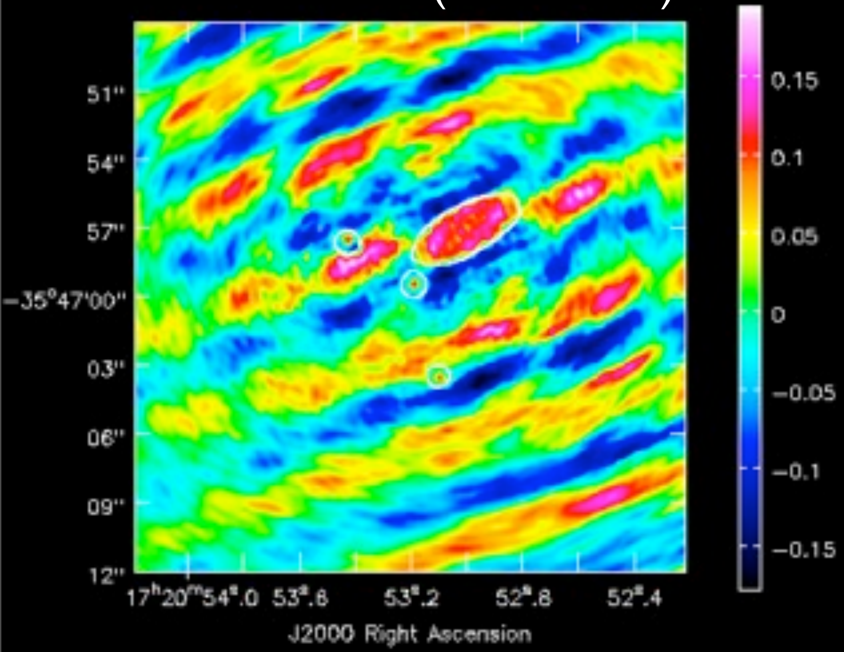


model convolved w/ restoring beam

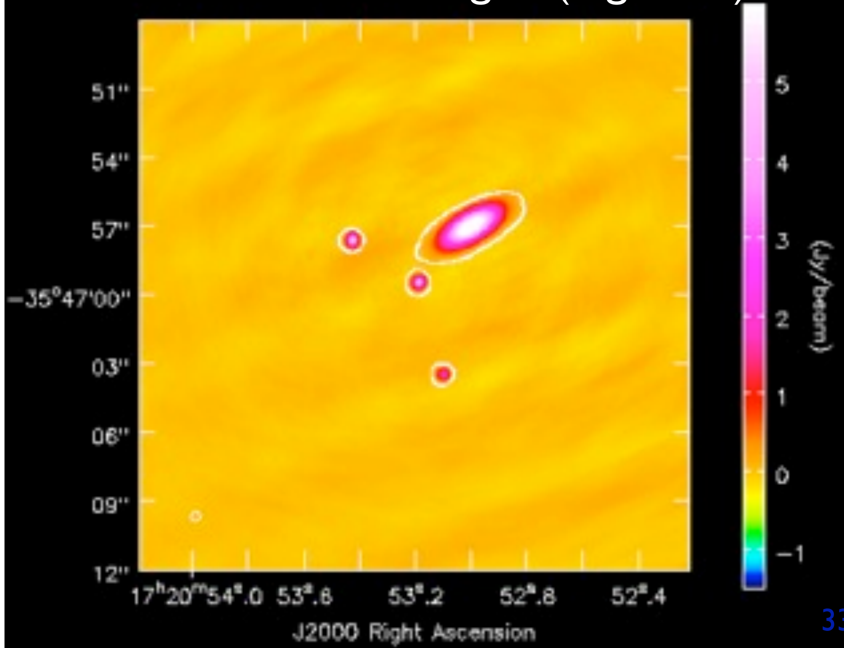


1500 iterations

residual (linear scale)



cleaned image (log scale)

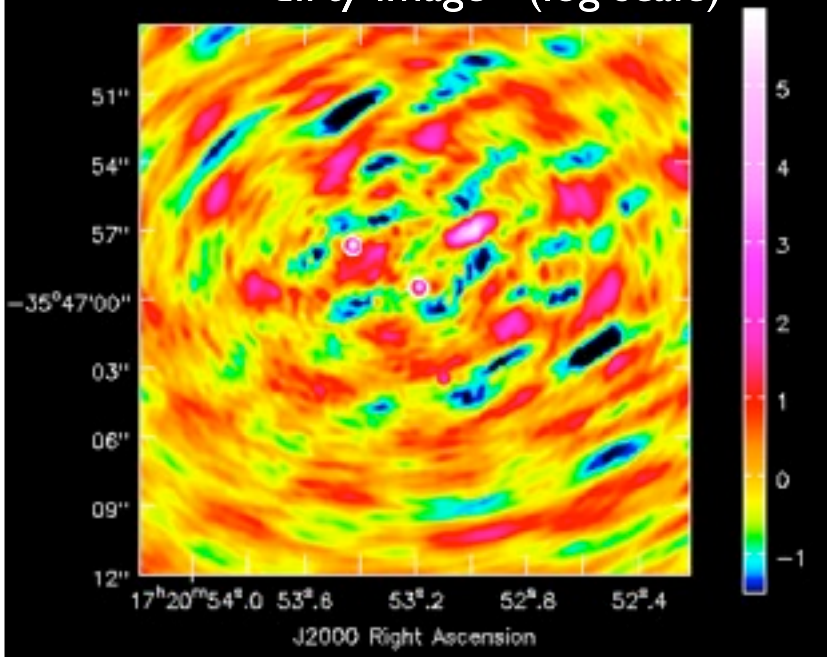


Deconvolution

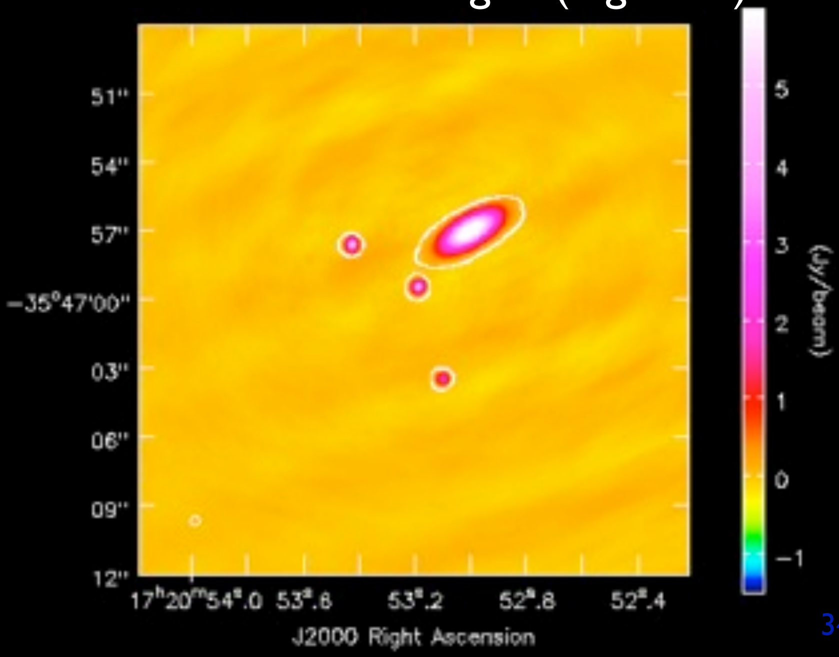
Results depend on:

- image parameters: size, cell, weighting, gridding, mosaic
- deconvolution parameters: algorithm, iterations, boxing, stopping criteria

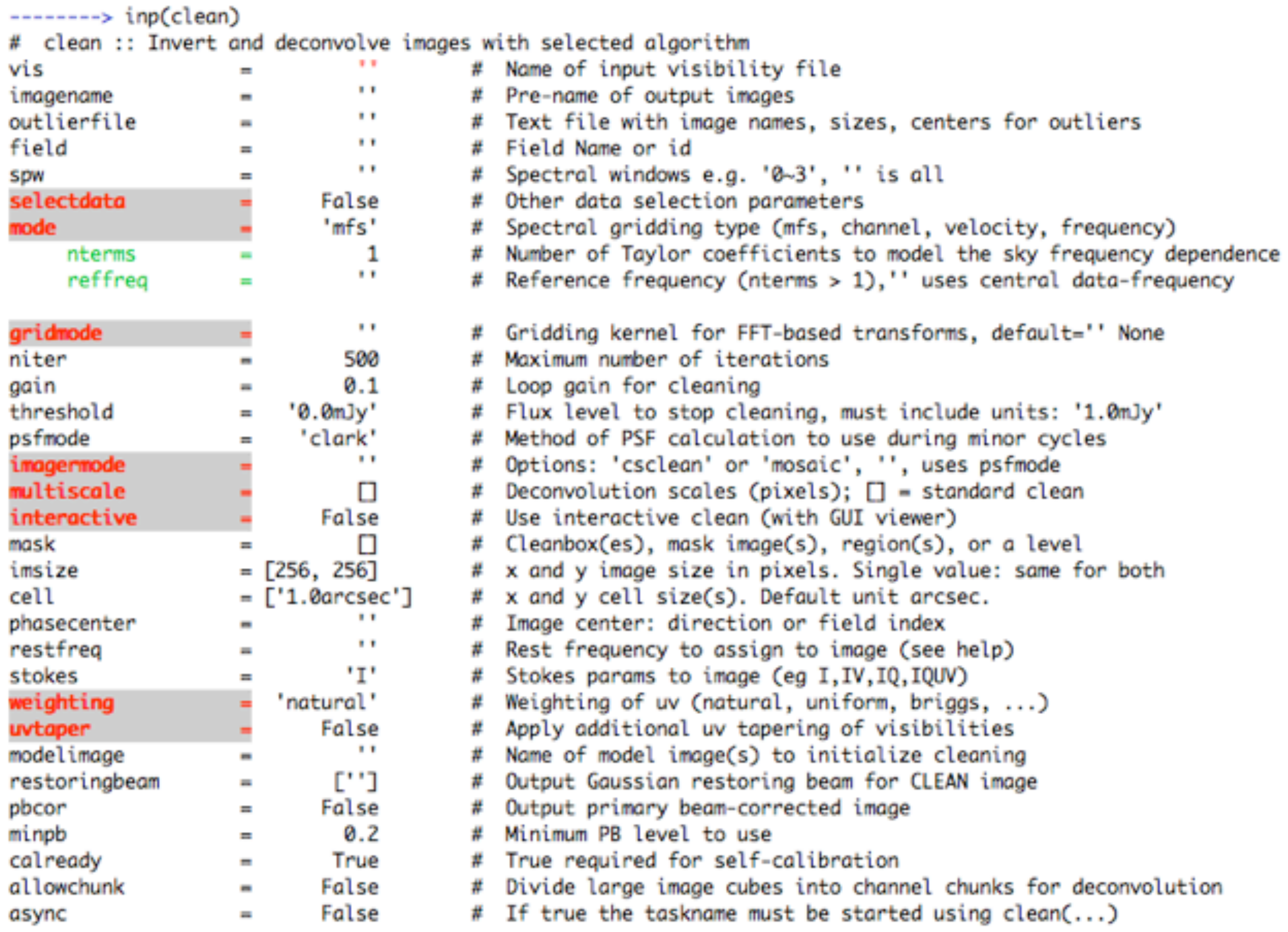
dirty image (log scale)



cleaned image (log scale)



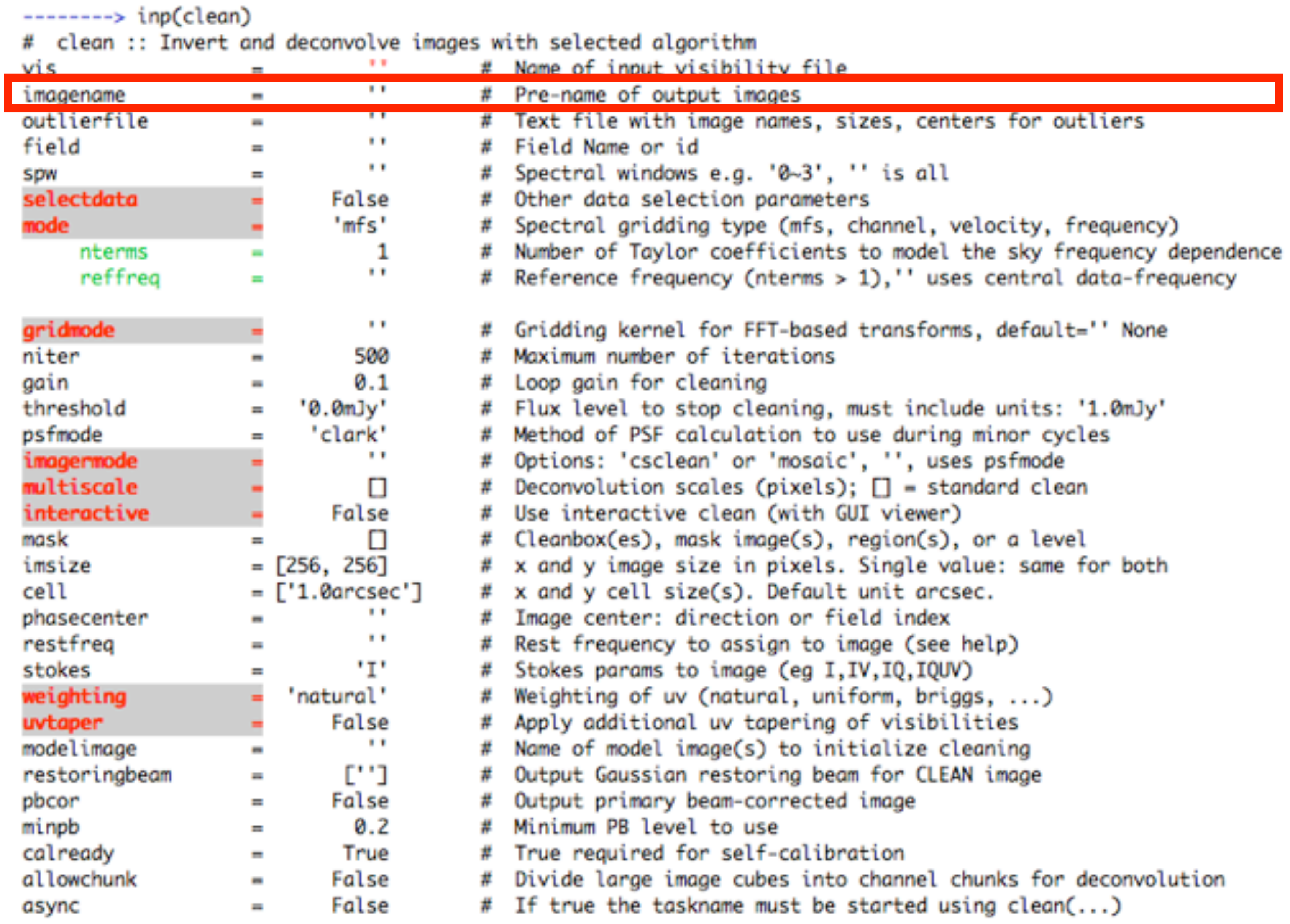
CASA: inputs to “clean”



```
--------> inp(clean)
#  clean :: Invert and deconvolve images with selected algorithm
vis                 =        ''        #  Name of input visibility file
imagename           =        ''        #  Pre-name of output images
outlierfile         =        ''        #  Text file with image names, sizes, centers for outliers
field               =        ''        #  Field Name or id
spw                 =        ''        #  Spectral windows e.g. '0~3', '' is all
selectdata          =     False        #  Other data selection parameters
mode                =     'mfs'        #  Spectral gridding type (mfs, channel, velocity, frequency)
     nterms         =         1        #  Number of Taylor coefficients to model the sky frequency dependence
     reffreq        =        ''        #  Reference frequency (nterms > 1),'' uses central data-frequency

gridmode            =        ''        #  Gridding kernel for FFT-based transforms, default='' None
niter               =       500        #  Maximum number of iterations
gain                =       0.1        #  Loop gain for cleaning
threshold           =  '0.0mJy'        #  Flux level to stop cleaning, must include units: '1.0mJy'
psfmode             =   'clark'        #  Method of PSF calculation to use during minor cycles
imagermode          =        ''        #  Options: 'csclean' or 'mosaic', '', uses psfmode
multiscale          =        []        #  Deconvolution scales (pixels); [] = standard clean
interactive         =     False        #  Use interactive clean (with GUI viewer)
mask                =        []        #  Cleanbox(es), mask image(s), region(s), or a level
imsize              = [256, 256]       #  x and y image size in pixels. Single value: same for both
cell                = ['1.0arcsec']    #  x and y cell size(s). Default unit arcsec.
phasecenter         =        ''        #  Image center: direction or field index
restfreq            =        ''        #  Rest frequency to assign to image (see help)
stokes              =       'I'        #  Stokes params to image (eg I,IV,IQ,IQUV)
weighting           = 'natural'        #  Weighting of uv (natural, uniform, briggs, ...)
uvtaper             =     False        #  Apply additional uv tapering of visibilities
modelimage          =        ''        #  Name of model image(s) to initialize cleaning
restoringbeam       =      ['']        #  Output Gaussian restoring beam for CLEAN image
pbcor               =     False        #  Output primary beam-corrected image
minpb               =       0.2        #  Minimum PB level to use
calready            =      True        #  True required for self-calibration
allowchunk          =     False        #  Divide large image cubes into channel chunks for deconvolution
async               =     False        #  If true the taskname must be started using clean(...)
```

imagename



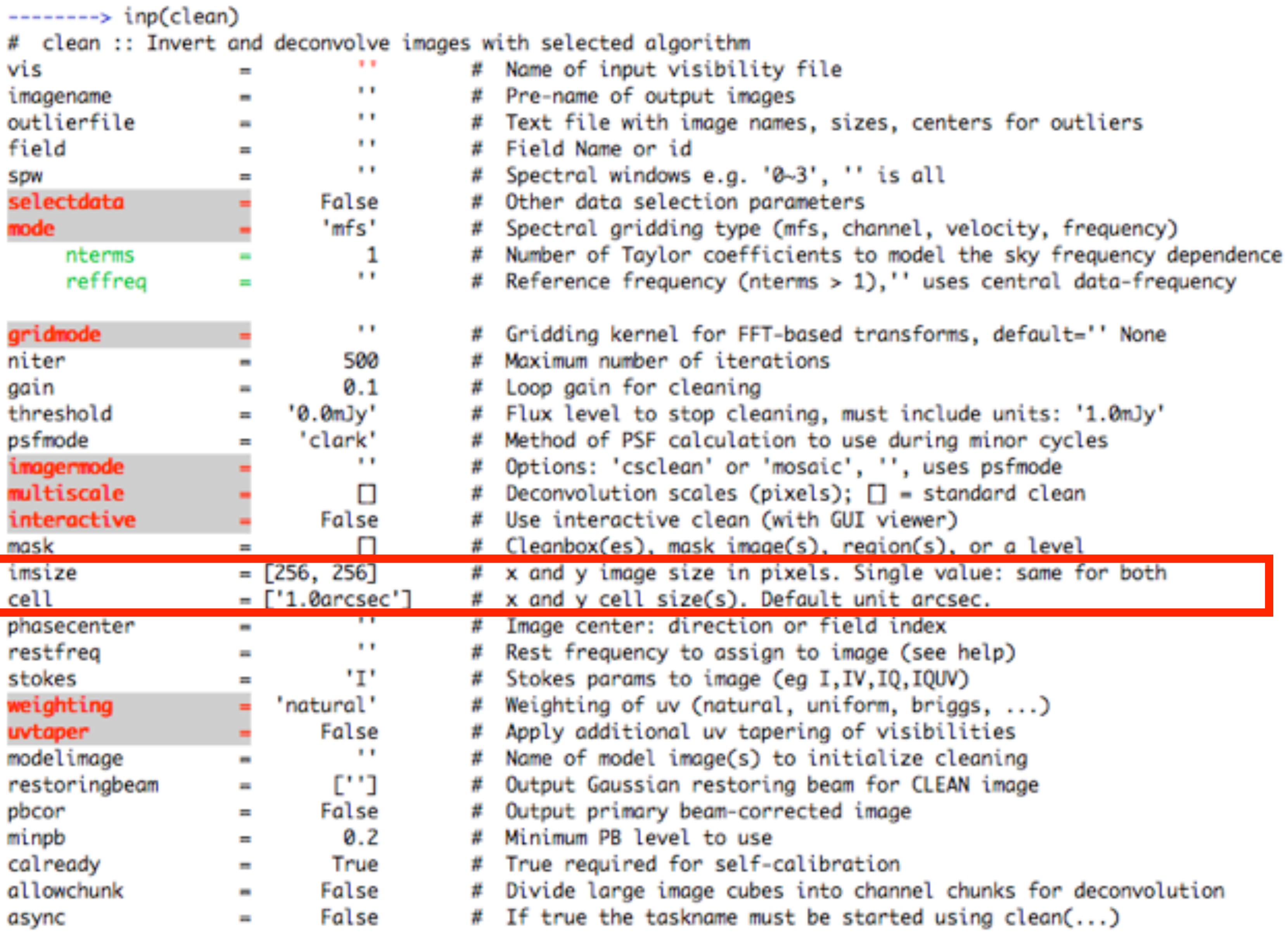
```
--------> inp(clean)
#  clean :: Invert and deconvolve images with selected algorithm
vis                 =         ''        #  Name of input visibility file
imagename           =         ''        #  Pre-name of output images
outlierfile         =         ''        #  Text file with image names, sizes, centers for outliers
field               =         ''        #  Field Name or id
spw                 =         ''        #  Spectral windows e.g. '0~3', '' is all
selectdata          =      False        #  Other data selection parameters
mode                =      'mfs'        #  Spectral gridding type (mfs, channel, velocity, frequency)
     nterms         =          1        #  Number of Taylor coefficients to model the sky frequency dependence
     reffreq        =         ''        #  Reference frequency (nterms > 1),'' uses central data-frequency

gridmode            =         ''        #  Gridding kernel for FFT-based transforms, default='' None
niter               =        500        #  Maximum number of iterations
gain                =        0.1        #  Loop gain for cleaning
threshold           =   '0.0mJy'        #  Flux level to stop cleaning, must include units: '1.0mJy'
psfmode             =    'clark'        #  Method of PSF calculation to use during minor cycles
imagermode          =         ''        #  Options: 'csclean' or 'mosaic', '', uses psfmode
multiscale          =         []        #  Deconvolution scales (pixels); [] = standard clean
interactive         =      False        #  Use interactive clean (with GUI viewer)
mask                =         []        #  Cleanbox(es), mask image(s), region(s), or a level
imsize              = [256, 256]        #  x and y image size in pixels. Single value: same for both
cell                = ['1.0arcsec']     #  x and y cell size(s). Default unit arcsec.
phasecenter         =         ''        #  Image center: direction or field index
restfreq            =         ''        #  Rest frequency to assign to image (see help)
stokes              =        'I'        #  Stokes params to image (eg I,IV,IQ,IQUV)
weighting           =  'natural'        #  Weighting of uv (natural, uniform, briggs, ...)
uvtaper             =      False        #  Apply additional uv tapering of visibilities
modelimage          =         ''        #  Name of model image(s) to initialize cleaning
restoringbeam       =       ['']        #  Output Gaussian restoring beam for CLEAN image
pbcor               =      False        #  Output primary beam-corrected image
minpb               =        0.2        #  Minimum PB level to use
calready            =       True        #  True required for self-calibration
allowchunk          =      False        #  Divide large image cubes into channel chunks for deconvolution
async               =      False        #  If true the taskname must be started using clean(...)
```

imagename

- <*imagename*>.image
  - final cleaned image (or dirty image if niter=0)
- <*imagename*>.psf
  - point spread function, or "dirty beam"
- <*imagename*>.model
  - image of the clean components
- <*imagename*>.residual
  - residual image after subtracting clean components
  - check to see if more cleaning is needed
- <*imagename*>.flux
  - relative sky sensitivity over field
  - pbcor=True divides the .image by the .flux
  - higher noise at edge of image

imsize and cell



```
--------> inp(clean)
#  clean :: Invert and deconvolve images with selected algorithm
vis                 =         ''        #  Name of input visibility file
imagename           =         ''        #  Pre-name of output images
outlierfile         =         ''        #  Text file with image names, sizes, centers for outliers
field               =         ''        #  Field Name or id
spw                 =         ''        #  Spectral windows e.g. '0~3', '' is all
selectdata          =      False        #  Other data selection parameters
mode                =      'mfs'        #  Spectral gridding type (mfs, channel, velocity, frequency)
     nterms         =          1        #  Number of Taylor coefficients to model the sky frequency dependence
     reffreq        =         ''        #  Reference frequency (nterms > 1),'' uses central data-frequency

gridmode            =         ''        #  Gridding kernel for FFT-based transforms, default='' None
niter               =        500        #  Maximum number of iterations
gain                =        0.1        #  Loop gain for cleaning
threshold           =   '0.0mJy'        #  Flux level to stop cleaning, must include units: '1.0mJy'
psfmode             =    'clark'        #  Method of PSF calculation to use during minor cycles
imagermode          =         ''        #  Options: 'csclean' or 'mosaic', '', uses psfmode
multiscale          =         []        #  Deconvolution scales (pixels); [] = standard clean
interactive         =      False        #  Use interactive clean (with GUI viewer)
mask                =         []        #  Cleanbox(es), mask image(s), region(s), or a level
imsize              = [256, 256]        #  x and y image size in pixels. Single value: same for both
cell                = ['1.0arcsec']     #  x and y cell size(s). Default unit arcsec.
phasecenter         =         ''        #  Image center: direction or field index
restfreq            =         ''        #  Rest frequency to assign to image (see help)
stokes              =        'I'        #  Stokes params to image (eg I,IV,IQ,IQUV)
weighting           =  'natural'        #  Weighting of uv (natural, uniform, briggs, ...)
uvtaper             =      False        #  Apply additional uv tapering of visibilities
modelimage          =         ''        #  Name of model image(s) to initialize cleaning
restoringbeam       =       ['']        #  Output Gaussian restoring beam for CLEAN image
pbcor               =      False        #  Output primary beam-corrected image
minpb               =        0.2        #  Minimum PB level to use
calready            =       True        #  True required for self-calibration
allowchunk          =      False        #  Divide large image cubes into channel chunks for deconvolution
async               =      False        #  If true the taskname must be started using clean(...)
```

imsize and cell

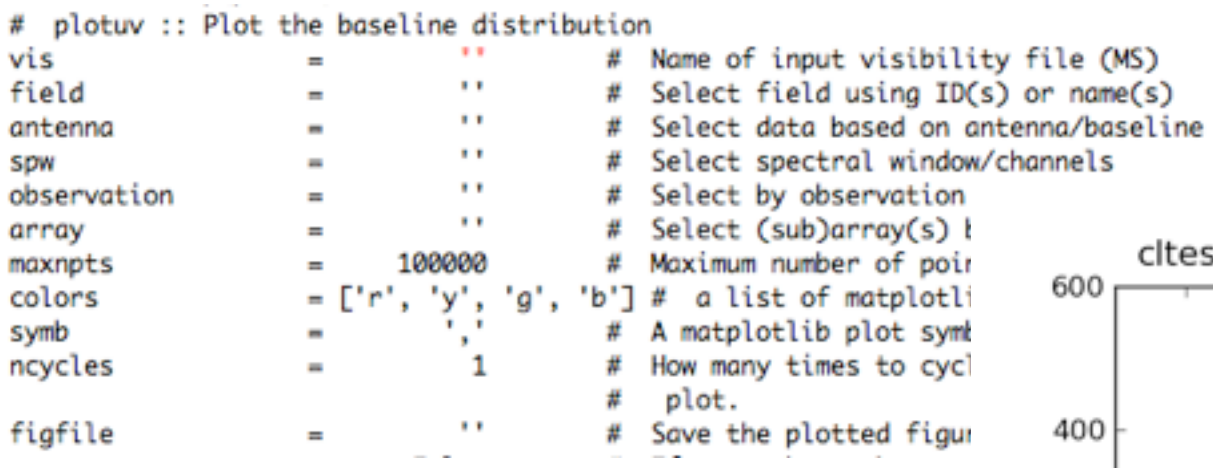
- cell
  - should satisfy sampling theorem for the longest baselines, $\Delta x < 1/(2\ u_{max}),\ \Delta y < 1/(2\ v_{max})$
  - in practice, 3 to 5 pixels across the main lobe of the dirty beam

- imsize
  - natural resolution in (u,v) plane samples FT{A(x,y)}, implies image size 2x primary beam
  - primary bean ~ $1.2 * \lambda/D$, D = telescope diameter
  - e.g., ALMA: 870 μm, 12 m telescope $\rightarrow$ 2 x 18 arcsec

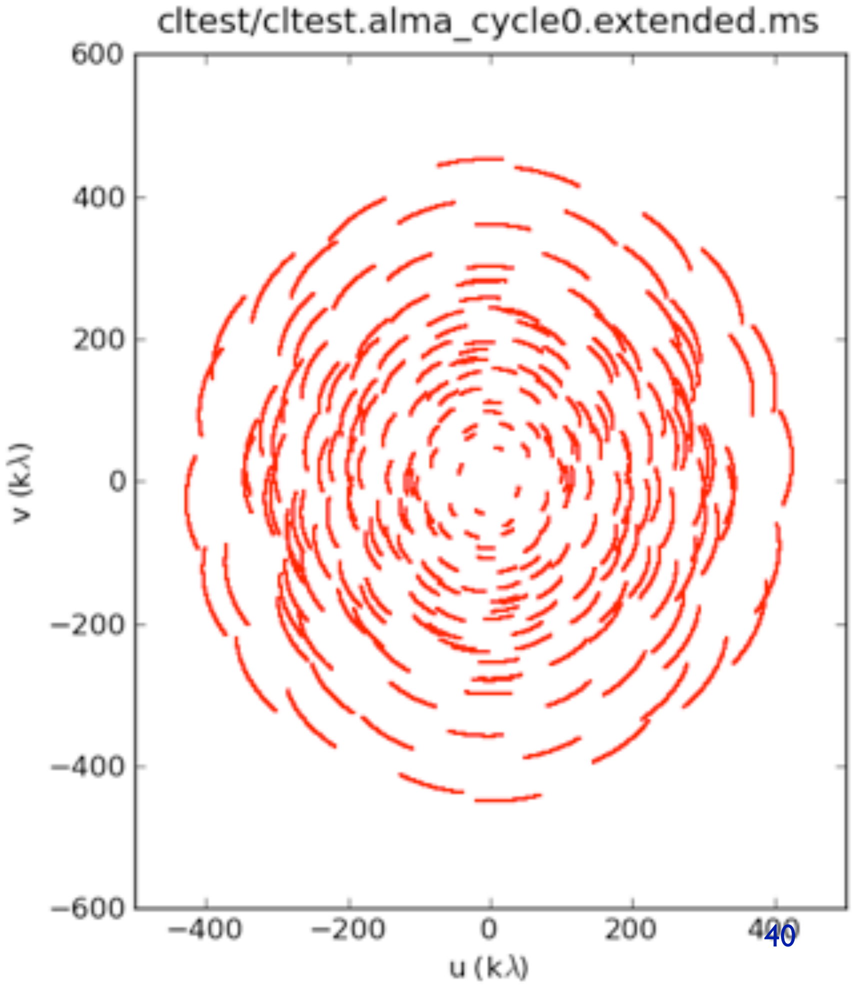
* not restricted to powers of 2 (in fact internal padding complicates)

* if there are bright sources in the sidelobes of A(x,y), then they will be aliased into the image (need to make a larger image)

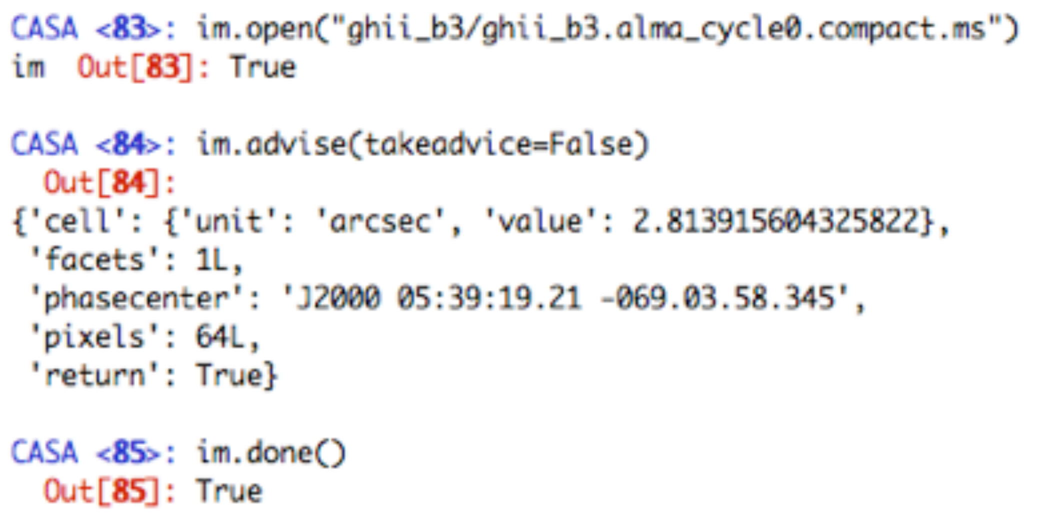
plotuv



```
#  plotuv :: Plot the baseline distribution
vis                 =          ''        #  Name of input visibility file (MS)
field               =          ''        #  Select field using ID(s) or name(s)
antenna             =          ''        #  Select data based on antenna/baseline
spw                 =          ''        #  Select spectral window/channels
observation         =          ''        #  Select by observation
array               =          ''        #  Select (sub)array(s) b
maxnpts             =      100000        #  Maximum number of poi
colors              = ['r', 'y', 'g', 'b'] #  a list of matplotli
symb                =         ','        #  A matplotlib plot symb
ncycles             =           1        #  How many times to cycl
                                         #   plot.
figfile             =          ''        #  Save the plotted figu
```



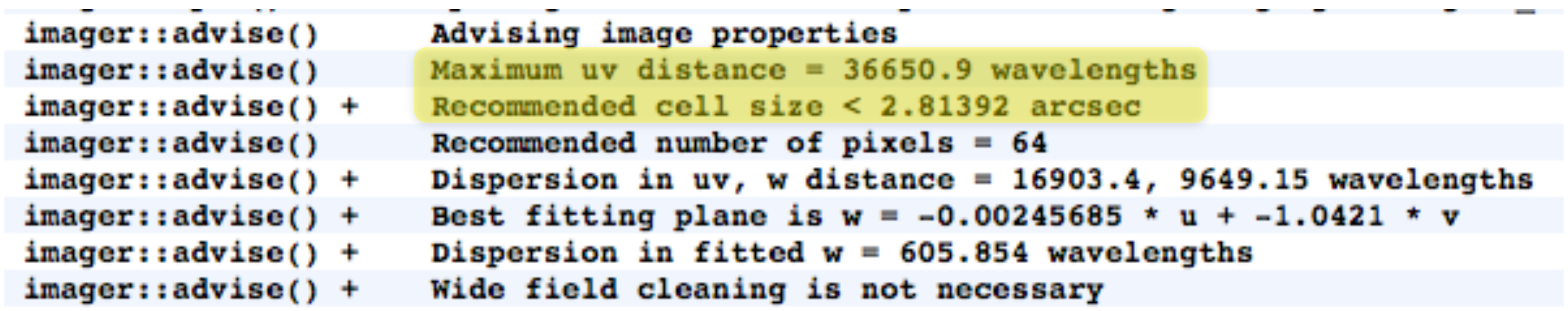
im.advise



```
CASA <83>: im.open("ghii_b3/ghii_b3.alma_cycle0.compact.ms")
im  Out[83]: True

CASA <84>: im.advise(takeadvice=False)
  Out[84]:
{'cell': {'unit': 'arcsec', 'value': 2.813915604325822},
 'facets': 1L,
 'phasecenter': 'J2000 05:39:19.21 -069.03.58.345',
 'pixels': 64L,
 'return': True}

CASA <85>: im.done()
  Out[85]: True
```

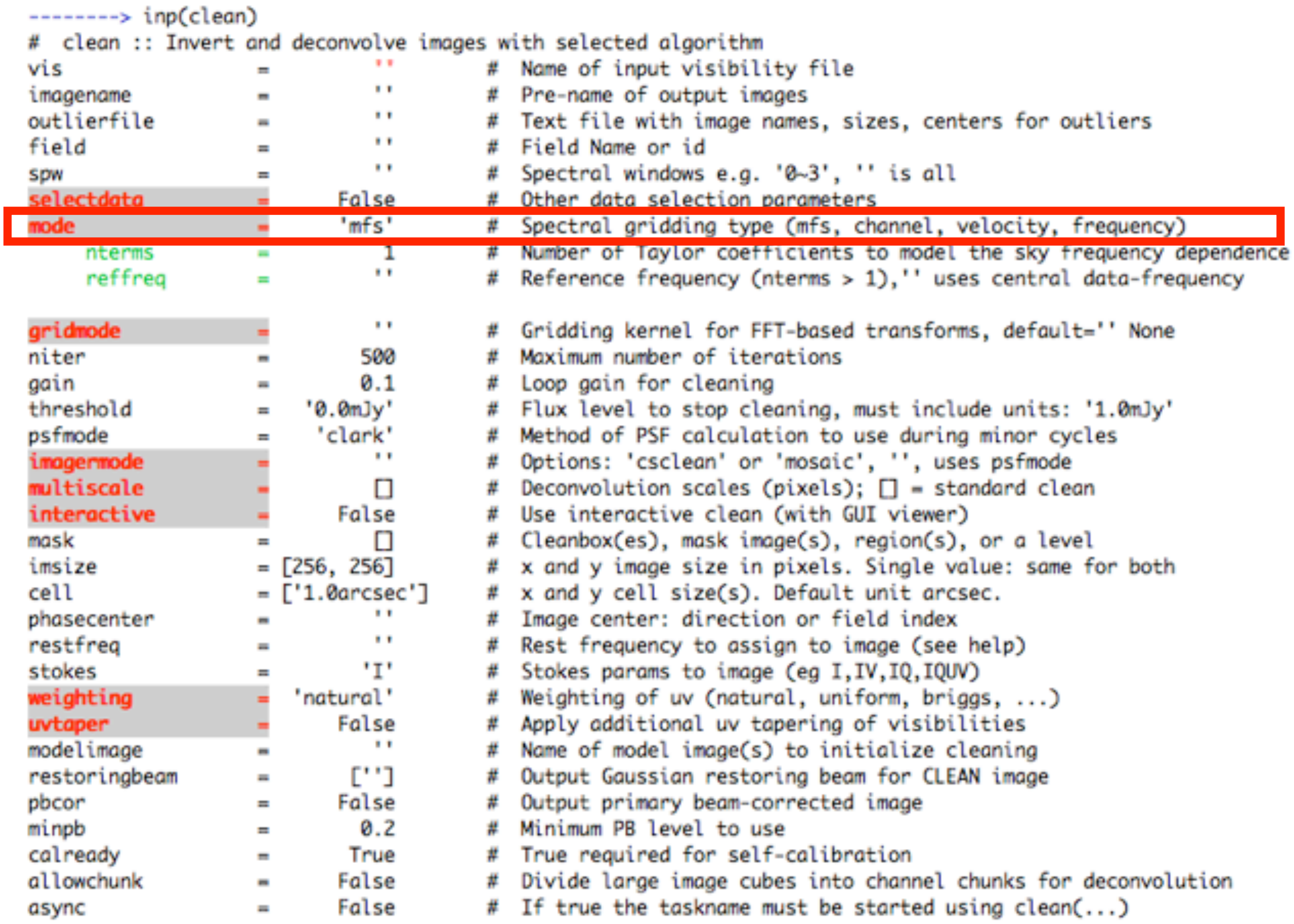


```
imager::advise()        Advising image properties
imager::advise()        Maximum uv distance = 36650.9 wavelengths
imager::advise() +      Recommended cell size < 2.81392 arcsec
imager::advise()        Recommended number of pixels = 64
imager::advise() +      Dispersion in uv, w distance = 16903.4, 9649.15 wavelengths
imager::advise() +      Best fitting plane is w = -0.00245685 * u + -1.0421 * v
imager::advise() +      Dispersion in fitted w = 605.854 wavelengths
imager::advise() +      Wide field cleaning is not necessary
```

0.5/maxuv – absolute minimum;

recommend < 0.2/maxuv = 1.1arcsec

mode



```
--------> inp(clean)
#  clean :: Invert and deconvolve images with selected algorithm
vis                 =         ''        #  Name of input visibility file
imagename           =         ''        #  Pre-name of output images
outlierfile         =         ''        #  Text file with image names, sizes, centers for outliers
field               =         ''        #  Field Name or id
spw                 =         ''        #  Spectral windows e.g. '0~3', '' is all
selectdata          =      False        #  Other data selection parameters
mode                =      'mfs'        #  Spectral gridding type (mfs, channel, velocity, frequency)
     nterms         =          1        #  Number of Taylor coefficients to model the sky frequency dependence
     reffreq        =         ''        #  Reference frequency (nterms > 1),'' uses central data-frequency

gridmode            =         ''        #  Gridding kernel for FFT-based transforms, default='' None
niter               =        500        #  Maximum number of iterations
gain                =        0.1        #  Loop gain for cleaning
threshold           =   '0.0mJy'        #  Flux level to stop cleaning, must include units: '1.0mJy'
psfmode             =    'clark'        #  Method of PSF calculation to use during minor cycles
imagermode          =         ''        #  Options: 'csclean' or 'mosaic', '', uses psfmode
multiscale          =         []        #  Deconvolution scales (pixels); [] = standard clean
interactive         =      False        #  Use interactive clean (with GUI viewer)
mask                =         []        #  Cleanbox(es), mask image(s), region(s), or a level
imsize              = [256, 256]        #  x and y image size in pixels. Single value: same for both
cell                = ['1.0arcsec']     #  x and y cell size(s). Default unit arcsec.
phasecenter         =         ''        #  Image center: direction or field index
restfreq            =         ''        #  Rest frequency to assign to image (see help)
stokes              =        'I'        #  Stokes params to image (eg I,IV,IQ,IQUV)
weighting           =  'natural'        #  Weighting of uv (natural, uniform, briggs, ...)
uvtaper             =      False        #  Apply additional uv tapering of visibilities
modelimage          =         ''        #  Name of model image(s) to initialize cleaning
restoringbeam       =       ['']        #  Output Gaussian restoring beam for CLEAN image
pbcor               =      False        #  Output primary beam-corrected image
minpb               =        0.2        #  Minimum PB level to use
calready            =       True        #  True required for self-calibration
allowchunk          =      False        #  Divide large image cubes into channel chunks for deconvolution
async               =      False        #  If true the taskname must be started using clean(...)
```

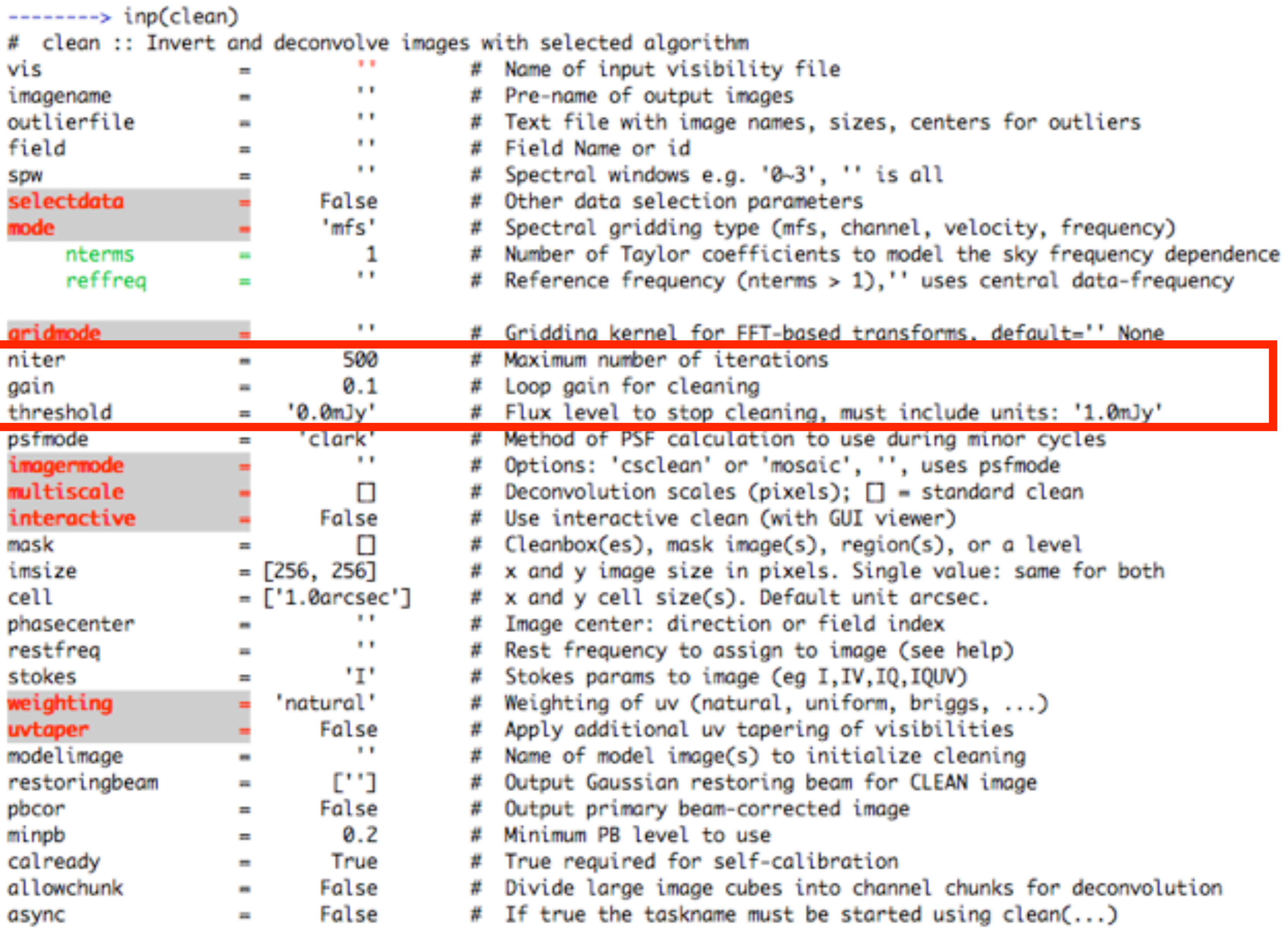
mode

- mode = **mfs** (multi-frequency synthesis)
  - sum each channel to make a continuum image
  - nterm: models the sky-frequency dependence of the continuum

- mode = **channel**, **velocity**, or **frequency**
  - data: taken in sky frequency (terrestrial, TOPO) frame
    - velocity: include doppler shifts from:
      - Earth rotation: few km/s (diurnal)
      - Earth orbit: 30 km/s (annual)
      - Earth/Sun motion w.r.t. LSR (e.g. LSRK, LSRD)
      - maybe galactic rotation to extragalactic frames
  - imaging applies doppler corrections on the fly, works in LSRK
  - you choose the subsequent output cube parameters
    - **nchan**, **start**, **width** specify which channels to image

  - can shift and regrid data before imaging using cvel

When to stop cleaning? : niter & threshold



```
--------> inp(clean)
#  clean :: Invert and deconvolve images with selected algorithm
vis                 =         ''        #  Name of input visibility file
imagename           =         ''        #  Pre-name of output images
outlierfile         =         ''        #  Text file with image names, sizes, centers for outliers
field               =         ''        #  Field Name or id
spw                 =         ''        #  Spectral windows e.g. '0~3', '' is all
selectdata          =      False        #  Other data selection parameters
mode                =      'mfs'        #  Spectral gridding type (mfs, channel, velocity, frequency)
     nterms         =          1        #  Number of Taylor coefficients to model the sky frequency dependence
     reffreq        =         ''        #  Reference frequency (nterms > 1),'' uses central data-frequency

gridmode            =         ''        #  Gridding kernel for FFT-based transforms, default='' None
niter               =        500        #  Maximum number of iterations
gain                =        0.1        #  Loop gain for cleaning
threshold           =   '0.0mJy'        #  Flux level to stop cleaning, must include units: '1.0mJy'
psfmode             =    'clark'        #  Method of PSF calculation to use during minor cycles
imagermode          =         ''        #  Options: 'csclean' or 'mosaic', '', uses psfmode
multiscale          =         []        #  Deconvolution scales (pixels); [] = standard clean
interactive         =      False        #  Use interactive clean (with GUI viewer)
mask                =         []        #  Cleanbox(es), mask image(s), region(s), or a level
imsize              = [256, 256]        #  x and y image size in pixels. Single value: same for both
cell                = ['1.0arcsec']     #  x and y cell size(s). Default unit arcsec.
phasecenter         =         ''        #  Image center: direction or field index
restfreq            =         ''        #  Rest frequency to assign to image (see help)
stokes              =        'I'        #  Stokes params to image (eg I,IV,IQ,IQUV)
weighting           =  'natural'        #  Weighting of uv (natural, uniform, briggs, ...)
uvtaper             =      False        #  Apply additional uv tapering of visibilities
modelimage          =         ''        #  Name of model image(s) to initialize cleaning
restoringbeam       =       ['']        #  Output Gaussian restoring beam for CLEAN image
pbcor               =      False        #  Output primary beam-corrected image
minpb               =        0.2        #  Minimum PB level to use
calready            =       True        #  True required for self-calibration
allowchunk          =      False        #  Divide large image cubes into channel chunks for deconvolution
async               =      False        #  If true the taskname must be started using clean(...)
```

When to stop cleaning? : niter & threshold

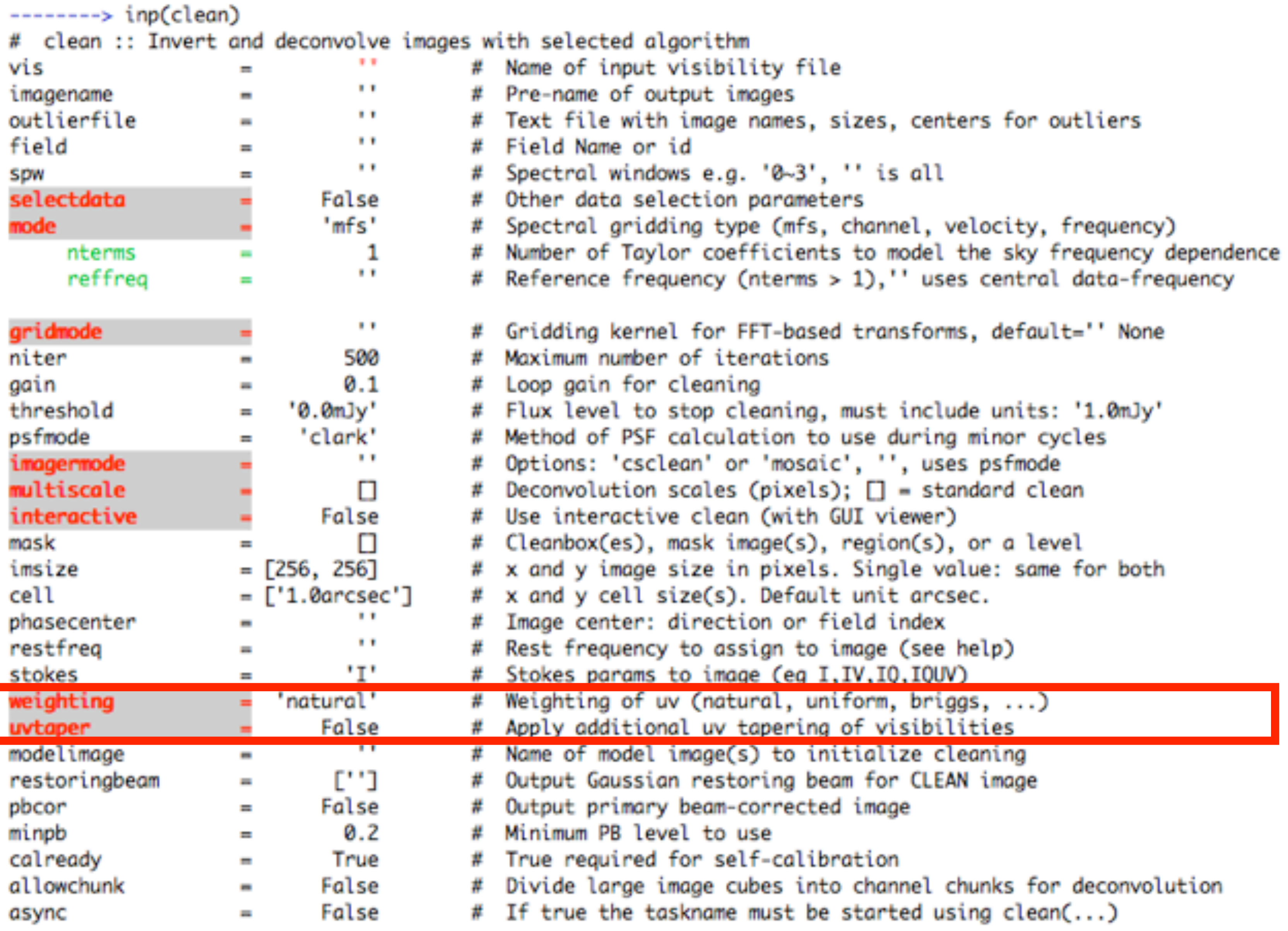
- niter
  - Maximum number of clean interations to perform.
  - Set niter=0 for dirty map OR set to large number
  - use "threshold" to terminate cleaning

- threshold
  - stop cleaning when peak *residual* has this value
  - set to multiple of rms noise if noise-limited

    OR

    fraction of brightest source peak flux if dynamic range limited

mode



```
--------> inp(clean)
#  clean :: Invert and deconvolve images with selected algorithm
vis                 =         ''        #  Name of input visibility file
imagename           =         ''        #  Pre-name of output images
outlierfile         =         ''        #  Text file with image names, sizes, centers for outliers
field               =         ''        #  Field Name or id
spw                 =         ''        #  Spectral windows e.g. '0~3', '' is all
selectdata          =      False        #  Other data selection parameters
mode                =      'mfs'        #  Spectral gridding type (mfs, channel, velocity, frequency)
     nterms         =          1        #  Number of Taylor coefficients to model the sky frequency dependence
     reffreq        =         ''        #  Reference frequency (nterms > 1),'' uses central data-frequency

gridmode            =         ''        #  Gridding kernel for FFT-based transforms, default='' None
niter               =        500        #  Maximum number of iterations
gain                =        0.1        #  Loop gain for cleaning
threshold           =   '0.0mJy'        #  Flux level to stop cleaning, must include units: '1.0mJy'
psfmode             =    'clark'        #  Method of PSF calculation to use during minor cycles
imagermode          =         ''        #  Options: 'csclean' or 'mosaic', '', uses psfmode
multiscale          =         []        #  Deconvolution scales (pixels); [] = standard clean
interactive         =      False        #  Use interactive clean (with GUI viewer)
mask                =         []        #  Cleanbox(es), mask image(s), region(s), or a level
imsize              = [256, 256]        #  x and y image size in pixels. Single value: same for both
cell                = ['1.0arcsec']     #  x and y cell size(s). Default unit arcsec.
phasecenter         =         ''        #  Image center: direction or field index
restfreq            =         ''        #  Rest frequency to assign to image (see help)
stokes              =        'I'        #  Stokes params to image (eg I,IV,IQ,IQUV)
weighting           =  'natural'        #  Weighting of uv (natural, uniform, briggs, ...)
uvtaper             =      False        #  Apply additional uv tapering of visibilities
modelimage          =         ''        #  Name of model image(s) to initialize cleaning
restoringbeam       =       ['']        #  Output Gaussian restoring beam for CLEAN image
pbcor               =      False        #  Output primary beam-corrected image
minpb               =        0.2        #  Minimum PB level to use
calready            =       True        #  True required for self-calibration
allowchunk          =      False        #  Divide large image cubes into channel chunks for deconvolution
async               =      False        #  If true the taskname must be started using clean(...)
```

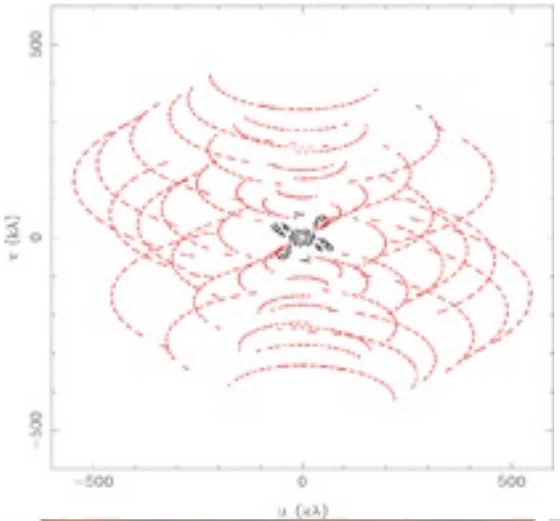
Dirty Beam Shape and Weighting

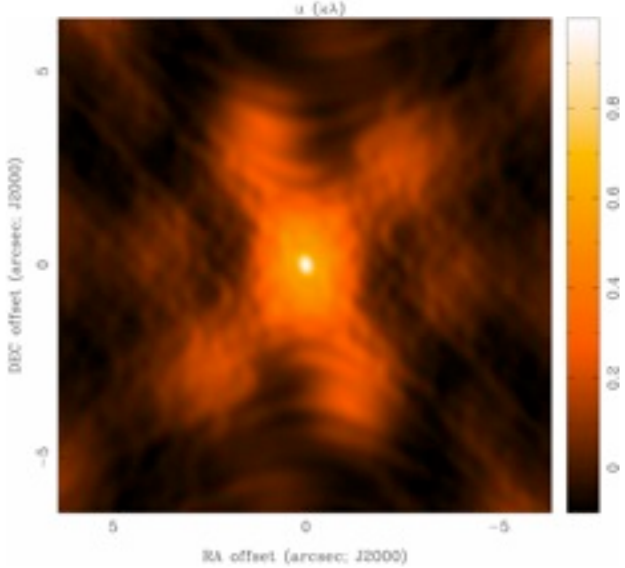
- introduce weighting function W(u,v)

$$S(l,m) = \mathcal{F}^{-1}\{W(u,v) \times S(u,v)\}$$

  - W modifies dirty beam

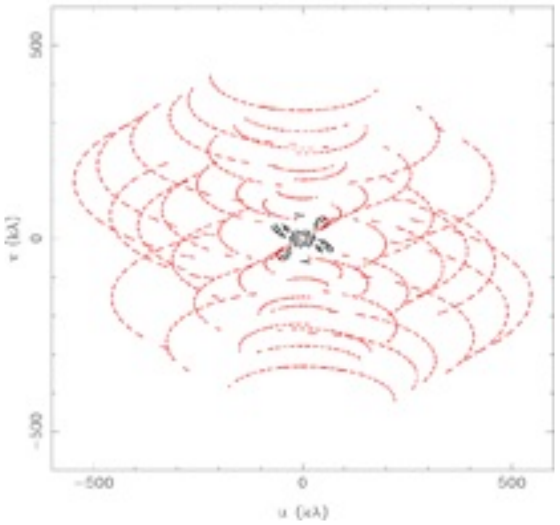
- "Natural" weighting
  - *W(u,v) = 1/σ^2(u,v)* at points with data and zero elsewhere, where σ^2(u,v) is the noise variance of the (u,v) sample
  - maximizes point source sensitivity
  - (lowest rms in image)
  - generally more weight to short baselines (large spatial scales), degrades resolution

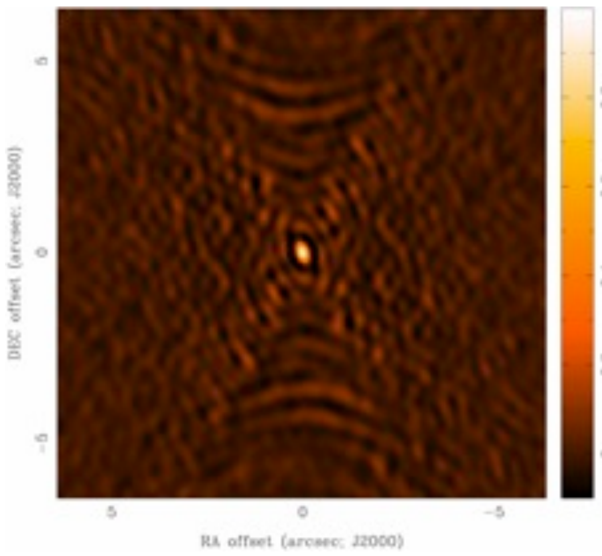




Dirty Beam Shape and Weighting

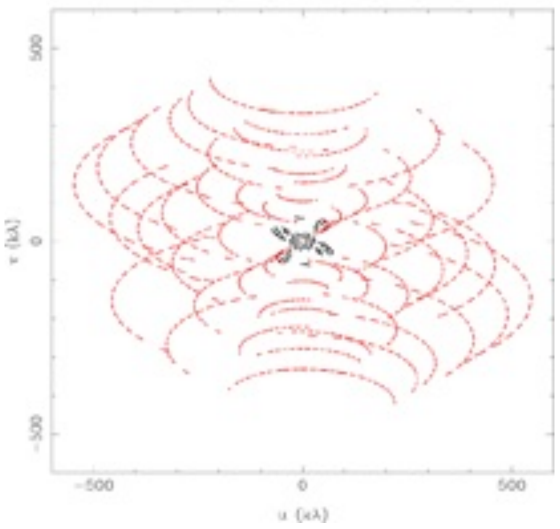
- "Uniform" weighting
  - *W(u,v)* is inversely proportional to local density of (u,v) points
  - fills (u,v) plane more uniformly, so (outer) sidelobes are lower
  - gives more weight to long baselines and therefore higher angular resolution
  - degrades point source sensitivity (higher rms in image)
  - can be trouble with sparse sampling: cells with few data points have same weight as cells with many data points

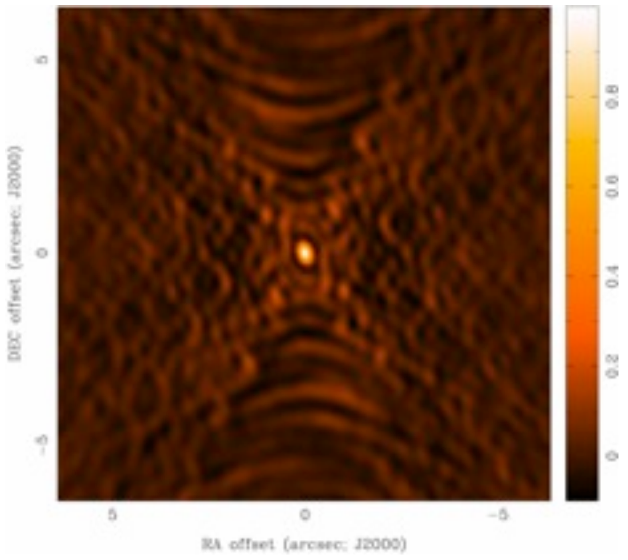




Dirty Beam Shape and Weighting

- "Robust" (Briggs) weighting
  - variant of "uniform" that avoids giving too much weight to cell with low natural weight
  - an adjustable parameter that allows for continuous variation between highest angular resolution and optimal point source sensitivity
  - large threshold → natural weighting
  - small threshold → uniform weighting
  - robust ~ 0 to 1 gives good results





$$w_i = \frac{1}{\sigma_i^2} \qquad w_i = \frac{w_i}{1 + W_k f^2} \qquad f^2 = \frac{(5 * 10^{-R})^2}{\frac{\sum_k W_k^2}{\sum_i w_i}}$$

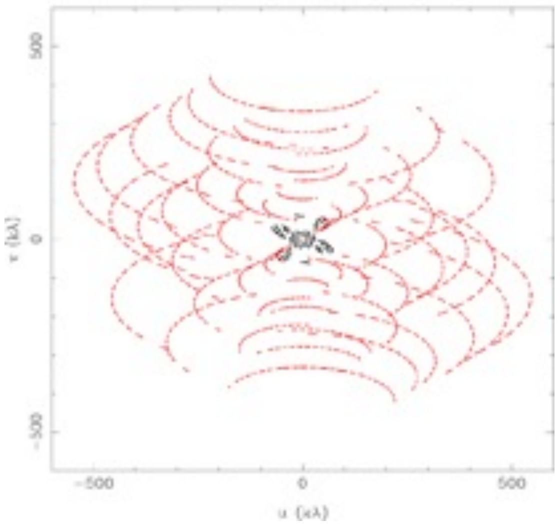
Dirty Beam Shape and Weighting

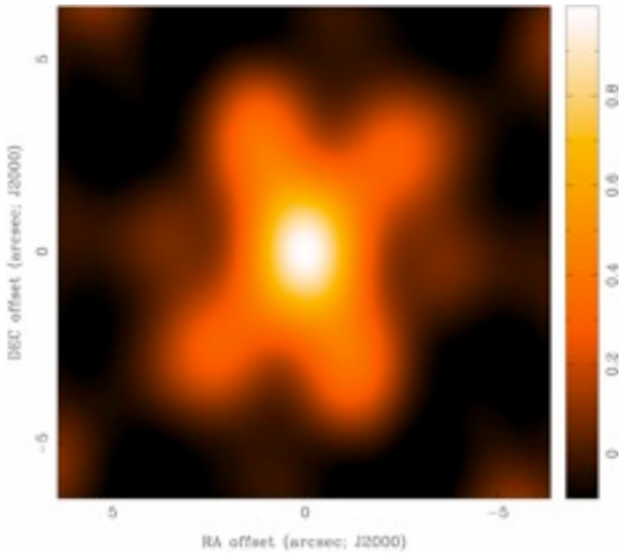
- "Tapering"
  - apodize the (u,v) sampling by a Gaussian

$$W(u,v) = exp\left\{-\frac{(u^2+v^2)}{t^2}\right\}$$

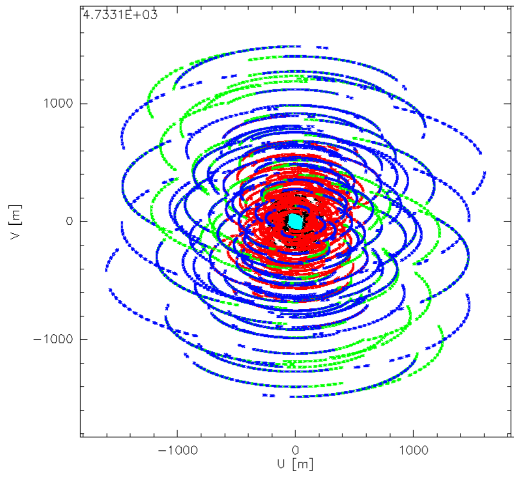
t = tapering parameter (in kλ; arcsec)

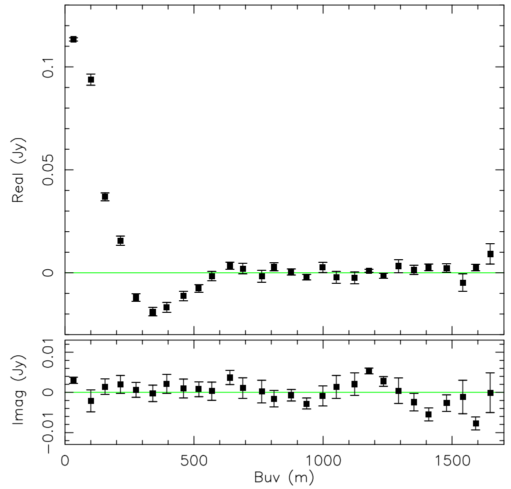
  - like smoothing in the image plane (convolution by a Gaussian)
  - gives more weight to short baselines, degrades angular resolution
  - degrades point source sensitivity but can improve sensitivity to extended structure

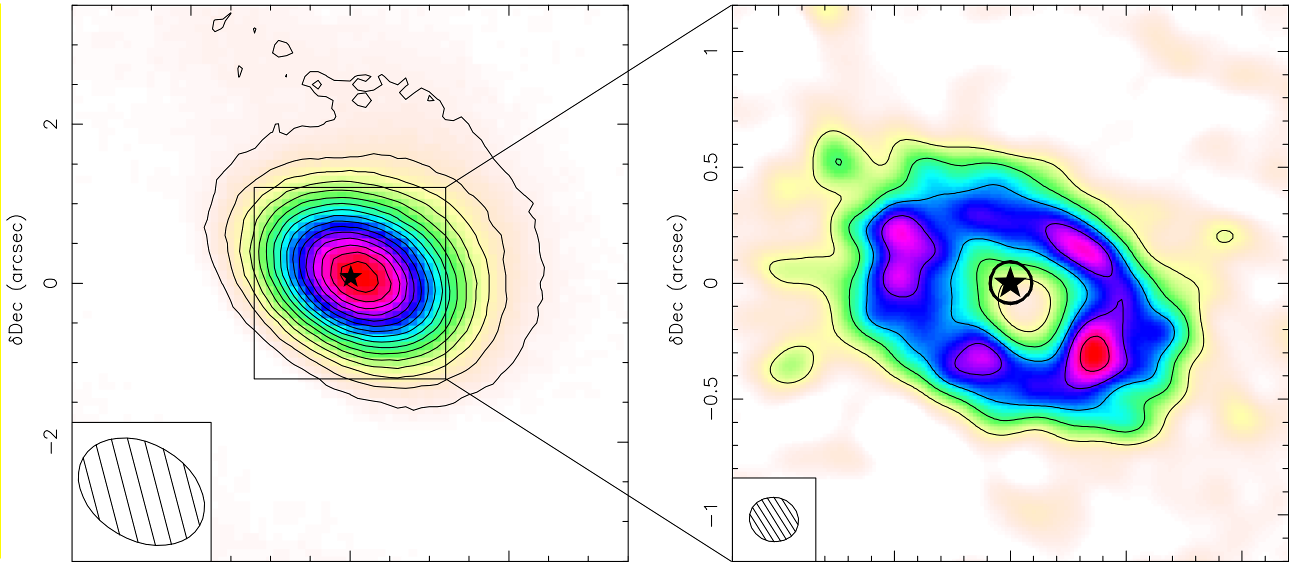




Example of Natural vs. Uniform Weighting







NATURAL WEIGHTING

UNIFROM WEIGHTING

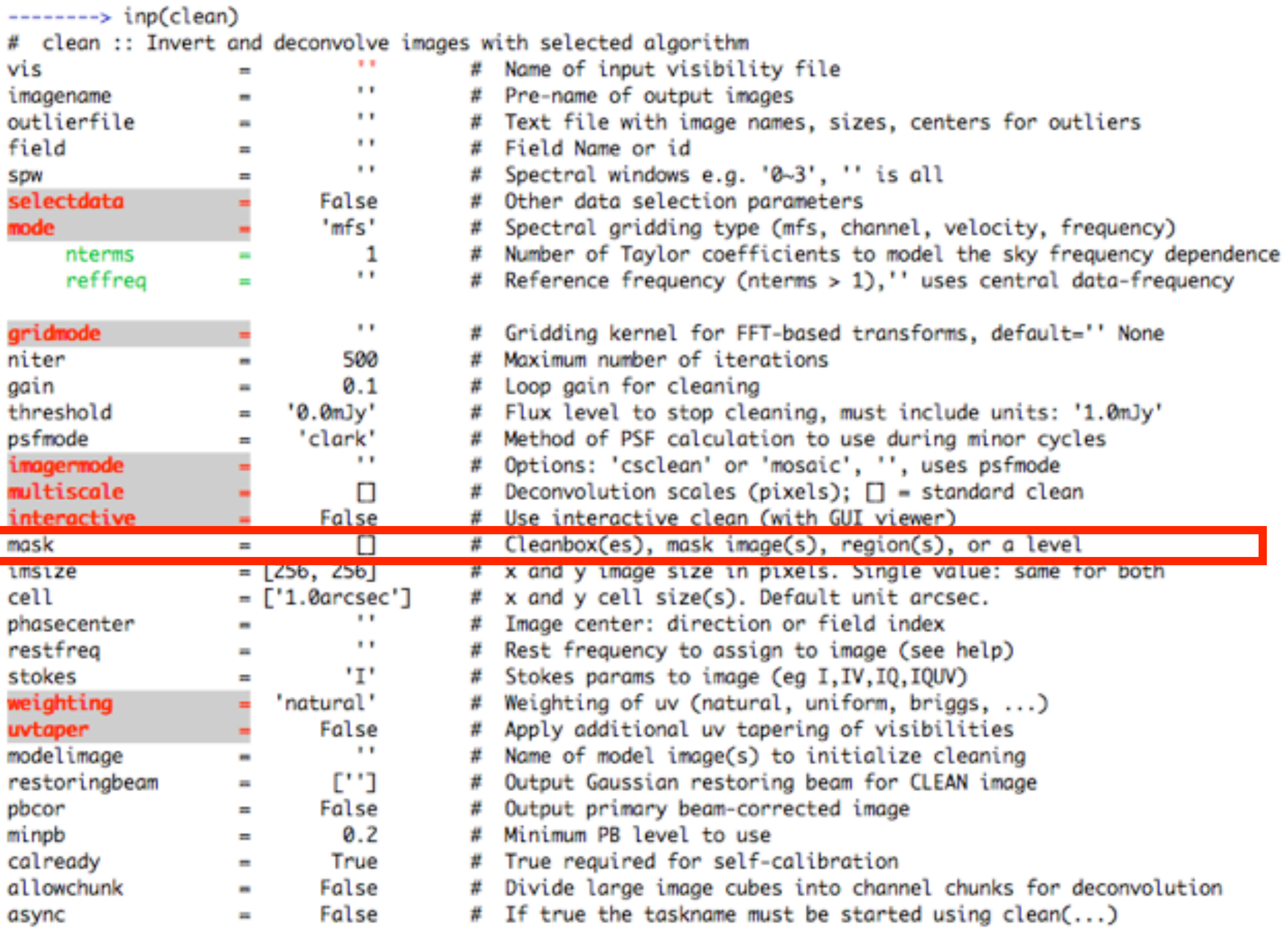
FWHM beam size = 1.7'' x 1.2''   0.21'' x 0.19''

Weighting and Tapering: Summary

- imaging parameters provide a lot of freedom
- appropriate choice depends on science goals

	Robust/Uniform	Natural	Taper
Resolution	higher	medium	lower
Sidelobes	lower	higher	depends
Point Source Sensitivity	lower	maximum	lower
Extended Source Sensitivity	lower	medium	higher

mask



```
--------> inp(clean)
#  clean :: Invert and deconvolve images with selected algorithm
vis                 =         ''        #  Name of input visibility file
imagename           =         ''        #  Pre-name of output images
outlierfile         =         ''        #  Text file with image names, sizes, centers for outliers
field               =         ''        #  Field Name or id
spw                 =         ''        #  Spectral windows e.g. '0~3', '' is all
selectdata          =      False        #  Other data selection parameters
mode                =      'mfs'        #  Spectral gridding type (mfs, channel, velocity, frequency)
     nterms         =          1        #  Number of Taylor coefficients to model the sky frequency dependence
     reffreq        =         ''        #  Reference frequency (nterms > 1),'' uses central data-frequency

gridmode            =         ''        #  Gridding kernel for FFT-based transforms, default='' None
niter               =        500        #  Maximum number of iterations
gain                =        0.1        #  Loop gain for cleaning
threshold           =   '0.0mJy'        #  Flux level to stop cleaning, must include units: '1.0mJy'
psfmode             =    'clark'        #  Method of PSF calculation to use during minor cycles
imagermode          =         ''        #  Options: 'csclean' or 'mosaic', '', uses psfmode
multiscale          =         []        #  Deconvolution scales (pixels); [] = standard clean
interactive         =      False        #  Use interactive clean (with GUI viewer)
mask                =         []        #  Cleanbox(es), mask image(s), region(s), or a level
imsize              = [256, 256]        #  x and y image size in pixels. Single value: same for both
cell                = ['1.0arcsec']     #  x and y cell size(s). Default unit arcsec.
phasecenter         =         ''        #  Image center: direction or field index
restfreq            =         ''        #  Rest frequency to assign to image (see help)
stokes              =        'I'        #  Stokes params to image (eg I,IV,IQ,IQUV)
weighting           =  'natural'        #  Weighting of uv (natural, uniform, briggs, ...)
uvtaper             =      False        #  Apply additional uv tapering of visibilities
modelimage          =         ''        #  Name of model image(s) to initialize cleaning
restoringbeam       =       ['']        #  Output Gaussian restoring beam for CLEAN image
pbcor               =      False        #  Output primary beam-corrected image
minpb               =        0.2        #  Minimum PB level to use
calready            =       True        #  True required for self-calibration
allowchunk          =      False        #  Divide large image cubes into channel chunks for deconvolution
async               =      False        #  If true the taskname must be started using clean(...)
```

mask

- Define region where you expect the emission to be
- CLEAN components will fall within the masked region
- can also define with interactive=True

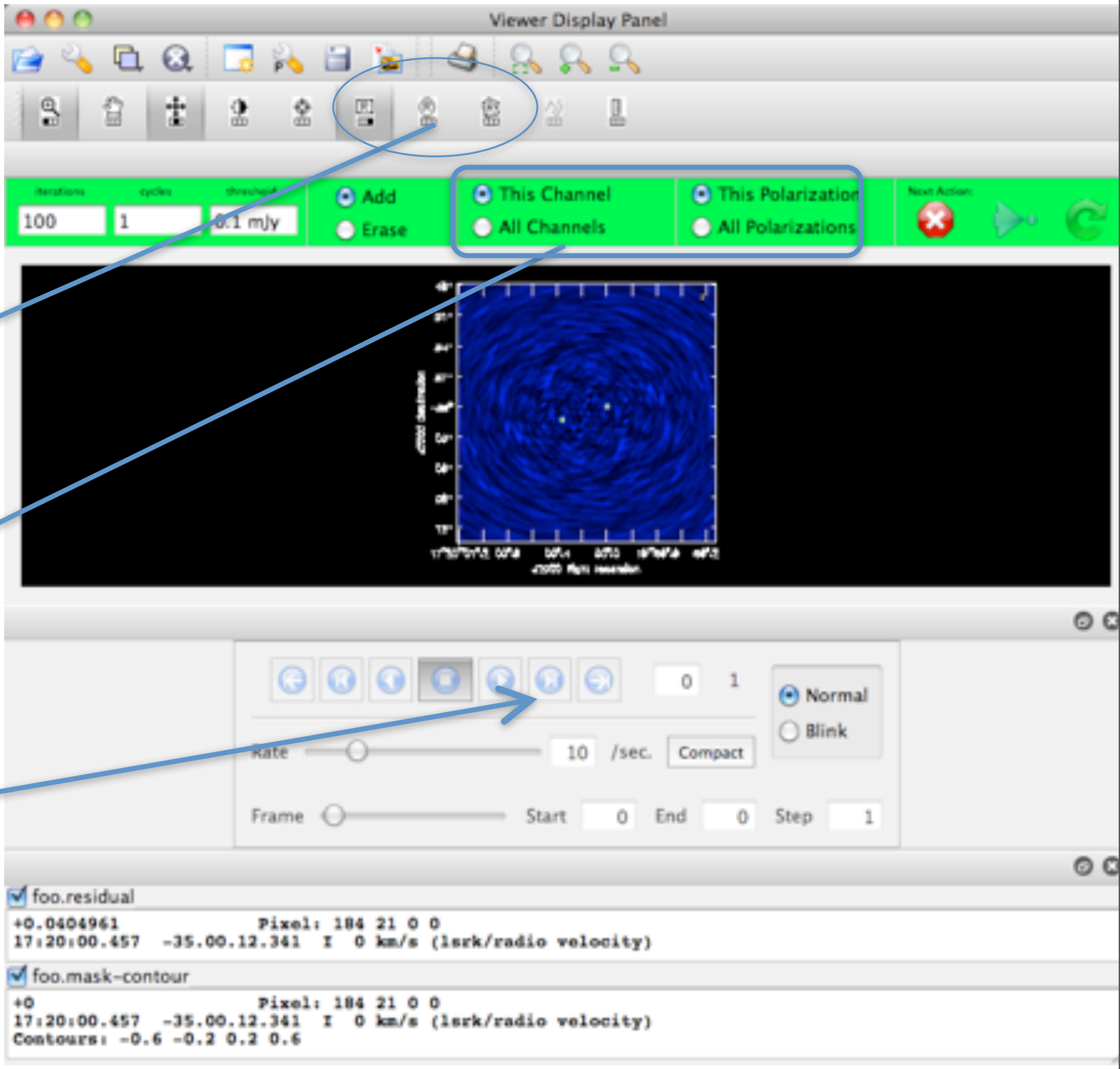
Interactive Clean

residual image in viewer

define a mask with R-click on shape type

define the same mask for all channels

or iterate through the channels with the tape deck and define separate masks



```
+0.0404961                    Pixel: 184 21 0 0
17:20:00.457   -35.00.12.341   I   0 km/s (lsrk/radio velocity)
```

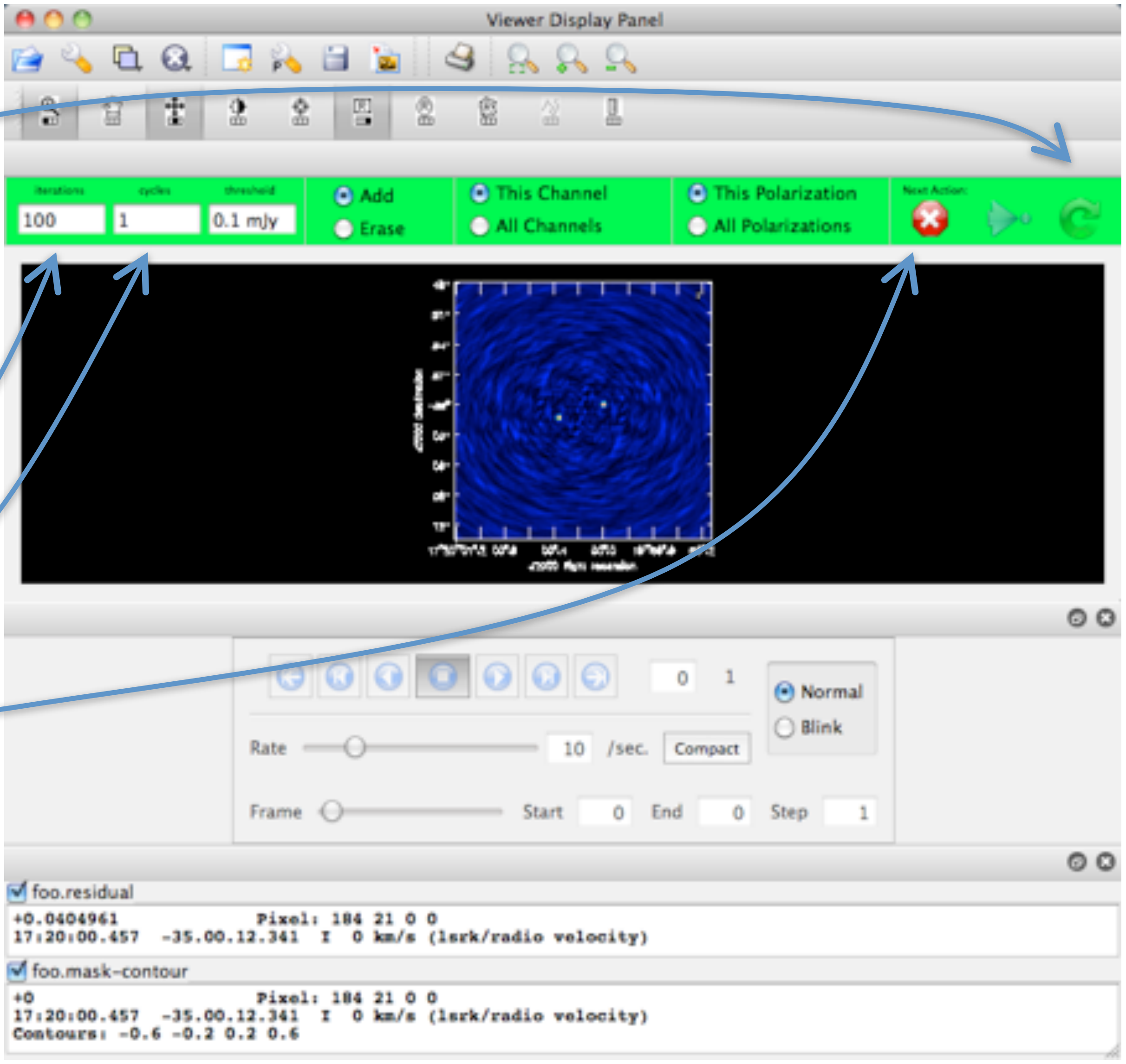
```
+0                            Pixel: 184 21 0 0
17:20:00.457   -35.00.12.341   I   0 km/s (lsrk/radio velocity)
Contours: -0.6 -0.2 0.2 0.6
```

Interactive Clean

perform N iterations

and return – every time the residual is displayed is a major cycle

continue until #cycles or threshold reached, or user stop



```
+0.0404961              Pixel: 184 21 0 0
17:20:00.457  -35.00.12.341  I   0 km/s (lsrk/radio velocity)
```

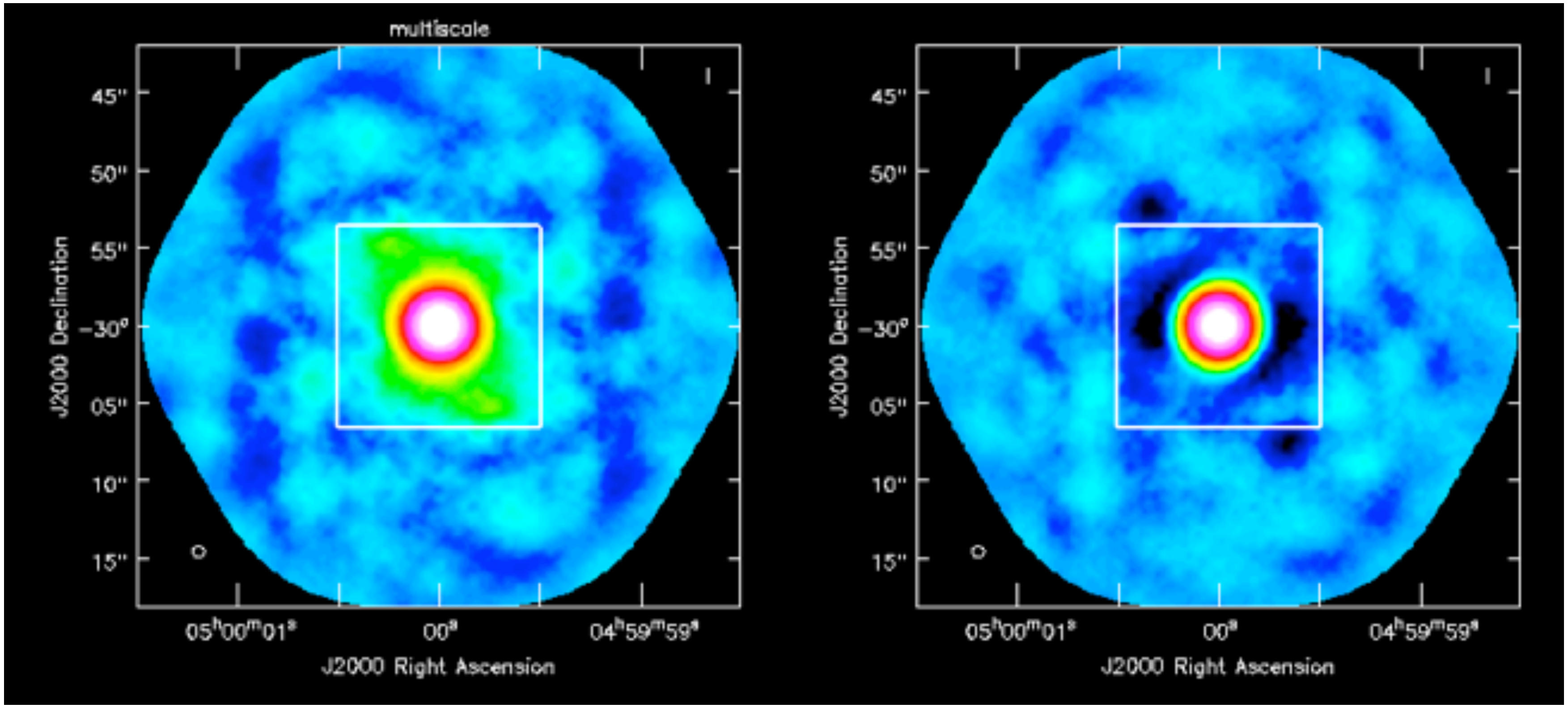
```
+0                      Pixel: 184 21 0 0
17:20:00.457  -35.00.12.341  I   0 km/s (lsrk/radio velocity)
Contours: -0.6 -0.2 0.2 0.6
```

multiscale clean

- Models the sky using components with different size scales
- multiscale = [0,5,15,45]
  - scales are in units of pixels
  - 0=point, typically 1-2x synthesized beam, then multiples of 2-3x that up to a fraction of the PB can be tricky to get to work right
- Yields promising results on extended sources
- See, e.g., Rich et al. 2008, AJ, 136, 2897 for comparison between clean and multiscale clean

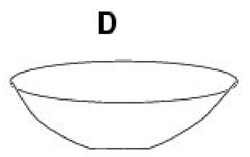
multiscale clean

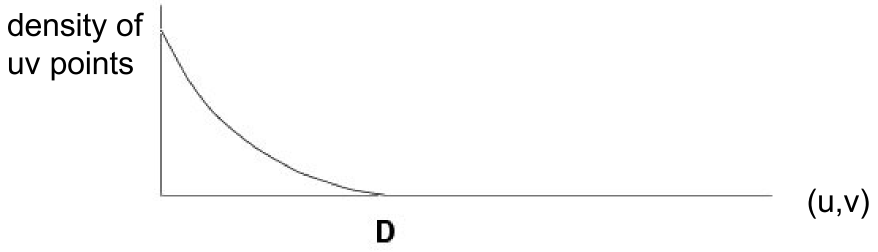
multiscale "classic" 1-scale



zero-spacing

- Large Single Telescope
  - make an image by scanning across the sky
  - all Fourier components from 0 to D sampled, where D is the telescope diameter (weighting depends on illumination)





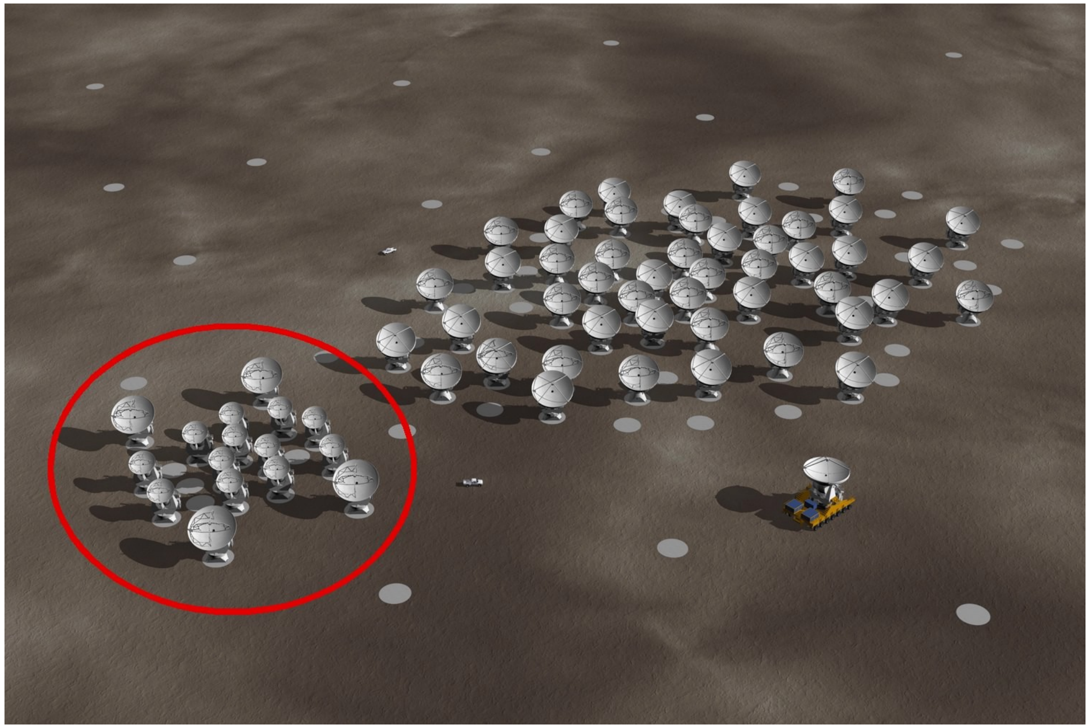
  - Fourier transform single dish map = T(x,y) ⊗ A(x,y), then divide by a(x,y) = FT{A(x,y)}, to estimate V(u,v)

$$\hat{V}(u,v) = \frac{[V(u,v)a(u,v)]}{\hat{a}(u,v)}$$

  - choose D large enough to overlap interferometer samples of V(u,v) and avoid using data where a(x,y) becomes small

zero-spacing

- separate array of smaller telescopes
  - use smaller telescopes observe short baselines not accessible to larger telescopes
  - shortest baselines from larger telescopes total power maps



ALMA with ACA

50 x 12 m:	12 m to 14 km
+12 x 7 m:	fills 7 to 12 m
+ 4 x 12 m:	fills 0 to 7 m

self calibration

- phase calibration is not perfect
  - gain solution interpolated from different time, and different location on the sky
- self-calibration corrects for gain errors from on-source observations
  - N complex gains, where N = number of antennas
  - N(N-1)/2 visibilities
  - over constrained problem is N is large and source structure is simple
  - required sufficient signal-to-noise at each time interval
- procedure
  - assume initial model
  - solve for time-dependent gains
  - form new sky model from corrected data using CLEAN
  - solve for new gains

mosaicking

- image multiple pointings of the antennas
- imagermode = ‘mosaic’
- mosweight = True
  - weights pointings by sensitivity in overlap regions
- field = “
  - selects fields to mosaic
- ftmachine
  - ftmachine=‘mosaic’
    - add in uv plane and invert together
  - ftmachine=’ft’
    - shift and add in image plane