MANCHESTER 1824

The University of Manchester

Introduction to Imaging:

General Principles

Adam Avison

EUROPEAN ARC
ALMA Regional Centre || UK

Further Reading

The slides from this talk are based on the fundamentals of interferometry which are explained in detail across:

- *"Interferometry and Synthesis in Radio Astronomy"* - Thompson, Moran & Swenson
- *"Synthesis Imaging in Radio Astronomy II"* – NRAO
- *"An introduction to Radio Astronomy"* – Burke and Graham-Smith (4th edition out soon Burke, Graham-Smith (& Wilkinson?))
- *"Tools of Radio Astronomy"* – Wilson, Rohfls & Hüttemeister
- *"The CASA Cookbook"* – Ott & Kern et al.

Outline

- What we measure
- Imaging the data
- The CLEANing process

An advanced warning, for a talk about imaging it is rather wordy to begin with.

What we measure

Correlator output

$$R_{x,y} = A_0 \left|V\right| \Delta\nu \cos(2\pi\nu \mathbf{b}_\lambda \cdot \mathbf{s}_0 - \phi_V)$$

Eq. 1

Visibility equation

$$V(u,v) = \int A(l,m) I(l,m)\, e^{-i2\pi(ul+vm)} \frac{\mathrm{d}l\mathrm{d}m}{\sqrt{1-l^2-m^2}}$$

Eq. 2

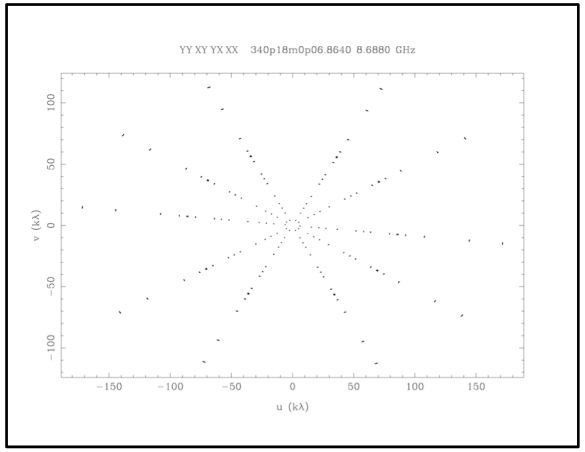
Dirty image

$$I^D(l,m) = \iint_{-\infty}^{\infty} S(u,v) V(u,v) e^{2\pi i(ul+vm)} du dv$$

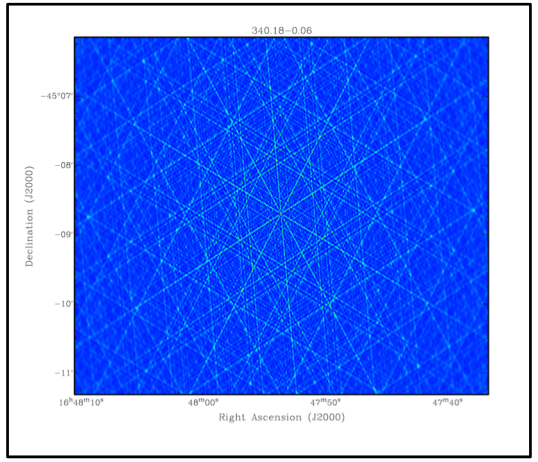
Eq. 3

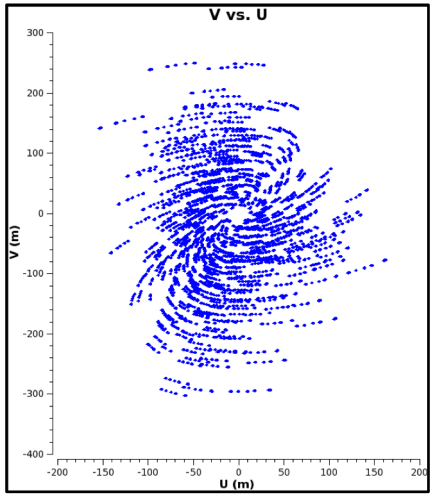
Filling the *uv*-plane

We want to fill the *uv*-plane because the *uv*-coverage is the **FT** of the synthesised beam, *B*. The greater the *uv*-coverage the better behaved the sidelobes are.

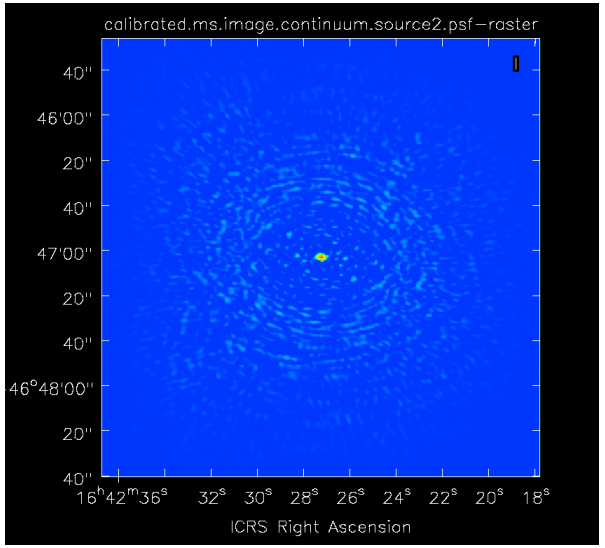


6 dishes
(~15 mins)





42 dishes
(~15 mins)



Sampling function

Let us call the sampling of the uv-plane (aka uv-coverage), S, the sampling function

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

Given this the synthesized beam, B, is $B = \mathbf{FT}(S)$.

And for each uv-point we have an observed visibility, $V(u,v)$, so we can define the *sampled visibility function* as:

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

So $V^s = SV$ and from earlier (eq. 3) $I^D = \mathbf{FT}(V^S) = \mathbf{FT}\,(SV)$. From which it follows that I^D is the measured sky brightness convolved with the synthesis beam, B.

This is a truncated version of Chapter 7 (Briggs, Schwab & Sramek) of *"Synthesis Imaging in Radio Astronomy II"*

Weighting

Given this we can introduce weighting functions to control the shape of the synthesised beam.

$$W(u,v) = \sum_{k=1}^{M} R_k D_k T_k \delta(u - u_k, v - v_k)$$

$\boldsymbol{R_k}$= Weights relating to data quality, i.e. down weight bad data. This is observation dependent and we have no post observation control over it (so ignore).

$\boldsymbol{T_k}$= Tapering function. Apply a tapering function (i.e. Gaussian), to the uv-coverage to for example downweight the outer uv-points lowering resolution.

$\boldsymbol{D_k}$= Density weighting... Next slide.

And as per the previous slide we can define the *weighted and sampled visibility function* as $V^W = WV$.

This is a truncated version of Chapter 7 (Briggs, Schwab & Sramek) of *"Synthesis Imaging in Radio Astronomy II"*

Density weighting description

Due to the nature of how arrays are typically built the uv-coverage density is typically higher toward the uv origin.

Two "extremes" of density weighting are typically used:

Natural weighting: $\boldsymbol{D_k}$= 1. All visibilities are treated the same.
- This gives the highest signal to noise possible within the final image
- The poorest angular resolution and higher sidelobe effects.

Uniform weighting: $\boldsymbol{D_k}$= 1/$\boldsymbol{N_s}$*(k).* Weight visibilities by the number of data points in a symmetric region, $\boldsymbol{s}$.
- Downweights data in dense regions.
- Higher resolution, lower sidelobes, but worse SNR.

There are then super and sub-uniform (see e.g. CASA cookbook), and...

Briggs weighting: A 'sliding scale' between Natural and Uniform controlled by the 'robust' parameter. With (in CASA) +2 = nearly Natural and -2 = nearly Uniform.

We'll come back to these later as we define these during CLEAN.

This is a truncated version of Chapter 7 (Briggs, Schwab & Sramek) of *"Synthesis Imaging in Radio Astronomy II"*

Imaging the data

We now have the sampled and weighted visibilities, $\boldsymbol{V^W}$.

In order to efficiently make an image of our target sky brightness distribution, $\boldsymbol{I}$, we need to take the Fourier transform of this using Fast Fourier Transforms (FFTs).

This requires the $\boldsymbol{V^W}$ data to be gridded on to a regular grid.

This is done by convolution with some suitable gridding function*. Leaving us ultimately with some weighted and gridded visibilities which can be FFT'd to give us our dirty image $\boldsymbol{I^D}$.

*Discussion of gridding algorithms is a little beyond the scope of this workshop, please check the references at the start of this talk for more information.

CLEAN-ing

We've seen that our dirty image $\boldsymbol{I}^D$ is the sky brightness distribution convolved with the synthesised beam $\boldsymbol{B}$.

To get a better representation of the sky brightness distribution we need to remove the artefacts introduced by $\boldsymbol{B}$. To achieve this we use CLEAN*

*As far as I know CLEAN is not an acronym, but it is usually captialised. I don't know why?!

Simple CLEAN overview

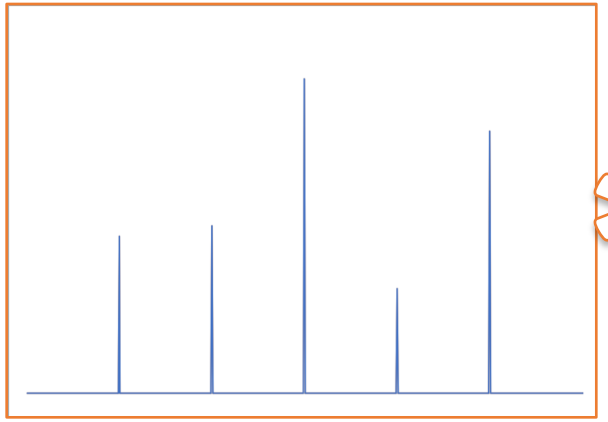
The Högbom (1974) CLEANing algorithm is the simplest CLEAN algorithm and is very illustrative of how CLEAN works in general.

In words the Högbom algorithm works as follows:

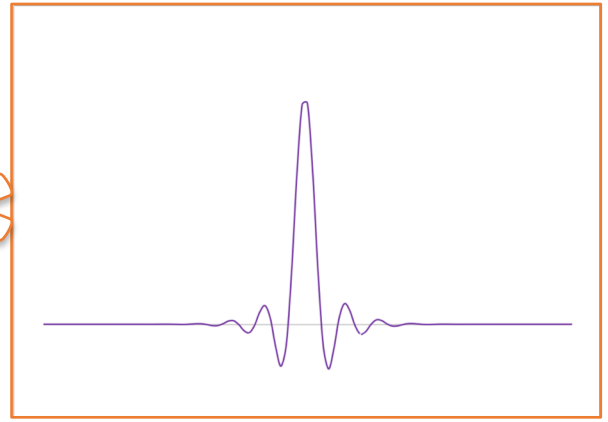
1) Find the magnitude and position of peak emission in the dirty image.
2) Subtract from the dirty image the dirty beam, B, scaled by some gain value (i.e. 0.1). Creating a 'residual' image.
3) Note the position and magnitude subtracted as a point in a model.
4) Repeat 1-3 until a user defined threshold is reached, either some noise limit (in the residual) or a given number of iterations.
5) Convolve the final model with an idealised beam. I.e. a beam based on the interferometer if it was a huge single dish.

Or in a 2D example on the next page

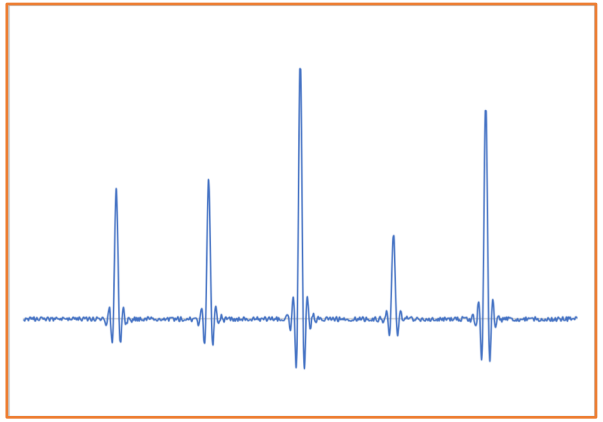
Högbom 2D example



True sky of 5 point sources

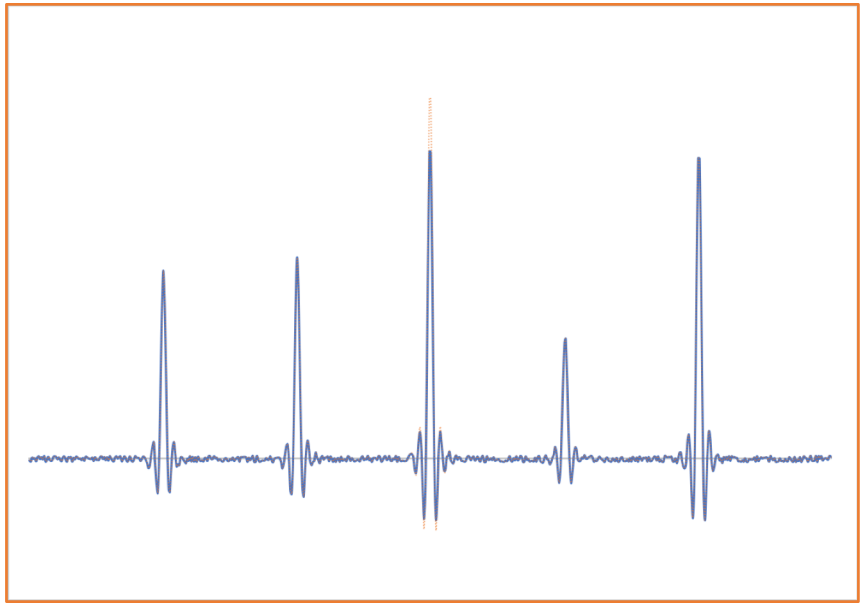


Dirty Beam (DB)

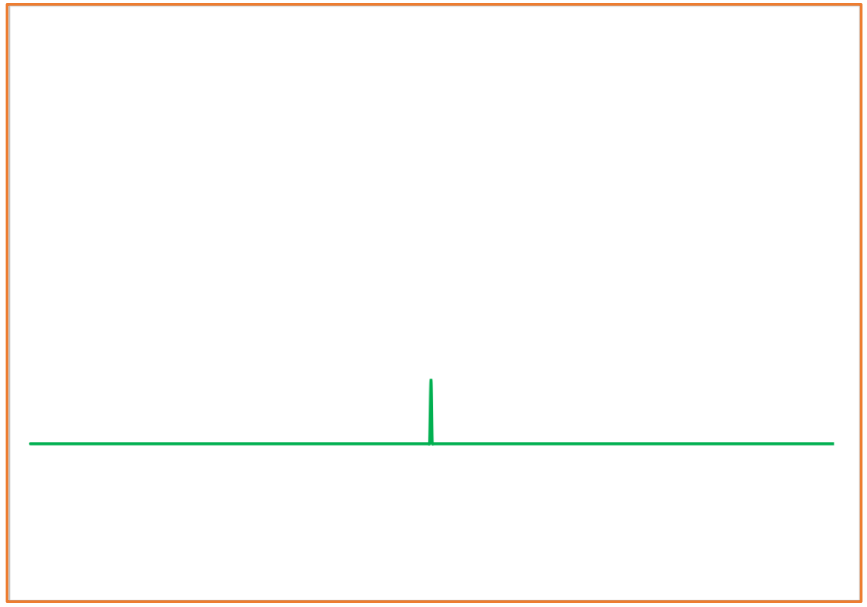


Dirty Image + noise, (DI)

Iteration 1

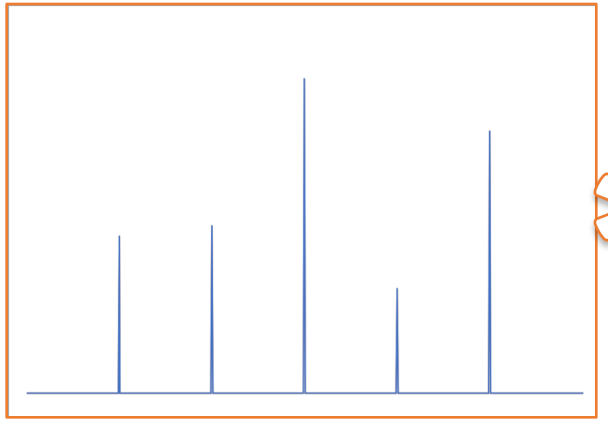


Residual Image after subtracting DB from peak in DI

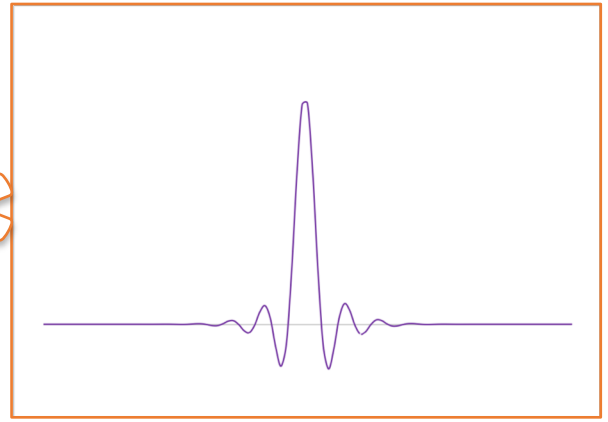


CLEAN components in Model Image

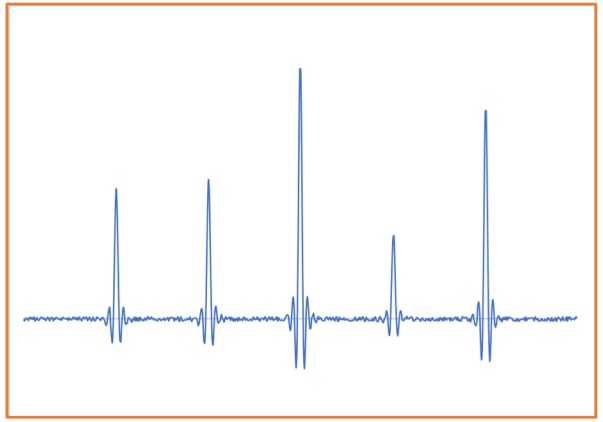
Högbom 2D example



True sky of 5 point sources

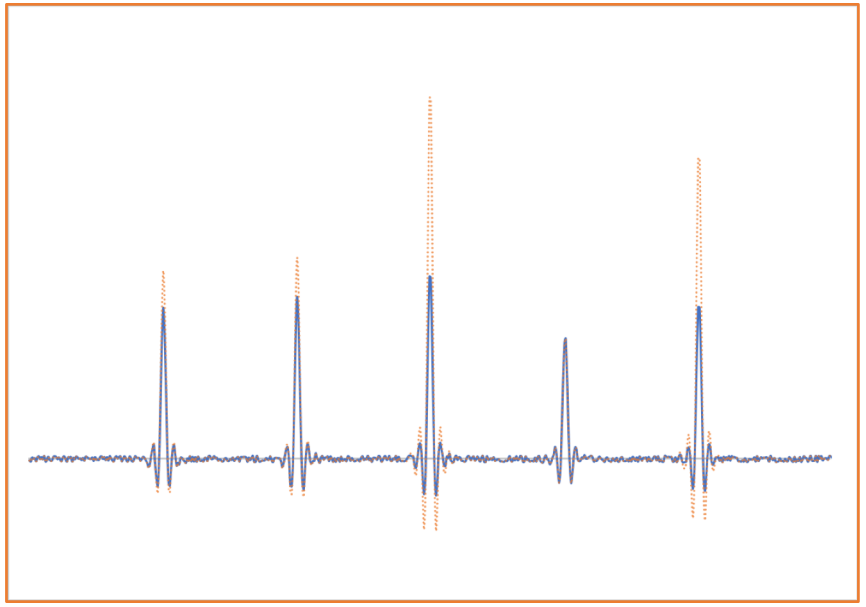


Dirty Beam (DB)

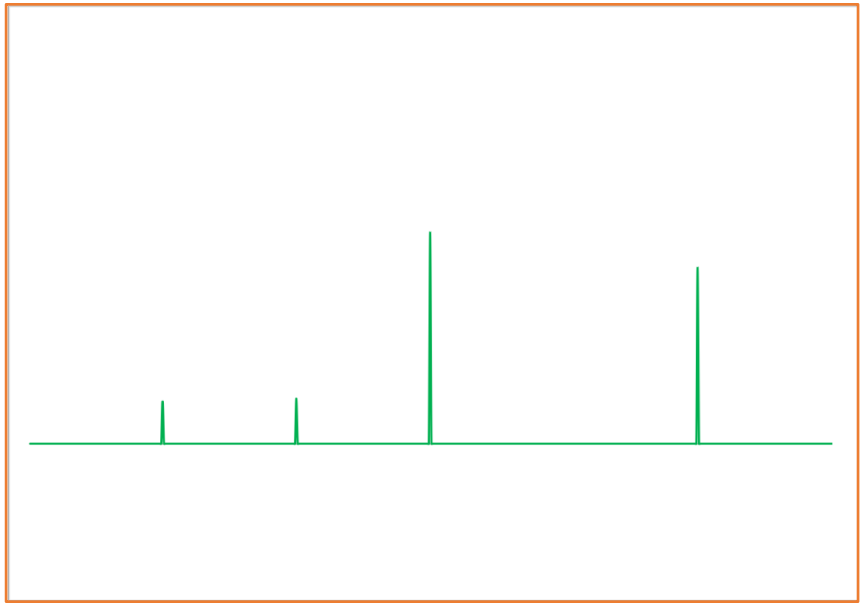


Dirty Image + noise, (DI)

Iteration *n*

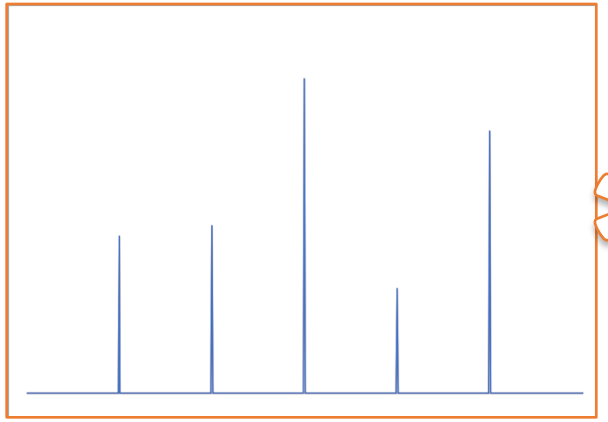


Residual Image after subtracting several DBs from located peaks in DI

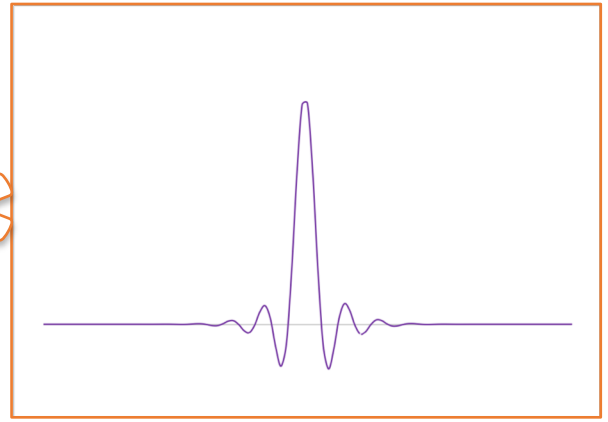


CLEAN components in Model Image

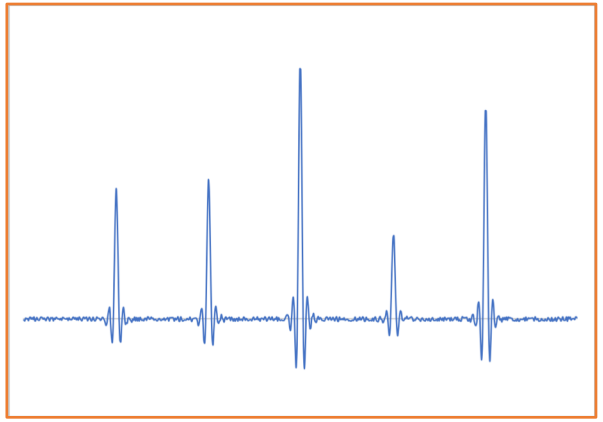
Högbom 2D example



True sky of 5 point sources

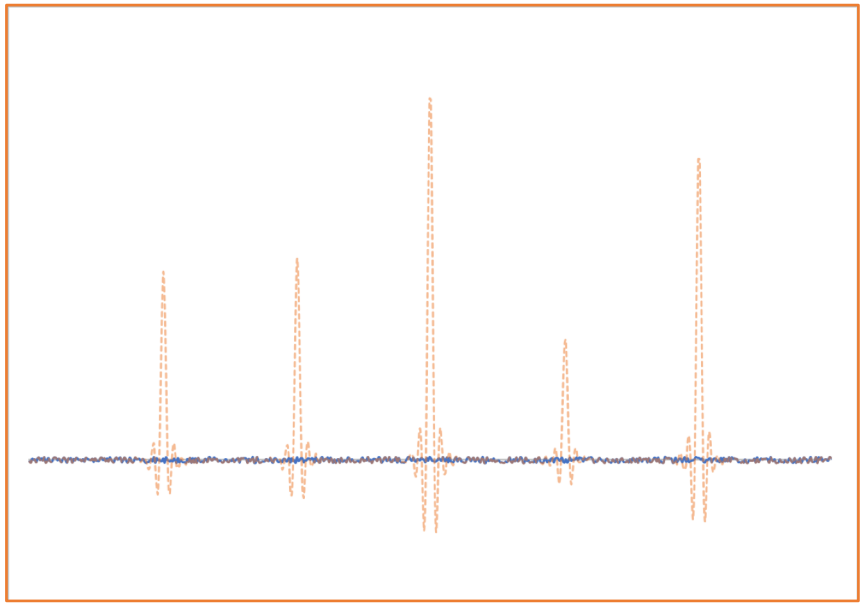


Dirty Beam (DB)

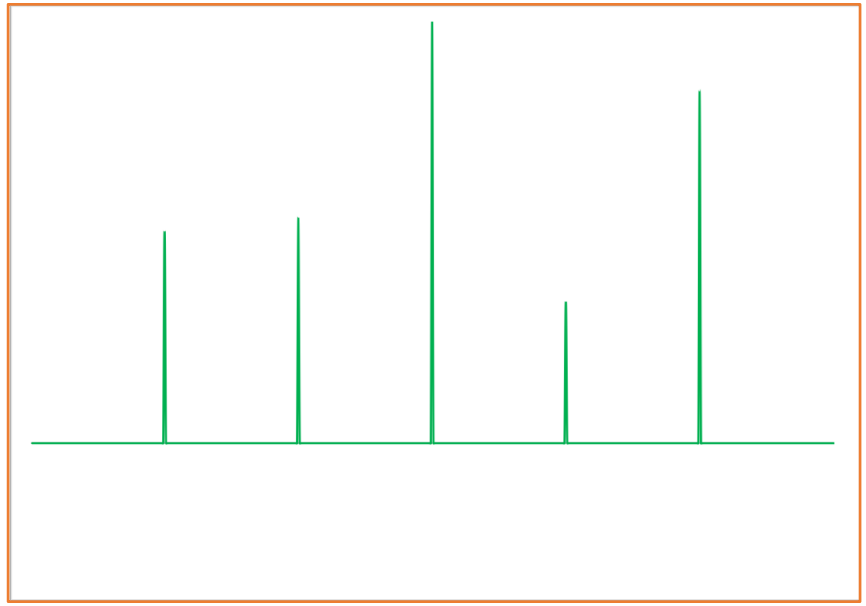


Dirty Image + noise, (DI)

Iteration *FINAL*

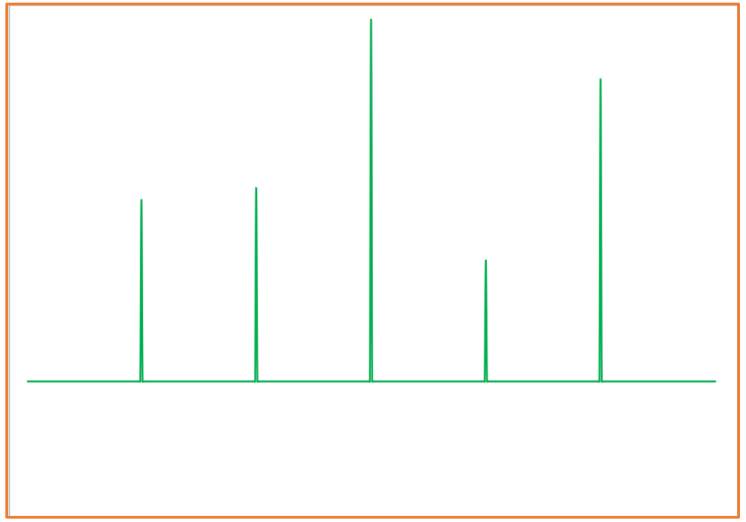


Residual Image after subtracting enough DBs from located peaks in DI until threshold met.

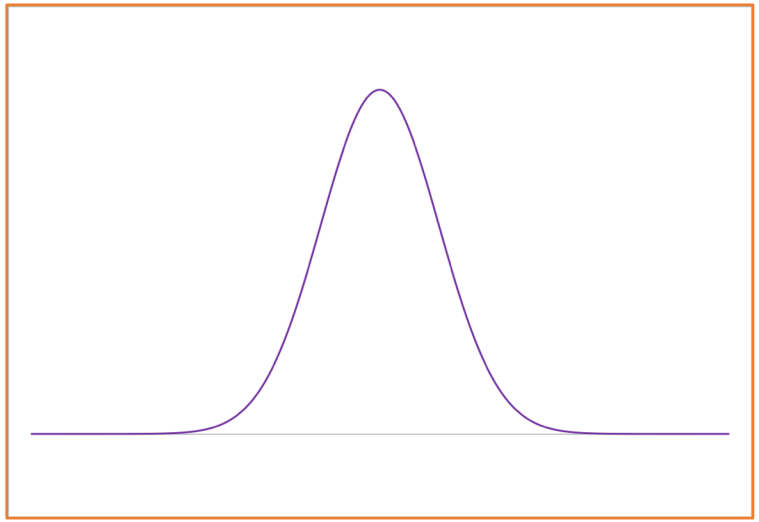


CLEAN components in Model Image after final CLEAN loop

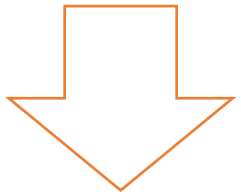
Högbom 2D example

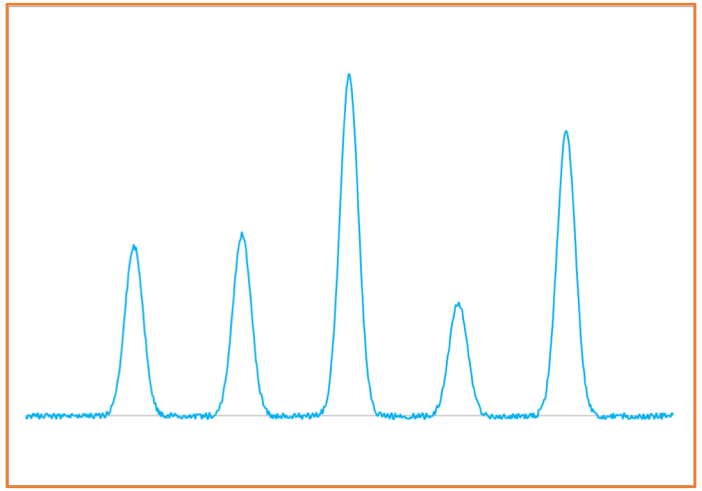


CLEAN components in Model Image after final CLEAN loop

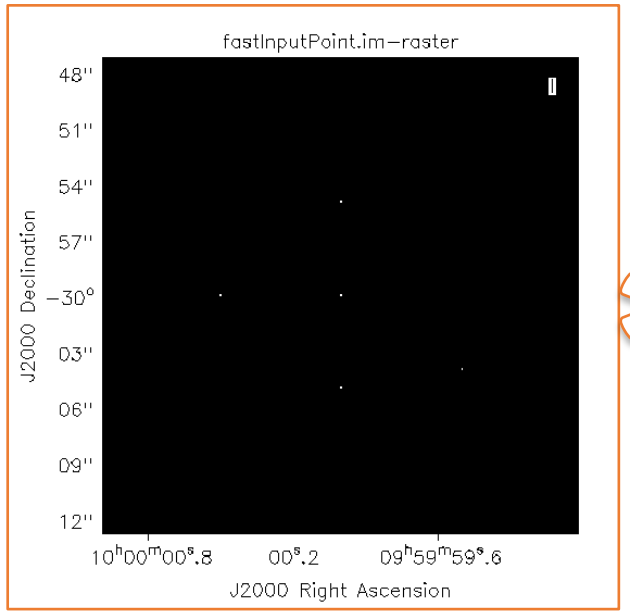


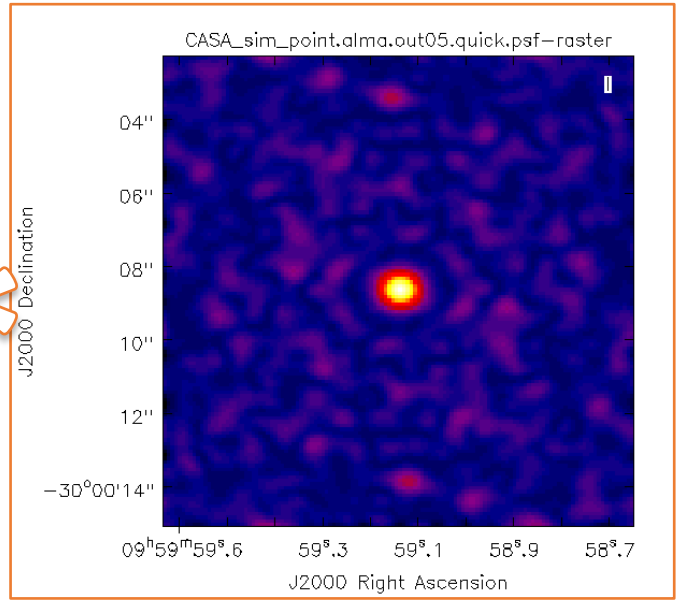
Idealized beam

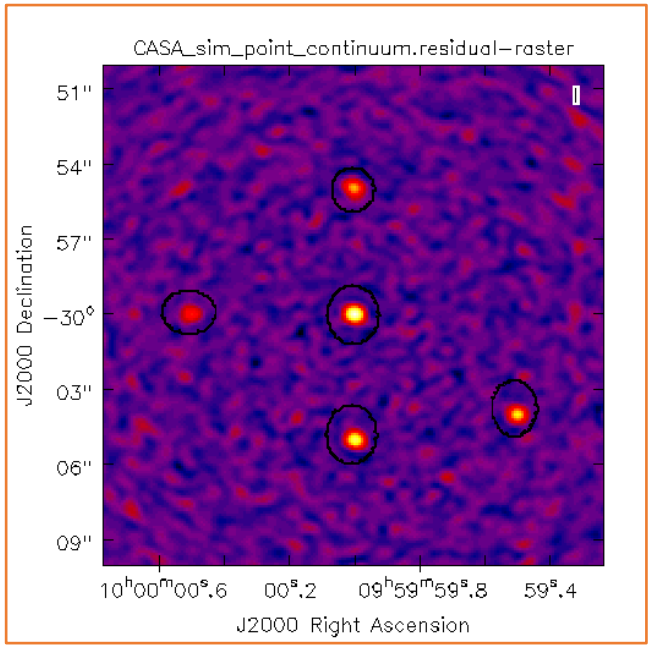


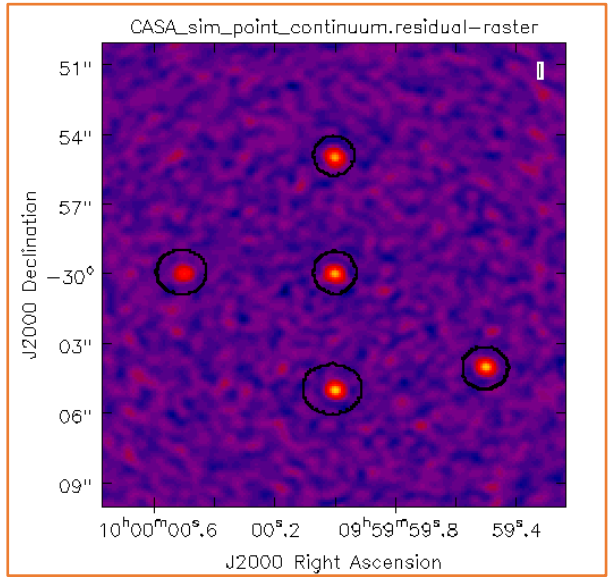


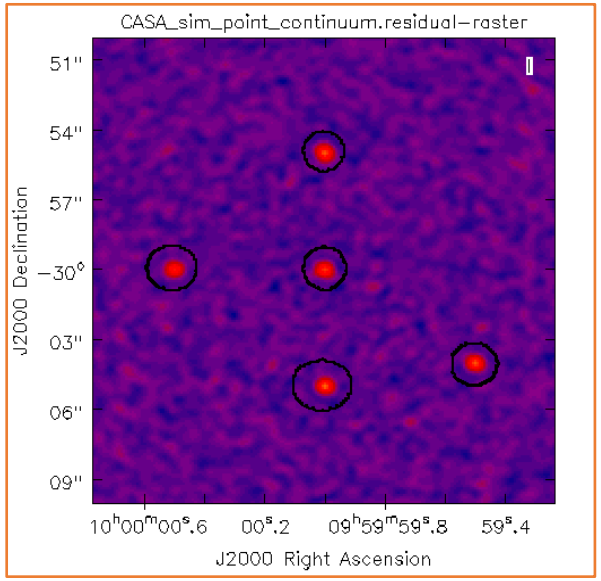
Final reconstructed image

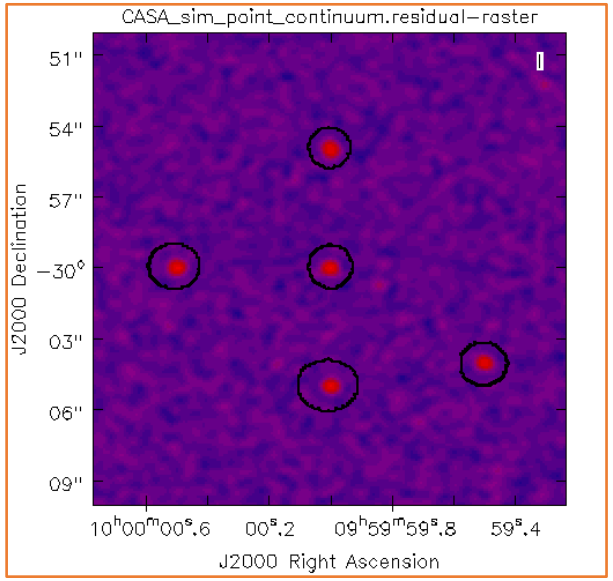


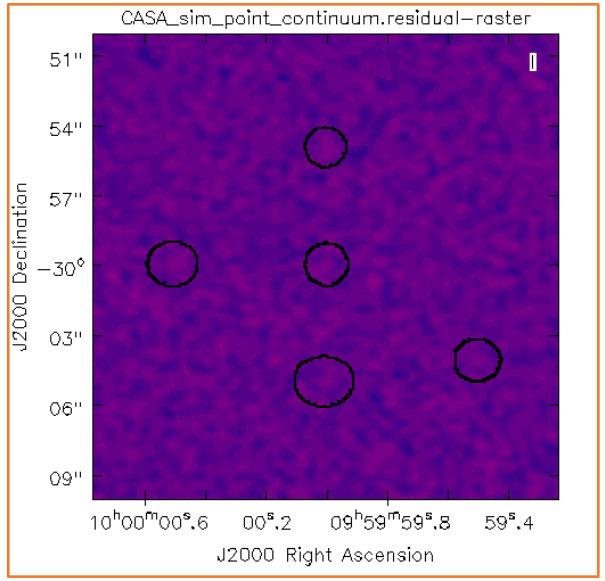












... and in 3D

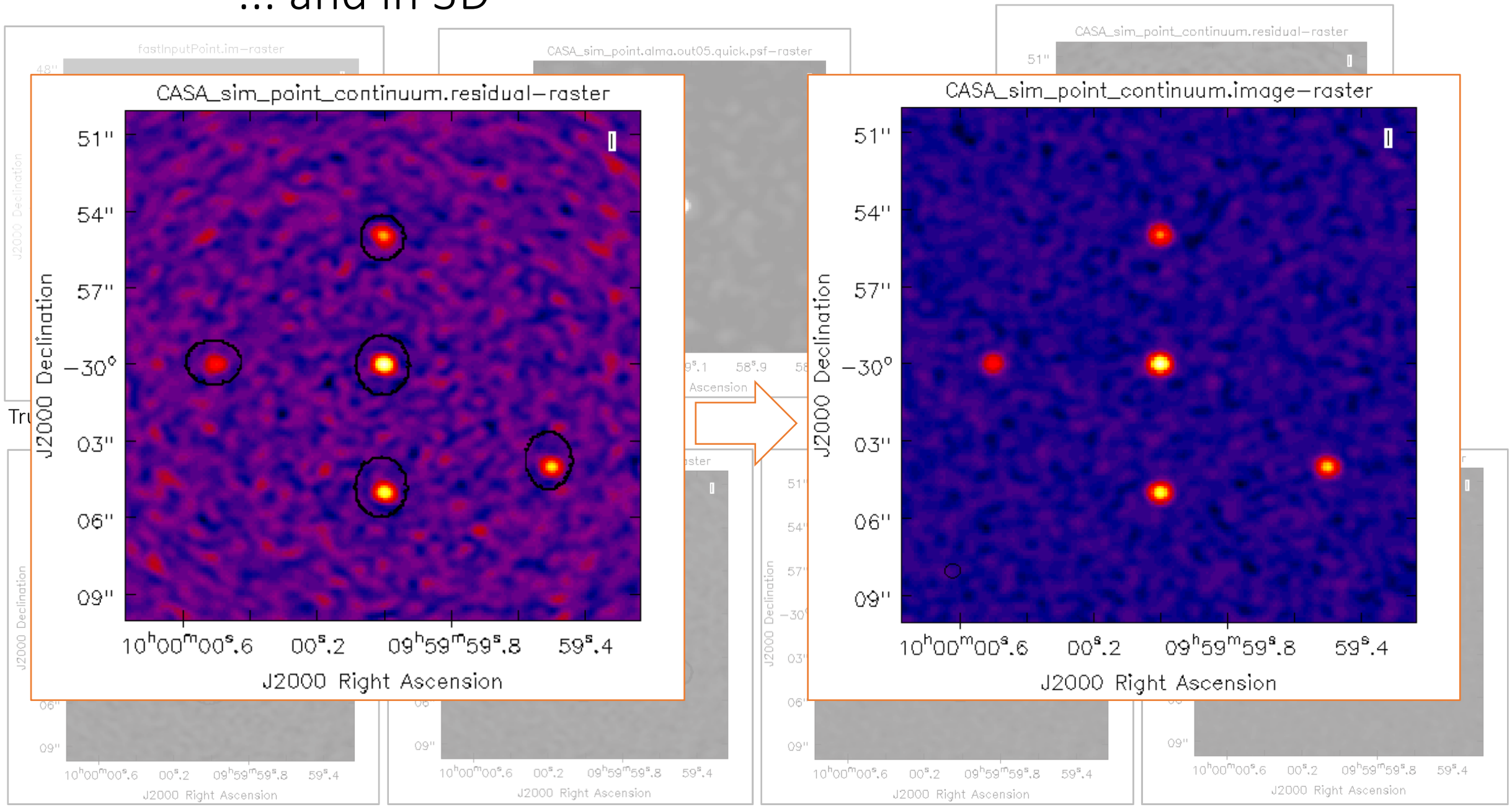
True sky of 5 point sources

Dirty Beam (DB)

Dirty Image + noise, (DI)

→ Increasing CLEAN cycles →

… and in 3D


CASA_sim_point_continuum.residual−raster
51''
54''
57''
−30°
03''
06''
09''
J2000 Declination
10h00m00s.6
00s.2
09h59m59s.8
59s.4
J2000 Right Ascension
CASA_sim_point_continuum.image−raster
51''
54''
57''
−30°
03''
06''
09''
J2000 Declination
10h00m00s.6
00s.2
09h59m59s.8
59s.4
J2000 Right Ascension

Other CLEAN algorithm

- **Clarke/Cotton-Schwab/ClarkeStokes:** Improvements and refinements on Högbom. In general these are preferred to Högbom. They rely on major and minor CLEAN cycles. (Next slide).

- **Multiscale:** Searches for clean components based on user defined size scales. Better for cleaning extended, diffuse, none point-like sources.

- **These are only a couple**

Major and Minor CLEAN cycles

- For the more ‘advanced’ CLEAN algorithms (and infact CASA’s version of Högbom). CLEAN runs what are termed Major and Minor cycles.

- During a Major Cycle the current model is used to generate model visibilities. Subtract these from the data generate residual visibilities. Convert the residual visibilities into a residual image. *(In the visibility domain).*

- During a Minor Cycle peak flux components are found and the model and residual image are updated. *(All in the image domain).*

Useful image from: https://casa.nrao.edu/casadocs/casa-5.3.0/synthesis-imaging/imaging-overview

Basic Sequence of Imaging Logic:

```
Data : Calibrated visibilities, data weights, UV sampling function
Input : Algorithm and iteration controls (stopping threshold, loop gain,...)
Output : Model Image, Restored Image, Residual Image,...

Initialize the model image
Compute the point spread function
Compute the initial residual image
While ( not reached global stopping criterion )          /* Major Cycle */
{
   While ( not reached minor-cycle stopping criterion )   /* Minor Cycle */
   {
      Find the parameters of a new flux component
      Update the model and residual images
   }
   Use current model image to predict model visibilities
   Calculate residual visibilities (data - model)
   Compute a new residual image from residual visibilities
}
Convolve the final model image with the fitted beam and add to the residual image
```

```
#  tclean :: Radio Interferometric Image Reconstruction
vis                 =          ''         #  Name of input visibility file(s)
selectdata          =        True         #  Enable data selection parameters
     field          =          ''         #  field(s) to select
     spw            =          ''         #  spw(s)/channels to select
     timerange      =          ''         #  Range of time to select from data
     uvrange        =          ''         #  Select data within uvrange
     antenna        =          ''         #  Select data based on
                                           #   antenna/baseline
     scan           =          ''         #  Scan number range
     observation    =          ''         #  Observation ID range
     intent         =          ''         #  Scan Intent(s)

datacolumn          = 'corrected'         #  Data column to image(data,corrected)
imagename           =          ''         #  Pre-name of output images
imsize              =       [100]         #  Number of pixels
cell                = ['1arcsec']         #  Cell size
phasecenter         =          ''         #  Phase center of the image
stokes              =         'I'         #  Stokes Planes to make
projection          =       'SIN'         #  Coordinate projection (SIN, HPX)
startmodel          =          ''         #  Name of starting model image
specmode            =       'mfs'         #  Spectral definition mode
                                           #   (mfs,cube,cubedata)
     reffreq        =          ''         #  Reference frequency

gridder             = 'standard'          #  Gridding options (standard,
                                           #   wproject, widefield, mosaic,
                                           #   awproject)
     vptable        =          ''         #  Name of Voltage Pattern table
     pblimit        =         0.2         #  >PB gain level at which to cut off
                                           #   normalizations

deconvolver         =    'hogbom'         #  Minor cycle algorithm (hogbom,clark,
                                           #   multiscale,mtmfs,mem,clarkstokes)
restoration         =        True         #  Do restoration steps (or not)
     restoringbeam  =          []         #  Restoring beam shape to use. Default
                                           #   is the PSF main lobe
     pbcor          =       False         #  Apply PB correction on the output
                                           #   restored image

outlierfile         =          ''         #  Name of outlier-field image
                                           #   definitions
weighting           =   'natural'         #  Weighting scheme
                                           #   (natural,uniform,briggs)
     uvtaper        =          []         #  uv-taper on outer baselines in uv-
                                           #   plane

niter               =           0         #  Maximum number of iterations
usemask             =      'user'         #  Type of mask(s) for deconvolution
                                           #   (user, pb, auto-thresh, auto-
                                           #   thresh2, or auto-multithresh)
     mask           =          ''         #  Mask (a list of image name(s) or
                                           #   region file(s) or region string(s)
                                           #   )
     pbmask         =         0.0         #  primary beam mask

restart             =        True         #  True : Re-use existing images. False
                                           #   : Increment imagename
savemodel           =      'none'         #  Options to save model visibilities
                                           #   (none, virtual, modelcolumn)
calcres             =        True         #  Calculate initial residual image
calcpsf             =        True         #  Calculate PSF
parallel            =       False         #  Run major cycles in parallel
```

Setting up CLEAN in CASA

In CASA the task to use for CLEANing is **'tclean'**, which is the current state of the art CLEAN.

It is a refactored replacement to the task **'clean'** .

There are a *LOT* of parameters in this task.

At the beginner level the most important to define are:

1. The measurement set to use
2. The field you want to image
3. Which SPW to use
4. The size of the image you want to make (next slide)
5. The pixel (cell) size you want in your image (next slide)
6. The image weighting
7. The cleaning threshold value

Rules of thumb:

- Cell size = At least 3 pixels across the synthesised beam. Defined in arcsec.

$$cell \sim \frac{\lambda}{b_{max}} \times \frac{1}{n}$$ (where n ~ 3-6)

- Image size = Cover at least the primary beam/field of view. Defined in pixels of size=$cell$.

$$imsize \sim \frac{\lambda}{D} \times \frac{1}{cell}$$ (where D is the dish diameter)

- Your threshold should be roughly:

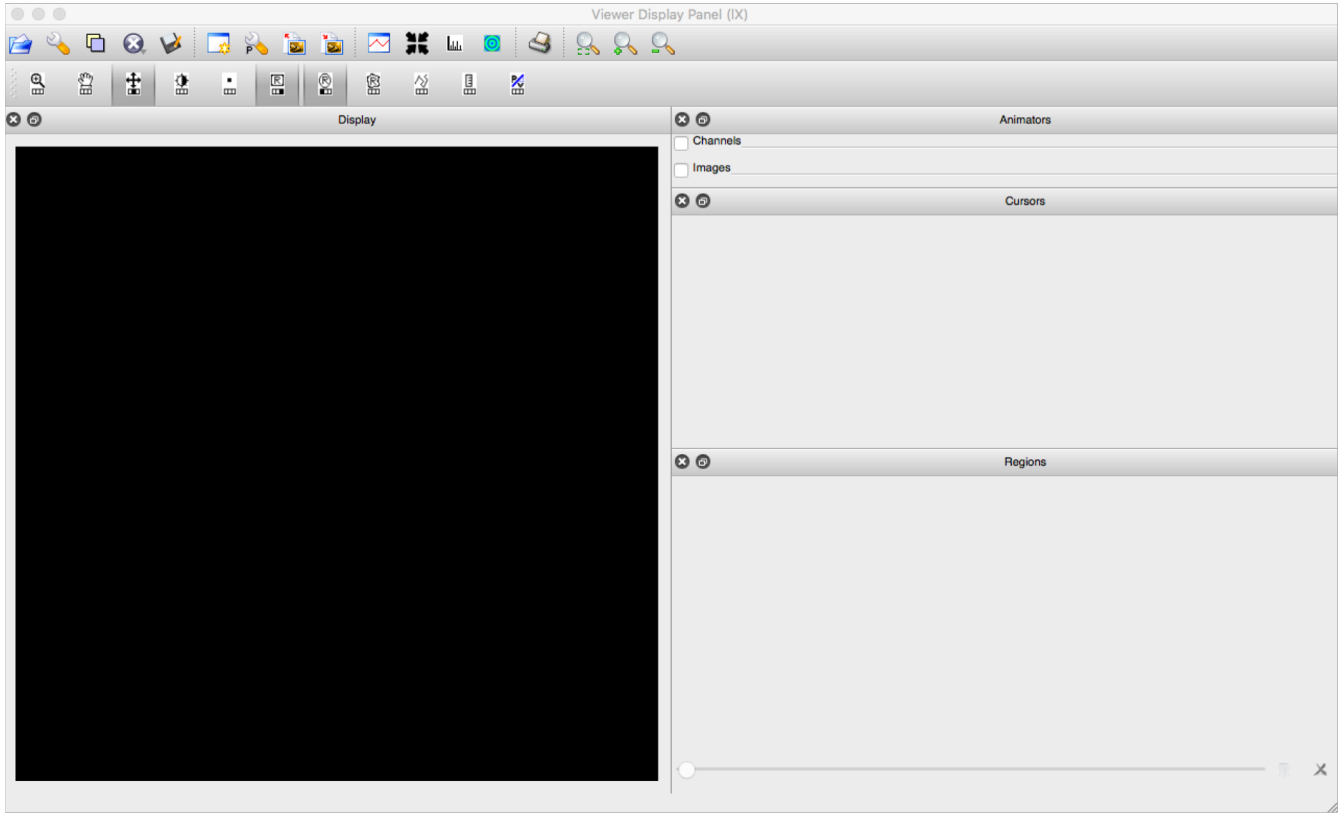
$$threshold \sim \frac{2kTsys}{A_e\sqrt{(N(N-1)\Delta\nu\Delta t)}}$$

The CASA viewer

- During the tutorials we have see the CASA viewer.

- Beyond simply allowing us to view images it can be used to preform image analysis.

- Can be started within CASA with the call `viewer()` or outside of CASA with `casaviewer` on the command line.



IMPORTANT INFORMATION:

In the future (potentially as early as December/January), a new “Viewer” known as CARTA will be released.

The bad news: Early versions of CARTA may not have all the functionality of Viewer.
(But the Viewer is going to stick around for a while!)

The good news: The early beta version of CARTA 1.0 we’ve seen are a significant improvement on Viewer in terms of speed, reliability and aesthetic.

CASA Tasks and Tools for image analysis

TASKS:
Front end, user friend command line functions for data reduction, manipulation in CASA.

Built upon the TOOL kit functions available in CASA.

Sometimes have a bit more functionality than functions available in GUIs e.g. viewer

vs

TOOLS:
'Under the hood' basic functions upon which tasks are built.

Preform simple tasks but can be useful in image manipulation.

Currently not the best documentation.

A non-exhaustive list of useful image analysis tasks available in CASA

immoments()
Compute moments from an image

imhead()
List, get and put image header parameters

imsubimage()
Create a (sub)image from a region of the image

specfit ()
Fit 1-dimensional gaussians and/or polynomial models to an image or image region

imfit()
Fit one or more elliptical Gaussian components on an image region

imval()
Get the data value(s) and/or mask value in an image

impv ()
Construct a position-velocity image by choosing two points in the direction plane

immath()
Perform mathematic operations on images

imstat()
Displays statistical information from an image or image region

Using the imaging toolkit functions

- The Imaging toolkit is more object oriented and Pythonic than using CASA tasks
- You will need to use multiple tools to achieve a single functions as you have to open and close the target image before doing anything to it.
- An example of a simple sequence of calls is given on the right.

```
CASA <2>: ia.open('myImage.image')
Out[2]: True

CASA <3>: ia.maxfit() #-- Find and fit max pixel in image
Out[3]:
{'component0': {'flux': {'error': array([ 0.,  0.,  0.,  0.]),
  'polarisation': 'Stokes',
  'unit': 'Jy',
  ... }}

CASA <4>: ia.close()
Out[4]: True

CASA <5>: ia.done()
Out[5]: True
```

A non-exhaustive list of useful image analysis toolkit functions available in CASA

ia.open()
Open a new image file with this image tool

ia.close()
Close the image tool

ia.done()
Destroy this image tool

ia.getchunk ()
Get the pixel values from a regular region of the image into an array

ia.coordsys()
Get the Coordinate System of the image

ia.findsources()
Find point sources in the sky

ia.maxfit ()
Find maximum and do parabolic fit in the sky

ia.newimagefromarray()
Construct a casa image from an array

ia.convolve2D()
Convolve image by a 2D kernel