Imaging ALMA data

Preparation + hands-on demo

Dan Walker

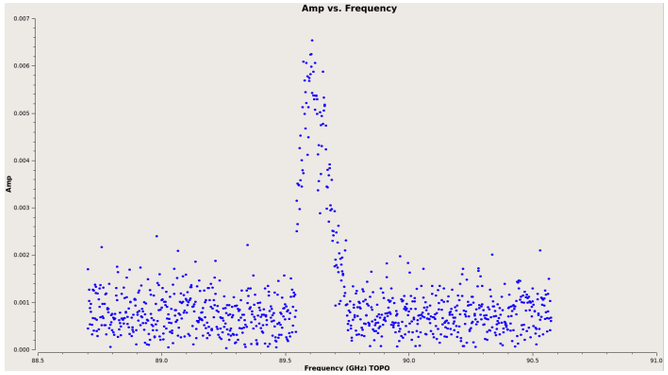
EUROPEAN ARC
ALMA Regional Centre || UK

MANCHESTER 1824

Spectral line imaging

- The process of cleaning line data is broadly similar to cleaning the continuum, but there are some extra steps and parameters:
    - We now have a third dimension (frequency/velocity) made up of channels, the spacing of which is related to the spectral resolution of the data
    - The emission changes from channel-to-channel, which makes cleaning and masking more complex
    - More data → longer processing time
    - The dust continuum level must be subtracted to ensure that we are imaging only the line emission

Splitting out the CO data



- For this particular dataset there are four SPWs, only one of which contains spectral line emission (recall this plot from earlier)

- This is CO (3-2) emission, which is contained in SPW 25

- To make the data more easily manageable, we can split out this SPW:

```
split(  vis                 = visfile,

        outputvis           = visfile+'.split',

        field               = 'PJ113921.7',

        spw                 = '25',

        datacolumn          = 'corrected')
```

Note: splitting out SPWs re-indexes them. In the output file there will only be one SPW, indexed as ‘0’

If CASA complains about having no CORRECTED datacolumn, change to ‘data’

Continuum subtraction

- Having split out the CO SPW, we need to subtract the continuum emission. This can be done after imaging, but it's generally recommended to do this beforehand.

- Use CASA task `uvcontsub` to subtract the continuum from the *uv* data

- contchans_CO = '0:88.72~89.46GHz;89.86~90.54GHz'

```
uvcontsub(  vis              = visfile+'.split',

            outputvis        = blah,

            field            = 'PJ113921.7',

            spw              = '0',

            fitspec          = contchans_CO,

            fitorder         = 1)
```

CO cube imaging

- Now we can start imaging the CO emission! Let's start by making a dirty image (0 clean iterations), just like we did for the continuum

```
tclean(  vis             = visfile+'.split.contsub',
         imagename       = 'PJ113921.7.CO.cube.dirty',
         spw             = '0',
         restfreq        = '345.79599GHz',
         specmode        = 'cube',
         imsize          = [1250, 1250],
         cell            = '0.09arcsec',
         deconvolver     = 'hogbom',
         niter           = 0,
         weighting       = 'briggsbwtaper',
         robust          = 0.5,
         gridder         = 'standard',
         interactive     = False)
```

CO cube imaging

- As for the continuum, we see that the emission is concentrated to the inner portion of the field, so we can crop the image when cleaning
- We also see that the line emission only appears in a small portion of the channels, so we can restrict the imaging to just these channels
- Once again, we want to increase the number of clean iterations, add a cleaning threshold, and add auto masking parameters …

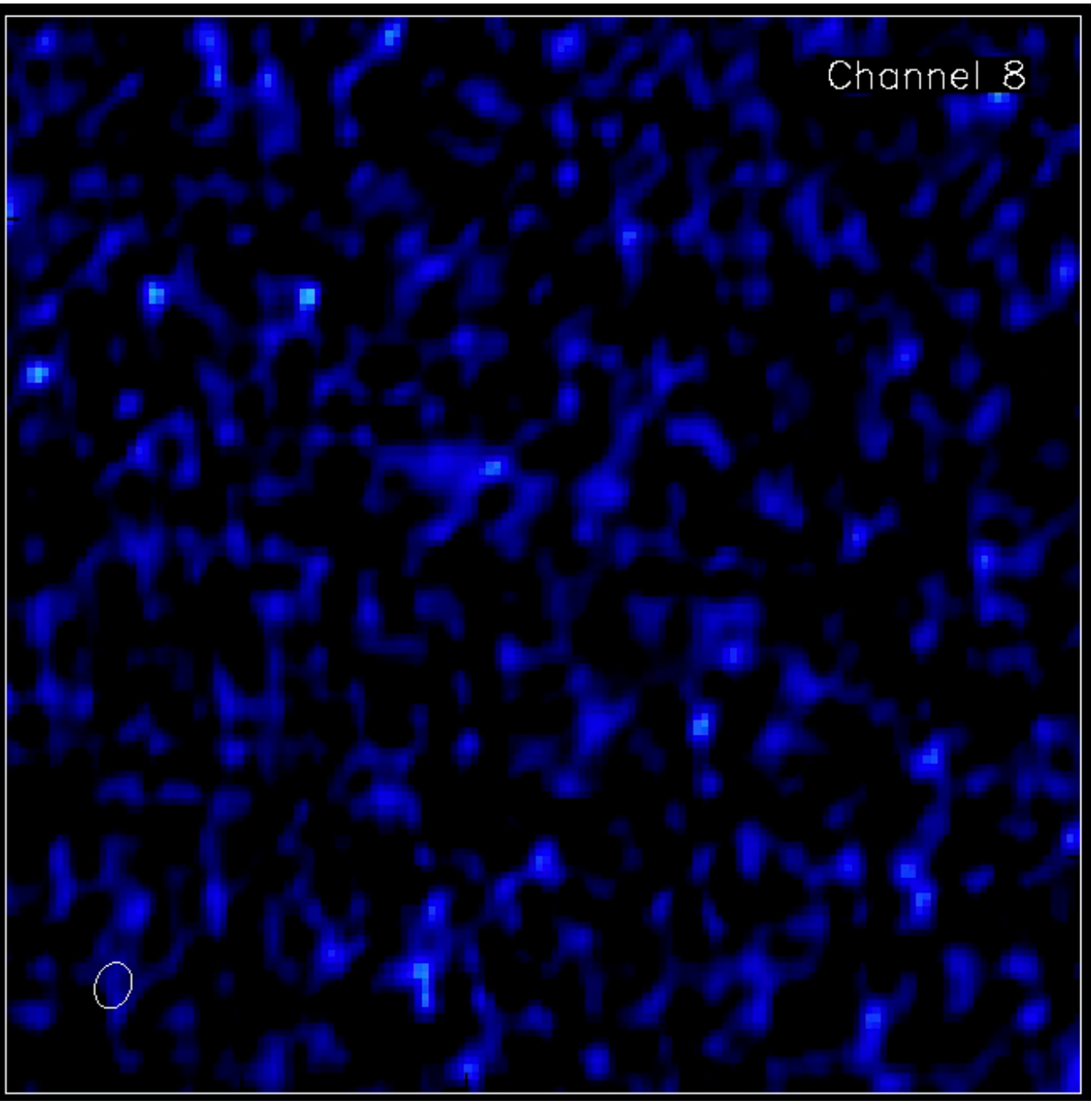
CO cube imaging

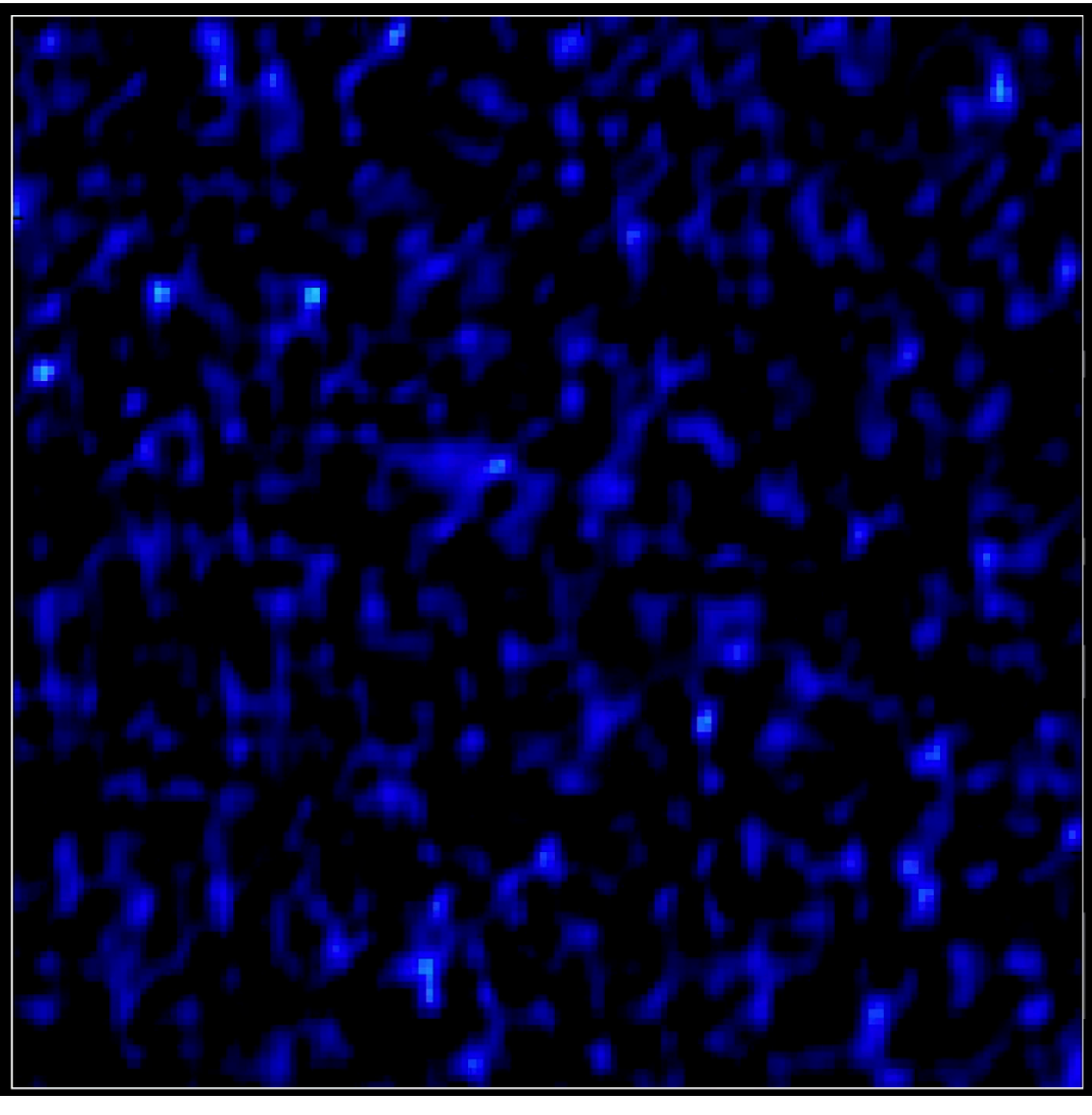
```
tclean(  vis                 = visfile+'.split.contsub',
         imagename           = 'PJ113921.7.CO.cube',
         phasecenter         = 'ICRS 11:39:21.728 20.24.51.838',
         spw                 = '0',
         restfreq            = '345.79599GHz',
         specmode            = 'cube',
         imsize              = [150, 150],
         cell                = '0.09arcsec',
         deconvolver         = 'hogbom',
         niter               = 1000000,
         weighting           = 'briggsbwtaper',
         robust              = 0.5,
         pbcor               = True,
         threshold           = '0.001Jy',
         nchan               = 100,
         start               = 200,
         width               = 1,
         interactive         = False,
         usemask             = 'auto-multithresh',
         sidelobethreshold   = 2.0,
         noisethreshold      = 4.0,
         lownoisethreshold   = 2.5)
```

CO cube imaging

Image

Residual



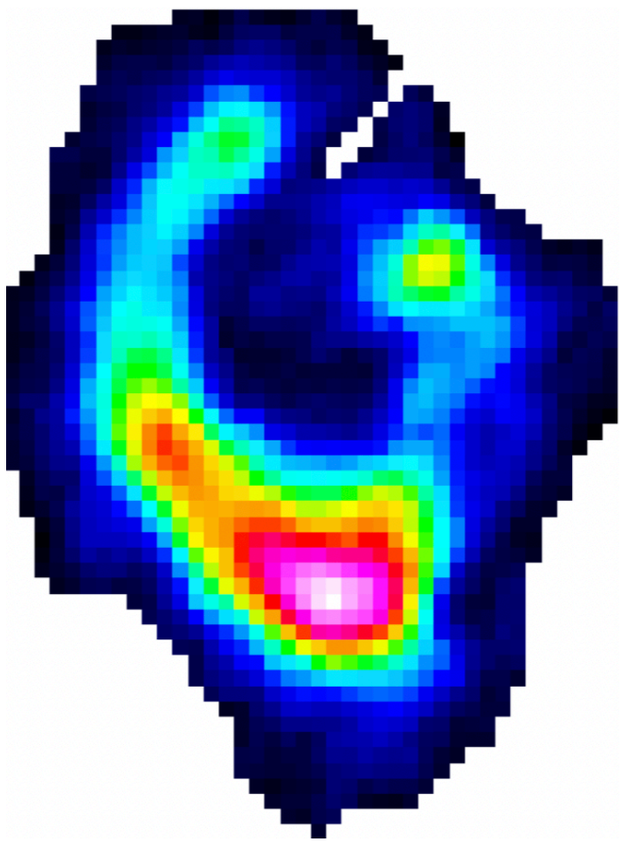


Moment maps

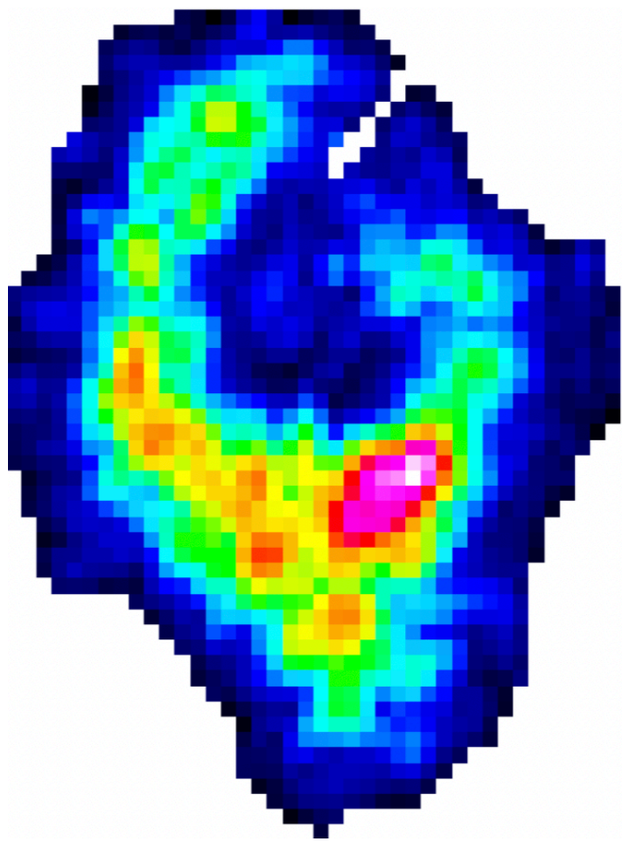
See documentation for full definitions

```
immoments(imagename     = 'PJ113921.7.CO.cube.image',
          moments       = [0, 1, 8],
          region        = 'moment_region.crtf',
          chans         = '10~65',
          includepix    = [0.0007, 100],
          outfile       = 'PJ113921.7.CO.cube.moment')
```

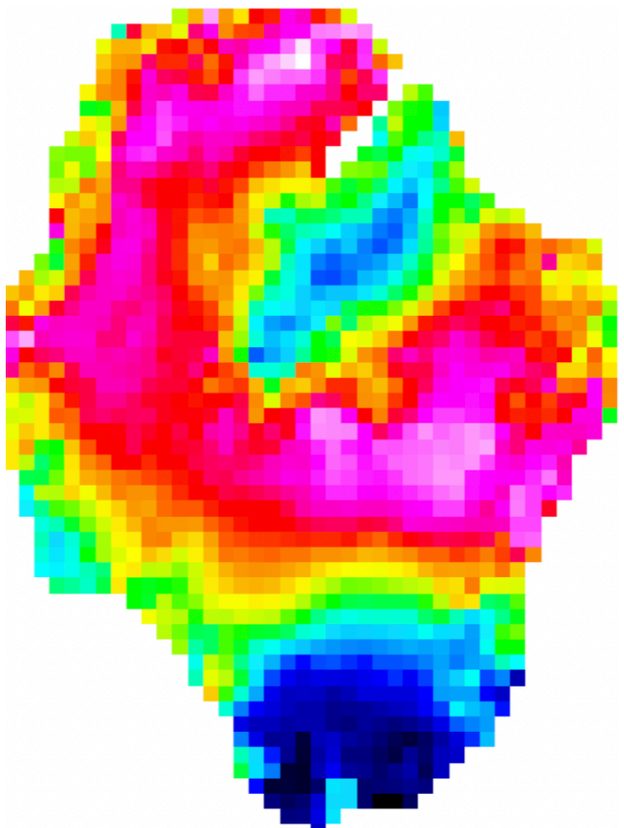
Integrated intensity (moment 0)



Maximum intensity (moment 8)



Velocity field (moment 1)



Alternative cube analysis tools

Spectral Cube (Python)

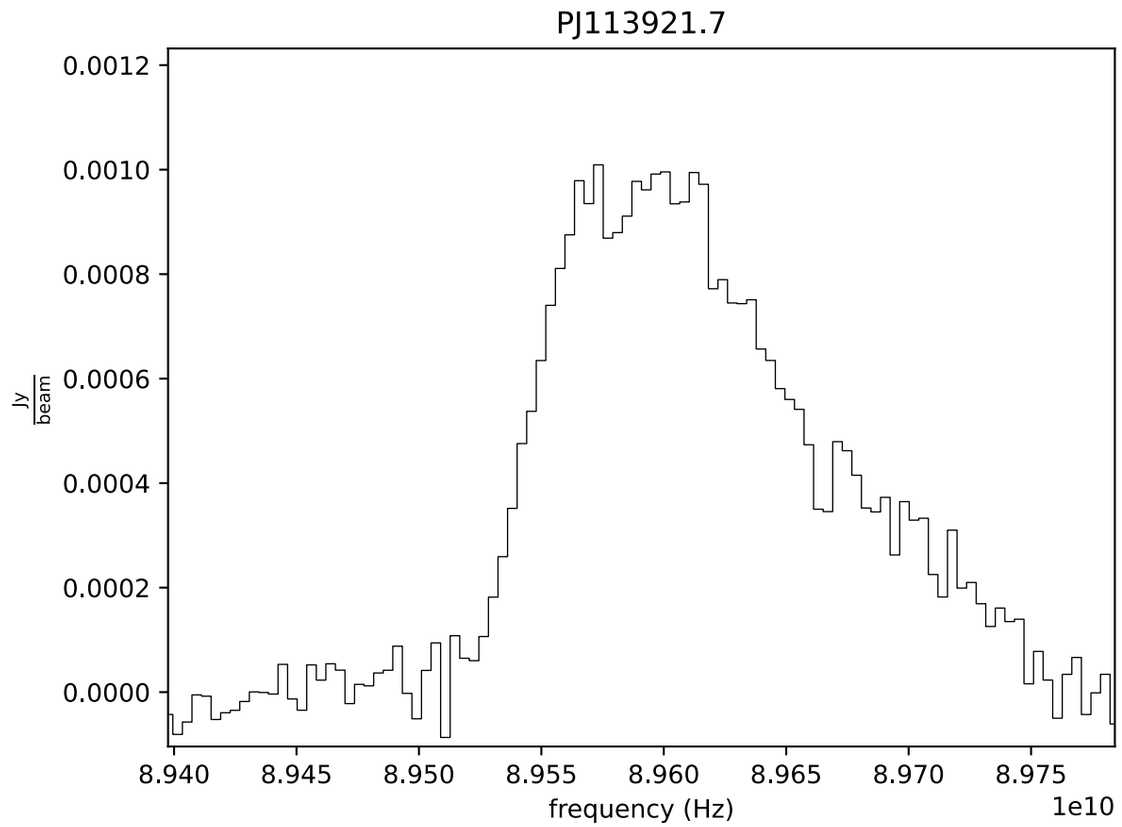
- Toolkit for reading, writing, manipulating, and analysing spectral cube data
- Create sub-cubes, moments, extract spectra etc.
- Designed to work with very large cubes that are too large to load into memory

Pyspeckit (Python)

- Analysis toolkit for analysing spectra
- Plotting, line fitting, line modelling, and more

Alternative cube analysis tools

```
import pyspeckit

import matplotlib.pyplot as plt

from spectral_cube import SpectralCube



filename = 'PJ113921.7.CO.cube.image.fits'

cube     = SpectralCube.read(filename)

meanspec = cube.mean(axis=(1,2))

meanspec.write(filename.replace('.fits', '.meanspec.fits'))



sp = pyspeckit.Spectrum(filename.replace('.fits', '.meanspec.fits'))

sp.plotter(color = 'k', linestyle='-')

sp.plotter.savefig(filename.replace('.fits', '.meanspec.png'))
```



Parallel processing (Linux only!)

You can run `tclean` in parallel across multiple cores in order to distribute the processing and speed things up:

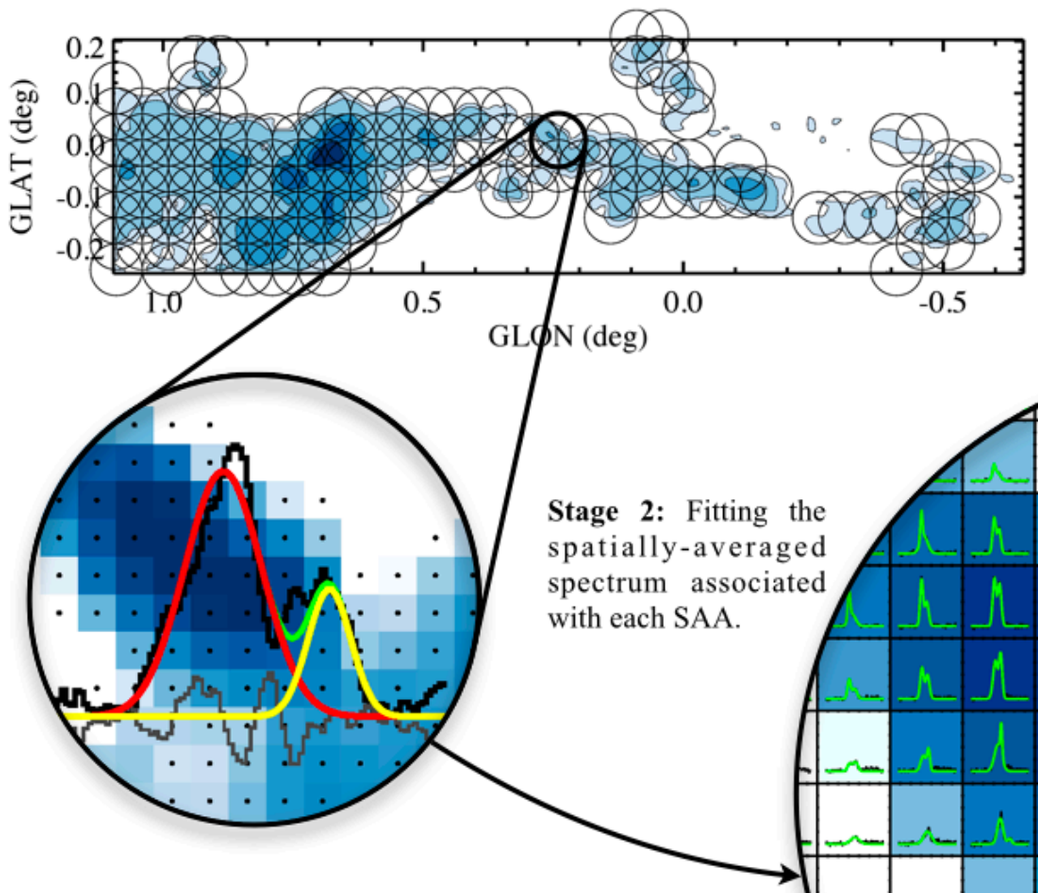
- In `tclean`, specify the parameter `parallel=True`
- Place your `tclean` command in a `.py` script
- Run your script via:

Number of cores

```
/path_to_casa/bin/mpicasa -n 8 /path_to_casa/bin/casa --
nologger -c ./imaging_script.py
```

[You can also place the above command into a .sh script and execute it in the background]

SCOUSE(py): Semi-automated multi-COmponent Universal Spectral-line fitting Engine



Stage 1: Identify the spatial area over which to implement SCOUSE - A grid of Spectral Averaging Areas (SAAs; black circles where 50% of the enclosed positions have a non-zero integrated intensity).

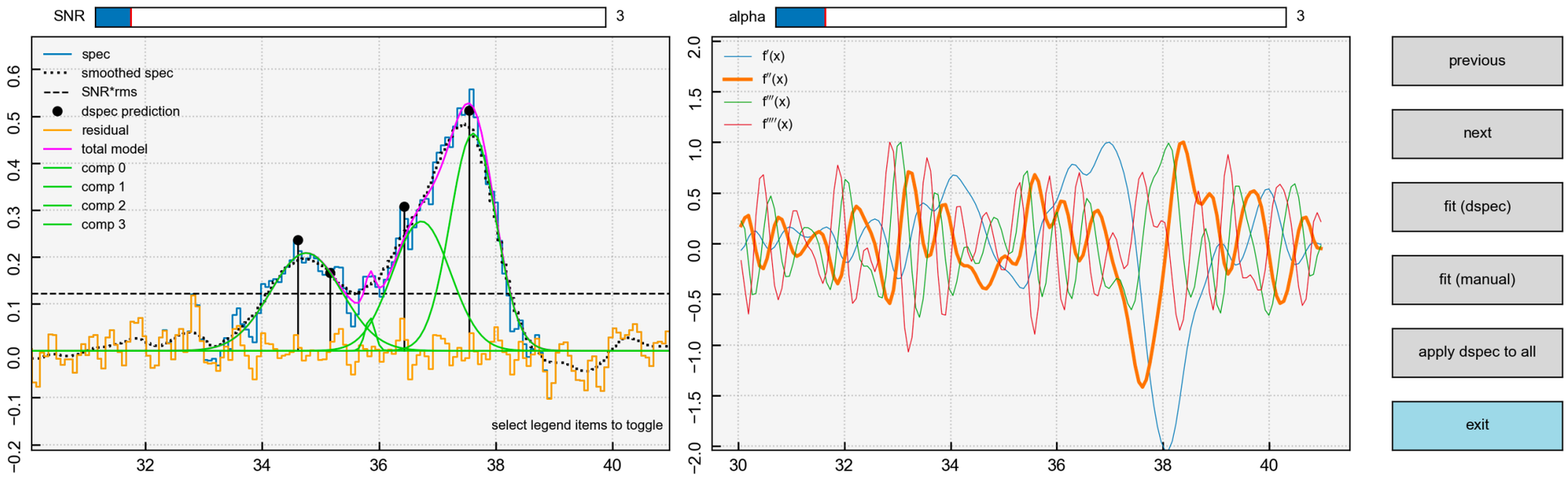
Stage 2: Fitting the spatially-averaged spectrum associated with each SAA.

Stages 3 & 4: Fitting the individual spectra using output parameters from stage 2 as free-parameter inputs, and selecting the "best-fits" to each spectrum.

Available for download: **https://scousepy.readthedocs.io/en/latest/**

For each averaged spectrum, scousepy provides an initial fit, which the user cycles through an interactive GUI to either accept or update the fit.

scousepy will then enter the fully automated stage, where it will take these averaged fits, and pass the parameters to fit the spectrum at every single pixel within each averaging area.



derivative spectroscopy information:
SNR: 3
alpha size: 3
pyspeckit fit information: **Fit has converged...**
number of components: 4
amplitude: 0.21 +/- 0.01; 0.07 +/- 0.04; 0.28 +/- 0.05; 0.46 +/- 0.12
shift: 34.75 +/- 0.05; 35.85 +/- 0.04; 36.73 +/- 0.24; 37.61 +/- 0.1
width: 0.61 +/- 0.06; 0.08 +/- 0.05; 0.5 +/- 0.18; 0.39 +/- 0.04
goodness of fit information:
chisq: 118.95
red chisq: 0.82
AIC: -1022.81

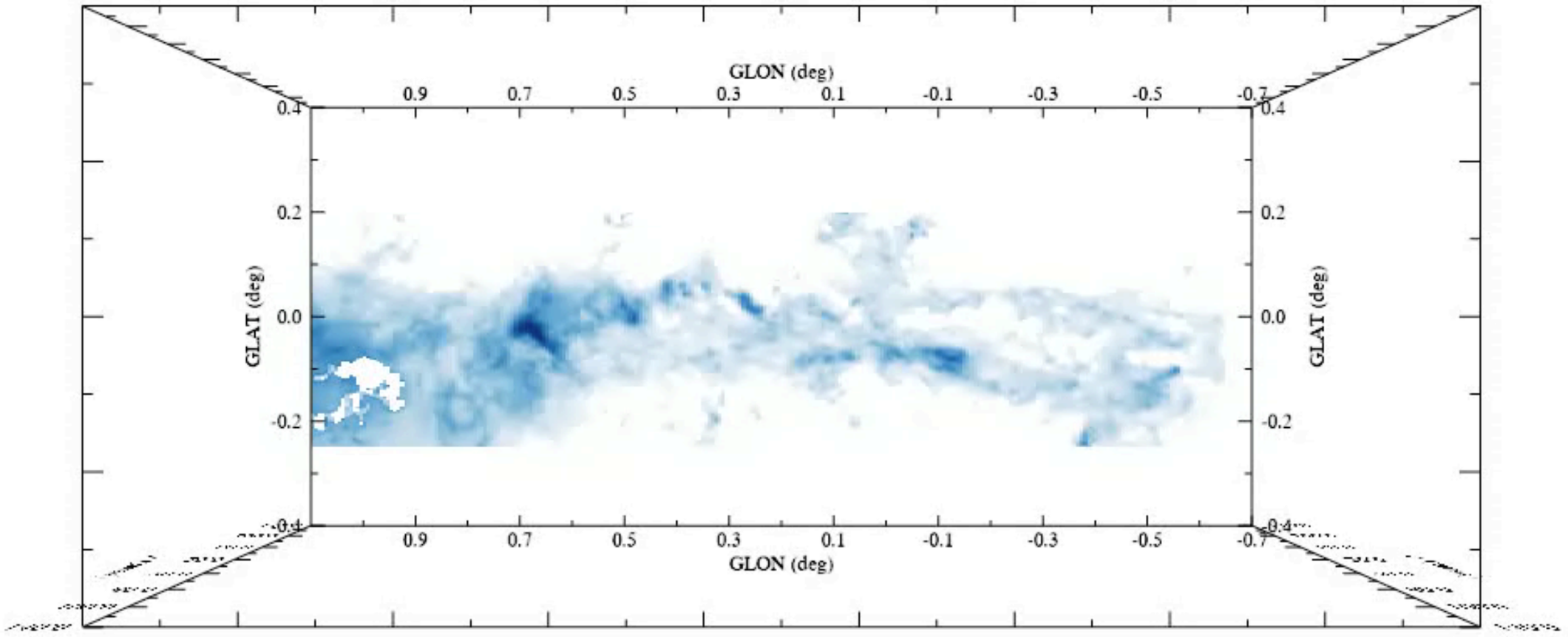
<< < 1 of 24 > >>

Once the automated fitting has completed, it will output an ascii file containing many measured parameters per pixel, e.g.:

- Number of components at that pixel location
- x
- y
- Amplitude
- Centroid velocity
- Velocity dispersion
- RMS

You can plot these data in various ways to make some cool plots …

SCOUSEpy …spectral line fitting of the inner ~250 pc of the Milky Way **Henshaw et al. 2016a**



“Large-scale” perspective; “Small-scale” detail

Henshaw et al. 2019

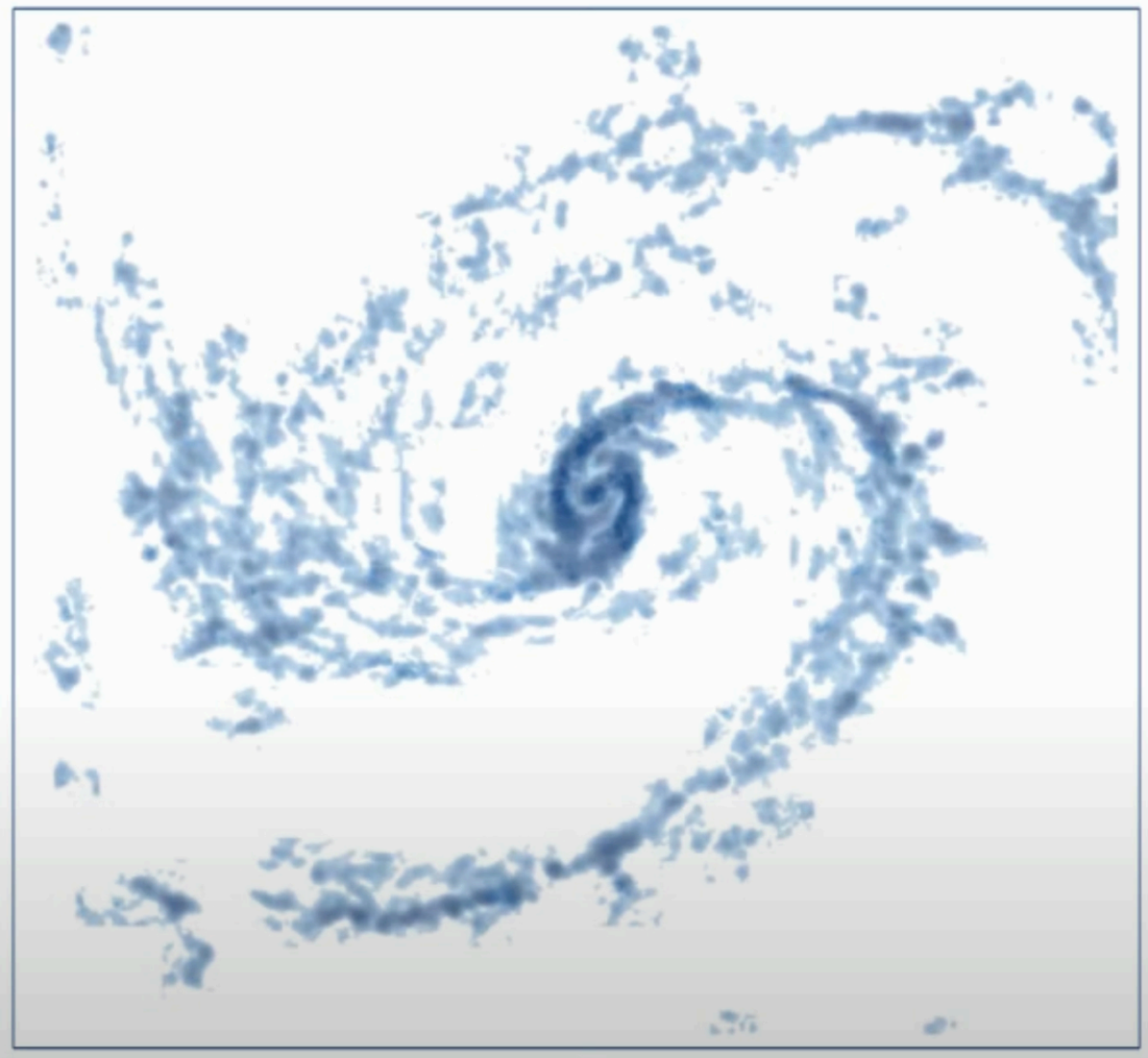
Ubiquitous Velocity Fluctuations throughout the Molecular Interstellar Medium

G0.253+0.016 a.k.a. "The Brick"

Jonathan D. Henshaw et al.

Henshaw et al. 2020

NGC 4321



as viewed on the plane of the sky (position-position view)

You can find a tutorial here if you fancy trying it yourself:

https://scousepy.readthedocs.io/en/latest/tutorial_v2.0.0.html