Data Comb 2019: M100

Peter Teuben (University of Maryland, USA)
David Rebolledo (Joint ALMA Observatory, Chile; National Radio Astronomy Observatory, USA)
Jens Kauffmann (MIT Haystack, USA)
Atefeh Aghababaei (University of Cologne, Germany)
Quentin Salomé (IRyA-UNAM, Mexico)
Toshiki Saito (MPIA-Heidelberg, Germany)

September 17, 2019

This memo was prepared as part of the workshop "Improving Image Fidelity on Astronomical Data: Radio Interferometer and Single-Dish Data Combination," held on 12-16 Aug 2019 at the Lorentz Center in Leiden, The Netherlands.

1 Dataset overview

The M100 group used the data from the *CASA Guide "M100 Band3 Combine 5.4"*. The data are all calibrated, but the interferometric data comes in around 4000 channels, whereas the total power in 70 channel. This will need a 26 GB download, or one can take the QAC benchmark data (5 km/s gridded between 1400 and 1745 km/s), which is a mere 105 MB and all datasets are now on a common spectral axis (70 channels). A script is available that shows how to trim this data. Noteworthy here is that the MS and TP data have their spectral axes in opposite directions, which needs to be corrected in order for SDINT to combine properly. The TP data is already ordered in increasing velocity, but the MS data cannot be reverted due to a bug in `mstransform()`. See `https://open-jira.nrao.edu/browse/CASR-57` for those who have access. Hence we reversed the order of the TP data using `imtrans()`. Sadly this resulted in the first or last channel to not properly combine..

Important observing parameters are in Table 1. These are the data that also form the basis for the M100 CASA Guide, `https://casaguides.nrao.edu/index.php/M100_Band3`. A further guide at `https://casaguides.nrao.edu/index.php/M100_Band3_Combine_5.4` describes the combination of 12m data, ACA observations, and TP data via feathering in CASA 5.4. The data were downloaded using links summarized in Sec 1 of `https://github.com/teuben/dc2019/blob/master/data/README_DC2019_data`.

2 Combination Methods

2.1 M100 benchmark

Since the M100 data have also been prepared in a small 2 minute benchmark form (the 5 km/s 70-channel dataset) via QAC, the first thing we did is confirm we all get the same flux values on the benchmark. Different CASA versions will get slightly different answers, perhaps even between Linux and Mac. See table below.

The benchmark can be run with QAC installed as follows:

```
cd QAC/bench
make bench
```

Table 1: Observational details

Parameter	Value
Object	M100 (NGC 4321)
Phase center	J2000 12h22m54.900s +15d49m15.000s
Rest frequency	115.271202GHz (*)
V_{LSR}	1575 km/s
ΔV (channel width)	-5 km/s
Velocity range imaged	[1745-1400] km/s
Map size	400 × 400 ″
12m-array pointings	47
7m-array pointings	23
12m-array beamsize	4.0 × 2.6 ″ (PA=-88°)
7m-array beamsize	14 × 11 ″ (PA=-86°)
Range of 12m baselines	[15 - 453] check-m
Range of 7m baselines	[9 - 49] check-m

where `QAC_STATS` lines report the *mean,dispersion,min,max,flux* for the listed map. The benchmark reports this for

```
bench/clean/tpint_2.tweak.image
0.004743185253331385 0.02181014437004293 -0.04345422238111496 0.42146903276443481
    ↪ 473.98086630998699
```

thus a total flux of 474 Jy km/s is found for CASA 5.6 in these 5 channels. We did not always exactly reproduce these values, as different CASA versions and tinkering with the benchmark data made this a futile exercise. Besides, we had better things to do.

2.2 Method 1: Feather

2.2.1 Basic run of Feather

We started by running the standard Feather procedure detailed in the CASA Guide M100 at `https://casaguides.nrao.edu/index.php/M100_Band3_Combine_5.4`. The executable script can be found at the DC2019 workshop website at `https://github.com/teuben/dc2019/blob/master/scripts/M100_combine1.py`. This script produces all the plots that are on the casaguide page, and some more.

This script takes the raw 12 m and 7 m data, and selects the right spectral windows where CO was observed. Figure 1 shows the pointing grids for both 12 m and 7 m observations.

When combining the 12 m and the 7 m, the weighting of each data set should be checked. The weighting is proportional to $1/\sigma^2$, and σ depends on, among other factors, the integration time and the effective dish area. For M100 data, the weighting ratio between 7 m and 12 m data should be ~ 0.19. Figure shows the weights after concatenation of the 12m and the 7 m is done. The mean value observed in Figure 2 is roughly consistent with the theoretical value of ~ 0.19.

Once both 12 m and 7 m datasets are combined, it is possible to run diagnostic plots to check that everything is in order. By looking at the amplitude vs u-v distance plot for example (Figure 3 left panel), we can see that the 7 m data is much noisier than the 12 m data. If we only consider the brightest channels, then we see a better correspondence in the amplitude vs. velocity plot (Figure 3 right panel).

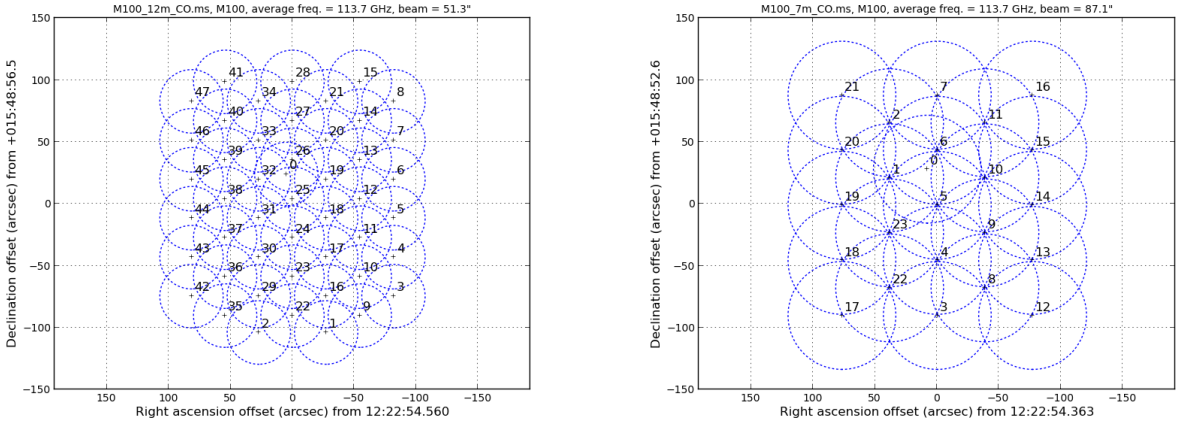

Figure 1: Pointings grids for the 12 m (Left) and 7 m (Right) mosaics.

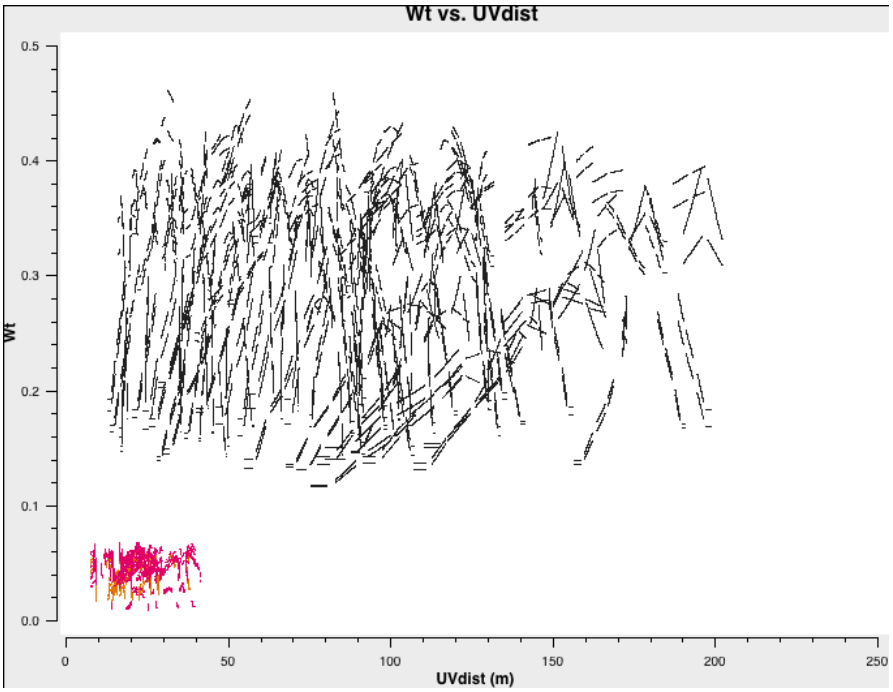

Figure 2: Weights of the 7 m and the 12 m after both datasets are concatenated.

2.2.2 Playing with `sdfactor` and `lowpassfiltersd` parameters

Here we show the impact of `sdfactor` and `lowpassfiltersd` parameters on the resultant feathered image. The defaults of these two parameters are `1.0` and `False` resp. Total flux values of all the tested combinations are listed in Table 1. Figure 5 shows 12m+7m+TP feathered moment-0 maps for M100 with different `sdfactor` from 0.1 to 1.5 and `lowpassfiltersd`=True. More fluffy, extended structures get emphasized as the `sdfactor` value increases. This is an expected trend since `sdfactor` modifies the flux scale of the `lowres` image (TP image in this case). This trend is clearly seen in the case with `lowpassfiltersd`=False. The total flux of the feathered image is consistent with that of the original TP image when we apply (1) `sdfactor`=1.0 and `lowpassfiltersd`=False or (2) `sdfactor`=1.2 and `lowpassfiltersd`=True.

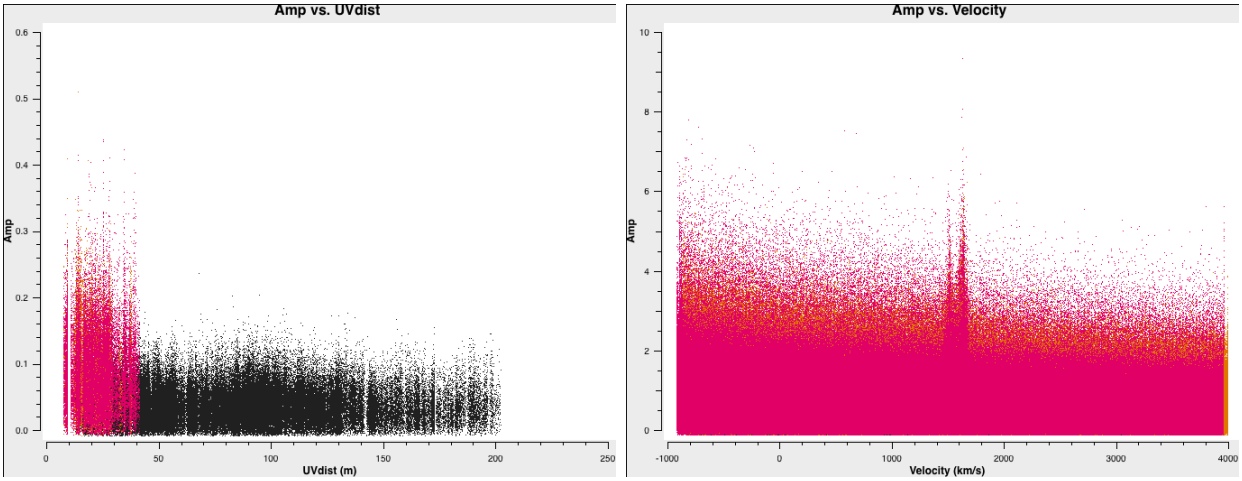

Figure 3: Left: Amplitude vs. u-v distance for 12 m and 7 m data. Right: Amplitude vs. velocity considering bright channels.

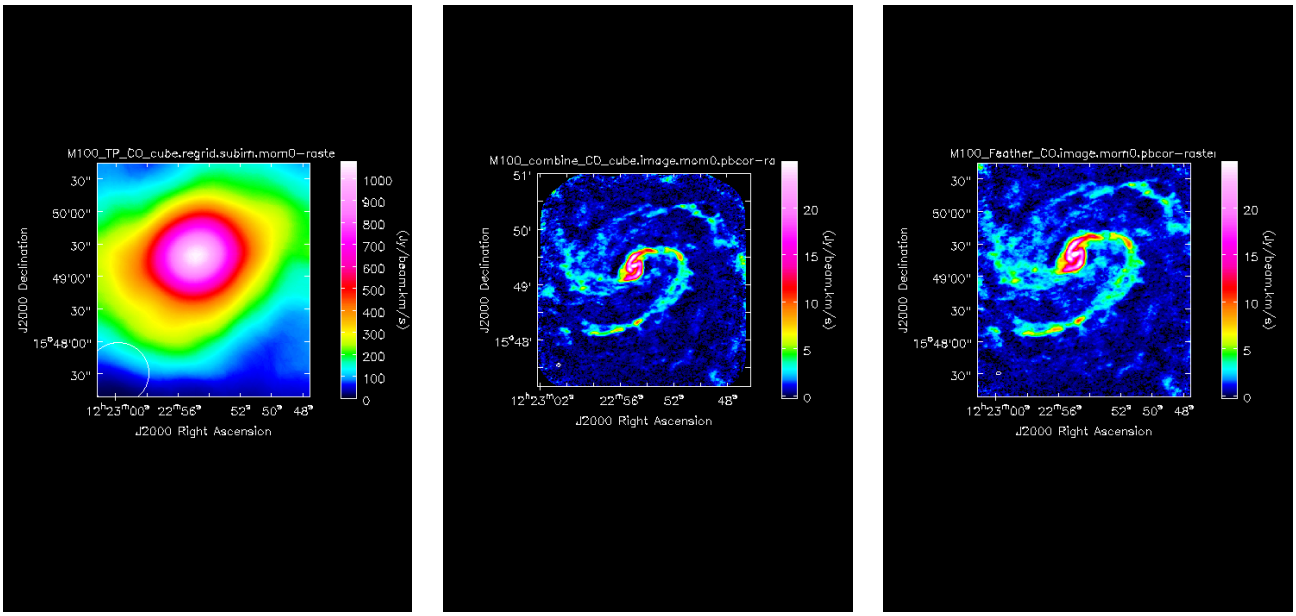

Figure 4: Image comparison between TP, 7m+12m and feathered.

Figure 6 shows a comparison between two moment-0 maps with `lowpassfiltersd`=True and False as well as a flux difference of them. We see fluffy structures around the north-eastern arm are more emphasized in the image with `lowpassfiltersd`=True. However, if one takes pixel-by-pixel flux differences as shown in the right panel of Figure 6, one can immediately realize that `lowpassfiltersd`=True does not only add extended emission at the north-eastern part, but also suppresses extended emission around the center and the north-western spiral arm at the same time.

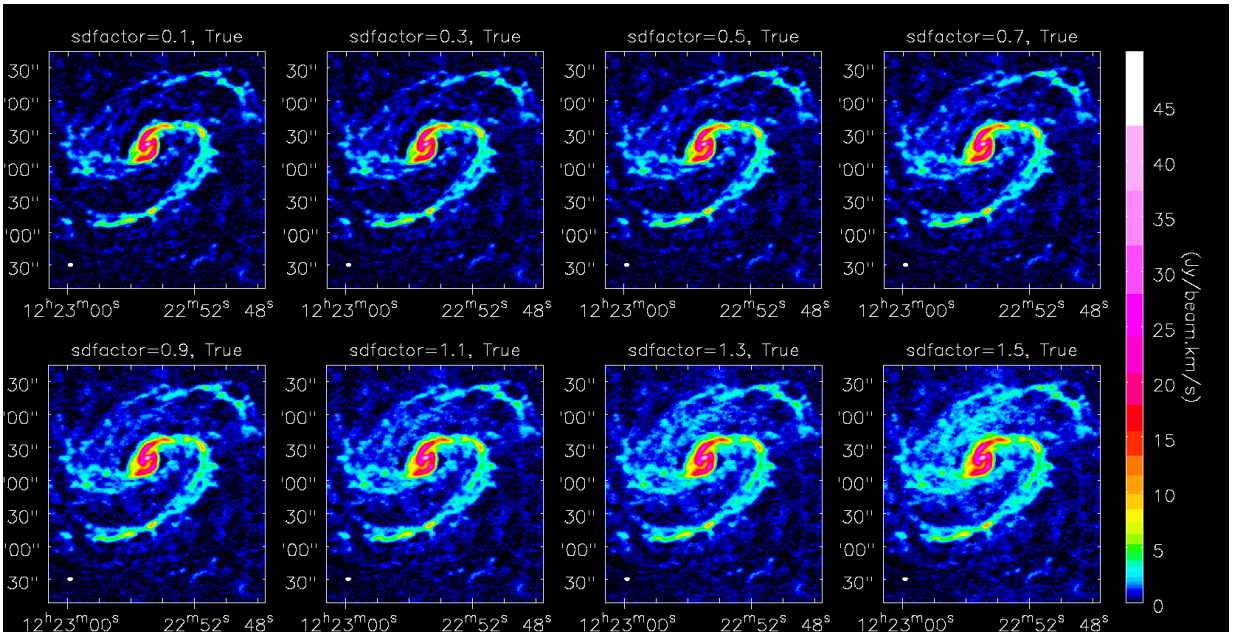

Figure 5: Feathered 12m+7m+TP Moment-0 maps for M100 with varying `sdfactor` from 0.1 to 1.5 (step 0.2). **[PT: Units are Jy.km/s.]**

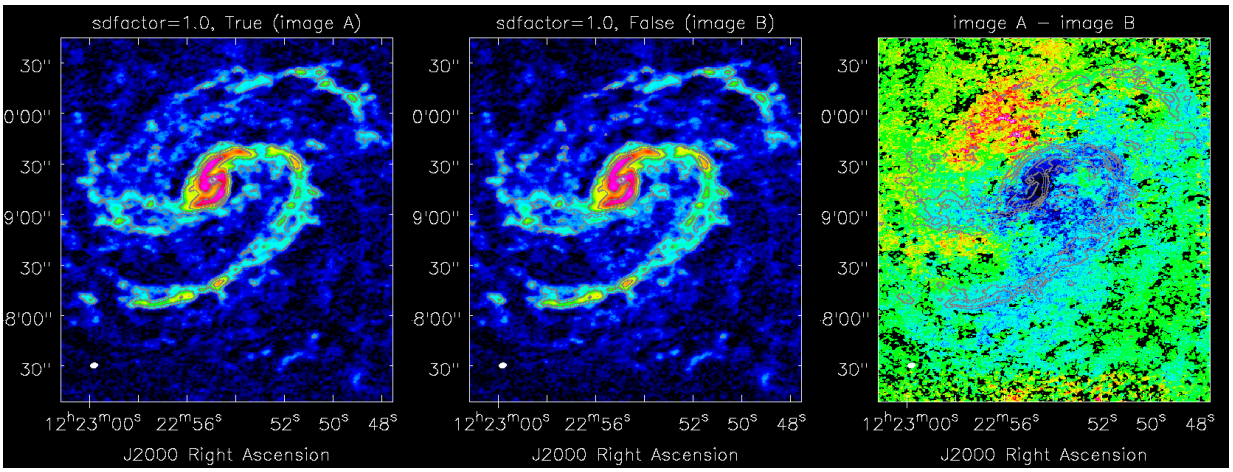

Figure 6: Feathered 12m+7m+TP Moment-0 maps for M100 with lowpassfiltersd=True (left) and lowpassfiltersd=False. The sdfactor is 1.0. The difference between them is shown in the right plot. The color scale is ranging from 1.5 to -1.5 Jy beam^{-1} km s^{-1}. **[TS: add color bars.]** **[PT: isn't the unit Jy.km/s ?]**

2.3 Method 2: Joint Deconvolution (tp2vis)

2.3.1 Presentation of the different steps

In this section, we followed the different steps of the online guide[1]. We worked with the benchmark data, of which details are given in table 1.

As a first step, we combine the ALMA-12m and ACA-7m data. To do so, we first produce a "dirty" image using `niter=0` to estimate the noise level. The noise level is then used as a threshold to clean

[1] `https://github.com/tp2vis/distribute/blob/master/example1.md`

Table 2: Image quality

`sdfactor`	Total flux `lowpassfiltersd`=True (Jy km s^{-1})	Total flux `lowpassfiltersd`=False [default] (Jy km s^{-1})
0.1	2095	2106
0.3	2230	2267
0.5	2383	2448
0.7	2559	2651
0.9	2764	2882
1.0 [default]	2880	3009
1.1	3007	3145
1.3	3294	3449
1.5	3632	3794

with a larger number of iterations (here `niter=10000`). This ALMA+ACA cube will be used to highlight the effect of spatial filtering by the interferometers.

We then produce the pseudo-visibility of the Total Power. The image datacube is converted into a MS table with `tp2vis`. We produced two MS tables, tapering off the edge emissions or not, in order to compare the two results.

Finally, we combine the ALMA-12m, ACA-7m and TP data using the same process as to combine the ALMA-12m and ACA-7m data. Note that, `tclean` requires the MS tables to be concatanated first with `concat`. For the ALMA+ACA+TP combination, we used `niter=1e6` in order to clean deep enough.

We compared the result of the ALMA+ACA+TP combination with the TP image used to produce the pseudo-visibility. When looking at the mean value per channel, it appears that tapering off the edge emission of the TP (using the parameter `winpix` within `tp2vis`) when producing the pseudo-visibility is required to accurately recover the profile (figure 7). However, the flux is higher higher by a factor about 1.5 (figure 8).

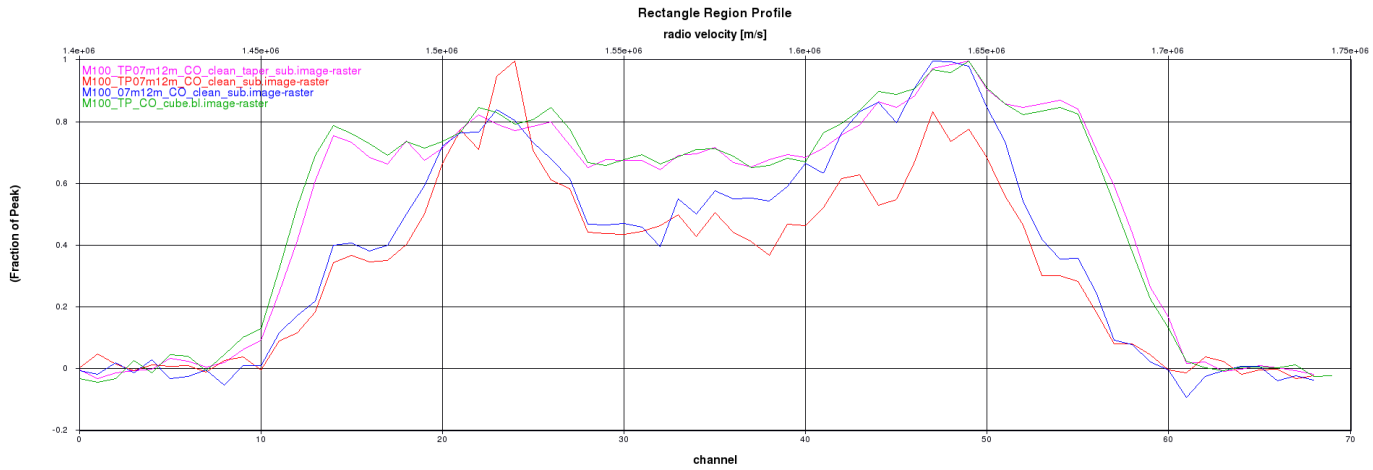

Figure 7: Profile of the mean value per channel for the TP image (green), ALMA+ACA (blue), ALMA+ACA+TP (red) and ALMA+ACA+TP with tapering (purple).

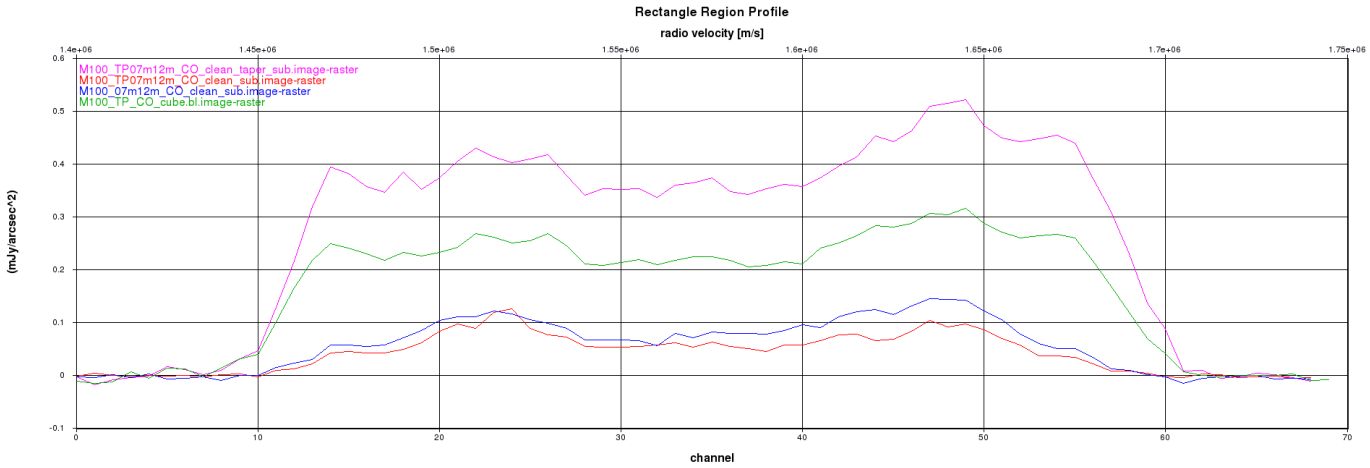

Figure 8: Mean flux per channel for the TP image (green), ALMA+ACA (blue), ALMA+ACA+TP (red) and ALMA+ACA+TP with tapering (purple).

2.3.2 Flux recovering for the Total Power

As presented above, the total flux within the ALMA+ACA+TP data is much higher than the total flux in the TP image. We hypothesised that such difference may come from issues in the `tp2vis` routines. To test this hypothesis, we apply the following method:

```
run tp2vis on the TP image
clean the pseudo-visibility
compare the total flux
```

The original TP image is 110×110 pixels so we first produced a clean image of the pseudo-visibility with a size of 120×120 pixels. The resulting image presents some elliptical structures on the edge (figure 9), which are likely produced by the Fourier transform which produce the image.

We therefore increased the number of pixels for the clean image. It appears that imaging within CASA is more stable when using a map size of 2^n. Using 256^2 pixels results in the same results therefore we produced clean images of 512^2 and 1024^2 pixels. The moment 0 map shows more or less the same structure, while increasing the map size makes appear an extented structure in the East (figure 10).

The map size as an visible effect on the total flux that is recovered (11). Indeed, when using a map size similar to the size of the original map, the flux is far from being recover, likely due to the elliptical structures produced by the Fourier transform. On the opposite, it appears that a map size which is too large tends to provide a total flux higher than in the original image. A map size of 512^2 pixels seems to be the best case.

In the previous section, we concluded that, while the total flux is not properly recovered, the spectral profile is well recovered when we taper off the edge of the TP image while producing the pseudo-visibility (with the parameter `winpix` in `tp2vis`). We here want to see the effect of such tapering on the total flux recovering for the TP flux. Figure 12 shows that there is no clear difference with and without tapering.

[PT: I'm also reaching a conclusion, based on just cleaning the TP.MS, that using cycleniter=10 or 100 improves the flux convergence a lot. I suspect this is due to that we do not have a good enough definition of the VP for the TP.]

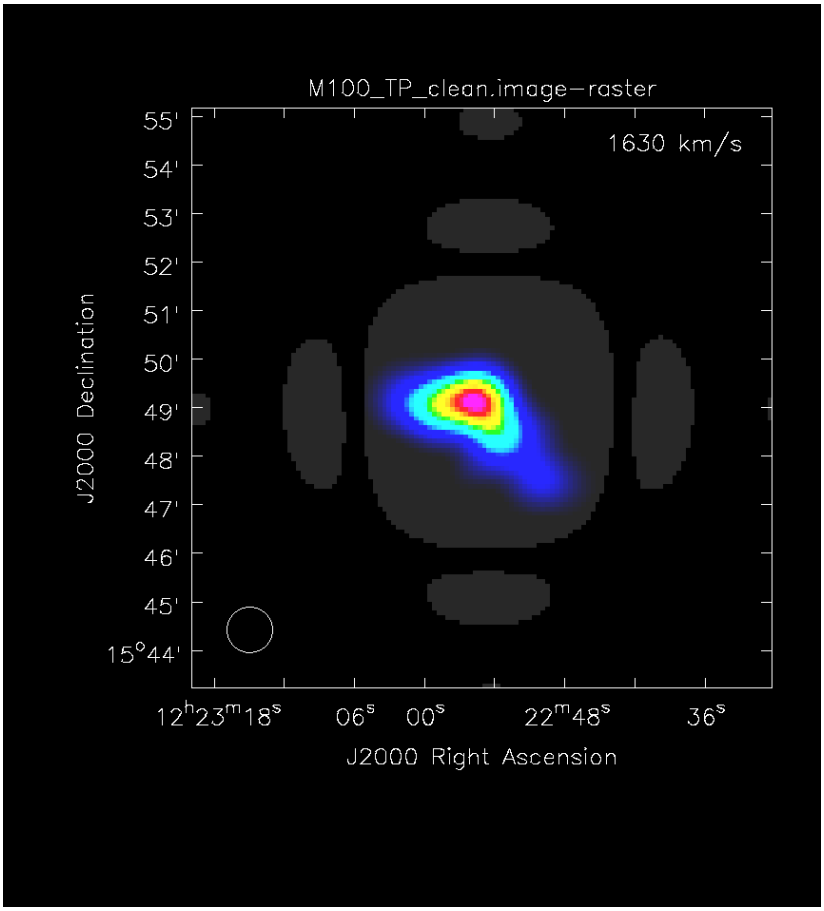

Figure 9: One channel map of the TP clean image with a size of 120×120 pixels. We clearly see elliptical images which are produced by the Fourier transform.

2.3.3 What is the right weight?

The crucial question with tp2vis is to set the right weight. Secondary is to tweak or not to tweak. If you set niter high enough, in theory the residuals go down to 0, and tweaking is not needed. In practice the clean procedure can be unstable, and tweak maps *should* converge faster. So we took an experiment (combine4a) where we used a series of niter's and weight's to cover the range from `beammatch` to `rms` to study this convergence. This is summarized in Figure 13. For a normal tclean niter=300,000 is beginning to show some striping problems , but adding threshold = 0.011 and nsigma=1.0 seems to control that better but give for slower convergence. We are clearly stepping in the tclean-tinkering domain, for which we officially don't have time.

2.4 Method 3 Joint Deconvolution (sdint)

We were able to get SDINT (Rau's method) to work as well, although the interface requires one to provide MS and TP to have their spectral axes. We used imtrans() to flip the spectral axis. See our `M100_combine3.py` script.

3 Comparison and evaluation

3.1 Notes

We have not compiled across people in this team yet. But when comparing with the casaguide Feather example, keep in mind their tclean only used niter=10000, which still has some clear structures in the residual map. Also their tclean used the new automasking technique. One could argue that

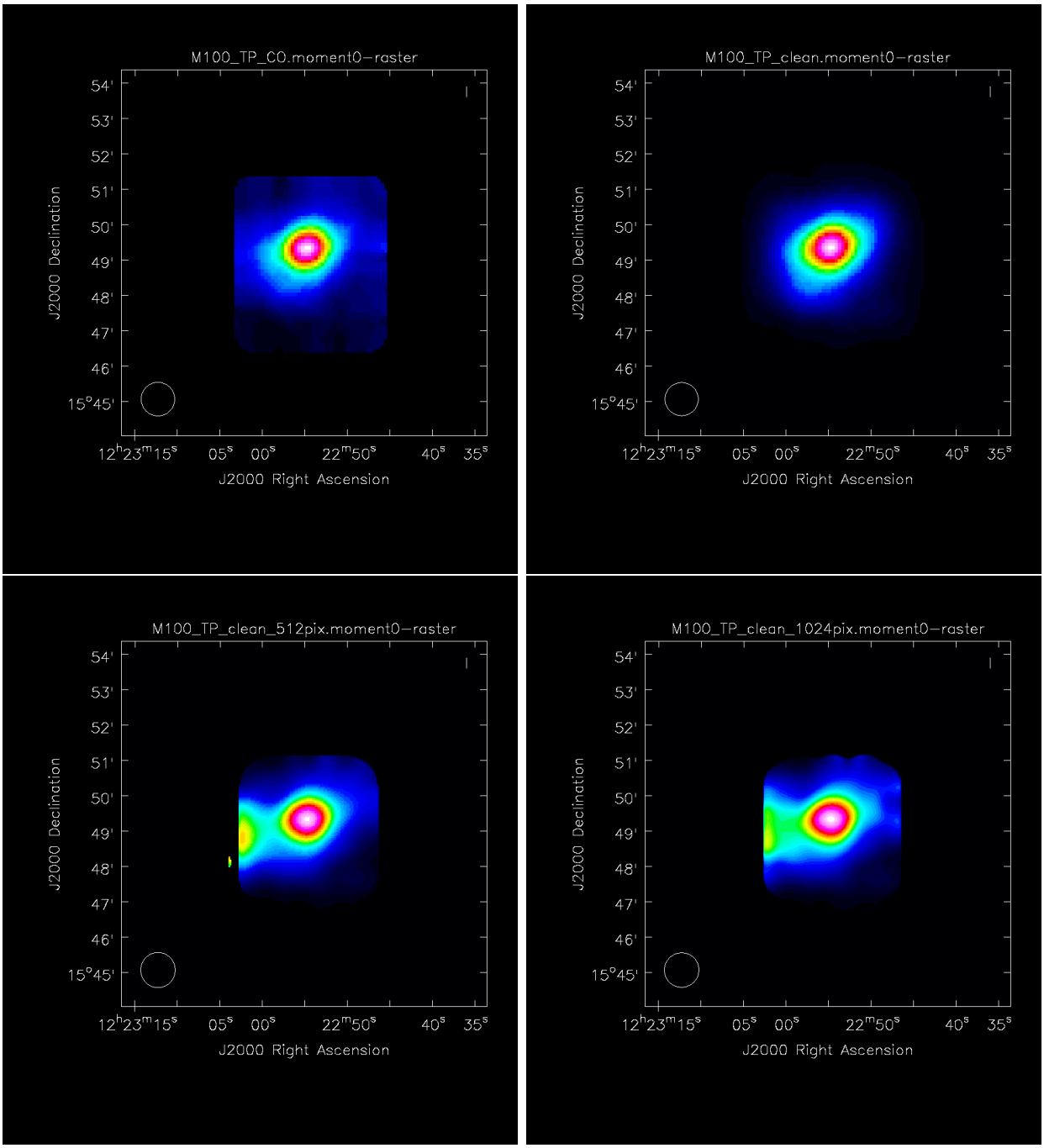

Figure 10: Moment 0 map of the TP image (*top left*) and the clean image of the pseudo-visibility produced by `tp2vis` for a map size of 120^2 (*top right*), 512^2 and 1024^2 pixels (*bottom*).

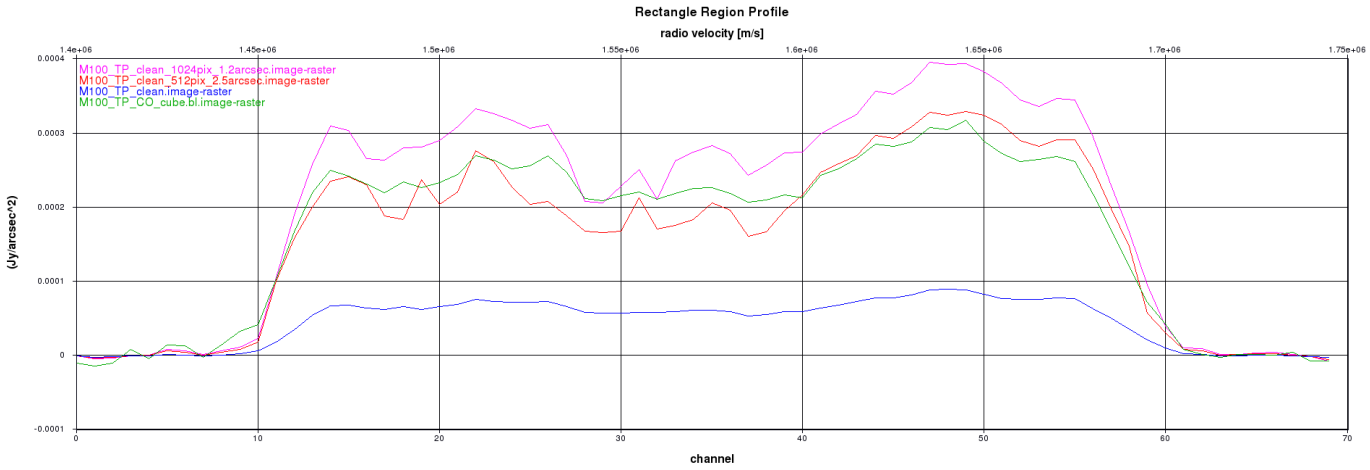

Figure 11: Effect of the map size. The green line shows the spectral profile of the original TP image, while the blue, red and purple lines are the profiles for the clean image of 120^2, 512^2 and 1024^2 pixels, respectively.

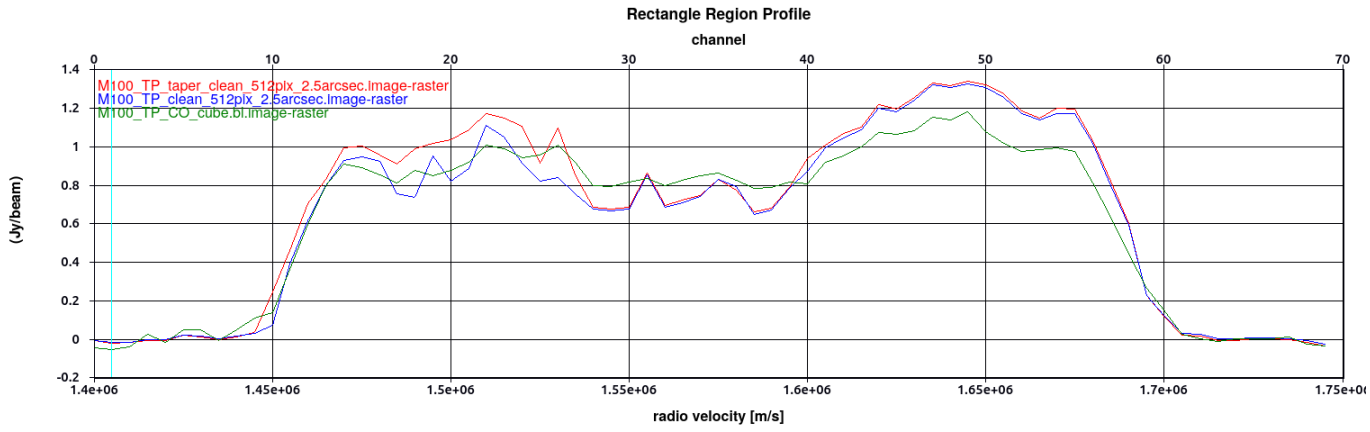

Figure 12: Effect of tapering off the edge emission of the TP emission while running `tp2vis`. The green line shows the spectral profile of the original TP image, while the blue and red lines are the profiles of a 512^2 pixels clean image, without and with tapering.

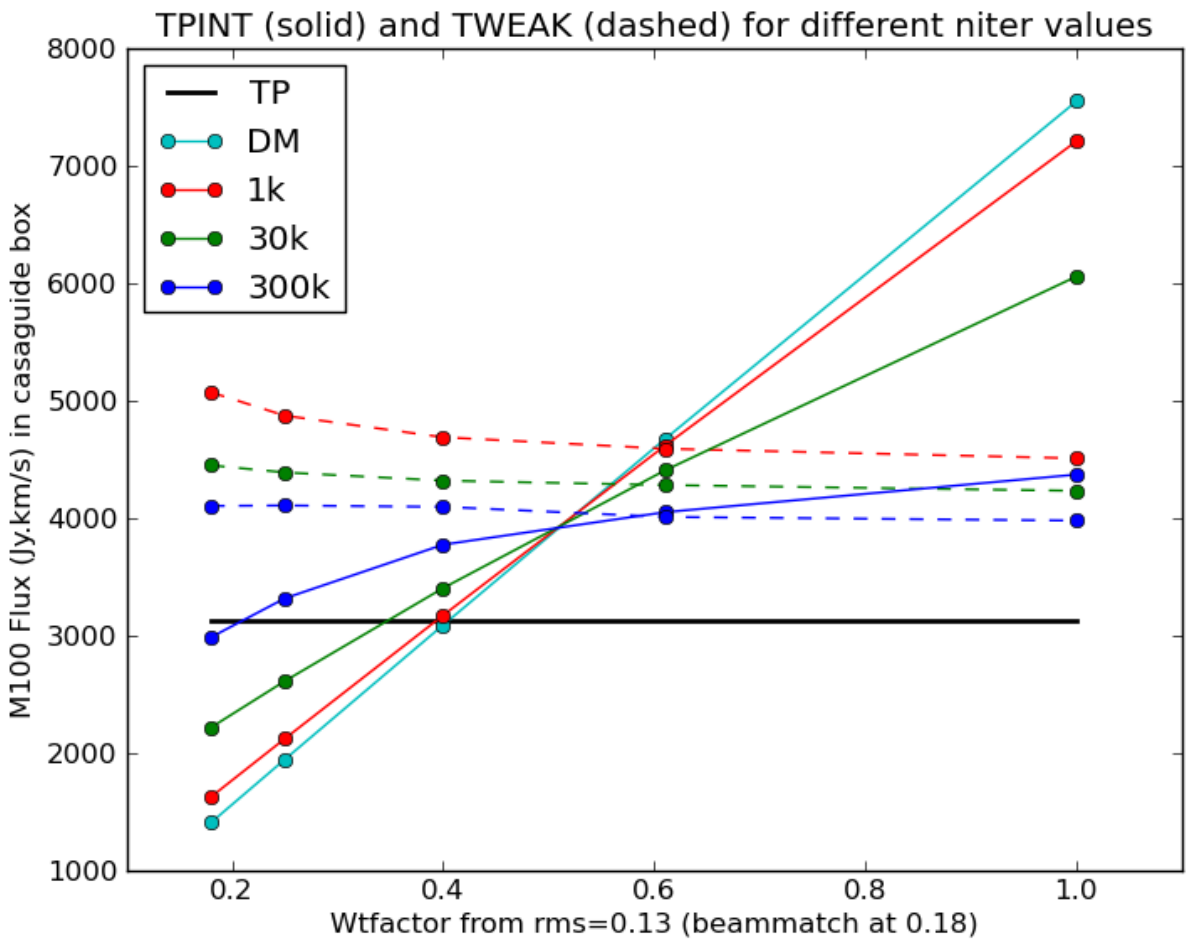

Figure 13: Fluxes of TPINT and TWEAK maps for different values of niter as function of the fractional weight based on rms weighting. On the right side 1.0 means rms weighting. In this case 0.18 means beammatch weighting. The "correct" flux is assumed 3118 Jy.km/s in the casaguide box, which is used here.

cleaning a tp2vis/sdint type combination converges more quickly to the correct answer, and thus a quantitative comparison in our current scripts is difficult.

Another source of uncertainty comes from the fake extrapolation structures in the TP map. The casaguide "solves" this by using a box, but doesn't state this clearly why their box is the correct box. The relatively large 10% errors in the BIMASONG total power flux makes it impossible to judge the box size.

3.2 Comparisons-1

The `M100_combine*py` scripts have all the implementations of the various codes, ranging from exactly the casaguide (with the new automasking feature) to experimental SDINT. Generally we use niter=10000 to at least compare them on a somewhat more equal footing. Perhaps since it's so easy, would be good to compare the niter=10000 feather case with a more straight up tclean.

The script `M100_final.py` tries to summarize some plots and total fluxes, which we summarize in Table 3 below.

[PT: Better figures are coming, these comparisons are flawed since one needs to be sure to compare them on a smaller box to prevent using the non-astronomical grid features near the edge. This makes the comparison with BIMASONG harder]

Table 3: Total Flux

`method`	Beam (arcsec)	Peak flux (Jy/beam)	Total flux (Jy km s^{-1})
TP	56.7	8.81	3118
TP.MS image	55.0 x 53.4	8.26	2281
INT 7+12	4.40 x 2.93	0.776	1018
Feather	4.39 x 2.93	0.791	3051
TP2VIS 7			
TP2VIS 7+12			
SDINT 7	14.6 x 11.3	3.16	2474
SDINT 7+12	4.41 x 2.93	0.797	2361

The imaged TP.MS map does not have a circular beam. This is some internal tclean issue with mosaics, as a single pointing gives round beams.

4 Additional notes, comments, challenges encountered

4.1 Looping of `tclean` over pointing positions

In a separate discussion, unrelated to M100, we wondered about certain implementation aspects of `tclean`. This discussion was motivated by cleaning procedures for mosaics in MIRIAD: that package woulver the pointings in a specific visibility file, collect all visibilities for a given pointing, produce an initial dirty map with data for that position, and initiate cleaning on that dirty map. Instruments like CARMA contained dishes from 3 m to 10 m diameter, and all telescopes would observe exactly the same pointing positions. In that case data from all telescopes are available for all poining positions. But this is different for ALMA, since the 12m array and the ACA observe different pointing grids. In that case, data for only one set of antennas (i.e., 12m array or ACA) is typically available per pointing position. If this is true, that would mean that cleaning of a given mosaic position would

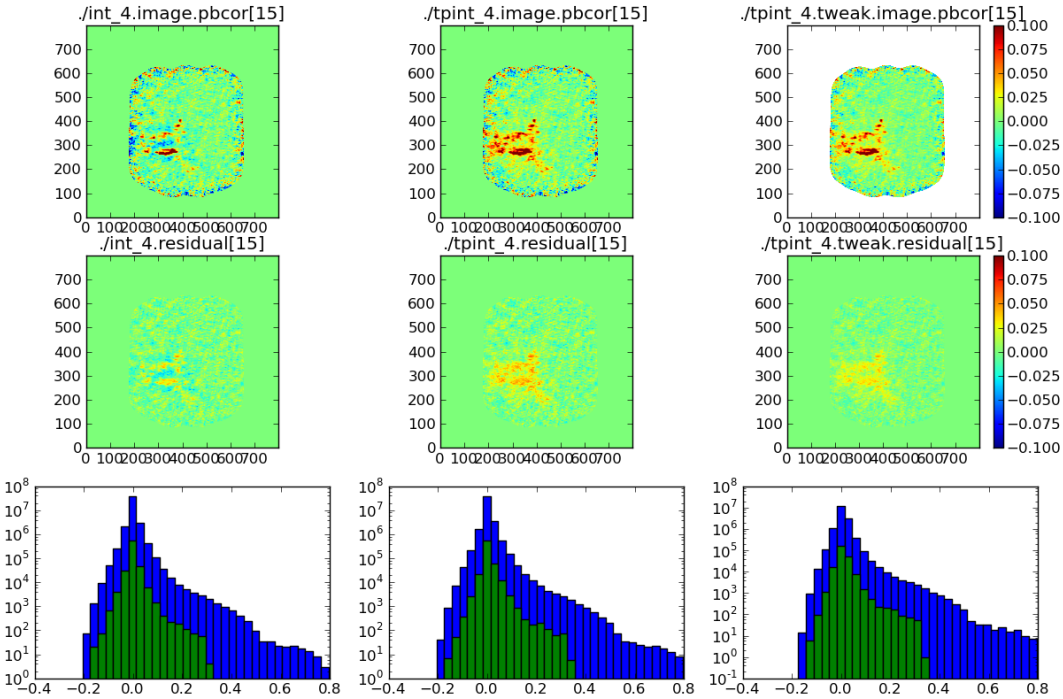

Figure 14: tp2vis: Maps and Residuals and Map histograms from INT, TPINT, and TWEAK. The total flux in the INT (7m+12m) map is 1107 Jy.km/s after 10000 tclean iterations, whereas tpint gives 6072 and tweaked 5083. **[PT: need to plot only the standard box area]**

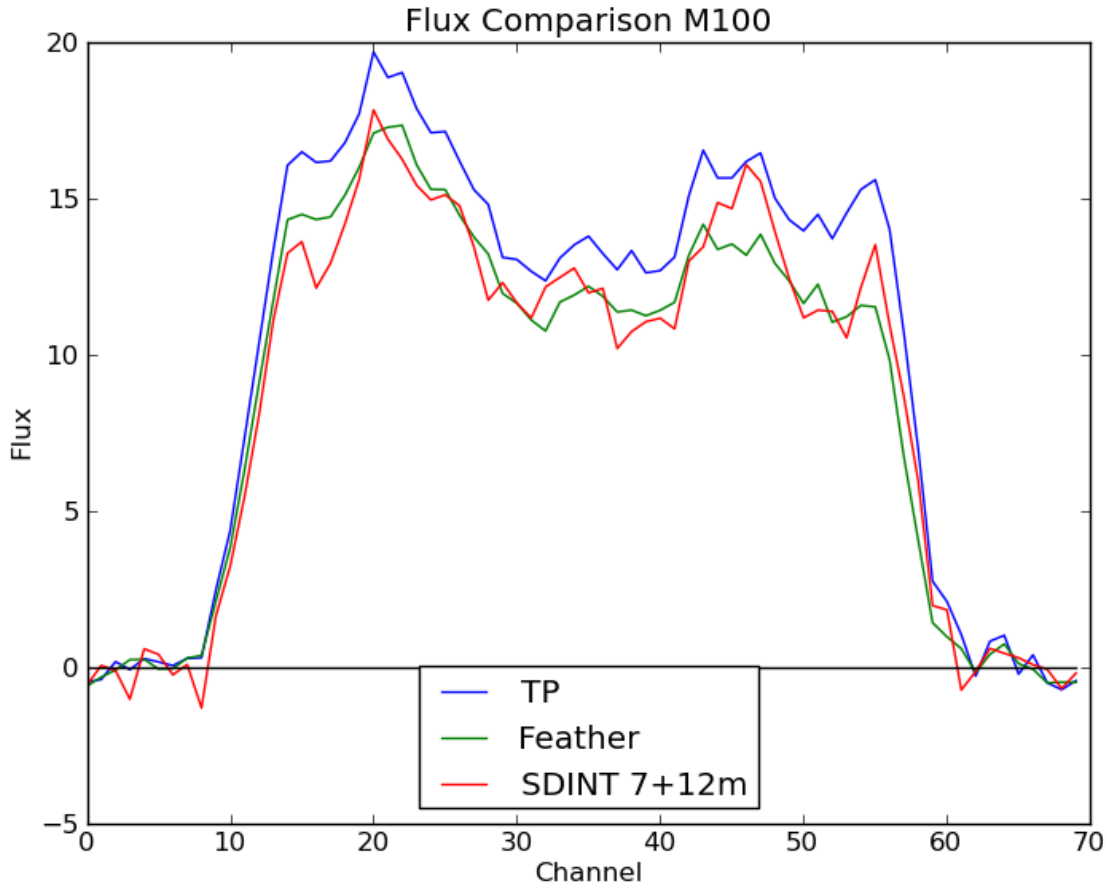

Figure 15: Comparing fluxes for different methods in each of the 70 channels. Still need to add the tp2vis, once decided what the right weight is.

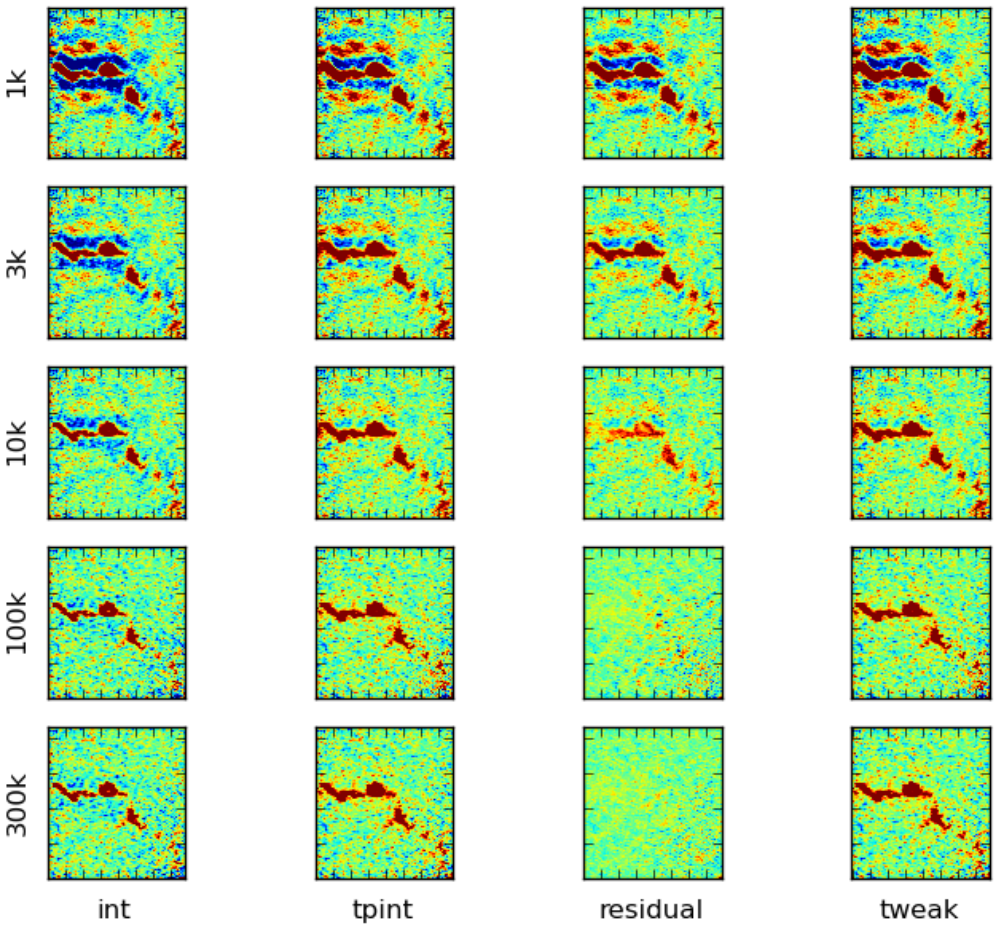

Figure 16: Rows showing the convergence of INT, TPINT, RESIDUAL and TWEAK maps for different values of niter. Notice the 3rd row (niter=10k) is the standard value in the casaguide. Display range is $\pm 5\sigma$.

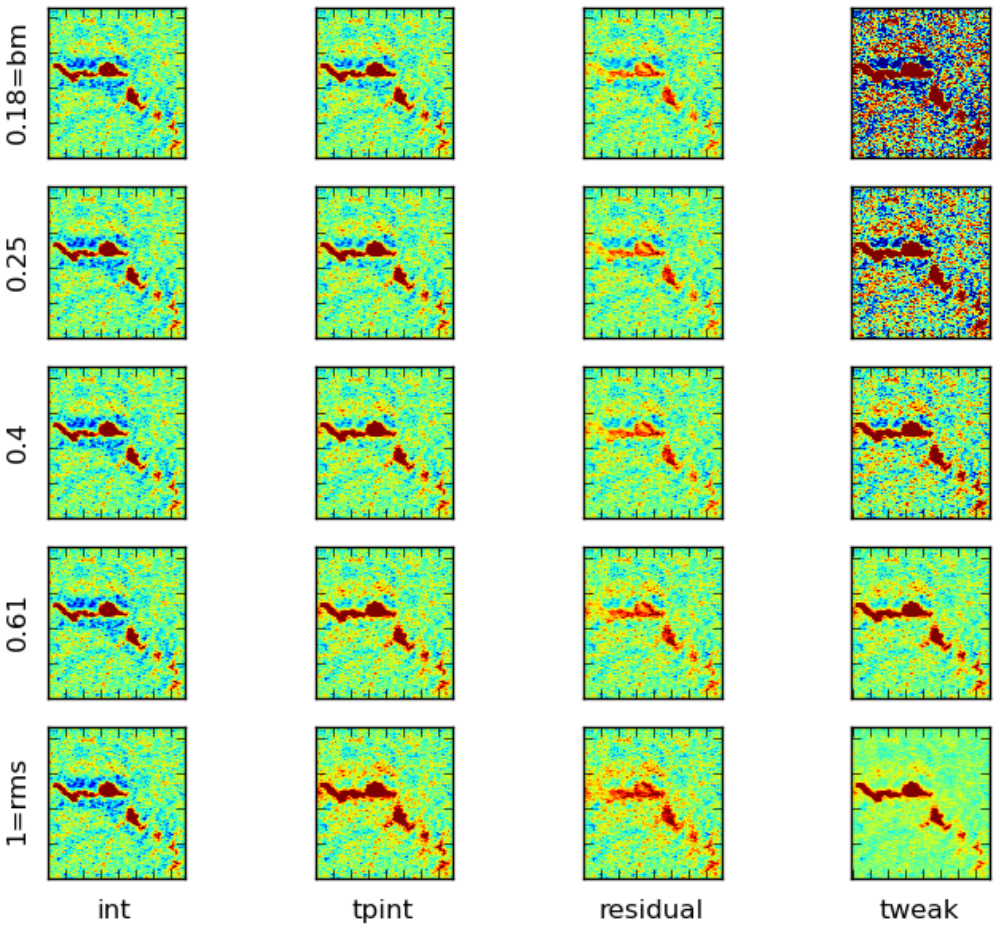

Figure 17: For a fixed value of niter=10k different rows show the INT, TPINT, RESIDUAL and TWEAK map for different weights. The top row matched the rms setting, the bottom row the beam-match. Display range is $\pm 5\sigma$.

not make use of the full range of spatial information available in the visibilities, and only the scales covered by either the 12m array or the ACA — but never the scales jointly covered by both arrays — would be used.

These implementation details are not clear from the documentation of `tclean`. Also, none of the persons present at Leiden was sufficiently aware of such details to resolve this question. It is in principle possible to interpolate the *uv*–data from the ACA onto the pointing grid of the 12m array. But it is not clear whether this is done. **[JK: Currently waiting for feedback from Urvashi. Will update accordingly.]**

Implementation of these details in the cleaning procedures can potentially be of fundamental importance for observing procedures at ALMA. Depending on cleaning procedures, it might be advisable to collect ACA data towards the exact locations of 12m array pointings. That would generally produce the best data overlap between the 12m array and ACA data, at the expense of observing inefficiencies with the ACA (which would have to sample many more pointing positions than done currently).

4.2 Gridding of single–dish data into interferometer spectral channels

The spectral channels of the single–dish and interferometer data typically not be identical. In that case it will be necessary to sample one of the data sets into the channels of the other. It makes no sense to sample the spectral data into channels finer than those of the interferometer, because the information at high angular resolution is not (individually) available for these fine spectral channels. Before merging of single–dish an interferometer data, all data must thus be sampled into spectral channels that are identical to or wider than those of the original interferometer data.

The function `imregrid` does offer *some* or the relevant functionality. It can grid a given image cube into the spectral channels provided by some template header. However, as the `imgregrid` documentation states, "the imregrid task currently finds the nearest input pixel center and interpolates to the output pixel center". It further contains a warning that "no averaging is done in any direction". This is a problem. Consider an interferometer observation with spectral channels that are much wider than those in the single–dish images. The function `imregrid` would in this case take the spectral channels from the interferometer, find the channel in the single–dish data closest to each channel in the interferometer data, and use that single channel in the single–dish data to produce the output re–gridded spectral channels. All the other channels in the single–dish data, that might contain very different data values and that can generally be used to reduce the noise level, are ignored (as far as a reading of the `imgregrid` documentation suggests). This is not a desirable approach.

It would instead be desirable to bin spectral channels into some other grid. CASA offers a function `imrebin`, but that function only bins in terms of integer pixels and channels, and it does not bin to a frame set by some template.

[JK: Need discussion here about spectral response per channel of ALMA, or generally interferometers. Contacted David Mehringer about that, will update accordingly.]

A desirable approach would be as follows, assuming that the interferometer data is obtained on a coarser spectral grid than the single–dish data.

1. Smooth the single–dish data to the spectral resolution of the interferometer observations. As discussed below, no ideal solution appears to exist in CASA for this operation.

2. Grid the smoothed data into the spectral channels of the interferometer data. As discussed below, this task can be implemented using `imregrid`.

The discussion above already showed that `imrebin` is not suitable for this operation. The function `sdsmooth` delivers some of the relevant functionality — as long as the single–dish data are available

in the form of a measurement set. The `sdsmooth` function can smooth data using boxcar and Gauss filters — with widths specified in integer channels. This restriction in the specification of kernel widths is a major limitation.

The `specsmooth` function does work on images, but it only offers Hanning and boxcar filters. Further, kernel widths again have to be specified in integer channel numbers. This limitation in kernel types and widths means that this function cannot generally solve the smoothing function.

[JK: Specify approach to smoothing via Python?]

[PT: A number of issues, including bugs in mstransform, as discussed in the `M100_trimdata.py` script]

4.3 Where is the edge of the TP map?

The total flux measured over all the TP map is clearly too high, e.g. compared to the BIMA SONG value. Visual inspection shows phantom features perpendicular to the edge of the map. In maps with slanted coverage this is even more apparent. The casaguide beats around the bush and takes a box within which the comparisons are done. It is not clear if the choosen box is the optimal one. An ALMA helpdesk ticket (16180) has been filed on this.

We also goofed a bit ourselves: the trimmed QAC benchmark data were based on the older CASA_4.3 guide, and the flux (including the edges) was 4001 Jy.km/s, whereas the CASA_5.4 guide resulted in 3562. If you ignore the edges and use the casaguide box, those values are 3145 and 3118 Jy.km/s resp, much better agreement between the old and new TP data. The BIMASONG value from the NRAO 12m dish is 2972 $\pm$ 319 Jy.km/s over a 6 arcmin area, is still in good agreement with either TP value.

5 Appendix: Code

5.1 Scripts used

First we summarize the scripts we used. They can be found in the dc2019/scripts directory:

```
M100_combine1.py - the original casaguide 5.4 feather script
                   (only modified to run in a subdirectory "M100")
M100_trimdata.py - how we trimmed data to the benchmark size
M100_combine2.py - like combine1, but now for the benchmark data + some QAC
M100_combine4.py - tp2vis script for the benchmark data
M100_combine3.py - sdint script for the benchmark data
M100_final.py    - plotting and printing various summaries
```

5.2 Data Files

The above mentioned scripts produce a lot of datasets. Be aware some are noise-flat (most .image) and some are flux-flat (the .image.pbcor). We summarize the important ones:

```
M100_TP_CO_cube.bl.image    - input TP cube

M100/M100_combine_CO_cube.image.pbcor
M100/M100_Feather_CO.image.pbcor

M100qac/test6/tp0/clean0/dirtymap.image.pbcor - tp.ms imaged
M100qac/M100sdint_7.joint.cube.image
M100qac/M100sdint_12.joint.cube.image
M100qac/M100_Feather_CO.image.pbcor
M100qac/test6/clean6/tpint_2.tweak.image

M100t2v/
M100t2v/
```

5.3 Method 1: Feather

We used two scripts, closely following the casaguide version. The original works on the large 26GB data, the second one assumes you have the benchmark data, and we also have a script that shows how the large 26GB data is trimmed to a more manageable 100MB dataset. The trimdata script also fixes a number of issues with these data, e.g. the ordering of channels, the missing (or wrong) rest frequency etc. See comments in the script.

```
We used M100_combine1.py here
```

5.4 Method 2: Joint Deconvolution (tp2vis)

[PT: For now this script is short, and does not yet exist in dc2019/scripts, so it can stay. Unclear where/how the file 12.ptg is created. I could not run this script "as is"]

```
### Define measurement set
twelvem='M100_aver_12.ms' # 12m measurement set
aca='M100_aver_7.ms' # 7m measurement set
TPcube='M100_TP_CO_cube.bl.image' # TP data cube (assume one polarization)


### Make initial 12m + 7m dirty
os.system('rm -rf '+'M100_07m12m_CO_dirty.*')

tclean(vis=[twelvem,aca],imagename='M100_07m12m_CO_dirty', niter=0,gridder='mosaic
    ↪ ',imsize=800,cell='0.5arcsec',phasecenter='J2000 12h22m54.9 +15d49m15',
    ↪ weighting='natural',threshold='0mJy',specmode='cube',outframe='LSRK',
    ↪ restfreq='115.271201800GHz',nchan=70,start='1400km/s',width='5km/s')

### Make initial 12m + 7m clean
os.system('rm -rf '+'M100_07m12m_CO_clean.*')
tclean(vis=[twelvem,aca],imagename='M100_07m12m_CO_clean', niter=10000,gridder='
    ↪ mosaic',imsize=800,cell='0.5arcsec',phasecenter='J2000 12h22m54.9 +15d49m15
    ↪ ',weighting='natural',threshold='0mJy',specmode='cube',outframe='LSRK',
    ↪ restfreq='115.271201800GHz',nchan=70,start='1400km/s',width='5km/s')

#make listobs file
listobs('M100_12m_CO.ms',listfile='12m.log')

#measure RMS
imstat(TPcube,axes=[0,1])['rms'][:6].mean()

#Load and run TP2VIS to generate a measurement set (MS) for TP
execfile('tp2vis.py') #If you install QAC before, you donot need to execute this
    ↪ file
os.system('rm -rf '+'M100_TP_CO.*')
tp2vis(TPcube,'M100_TP_CO.ms','12.ptg',nvgrp=5,rms=0.15)

#Run concat to make a single .ms file of all three 12m , 7m and total power cubes
os.system('rm -rf '+'12m7mTP_cocat.*')
concat(vis=[twelvem,aca,'M100_TP_CO.ms'], concatvis='12m7mTP_cocat.ms',
    ↪ copypointing=False)

#Run tclean to make dirty map
os.system('rm -rf '+'M100_TP07m12m_CO_dirty.*')
tclean(vis='12m7mTP_cocat.ms',imagename='M100_TP07m12m_CO_dirty', niter=0,gridder
    ↪ ='mosaic',imsize=800,cell='0.5arcsec',phasecenter='J2000 12h22m54.9 +15
    ↪ d49m15',weighting='natural',threshold='0mJy',specmode='cube',outframe='LSRK
    ↪ ',restfreq='115.271201800GHz',nchan=70,start='1400km/s',width='5km/s')
```

```
#Run tclean to make clean map
os.system('rm -rf '+'M100_TP07m12m_CO_clean.*')
tclean(vis='12m7mTP_cocat.ms',imagename='M100_TP07m12m_CO_clean', niter=10000,
    ↪ gridder='mosaic',imsize=800,cell='0.5arcsec',phasecenter='J2000 12h22m54.9
    ↪ +15d49m15',weighting='natural',threshold='0mJy',specmode='cube',outframe='
    ↪ LSRK',restfreq='115.271201800GHz',nchan=70,start='1400km/s',width='5km/s')
```