Deconvolution and Imaging

ASTR 240: In-class activity, October 8, 2018

In this activity, we will use calibrated visibilities from the Submillimeter Array to create an image of a disk around a nearby young star. The goal of this exercise is for you to get some familiarity with the process of creating an image from a set of visibilities, including the CLEAN algorithm that deconvolves the dirty beam from the dirty image. The questions are not rhetorical! Discuss them amongst your group, but please ask me if you can't figure out the answer.

Setup

1) Download and unpack the tar file containing the data we will use for the exercise. I have posted it on the web at:
   http://amhughes.web.wesleyan.edu/download/data.tar
   Once you have downloaded it, you can unpack it by typing "tar –xvf data.tar" in whatever directory you stored it.
2) Here is a brief introduction to the software we'll be using for deconvolution and imaging, known as MIRIAD (Multichannel Image Reconstruction Image Analysis and Display). You can simply type commands on the command line of a terminal; the syntax is:
   task keyword1=value keyword2=value (etc.)
   For example, if you wanted to inspect the header of the file "filename.vis," you could type: "prthd in=filename.vis" ("prthd" stands for the "print header" task, and the only required keyword is the filename "in").
   This web page has compiled a list of MIRIAD tasks. Click on a task to see a description of the task and an explanation of the various keywords associated with that task (rather than typing in the address you can just google "MIRIAD task index" and it should be the first result):
   http://www.atnf.csiro.au/computing/software/miriad/taskindex.html
   (note: MIRIAD is also available on all your individual student machines if you don't finish this exercise today. You will need to type "source /soft/miriad/miriad_start.csh" or add an appropriate line to your .cshrc file).

Getting familiar with the data

3) Inspect the files. There should be two sets of visibilities: one is a set of observations in the compact configuration (compact.vis), and the other is a set of observations in the extended configuration (extended.vis). You can view the header information by typing "prthd in=[filename].vis"
   This should provide information like the telescope, object of observation, number of antennae, frequency and bandwidth of the observation, and coordinates of the source. Take special note of the position of the source and the frequency of the observation (i.e., write them down so you can refer to them later).
4) Now, look at the (u,v) tracks. Plot $\varepsilon(u,v)$ for each data set by typing
   uvplt vis=extended.vis axis=uc,vc options=nobase,avall device=/xs

(Note that this plots u,v data for the file "extended.vis," with u on the x axis and v on the y axis; options "nobase" and "avall" tell it not to separate out the different baselines into different plots and to time-average all the data; /xs tells it to use a pgplot sindow as a plotting device.)
Do you see how there are two parallel sets of u,v tracks for each observation? This is showing you the two different sidebands (remember that u is defined as x/λ, so if you change the frequency of observation you change the separation of the baselines in the u,v plane). Which set of (u,v) tracks corresponds to the upper sideband (higher frequency), and which to the lower sideband (lower frequency)? Write down the (approximate) lengths of the longest and shortest baselines in this data set (in kilolambda).
Make the same plot for the file "compact.vis." Note the different baseline lengths (i.e., pay attention to the ranges of the x and y axes and how they change between plots). Write down the longest and shortest baselines in this data set.
Now, plot the (u,v) coverage of both data sets by typing:
```
uvplt vis=compact.vis,extended.vis axis=uc,vc options=nobase,avall device=/xs
```
They should show up in two different colors. Which color represents the compact configuration, and which represents the extended configuration?

5) Now, try fitting an elliptical Gaussian to the visibilities. There is a handy built-in task to do this called "uvfit." Type:
```
uvfit vis=compact.vis object=gauss
```
Check the flux, position, major and minor axes, and position angle of the fit. Write these down for later reference. Sketch what you expect the image to look like.
Try the same thing with the extended array data. Do the two data sets give you the same results? What do you think might explain any differences? Which fit would you trust more, and why? (These latter questions might become clearer after step 6.)
6) Plot the amplitudes as a function of baseline length (or u,v distance):
```
uvamp vis=compact.vis,extended.vis device=/xs bin=14,10,klam
```
This command plots the visibilities as a function of distance from the phase center of observation, in 14 bins that are each 10 kilo-wavelengths wide (remember that baseline length is x/λ, and so a natural unit is the number of wavelengths that fit across the baseline, or for realistic data sets, thousands of wavelengths across the baseline).
Is the source detected? That is, do there seem to be some bins, particularly on the short baselines, in which a significant amount of flux is detected considering the scatter in the data (the error bars represent 1-sigma scatter)? Estimate the total flux of the source. (Do you know how to do this? If not, ask!)
Is the source spatially resolved? Recall that we say that a source is "resolved" if its visibilities do not resemble those of a point source (which is, by definition, unresolved).
What is the longest baseline on which you're sure the source is detected? How does this compare to the shortest baseline in the extended array configuration?

Create a dirty map

7) Now we begin the process of creating an image. Take the Fourier transform of the sampled visibilities (ε(u,v)V(u,v) in the notation we used in class) to create the dirty map. You can do this using the command:
   invert vis=compact.vis map=compact.mp beam=compact.bm cell=0.3 imsize=256 options=systemp,mfs robust=2
   This command takes the Fourier transform of the visibilities in the file "compact.vis," and creates a new file containing the dirty map called "compact.mp" while simultaneously creating a file that contains the dirty beam called "compact.bm" (we could have called the dirty map and dirty beam files anything we wanted – this is just my particular file system organization scheme). The "systemp" option weights the visibilities by the system temperature (so that noisier cells get less weight – you should ponder amongst your group why some visibilities would have higher system temperatures than others). The "robust=2" keyword specifies natural weighting (we'll go into more detail about this at the end of the exercise). Note that I have specified the size of a pixel in arcseconds (0.3) and the number of pixels across the image (256). In general, we want the size of a pixel to be a moderately small fraction (e.g., 1/3-1/10) of the synthesized beam size, and we want the number of pixels to make the image big enough so that it is at least somewhat larger than the size of the primary beam (and powers of two are nice for computation). Estimate the size of the synthesized beam using the length of the longest baseline in this data set. What is the size of the primary beam, given that the observations were conducted with 6m antennas, and given the frequency you read from the header? Are these choices of pixel size and number appropriate? If not, change them! (You will need to delete the old files you created first by typing "rm –rf compact.bm" and "rm -rf compact.mp".)
   Go through the same process for the extended array data:
   invert vis=extended.vis map=extended.mp beam=extended.bm cell=??? imsize=??? options=systemp
   … inserting appropriate values for the cell and image sizes, respectively, using the information I gave you above.
   Finally, create a combined map and dirty beam for the two sets of visibilities:
   invert vis=compact.vis,extended.vis map=comext.mp beam=comext.bm cell=??? imsize=??? options=systemp
   (Again, inserting appropriate values for the cell and image sizes.)
8) Look at the dirty map and the dirty beam (compact data first, followed by extended data, followed by combined data). In order to display an image (including dirty maps or dirty beams, use the command "cgdisp." For example:
   cgdisp in=compact.bm device=/xs labtyp=arcsec
   (Note that the labtyp=arcsec keyword just tells it to plot the axes in arcseconds rather than pixels)
   There are many different ways to display an image – if you're interested, I suggest reading about the "cgdisp" task in the MIRIAD task index website from item 2.
   When you look at the beam, notice whether it is elongated in any direction. Given what you know about the position of the source (and given the u,v tracks

you plotted earlier), why do you think this might be the case? Also notice the locations of the largest sidelobes, positive and negative.
When you look at the dirty map from the compact data, notice whether the source appears to be elongated in any particular direction. Is this the same direction as the elongation of the dirty beam? Why might these differ? How does the direction of elongation compare to the properties of the Gaussian fit you conducted in item 5? Can you spot artifacts in the image due to the positive and negative sidelobes in the beam?
Do the same thing for the extended data. Do you see any signal in the map of the extended data? Given what you noticed about the baselines on which signal was detected in item 6, and the baseline lengths you recorded in item 4, why might this be the case? Can you think of why the extended data might be useful even if there is no signal in the map? Hint: If the source were very compact compared to the beam size, would you expect to detect the source in the extended data? What if it instead were large compared to the beam size, so the flux was divided among several beams?
Finally, look at the combined compact/extended data. Can you see that the dirty beam has a bright, narrow core caused by the extended data, and a fluffier halo caused by the compact data? When you look at the dirty map, does the source still appear to be centrally peaked? (i.e., is the brightest pixel at the center of the image?)

CLEAN the map

9) For this step, let's focus only on the combined (compact + extended) data. The CLEAN procedure is implemented in MIRIAD in the following way:
clean map=comext.mp beam=comext.bm out=comext.cl niters=300
This uses the dirty map (comext.mp) and the dirty beam (comext.bm) that we generated using the "invert" task, and cycles through the CLEAN algorithm creating a clean component map that is stored in "comext.cl." This is the record of the locations of all the point sources that were detected as part of the CLEAN procedure.
Display the clean component map using the "cgdisp" command. Where are the brightest CLEAN components? Are they all concentrated in the middle, or does it look like there are two peaks instead?
Play with the number of iterations (niters) of the CLEAN procedure until you think you have just barely cleaned down to the noise. You will have to delete the CLEAN component map before each time you run the "clean" task (remove the file by typing "rm –rf comext.cl").
10) Finally, restore the CLEANed image by convolving with the "clean beam" using the "restor" task:
restor map=comext.mp beam=comext.bm model=comext.cl out=comext.cm
This task uses the dirty beam (comext.bm) to generate a smoother "clean beam," which it then convolves with the clean component map (comext.cl) and adds the residuals from the dirty map (comext.mp) to generate the cleaned map (comext.cm). Display the CLEANed map using the cgdisp command. Then try re-displaying the dirty map using the cgdisp command. What differences do you

notice? If you still see residuals from the beam, try the process again using more or fewer iterations of the CLEAN algorithm. Did you not CLEAN down to the noise? Did you CLEAN so much that you started confusing beam artifacts with signal?

11) Congratulations! You have just made an image of a debris disk around a young star! The double-peaked structure that we are able to detect by combining the compact and extended array data from the Submillimeter Array tells us that the disk has a central hole – i.e., it's a thin ring rather than a broad torus. What do you think could be clearing out that inner cavity in the disk???

If you finish early

12) Go back to step 7 and start playing with some of the adjustable keywords for the "invert" task. The most important one is the "robust" keyword. This allows you to toggle back and forth between "natural weighting" (robust=2) – which weights the visibilities according to the inverse of their variance (so that noisier cells in the (u,v) plane get lower weights) – and "uniform weighting (robust=-2) – which weights every (u,v) sample equally regardless of its quality. In practice, natural weighting provides the greatest point source sensitivity but lower spatial resolution, while uniform weighting provides the highest spatial resolution (because it places more emphasis on the longer baselines) but lower sensitivity. The "robust" keyword in the "invert" task assigns an intermediate value that allows you to approximate a spectrum of possible weights between the extremes of natural and uniform weighting. (This is known as the Briggs weighting scheme. The guy who invented it, Dan Briggs, died in a skydiving accident shortly after finishing his PhD. It's a terrible story – please think twice before skydiving!) Try some different values of the "robust" parameter, making sure to try both natural and uniform weighting and a few values in between, and see what gives you the nicest image (this is a subjective process!). Don't forget to delete the files in between tests, or use different names for the files (otherwise MIRIAD will complain that the "File exists"). I rather like a value of the "robust" parameter around 0.5 for this data set. To estimate the rms noise in the data, you can use the "imstat" task on the restored CLEANed image:
    ```
    imstat in=comext.cm
    ```
    Keep track of the rms noise for the different images you make. Would you expect the rms noise to be higher for a uniform or natural weighting scheme?

13) If you've thoroughly explored the Briggs weighting scheme, another thing you can try is using a Gaussian taper to place higher weight on the short baselines and lower weight on the long baselines – this is controlled by the "fwhm" keyword in the "invert" task (check the MIRIAD task index for an explanation of what this keyword does). Continue to keep track of the rms noise in your images – how does tapering the visibilities affect the noise in the image? Can you think of why this might be?