Reformatter Code – Planning the Solution

Both Line 2022 and Line et. al. 2024 are references to the solutions.

Input Parameters:

- 3D array containing brightness temperature values in h^{-1}Mpc x h^{-1}Mpc x h^{-1}Mpc box
- Maximum allowed frequency resolution
- Staring transverse comoving distance
- Length of box in each dimension (assuming uniform bin sizes)
- Phase reference point (RA, Dec)
- Flag to always interpolate and re-grid the t-dimension or only when maximum freq. resolution is met

Steps:

It is convention to equate the Hubble parameter as equal to 100 h, with h being the comoving parameter.

1. Convert each voxel's l,m-dimensions from comoving distance into flat angular resolution

Using the equation (Line et. al. 2024): $\Delta\theta = \Delta x/\mathcal{D}(z)$ and $\theta = x/\mathcal{D}(z)$

Note that θ is a flat-space angle, meaning it does not account for the spherical curvature of the night sky, this is because of the small angle approximation being employed and that θ is specifically representative of an angular resolution on the l,m-plane, not sky coordinates.

2. Convert t-dimension from comoving distance to redshift

As from Line 2022, the Hubble constant is estimated to be 68, or 100 h. This number is used as, because we are dealing with the early universe, the Hubble constant as approximated by CMB BAOs is more accurate to use than other lower redshift methods e.g. TRGB/SNe Ia. Thus $H_0 < 70$. h is defined to be 0.68 for the modern day due to this and the last convention.

Thus, using a cosmological ΛCDM model with (Line et. al. 2024, Greig et. al. 2022, Planck Collaboration et al. 2020) with parameters: $(H_0, \Omega_M, \Omega_\Lambda, \Omega_b) = (100, 0.31, 0.69, 0.048)$.

Astropy can calculate the redshift from the comoving distance:

```
import astropy.units as u
from astropy.cosmology import FlatLambdaCDM as fmodel, z_at_value as getz

cosmo = fmodel(H0=100, [model parameters])
getz([function params])
```

https://docs.astropy.org/en/stable/cosmology/index.html

3. Convert redshift into frequency

Frequency can be determined using the following equation, with the initial emitted wavelength set to 21cm or 1.42 GHz:

$$\nu = \frac{1.42 \times 10^9}{z + 1}$$

4. Convert Brightness Temperature into Janskies per pixel

To convert between the two, we need the following equations:

$$T_B = \frac{S_\nu c^2}{2k_B \nu^2}$$

$$L_\nu = \Omega S_\nu$$

Where L_ν is the luminous intensity in Jy pix^{-1}. Note that the solid angle can be calculated by multiplying $\Delta\theta$ and $\Delta\phi$ together (i.e. the two flat-space angles for the l,m-dimensions).

Thus,

$$L_\nu = \frac{2k_B \nu^2 T_B \Omega}{c^2}$$

5. Convert flat angular resolution into l,m-coordinates

This is a simple method: $l = \frac{\theta}{2\pi}$. This converts our radian unit into something physical, though note the small angle approximation is being used here as well.

6. Convert l, m coords into RA, Dec

From Line 2022, we can derive the RA and Dec using the following equations:

$$\Delta\delta = \sin^{-1} l$$

$$\Delta\alpha = \sin^{-1}\left(\frac{l}{\cos(\sin^{-1} l)}\right)$$

Where the deltas indicate that this is an RA, Dec deviation from the phase reference point, thus, to calculate the real RA, Dec, we must add these values to the phase reference point provided as input.

7. Resample frequency bins if resolution too large

The designed algorithm should then check the frequency bandwidth of the largest frequency bin. If the bandwidth is greater than the maximum frequency resolution, then we MUST interpolate our data and re-grid along the t-dimension.

We can, by default, interpolate and re-grid the data regardless, this is going to take the form of a parameter that can be fed the top-level function.

What can be done is a Multidimensional Spline interpolation of the effective frequency vs intensity histogram created by our data. This can be done using scipy:

https://stackoverflow.com/questions/69630248/spline-interpolation-of-a-2d-histogram-python

https://docs.scipy.org/doc/scipy/reference/interpolate.html

After this interpolation we can re-grid the data using the derived interpolation and a set of equal frequency bins (with the same number of bins/indices as prior).

8. Save data into OSKAR-friendly .osm file

Finally, we want to save our data into an OSKAR-friendly format. The specific file type OSKAR relies on is a .osm file listing a series of point sources in a CSV-type format with four parameters we care about:

- RA (deg)
- Dec (deg)
- Stokes I (Jy)
- Reference frequency (Hz)

An example of an actual line in the data would be:

```
# RA  Dec   I     Q   U   V   freq0
0.300 70.10 0.081 0.0 0.0 0.0 120e6
```

For every voxel in the array, we use the central (mean) bin frequency, RA, and Dec for that voxel to represent their corresponding parameters, and then the value inside of the voxel element as the stokes I parameter.

Output Parameters:

- 3D array containing point source Jansky per pixel values in rad (RA) x rad (Dec) x Hz box
- Index of array voxel bin sizes and values for all three dimensions
- Array must be interpolated and resized in t-dimension if necessary (might do it by default - debatable)
- A .osm file containing all the above information in a format that is digestible for OSKAR to use