**MAIN.PY**

Definitions:

*21cm refers to the signal generated by 21cmfast*

*image refers to the image created by the array*

*sigma is the error referring to each baseline vector*

input 21cm real space box : defines z, Realspace size (Mpc), Resolution (Mpc) with corresponding theta using cosmo function.

wavelength of 21cm signal defined using z (in m)

Experiment parameters defined:

- sampling time

- inclination of telescope

- background system temperature (mK)

- bandwidth (Hz) - based on an individual antenna

import antenna positions: func.importarray - turns into baselines(m) / wavelength(m)

rotation matrix on antenna positions - defines UV coverage : func.rotationmatrix

- rotates and stores each baseline vector. Saves how often a vector has been sampled

- final matrix used as single UV coverage

- angle is scaled using variable “scaling” to give the UV plane in its units corresponding to

21cm box <— in inverse space

-

create 3D fourier space box for 21cm and empty equivalent for the error and image box.

**FFT CONVENTION**: *Real space of resolution dtheta goes to an inverse space spanning 1/dtheta*

Sampling - Working with slices of 21cminverse:

*NB: while in real projects the UV sampling would change slightly from each slice due to slightly different wavelengths, here we assuming a roughly constant wavelength for each z box*

- for each point where the UV count is not zero, an std (noise) with a 1/rootN dependence

is introduced - used to add gaussian noise to real and imaginary baseline sample

seperately.

After sampling - combine all slices to give 3d inverse space box.

func.EOR window works on inverse space box.

- ktorratio flicks within r (index units) and k space units. k space is 1/2dl large (as it is

centred in the middle and ranges over a total 1/dl

- define window shape in index units using ktorratio - need to verify equations/values

- take out 3 by 3 by 3 centre window to be replaced afterwards to avoid DC issues

Real space box created using fft.ifftn which includes UV sampled noisy data.

Real space box for Windowed version

Visualising slice by slice function

Powerspectrum function analyses inverse space boxes. (compares 3 boxes).

- calculates powerspectrum with option of changing the k space steps

- uses func.powerspectrum3D

- prints the lines - can also compare to 21cmfast output to check results against “true”

results

func.Powerspectrum3D

- spherically averaging in shells of variable width to find |P(k)|^2 and k^3 |P(k)|^2 - need to

find out if pi factors are required or part of volume

CUTOFF - defines the fraction of the average temperature below which the algorithm defines a location as ionised.

ITERATIONS - number of random paths taken in mean free path advance

Mean free path analysis - uses func.secondbubbledistributioncalculator

- finds random ionised point

- generates random isotropic direction

- iterates in that direction until unionised point is found or reaching a boundary

- length is stored - estimates an ionised volume (Mpc^3) relating to that length

- this gives us an average volume given a point in an ionised bubble - this skews the result as there are more points in large ionised bubbles to chose from. Counteract by dividing by volume? This would give us the bubble size distribution.

bubble size distribution analysis - uses func.bubblesizedistributioncalculator

- turns points to ones or zeros depending on CUTOFF threshold

- outputs neutral fraction to compare to 21cmfast

- goes through and counts sizes of connected regions (removes them afterwards).

- could include two bubbles that have been connected by a perturbation - (too precise?)

further functions include an rms calculator to compare image and 21cm, a phase difference calculator and a slice by slice visualiser, as well as the ability to save boxes or slices