XST Data

Array correlation/crosslet statistics statistics (XSTs) describe the autocorrelation between different antenna for a single subband at a given point in time (visibilities, covariances). A single XST file contains a single subband's data, which differs from a ACC file which loops over a range of subbands. They can be used to generate all-sky maps, a representation of the sky birghtness distribution of the station at a given frequency.

Generating XSTs

XSTs are generated using the `rspctl` command while the station is in `swlevel 2` , a known mode is set by `rspctl --mode=N` or `rspctl --band=N_M` and the station is expected to be in bitmode 16 as no other observations can be performed simultaneously. It will generate an output intergrated over the last N_{sec} to a file in a given *local_folder* . The overall syntax to generate an output for a given *subband* is

```
user1@lcu$ rspctl --xcsubband=subband
user1@lcu$ rspctl --xcstatistics --duration=n_observation_seconds --integration=n_sec --directory=local_folder
```

It is recommended to run this in combination with a script that will move the file to a new location after $N_{observation\_sec}$ as the file contains no metadata of the mode or subband used for the observation. Moving to a specified *modeN/sbM/* subband is highly recommend as a result. A sample observing script, assuming the station is in `swlevel 2` can be found here.

A note on HBA all-sky observations

Due to the rigid tile structure of the HBAs aligning the side lobes, performing all-sky observations with the entire HBA array is a hopeless endeavour.

However, at the GLOW stations, James Anderson worked around this by activating a single antenna in each HBA tile in a pesudo-random fashion to minimise sidelobe collision. Further work on the methodology by Menno Norden and Michiel Brentjens optimized this method and resulted in the script found here, slightly modified for use on the Irish station, which will automate the process for you. It should be run before any attempts to perform all-sky observations with the HBA tiles.

XST Data Format

XSTs are antenna-majour files that are written to disk every integration period. They do not come with any metadata outside of the starting time, which is present in the file name.

Each sample is 4 x $N_{antenna}$ x $N_{antenna}$ long (by default, an international station has 96 antenna). Each antenna generates a real and imaginary sample for it's correlation with every other antenna (Complex128 type, 2 Float64s, for each polarization). As a result, at the (n,n) indictes we generate the autocorrelation of the antenna with no lag. The overall format can be thought of as a 3D cube, with (n,n) squares stacked for *m* time samples.

(N0,N0)	(N0,N1)	...	(N0,Nn)
(N1,N0)	(N1,N1)		
...		...	
(Nn,N0)			(Nn,Nn)

All-Sky Plotting

Some sample mode 3, subband 210 data from IE613 can be found here. it is highly recommended to use an existing implementation for generating all-sky images, I provide the example David worked on here, but there is also Griffin Foster's SWHT, a module inside Tobia Carozzi's iLiSA and some scripts floating around the LOFAR community.

The method for generating all-sky maps is heavily involed and as a result we will only outline the steps involved here.

- Acquire XST data, antenna locations and (optionally) calibrations files for your station
- Load your data and metadata into an eniroment, samples for XST and calibrtion data,

```
xstData = np.fromfile(dataRef, dtype = np.complex128)
reshapeSize = xstData.size / (192 ** 2)
xstData = xstData.reshape(192, 192, reshapeSize, order = 'F')


calData = np.fromfile(calRef, dtype = np.complex128)
rcuCount = calVar.size / 512
calData = calData.reshape(rcuCount, 512, order = 'F')
calData = calData[:, subband_of_interest]


calXData = calData[::2]
calYData = calData[1::2]
```

```
calData = np.dstack([calXData, calYData])
```

- Apply your calibrations if needed to each of the X/Y polarizations

```
calSubbandX = calData[:, subband, 0]
calSubbandY = calData[:, subband, 1]


calMatrixXArr = np.outer(calSubbandX, np.conj(calSubbandX).T)[..., np.newaxis]
inputCorrelations[::2, ::2, ...] = np.multiply(np.conj(calMatrixXArr), inputCorrelations[::2, ::2, ...])


calMatrixYArr = np.outer(calSubbandY, np.conj(calSubbandY).T)[..., np.newaxis]
inputCorrelations[1::2, 1::2, ...] = np.multiply(np.conj(calMatrixYArr), inputCorrelations[1::2, 1::2, ...])
```

- Generate a range of pointings from sin(-pi/2) -> sin(+pi/2) across a grid, lower values will limit your field of view ((l,m) grid size)
- Calculate your wavelength and corresponding wavenumber for your subband (`k = (2 * np.pi) / wavelength`)
- Form a weight matrix (and it's conjugate) for your pointing using the x-coordinates and y-coordinates of your antennae

```
wX = np.exp(-1j * k * posX * lVec) # (l, 96)
wY = np.exp(-1j * k * posY * mVec) # (m, 96)
weight = np.multiply(wX[:, np.newaxis, :], wY[:, :, np.newaxis]).transpose((1,2,0))[..., np.newaxis] # (l,m,96,1)
conjWeight = np.conj(weight).transpose((0,1,3,2)) # (l,m,1,96)
```

- For each sample of correlations, find the dot product of the correations with the complex conjugate of your weight matrix, and then the weight matrix

```
for frame in np.arange(frameCount):
	correlationMatrixChan = correlationMatrix[..., frame] # (96,96)
	tempProd = np.dot(conjWeight, correlationMatrixChan) # w.H * corr # (l,m, 1, 96)
	prodProd = np.multiply(tempProd.transpose((0,1,3,2)), weight)
	outputFrame[..., frame] = np.sum(prodProd, axis = (2,3))
```

Your output image will need to be rotated by the same mount as your station is rotated. You can find that information within the new iHBADeltas.conf files.

Revision #5
Created 1 July 2020 16:44:15 by David
Updated 3 July 2020 19:47:47 by David