BST Data

Beamlet statictics (BSTs) are effectively dyanmic spectra produced by the station at a regular, relatively low cadence, interval.

Generating BSTs

BSTs are generated using the [redacted] command in conjunction with a running [redacted] command (as a result, the station must be in at least [redacted]). It will write an output summary of the last N_{sec} (int) of antenna correlations to disk in a given *folder_location*. The overall synctax to call a [redacted] command is

[redacted]

The [redacted] command will generate data for $N_{observation\_sec}$ or until the process is killed. As a result, the process must be kept active in a screen or by trailing the execution with an ampersand to send it to the background.

Enabling BST generation with the [redacted] command will disable the CEP packet data stream, which can be re-enabled afterwards by calling [redacted] . You can verify this worked by calling [redacted] to get the current status of the datastream.

BST Data Format

BSTs are frequency-majour files that are written to disk every integration period. They do not come with any metadata outside of the stating time, output ring (only 0 available to international stations) and antenna polarization which are visible within the file name.

Each beamlet controlled by a [redacted] command will generate a single output sample per intergration. So the output array dimensions will depend on your observation, and may be up to 244 in 16-bit mode, 488 in 8-bit mode and 976 in 4-bit mode.

The output data is a float, 4 times the size of your input.

Bitmode	Output (float)
4	float32 (verify)

8	float64
16	float128

BST Plotting

A fast way to plot BSTs in Python can be achieved with the *numpy* and *matplotlib* libraries. If you want to test this out for yourself, there are a large number of BSTs available on the data.lofar.ie archive which contain 488 subband observations from our Solar monitoring campaign.

```
import numpy as np
import matplotlib.pyplot as plt

bstLocation = "/path/to/bst.dat"
#bstDtype = np.float32 # 4-bit obs
#bstDtype = np.float64 # 8-bit obs
#bstDtype = np.float128 # 16-bit obs
rawData = np.fromfile(bstLocation, dtype = bstDtype)

#numBeamlets = 976 # 4-bit obs
#numBeamlets = 488 # 8-bit obs
#numBeamlets = 244 # 16-bit obs
rawData = rawData.reshape(-1, numBeamlets)



plt.figure(figsize=(24,12))
plt.imshow(np.log10(rawData.T), aspect = "auto")
plt.xlabel("Time Samples"); plt.ylabel("Subband"); plt.title("BST Data {}".format(bstLocation))
plt.show()
```

Revision #5
Created 1 July 2020 16:44:23 by David
Updated 25 April 2022 23:21:10 by David