#### Low frequency wide bandwith interferometry

Above the ionospheric cutoff at ~20 MHz, sky noise from astronomic sources is very strong. For instance, at 50 MHz sky noise averages about 7000 K, while at 408 MHz it averages 22 K. At lower frequencies, ground noise is not significant, relaxing antenna and preamplifier requirements.

Simple dipole antennas can provide sky noise limited performance over a wide frequency range (say 40 to 60 MHz).

Now with more capable software defined radios, significant increases in observing power are within the grasp of amateur radio astronomers.

Inspired by approaches taken by the LOFAR and LWA low frequency radio astronomy projects, I have put together a low frequency interferometer that should be capable of doing some interesting science.

See these important and interesting references:

www.fringes.orgSound card based digital interferometerwww.lofar.orgLOFAR – lots of interesting info herehttp://lwa.phys.unm.edu/Interesting, real time, whole sky imaging data

# System schematic for my interferometer



- 1. 75 Ohm RG-6U used to connect antennas.
- 2. Ham designed LNAs probably wont work well due to their fairly narrow passband.
- 3. The Channel Master 7777 is great because it is wide band and has a FM trap built in to block FM radio stations (and they are cheap).
- 4. LOFAR-sytle inverted V antennas work well. Big beam width is a plus.

# Two simple inverted V dipoles cut for 50 MHz



Channel Master 7777 is now at base of antenna (not shown here)



Left: Inverted V dipole with 1:1 coax balun

Below: 145 meter baseline in semi-remote rural setting



Being out in the country is critical. The broadband Channel Master 7777 is much better than an EME type LNA and is waterproof and is powered through coax.

### Why low frequency interferometry?

- 1. Radio sources are very strong at low frequencies
- 2. Simple, low gain antennas can detect a wide range of sources
- 3. Digital signal processing approaches can allow for whole sky imaging from simple antennas. No physical tracking needed.
- 4. With high sample-rate SDRs, large fractional bandwidths are achievable at low frequencies allowing for wideband synthesis and good uv plane coverage with fewer antennas

## **General Strategy**

- 1. Record simultaneous streams of IQ data from the east and west antennas
- 2. Align sample streams from each radio using cross correlation and an inverse FFT based approach
- 3. Perform FFTs on IQ data streams from each antenna
- Cross multiply FFT blocks from each radio to generate a correlation product in the digital domain using Python and NumPy
- 5. Average the data and save as a 2D array of complex values this correlator data becomes the currency for subsequent analysis
- 6. Variations on this approach could be used to identify transient events

## Some notes:

# **Computer:**

- 1. Use up-to-date software. BladeRF drivers, firmware and FPGA images are much more stable now. Dropping samples is very undesirable.
- 2. Use a powerful computer with an i7 core (e.g. i7-2700k)
- 3. Disable any CPU bandwidth scaling run at full power
- 4. Expand the communication buffer settings for the BladeRF (see my Gnuradio Flow diagram)
- 5. Use solid state drives, even for lower bandwidth work if recording data for later analysis.
- 6. Use ext4 formatted drives to take advantage of precision file time stamping. Good for synchronizing data streams.
- 7. Linux/Ubuntu 14.04 is probably a necessity. Windows based executables are probably too old and/or buggy, but this remains untested by me.

# **RF sections:**

- 1. Use RG-6U cable with quality compression F connectors and F-to-SMA adapters. Avoid BNC. Better connectors or coax is unecessary.
- Use a Channel Master 7777 mast mount preamp. Sky noise limited performance exists from ~40 to 85 MHz and it has an FM trap which is a major plus.
- **3. Perhaps most important:** Setup in a remote, rural location. Urban and suburban environments have way to much RFI.

#### Gnuradio flowgraph to simultaneously capture IQ data streams from two BladeRFs



Attempts to cross correlate data in Gnuradio isn't working right now. Sample streams aren't aligned between radios and delay varies each time flow graph is started.

#### Aligning data:

If you are recording from two BladeRFs using Gnuradio at 2 MSPS, you will fill up a 1 TB SSD in about 8 hours. A random ~ 200 millisecond delay between each IQ sample stream can be expected.

Using approaches from the radar world, cross correlation, followed by an FFT can be used to align data. I did a lot of earlier troubleshooting using ATSC digital TV signals for experimentation which produces very clear indications for sample alignment. Unfortunately, astronomical signals are harder to align. Alignment pulses may be needed to make this step easier.

Basic aproach for alignment (see Python code for the gory details): Place IQ data chunk from east radio into a NumPy array (east) Place IQ data chunk from west radio into a Numpy array (west) Feast = FFT(east) Fwest = FFT(west) Filter out DC spike and other strong RFI carriers Cross correlation = Feast \* np.conj(Fwest) Alignment point = IFFT (Cross correlation)

#### Python coding used for aligning IQ data streams

```
import numpy as np
import math, struct
import pyfits, pyfftw, scipy
import matplotlib.pyplot as plt
corrout = np.zeros((4800,1000),dtype=float)
corradd = 0
windowfunc = np.kaiser(10000.5)
windowfunc = windowfunc.astype('complex')
f = '/media/david/New Volume/Interferometry/11-27-2014/east.bin'
g = '/media/david/New Volume/Interferometry/11-27-2014/west.bin'
fin = open(f,"rb")
gin = open(g,"rb")
pvfftw.interfaces.cache.enable()
graw=gin.read(200*16) # move the west stream ahead some
for zz in range(0,20): # move into data a litle bit
    fraw = fin.read(80000*16)
    graw = gin.read(80000*16)
for zz in range (1,4799):
    print "block #". zz
    fraw = fin.read(16)  # move ahead one sample at a time
    for xx in range (0,600): # average 600 FFT blocks
        fraw = fin.read(80000) # read 10000 IQ sample pairs
        graw = gin.read(80000)
        fa = np.array(struct.unpack('20000f', fraw))
        ga = np.array(struct.unpack('20000f',graw))
        eastdata = fa.astype(np.float32).view(np.complex64)
        eastdata = eastdata * windowfunc
        westdata = ga.astype(np.float32).view(np.complex64)
        westdata = westdata * windowfunc
        scieast = pyfftw.interfaces.numpy fft.fft(eastdata)
        scieast = np.fft.fftshift(scieast)
        sciwest = pyfftw.interfaces.numpy fft.fft(westdata)
        sciwest = np.fft.fftshift(sciwest)
        # cross correlation done on line below
        corradd = corradd + (scieast[5100:6100] * np.conj(sciwest[5100:6100]))
    corrinv = pyfftw.interfaces.numpy fft.ifft(corradd)
    corrinv = np.fft.ifftshift(corrinv)
    maxx = np.argmax(np.abs(corrinv))
    print "max " , maxx,np.abs(corrinv[maxx])
    corrout[zz,:] = np.abs(corrinv)
    corradd = 0
```

# Using the inverse FFT of the cross correlation product to find the alignment point for synchronizing IQ data streams from two radios



The east radio signal stream has been shifted one sample at a time, cross correlated and IFFTed. When the signal peak is at 500, signal is aligned. The program on the previous page shows the details of this process.

#### The FX correlator

Correlators form the heart of an interferometer and the correlated cross product is the currency from which time averaged data can be accumulated

At narrow bandwidths a correlator multiplies the east signal by the west signal to generated a cross correlation product.

At wider bandwidths, fringe smearing would occur, requiring the generally historical XF (lag) correlator or the now more popular FX correlator to be used. FX simply means do an FFT, then cross correlate the frequency bins element-wise.

Every frequency bin in the FFT is treated as its own channel, so instead of having say a single 3 kHz channel, in 30 MHz one would have 10000 3 kHz channels. Sensitivity would improve by sqrt(10000) in this case, or 100 times over a narrow band configuration.

Another benefit is that the baseline according to wavelengths at the low and high frequency bins is different. This is equivalent to having a series of antennas all spread out along a baseline all at once.

Python based coding to do the calculations is provided on the next slide.

#### Program to correlate IQ data streams from east and west receivers

```
import numpy as np
                             High speed FFTs/IFFTs
import math, struct
import pyfits, pyfftw, scipy
import matplotlib.pyplot as plt
corrall = np.zeros((650,10000),dtype=complex)
corradd = 0
windowfunc = np.kaiser(10000,5)
windowfunc = windowfunc.astvpe('complex')
                                                                         Big, raw IQ
f = '/media/david/New Volume/Interferometry/11-27-2014/east.bin'
                                                                         data files
q = '/media/david/New Volume/Interferometry/11-27-2014/west.bin'
pyfftw.interfaces.cache.enable() # for faster FFTs/IFFTs
fin = open(f, "rb")
qin = open(q, "rb")
fraw=fin.read(378*16) # 378 sample offset identified
                      # by alignment program
for zz in range (0,649):
   print "block #", zz
    for xx in range (0,6000): # integrate 6000 FFTs
        fraw = fin.read(80000) # 30 seconds
        graw = gin.read(80000)
        fa = np.array(struct.unpack('20000f', fraw))
        ga = np.array(struct.unpack('20000f',graw))
        eastdata = fa.astype(np.float32).view(np.complex64)
        eastdata = eastdata * windowfunc
                                                                    Do FFTs
        westdata = ga.astype(np.float32).view(np.complex64)
        westdata = westdata * windowfunc
        scieast = pyfftw.interfaces.numpy fft.fft(eastdata)
        scieast = np.fft.fftshift(scieast)
        sciwest = pyfftw.interfaces.numpy fft.fft(westdata
                                                                   Cross correlation
        sciwest = np.fft.fftshift(sciwest)
        corradd = corradd + (scieast * np.conj(sciwest))
    corrall[zz,:] = corradd
    corradd = 0
fin.close
gin.close
                                                             Save complex data as a
hdu = pyfits.PrimaryHDU(np.real(corrall))
hdw = pyfits.PrimaryHDU(np.imag(corrall))
                                                             pair of FITs files
hdu.writeto('/home/david/gnuflow/bistatic/50real.fit')
hdw.writeto('/home/david/gnuflow/bistatic/50imag.fit')
```

#### Preliminary digital "wideband" interferometry data

East and west inverted V dipoles separated on a 145 meter baseline

Two BladeRF SDRs operating in MIMO master/slave configuration with upconverting mixers to cover lower frequencies

Center frequency 50 MHz 2 MSPS sample rate 2 MHz band pass filtering

204 minutes of data from each radio recorded to solid state drive

Data below is processed in an FX correlator mode

- 1) FFT / IFFT used to align data from each radio tedious process more details soon. Single data stream mismatches decorrelate data as expected.
- 2) Aligned IQ data cross correlated (east \* conj(west)) and averaged
- 3) Data is contained in a 2D complex number numpy array
- 4) Complex cross product data array visualized in different ways see below

http://www.cv.nrao.edu/course/astr534/Interferometers1.html - background info



145 meter east – west antenna separation, 6 meter wavelength 18.5 minutes per fringe (204 minutes/ 11 fringes)

Fringe rate for object at 0 DEC should be equal to 9.48 minutes per fringe Object predicted to be at ~59 DEC ( acos(9.483/18.5)\*360/3.14159/2) in above data The radio source Cas A is at 58.6 DEC and was generally overhead (with Cyg A in view at the beginning of recording)

#### What's next?

A wideband low frequency interferometer should have the power to detect numerous sources and do some interesting science

Whole sky imaging with low gain antennas is possible, allowing for the possible detection of transient radio sources, for analysis of the ionosphere or study of scintillation events

The next steps for me will be to work on phase corrections and some attempts to perform delay/delay rate based imaging. Most of the professional literature focuses on fringe fitting data that has already been roughly aligned within the bounds of a single IQ sample unit.

The addition of another BladeRF/antenna unit will allow me to explore phase closure and better filling of the uv plane for image synthesis