Fixing Data Taken with the Wrong Sideband and a Frequency Offset

J. R. Fisher

NRAO, 520 Edgemont Rd., Charlottesville, VA 22903

`rfisher@nrao.edu`

August 16, 2005

On June 23, 2005 Vinayak Nagpal and I recorded a few data sample sets using signals from the 0.68-0.92 GHz receiver on the GBT and from the log-periodic antenna on the RFI monitor station near the 45-foot telescope. We recorded 8-bit A/D samples of the IF signals from the two 5-MHz passbands at 10 MS/s using an ADLink PCI-9812 data acquisition card. The passbands were roughly centered on an arbitrarily chosen TV signal carrier. The two signal paths used completely different amplifier, filter, and mixer components, and the resulting sidebands were opposite. Also, I forgot that the last local oscillator used in the monitor station receiver was not referenced to the site maser so we expected a small frequency offset in this signal with respect to the GBT signal. Hence, the monitor station data needed to be conditioned to reverse its sideband and shift its frequency before the two signals could be used in an adaptive filter experiment.

Sideband Flip and Conversion from Real to Complex Data

The signal sideband can be flipped in the real data samples by simply multiplying every second sample by minus one (cosine(pi)). This is the analog equivalent of mixing the signal with a 5 MHz LO at the top edge of the recorded 0 to 5 MHz passband. However, since we want to do further signal processing on complex samples of the same data, I chose to do a real-to-complex data conversion on both data channels first. The sideband flip is then accomplished by taking the complex conjugate of the monitor station samples. This has the effect of rotating the phase of all signals in the recorded passband in the opposite direction as time moves forward, which, again, is the analog equivalent of mixing the signal with a 5 MHz LO.

The real-to-complex data conversion can be done either with a Hilbert transform or by doing a real-to-complex Fourier transform of the data into the frequency domain and then a complex-to-complex inverse Fourier transform back to the time domain. Since I am more familiar with Fourier transform tools, I chose the latter. This leads to a practical problem

of transforming a very long data set, which may not fit into available computing memory. You can do the double transform on short sections of data and splice them together, but you have to worry about errors near the ends of the sections due to the finite length of the transforms. I experimented with various section lengths and found that for data lengths longer than a few tens of thousands of samples the difference between overlapping complex data samples computed in adjacent sections is a very small fraction of the A/D level interval. I also checked the validity of the conversion by comparing the real component of the complex samples with the even-numbered (beginning at 0) original real data samples. The amplitude agreement was better than 0.1% in the worst case of the end data sample.

I used a 32K real data sample length (16K complex transform length) and a 256 complex sample overlap between adjacent sample runs with a linear taper from one run to the next to avoid any residual steps at the boundaries. Keep in mind that small end-effect residuals may be introduced in the data that would show up as 305 Hz spurious signals (5 MHz / 16K) plus a few harmonics. Note, too, that any DC component in the original real samples is lost in this real-to-complex conversion. This component could be restored in a separate mathematical operation, but we have no need for it.

The first step in this signal conditioning process was to write two files that contain the complex sample equivalent of each of the two channels in the original real-sample data file. The Python code for doing this can be found in the data acquisition computer, datacq, directory, /data4/rfisher, as the file real2complex.py. The line of code for flipping the sideband is included in the routine for shifting frequency described in the next section.

Frequency Shift

A frequency shift in sampled data is the arithmetic equivalent of a mixer in RF electronics, which is the multiplication of the signal with a sine wave whose frequency is the desired shift. In a complex data set this is the same as imposing linear phase gradient across the full data set. The phase increment per complex data sample is given by

$$\Delta\phi = 2\pi\Delta f \Delta t, \tag{1}$$

where Δf is the frequency offset in Hz, and Δt is the complex sample period in seconds (inverse of sample rate in Hz). Note that the frequency spectrum of the resulting data set will be wrapped, i.e. for a positive frequency offset the high frequency end of the spectrum will be pushed around to the low frequency end.

The Python code for flipping the sideband and performing a frequency shift on complex data can be found in /data4/rfisher/condx_data.py.

By comparing spectra with a resolution of about 2.5 Hz of the two channels in the June 23 data set, data301.dat, I found that the frequency offset was about 64.0 Hz. There is a very narrow signal at about 3.69 MHz that is of nearly equal amplitude in the two channels that I used to determine this offset. Most other signals in these spectra are stronger in the GBT channel than in the monitor station channel.

Conversion of Complex to Real Data

If you need real data for further analysis, the complex samples can be converted to real in two steps. There will be twice as many real samples as complex samples, and the even numbered real samples, beginning with 0, will be the real components of the complex data. The odd-numbered real values can be computed by Fourier transforming the complex data into the frequency domain, putting a 0 to π phase gradient across the complex spectrum, and transforming back to the time domain. The real components of these modified complex samples will be the odd-numbered real values of the real sampled data. The Python code for this operation can be found in the 'datacq' directory, /data4/rfisher/complex2real.py.

Summary

In the data acquisition computer, 'datacq', directory, /data4/rfisher, I have converted the data in the file data301.dat to complex in the two files data301.dat.ch1 and data301.dat.ch2. The format in these files is 32-bit floating point with alternating real and imaginary values, beginning with real. The data for channel 2 from the rfi monitor station has been flipped in sideband and shifted by 64 Hz, and the resulting complex samples are in data301.dat.ch2x. The complex samples in data301.dat.ch1 and data301.dat.ch2x have been converted to floating point real samples in files data301.dat.ch1r and data301.dat.ch2xr. You can convert a different data file to complex samples, flip the sifdeband of channel 2, offset the frequency of channel 2 by 64 Hz, and convert back to real data with the following linux command in that directory:

python convert.py file_name

Change the frequency offset by editing the Python file, convert.py.