The Proper Production of Full-Stokes Spectra Using the Fully-Functioning Full-Stokes Mode of the GBT Spectrometer

Tim Robishaw & Carl Heiles

UC Berkeley

`robishaw@astro.berkeley,heiles@astro.berkeley.edu`

ABSTRACT

Before retiring in 2005 December, Ray Escoffier augmented the GBT Auto-Correlation Spectrometer to work in cross-correlation mode. We report on our laboratory and sky tests of this mode. The executive summary of this technical memo is: (a) except for sporadic lag dropouts, the cross-correlation mode works splendidly and should be made available to the public; (b) the data product currently being delivered to the public via `SDFITS` is *not* correct and needs to be fixed as soon as possible. This should not preclude the release of this observing mode to the public since the hardware is behaving as it should.

Here is the executive summary of what is needed for cross-correlation spectropolarimetry to be a viable observational technique using the GBT Spectrometer:

1. In `SDFITS`, we believe that the van-Vleck and scale factors that are currently being applied to the correlation functions are correct. They should be retained.
2. In `SDFITS`, the symmetrization of the correlation functions and the calculation of power spectra are *not* correct. We suggest scrapping the Python code that currently does these tasks and replacing it using the algorithms described in §4 of this memo. In particular, heed the comment under §4.3.2.
3. In `SDFITS`, several important pieces of information are not present and should be inserted. Details are in §2.

Contents

1 Introduction **2**

2 A Few Outstanding Problems with SDFITS **3**

3 End-to-End Tests of the Spectrometer in Full-Stokes Mode: It Works! **3**

3.1 Laboratory Tests 4

3.2 On-Sky Telescope Tests 4

3.3 Astronomical Results! 4

3.4 Available Full-Stokes Modes 5

3.5 Lag Dropouts 5

4 A Proper Way to Produce Spectra from the Measured Autocorrelation and Cross-Correlation Functions: Verbal Description **7**

4.1 Obtaining the Whole Correlation Function (WCF) from the Measured Correlation Functions 8

4.1.1 The *invented value* for the ACF 8

4.1.2 The *invented value* for the CCFs 8

4.2 Cookbook for Obtaining the WCFs from the Measured Correlation Functions 9

4.3 Taking the FFT of the WCFs and Obtaining the Power Spectra 10

4.3.1 The Method 10

4.3.2 → **AN IMPORTANT POINT!!!** ← 10

5 Producing Spectra from the Measured Autocorrelation and Cross-Correlation Functions: IDL **11**

5.1 Producing the WCF from the Measured Correlation Functions 11

5.2 Taking the FFT of the WCFs and Obtaining the Power Spectra 12

6 The Current SDFITS **12**

6.1 Problems with Cross-Correlation in the Current SDFITS 12

6.2 The Van-Vleck Correction and Power Scaling: Done Correctly in the Current SDFITS 14

6.3 The Current SDFITS: Recommendations for Cross-Correlation 15

7 Conclusion **16**

A An IDL Fix for the Current Cross-Correlation SDFITS Output **17**

1. Introduction

The GBT Autocorrelation Spectrometer (ACS; hereafter the Spectrometer) was upgraded in 2005 by Ray Escoffier to produce cross-correlation output. This will now allow full-Stokes observations to be made with the GBT. It had been possible to do so using the Spectral Processor, and in some cases this backend might still be preferable due to its higher dynamic range, however a number of factors prompted the need to have a functioning cross-correlation mode for the Spectrometer: (*a*) the Spectral Processor is on its last legs (with replacement parts nearly exhausted); (*b*) the Spectrometer is the main backend for the GBT; (*c*) the Spectral Processor is limited in bandwidth to less than 5 MHz; (*d*) the Spectral Processor can handle only 4 simultaneous sky frequencies, but the limited number of channels usually means no more than 2 are useful; (*e*) Ray Escoffier, the designer of the Spectrometer, retired in 2005 December, placing a firm deadline for a functioning cross-correlation mode.

We therefore investigated the response of the new cross-correlation mode both in the laboratory and on the sky using linearly-polarized astronomical sources. In the process we discovered that the cross-correlation mode is functioning but that the data product provided by `SDFITS` does not store the cross-correlation spectra correctly. In addition, vital information is missing from these data files. § 2 directly addresses these problems. §§ 3.1–3.2 describe the lab and sky tests. We provide astronomical polarization calibration results in § 3.3 that demonstrate that the GBT Spectrometer cross-correlation mode is fully operational. We describe the proper way to obtain power spectra from the GBT Spectrometer output in §§ 4–5. Finally, we explain what is wrong with the current version of `SDFITS` in § 6.

2. A Few Outstanding Problems with SDFITS

We begin this memo with a request that `SDFITS` be upgraded. To obtain the results in this memo, we observed polarization calibrators using the observatory-supported observing scan type "Spider" (which we describe in §3.2). We tried to analyze our results using only the information stored in the `SDFITS` files. We were not able to do so. In order to produce the results presented in this report, we had to rely on information not stored in `SDFITS`. This is not acceptable.

We *strongly* suggest that effort be made to correct the following in `SDFITS`:

1. In the current `SDFITS`, the cross-correlation spectra are not correct. § 6 describes what's wrong and § 4 offers the recipe to create proper cross-correlation spectra.
2. In the current `SDFITS`, the (RA, DEC) and Equinox of the requested source position are not stored in `SDFITS`. This is a huge problem for any observing that uses position offsets. For example, for the Spider scan type, the user specifies a source, say 3C286, and the source's position in the specified equinox is tracked. As this is happening, the telescope is driven through the source position in such a way as to create a straight line in a specified reference frame. Now, in order to analyze the polarized beam properties, we need to calculate the offsets in azimuth and elevation from the source position. However, all that `SDFITS` provides are the equinox and position of the telescope's current position; we cannot calculate the offsets because the source position is not provided. This is not peculiar to the Spider scan: many people make maps in an offset mode.
3. In the current `SDFITS`, there is no indication of whether the high or low cal is used. We live and die by whether the high cal is set since we do not rely on the winking cal. We would really prefer if a binary keyword named HIGH_CAL were stored in `SDFITS`. This can be set based on the value of the HIGH_CAL header keyword of the `IF` instrument `FITS` file. It's certainly been the case that we have accidentally observed with the low cal set. It might be true that we have access to the band-averaged cal value stored in the variable `TCAL` in `SDFITS`, but to know whether the high cal was set for *any* receiver based solely on this value would require *a priori* knowledge of what the typical values of the high cal are as a function of frequency; for instance, when observing at the high-frequency end of X band, the high-cal temperatures are not very much larger than the low-cal temperatures: how would we know from the `TCAL` value alone whether the high cal was set? Moreover, the band-averaged cal temperature might not be an appropriate value for a particular observing method and the user might wish to access the cal tables directly; in this case, we would need to know whether the high cal was set.

We also make an operational request. During an observing session, a collaborator observing for us accidentally set the scan number back to 1 after finishing the 95th scan. There should be a keyword to `SDFITS` that allows one to pass a time-stamp; this time-stamp would disregard any scans made before the corresponding time. This could break the ambiguity. It would be better if the observing software prevented this from ever happening! It's tough to think of why one would *ever* want repeating scan numbers in the same observing session.

3. End-to-End Tests of the Spectrometer in Full-Stokes Mode: It Works!

We have taken two sets of polarized data with the Spectrometer and carried through the analysis to obtain all four Stokes spectra. One dataset was in the lab, the other on the telescope. As it happened, only the telescope data were useful. Below we discuss the lab tests, the on-sky telescope tests in which we correctly derive the astronomical position angle of the linearly polarized calibrator 3C286, and the "lag dropout" problem that was seen in our results. Our analysis demonstrates that the full-Stokes mode of the Spectrometer is fully operational.

3.1. Laboratory Tests

Rich Lacasse and Nathan Sharp produced test fixtures for the Spectrometer cross-correlation mode in the Jansky Lab at Green Bank in the summer of 2005. Rich conducted laboratory tests of the modes listed in Table 1 in 2005 May and 2005 August. We analyzed the resulting Spectrometer `FITS` files. We found the cross-power spectra of the correlated noise did not have the expected shapes and found a large phase slope across the 12.5 MHz band. It was unclear to us what was causing these anomalies, but our hypothesis was that the filters designed for the cross-correlation test fixtures were not closely matched as they are on the Spectrometer. Rich agreed this was likely the case. We decided it was necessary to test the modes on the sky.

Table 1: GBT Spectrometer Modes Tested in the Lab

Bandwidth (MHz)	No. of Samplers	No. of Quadrants	Levels
12.5	2	4	9
50	2	4	9
12.5	8	4	3
50	8	4	3
200	4	1	3
800	2	4	3

3.2. On-Sky Telescope Tests

Sky tests were conducted from the period 2005 October 12 through 2005 November 26. The observing method used to determine the polarization characteristics of the Spectrometer has come to be known as the *Spider pattern.* The pattern is made up of four continuously-sampled scans centered on the nominal pointing position that create an eight-armed symmetric pattern (hence, *Spider*); two of the scans are aligned with the azimuth and elevation axes and the other two scans are perpendicular to one another and oriented 45° to the azimuth-elevation axes. The extent of each scan is three half-power beamwidths (HPBWs) on each side of the pointing position. Karen O'Neil and Amy Shelton developed the `Spider.py` Python script, which we helped test for both the Spectrometer and Spectral Processor, and there is now an observatory-supported scan type named "Spider." The `Spider.py` script allows the user to customize the parameters for this pattern, including the number of legs (possibly making the name Spider a misnomer for anything but the default [if you believe the original looked anything like a spider in the first place!])

Table 2 lists the Spectrometer modes that we tested by running Spider scans on the archetypal linearly-polarized source 3C286 through the zenith to maximize the parallactic angle swing.

3.3. Astronomical Results!

We emphasize, first, that our results were obtained using the data product created by `SDFITS` and were corrected to produce proper cross-correlation spectra as described in § 4. The current `SDFITS` software doesn't work and must be modified; see §§ 2 & 6.

Table 2: GBT Spectrometer Modes Tested on the Sky

Bandwidth (MHz)	No. of Samplers	Levels
12.5	4	3
12.5	4	9
12.5	8	3
50	4	3
50	4	9
50	8	3
200	2	3
800	2	3

Appendix A describes how to modify current `SDFITS` output to produce proper results. Applying that fix, we analyzed the 3C286 Spider scans using our IDL package for polarization calibration.[1] Figure 1 shows the results from Spider scans on 3C286 at L band. The observed cross-correlation products show the exact sinusoidal variation that we expect to see as a function of parallactic angle for a linearly-polarized source! The Mueller matrix is determined by fitting these data and we find that the position angle of 3C286 is $32\overset{\circ}{.}2$. Similarly, in Figure 2 we see the same plot for observations made at C band. These results are within 1° of the actual position angle.

3.4. Available Full-Stokes Modes

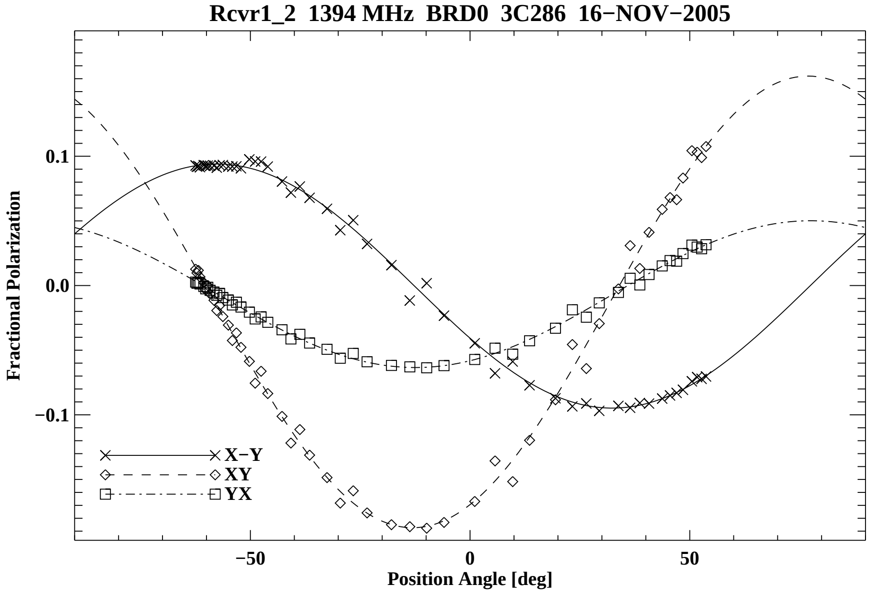
While the Spectrometer could in theory produce results for 32 samplers, or 16 sky frequencies, the IF is limited to 8 sky frequencies. This is a real shame for those wanting to do RRL work, but we understand this is unlikely to change any time soon. Given that all the modes of Table 1 appeared well-behaved in cross-correlation when tested in the lab and all of the modes of Table 2 have been shown to behave as expected with sky tests, we feel confident that *any* configuration of the GBT Spectrometer in full-Stokes mode will produce acceptable results. Of course, it would be our recommendation that any observer hoping to use a non-standard (or non-tested) Spectrometer configuration in full-Stokes mode actually test the polarization response by first running a set of Spider scans on 3C286. This will provide the Mueller matrix needed to properly correct one's data in addition to putting one's mind at ease that the Spectrometer will behave as expected.

3.5. Lag Dropouts

As part of project GBT05C-019, we made a very large (∼500 sq deg) map near the NCP using the Spectrometer in full-Stokes mode while observing the HI and OH lines simultaneously. This fully taxed the Spectrometer and in our understanding of how the Spectrometer is configured (which might well be too simplistic), we ended up employing bad LTA cards that were hidden in not-usually-found quadrants of the Spectrometer. Regardless of the true details of the physical hardware that produced them, we discovered that our map is peppered with what have come to be known as "lag dropouts." These are places where a grouping of correlator lags is offset from its neighbors. An algorithm for correcting these dropouts has been developed and will be implemented in `SDFITS` according to the specifications in Software Modification Request 15C306.

---

[1] It should be noted that this was all carried out inside of GBTIDL without any problems!



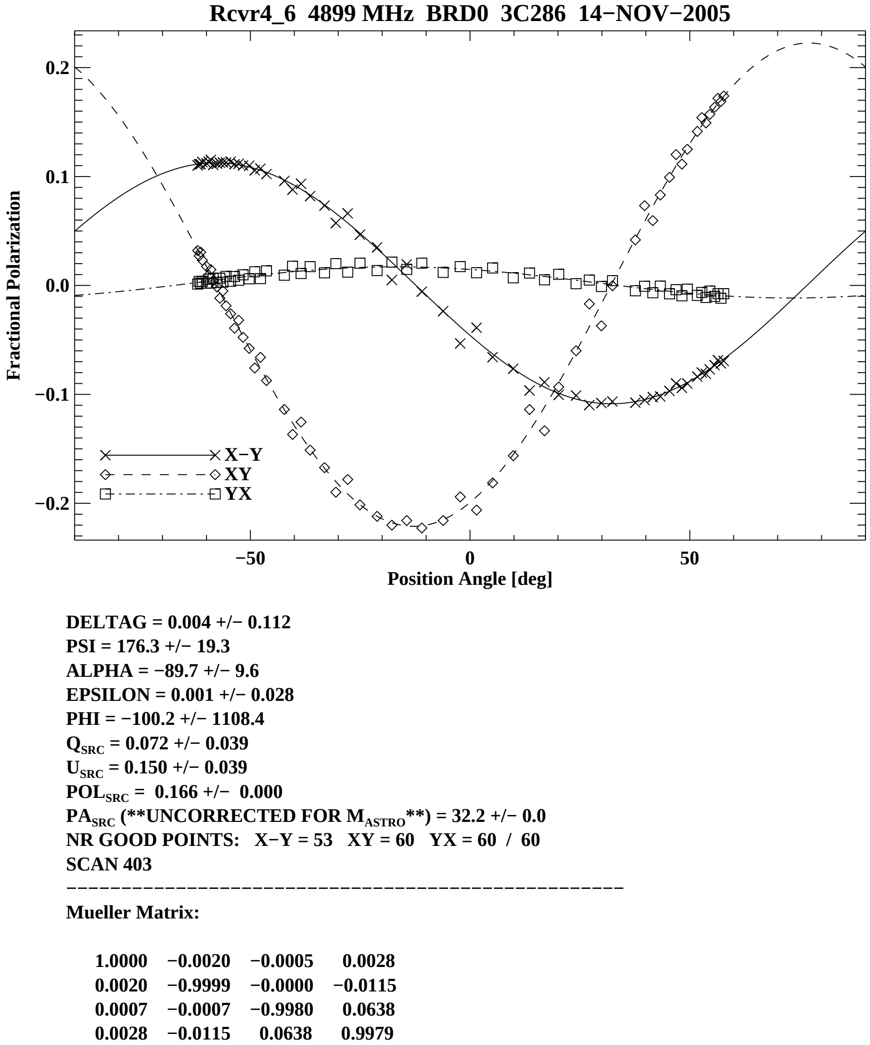
```
DELTAG = −0.001 +/− 0.089
PSI = −162.0 +/− 18.5
ALPHA = −89.4 +/− 9.2
EPSILON = 0.007 +/− 0.022
PHI = 9.7 +/− 180.0
QSRC = 0.060 +/− 0.032
USRC = 0.125 +/− 0.032
POLSRC =  0.139 +/−  0.000
PASRC (**UNCORRECTED FOR MASTRO**) = 32.1 +/− 0.0
NR GOOD POINTS:   X−Y = 51   XY = 52   YX = 52  /  52
SCAN 331
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Mueller Matrix:

   1.0000    0.0004    0.0141   −0.0024
  −0.0004   −0.9998   −0.0000   −0.0219
  −0.0127    0.0068   −0.9509   −0.3093
  −0.0067   −0.0208   −0.3094    0.9507
```

Fig. 1.— Spider scan results from polarization calibrator 3C286 at L band. The cross-correlation mode of the GBT Spectrometer is functioning impeccably.

We strongly reiterate the recommendation regarding bad lags in MR15C306. It suggests that the keyword `-fixbadlags` be made available that will, if set, correct the dropouts. However, for some observing situations—and engineering tests—the ability to recover the original measured correlation functions is crucial, even if they are incorrect. This is why it is important to be able to *not set* the `-fixbadlags` keyword, because then one can regain the original correlation functions by inverse-Fourier transforming the Stokes power spectra.

This brings us to proper and thorough documentation. It is important to make the algorithm for this fix accessible to users. Similarly, the `SDFITS` documentation should be more detailed and more easily accessible. We found it difficult to find information about what exactly is stored in `SDFITS` and what the ZEROCHAN value is actually storing. In order for an observer to recreate the measured lags output from the Spectrometer using the `SDFITS` spectra, it will be necessary to have accessible documentation outlining how the data are stored and how the spectra were created.



```
DELTAG = 0.004 +/− 0.112
PSI = 176.3 +/− 19.3
ALPHA = −89.7 +/− 9.6
EPSILON = 0.001 +/− 0.028
PHI = −100.2 +/− 1108.4
Q_SRC = 0.072 +/− 0.039
U_SRC = 0.150 +/− 0.039
POL_SRC =  0.166 +/−  0.000
PA_SRC (**UNCORRECTED FOR M_ASTRO**) = 32.2 +/− 0.0
NR GOOD POINTS:   X−Y = 53   XY = 60   YX = 60  /  60
SCAN 403
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Mueller Matrix:

    1.0000   −0.0020   −0.0005    0.0028
    0.0020   −0.9999   −0.0000   −0.0115
    0.0007   −0.0007   −0.9980    0.0638
    0.0028   −0.0115    0.0638    0.9979
```

Fig. 2.— Spider scan results from polarization calibrator 3C286 at C band.

4. A Proper Way to Produce Spectra from the Measured Autocorrelation and Cross-Correlation Functions: Verbal Description

In this section, we shall explain the details of creating power spectra from the autocorrelation and cross-correlation functions that are output by the GBT Spectrometer. First we define two terms: the *measured correlation function*, which is the set of numbers provided by the Spectrometer and which has entries for only positive lags; and the *Whole Correlation Function (WCF)*, which has both negative and positive lags.

Whole Autocorrelation Functions (WACFs) are inherently *symmetric* about zero lag. The Spectrometer produces two measured autocorrelation functions that have only positive lags; we denote these as XX_{lags} and YY_{lags}. In contrast, cross-correlation functions (CCFs) are inherently *nonsymmetric* about zero lag. The Spectrometer produces two cross-correlation functions that have only *positive* lags; we denote these as XY_{lags} and YX_{lags}. We can regard XY_{lags} as the *positive*-lag portion and YX_{lags} as the *negative*-lag portion of the Whole Cross-Correlation Function (WCCF).

The power spectrum is the Fourier transform of the WCF. However, Fourier transforms produce complex output, and power spectra must be real—they cannot be complex. Without delving into theoretical details, it is clear that we must arrange the WCFs and their corresponding Fourier transforms to produce real output.

4.1. Obtaining the Whole Correlation Function (WCF) from the Measured Correlation Functions

The measured correlation function values exist only for positive lags. We need to combine these positive-lag measurements into Whole Correlation Functions (WCFs) that have both negative and positive lags. The following two subsections discuss this process.

4.1.1. The invented value for the ACF

WACFs are real and are inherently symmetric about zero lag, i.e., they are Hermitian, so the Fourier transform is automatically real. Computationally, we must generate a symmetric WACF from the measured values at positive lags. The measured ACF has only positive lags with N channels: the lags range from 0 to $(N-1)\Delta\tau$, where $\Delta\tau$ is the sample interval (the delay time from one channel to the next). If we generate a symmetrized WACF by duplicating the nonzero positive-lag values at negative lags—and there are only $(N-1)$ nonzero positive lags—then the number of channels in the symmetrized WACF is not $2N$, but rather $(2N-1)$: the lags range from $-(N-1)\Delta\tau$ to $+(N-1)\Delta\tau$. This is awkward because we wish to retain an even number of channels for the FFT, and for other reasons.

Therefore, we must *invent* a new ACF value, which will be the value at lag $-N\Delta\tau$. Note that this value is *also* the value at lag $+N\Delta\tau$, because this is the nature of discrete Fourier transforms. The numerical value for this lag should be chosen to minimally impact the derived power spectrum. This means that the *invented* value at $-N\Delta\tau$ should be equal to the measured value at $(N-1)\Delta\tau$; otherwise there will be a high-frequency ripple across the power spectrum. With this *invented* value, the symmetrized WACF has $2N$ values and we can employ the FFT. Note that N of these values were measured and one was *invented*; the symmetrized WACF contains $N+1$ independent numbers. The FFT of the symmetrized WACF is real, so the values at positive frequencies are equal to those at negative frequencies. Because $N+1$ independent numbers went into the symmetrized WACF, the power spectrum also has $N+1$ independent numbers. These correspond to frequencies 0 to $\pm N\Delta f$, where $\Delta f = \frac{1}{N\Delta\tau}$.

4.1.2. The invented value for the CCFs

A WCCF has no inherent symmetry about zero lag, so the Fourier transform is complex. The two cross-power spectra are the real and imaginary portions of the Fourier transform. Computationally, we must arrange the $2N$ measured values of the two CCFs, each of which has only positive lags, as a $2N$-long WCCF with both negative and positive lags. To achieve this we follow a procedure similar to that for the WACF described above.

Specifically, we generate a WCCF by considering the N values of XY_{lags} as positive lags, running from 0 to $(N-1)\Delta\tau$, and the N values of YX_{lags} as negative lags, running from 0 to $-(N-1)\Delta\tau$. The zero-lag values of XY_{lags} and YX_{lags} are, in principle, identical; in practice they might be a bit different because of hardware imperfections,[2] so in the WCCF we make the zero-lag channel equal to the average of the zero-lag channels of XY_{lags} and YX_{lags}. Again, as with the WACF, the number of channels in the WCCF is not $2N$, but rather $(2N-1)$; the lags range from $-(N-1)\Delta\tau$ to $(N-1)\Delta\tau$.

---

[2] In this treatment, we sweep this difference under the rug by averaging. In practice, if the difference is nonzero it's a hardware problem that should be investigated! But also, if the difference is small, it doesn't matter much.

So again, as with the WACF, we must *invent* a new WCCF value, which will be the value at lag $-N\Delta\tau$ (which, again, is also the value at lag $+N\Delta\tau$). Again, this *invented* value should minimally impact the derived cross-power. The proper choice is the average of the highest lag values (the $(N-1)\Delta\tau$ values) of the measured CCFs, XY_{lags} and YX_{lags}.

4.2. Cookbook for Obtaining the WCFs from the Measured Correlation Functions

We can distill the above discussion to the following simple rules for combining two measured correlation functions, each of which has only positive lags, into what we are calling the Whole Correlation Function, which has both negative and positive lags. The WCF has $2N$ channels, with indices running from 0 to $(2N-1)$ and lags running from negative to positive delay. Thus, index 0 of the WCF array has lag $-N\Delta\tau$, index N has lag 0, and index $(2N-1)$ has lag $+(N-1)\Delta\tau$.

1. The WCF is generated from two correlation functions, one regarded as having positive and one as negative lags (denoted as POS and NEG respectively). Each array has N elements with lags increasing in absolute value with index. Thus, for POS, index 0 has delay 0 and index $(N-1)$ has lag $+(N-1)\Delta\tau$; for NEG, index 0 has delay 0 and index $(N-1)$ has lag $-(N-1)\Delta\tau$. We generate these arrays from the measured correlation functions as follows:
    (a) For a measured ACF of, say, XX_{lags}, $POS = XX_{lags}$ and $NEG = XX_{lags}$.
    (b) For a measured CCF, $POS = XY_{lags}$ and $NEG = YX_{lags}$.
2. Set the zero-lag value of WCF equal to the mean of the zero-lag values of POS and NEG.
3. Set the $(N-1)$ nonzero positive-lag values of WCF equal to the $(N-1)$ nonzero lag values of POS.
4. Set the $(N-1)$ nonzero negative-lag values running from $-(N-1)\Delta\tau$ to $-\Delta\tau$ of WCF equal to the $(N-1)$ nonzero lag values of NEG.
5. Set the $-N\Delta\tau$ lag value of WCF equal to the mean of the $(N-1)\Delta\tau$ values of POS and NEG.

We can represent this prescription as follows:

$$WCF = \left[\frac{POS[N-1]+NEG[N-1]}{2},\ NEG[N-1:1],\ \frac{POS[0]+NEG[0]}{2},\ POS[1:N-1]\right] \quad (1)$$

where the notation $NEG[N-1:1]$ denotes the reverse of the array NEG excluding the zeroth channel. Note that the *invented* value, discussed above, is $(POS[N-1]+NEG[N-1])/2$. We find that the whole autocorrelation functions are then:

$$\begin{aligned} WACF_{XX} &= [\ XX_{lags}[N-1],\ XX_{lags}[N-1:1],\ XX_{lags}[0:N-1]\ ] \quad (2)\\ WACF_{YY} &= [\ YY_{lags}[N-1],\ YY_{lags}[N-1:1],\ YY_{lags}[0:N-1]\ ] \quad (3)\end{aligned}$$

And the whole cross-correlation function becomes:

$$\begin{aligned} WCCF = \Bigg[\ & \frac{XY_{lags}[N-1]+YX_{lags}[N-1]}{2},\ YX_{lags}[N-1:1],\\ & \frac{XY_{lags}[0]+YX_{lags}[0]}{2},\ XY_{lags}[1:N-1]\ \Bigg] \quad (4)\end{aligned}$$

4.3. Taking the FFT of the WCFs and Obtaining the Power Spectra

4.3.1. The Method

Having generated a WCF, the proper way to extract its power spectrum is to take the FFT. This produces $2N$ power values for frequencies ranging from $\frac{-N}{N\Delta\tau}$ (which is, of course, equal to $\frac{-1}{\Delta\tau}$!) through 0 (DC) to $\frac{+(N-1)}{N\Delta\tau}$. There are $(N+1)$ independent powers at frequencies 0 to $\pm N\Delta f$, where $\Delta f = \frac{1}{N\Delta\tau}$. Finally, perform the following manipulations:

1. If it's a WACF, *sum* (not average!) the positive and negative frequencies. Given that the powers at $-N\Delta f$ and $+N\Delta f$ are equal (because of the nature of discrete Fourier transforms), the sum is simply twice the power at $-N\Delta f$. This provides powers at $(N+1)$ frequency values (ranging from 0 to $N\Delta f$), as befits the presence of $(N+1)$ independent numbers in the WACF. Do not sum the zero-frequency values with itself.
2. If it's a WCCF, the power spectrum is complex. Because the input numbers in the WCCF are real, the power spectrum is Hermitian.
    - (a) The XY cross-power spectrum is obtained from the *real* part of the complex spectrum exactly as for the ACF above, namely *sum* the negative and positive frequency powers.
    - (b) The YX cross-power spectrum is obtained from the *imaginary* part of the complex spectrum by *differencing* the negative and positive frequency parts. These two parts are the negatives of each other because the complex power spectrum is Hermitian, so taking the difference is like taking the sum of the absolute values.

4.3.2. → ***AN IMPORTANT POINT!!!*** ←

Above we italicized the words *sum* and *difference.* Why?

With the complex power spectrum, when summing or differencing, the zero-frequency power is not summed with itself, so all of the nonzero powers get multiplied by two with respect to the spectrum for only positive frequencies. *This factor-of-two scaling* is *the correct scaling.* So, for example, for a WACF it is not correct to take only the zero and nonzero positive-frequency portions of the power spectrum and discard the nonzero negative-frequency portions, because doing so incorrectly discards almost half the power—in particular, it demagnifies the nonzero-frequency components by a factor of two with respect to the DC component.

Moreover, for the WCCF, the zero-frequency value of the real component of XY_{lags} (which is symmetric about zero frequency) can be nonzero. In contrast, the imaginary component of YX_{lags} is odd and its zero-frequency component *must* be zero.

Here, we have made a big deal about getting the zero-frequency (DC) power correct with respect to powers at nonzero frequencies. Formally this is important, but in practice it is not very important because the DC power is generally a worthless quantity for the science. If the system is AC-coupled, then there should be no DC power; the presence of DC power in an AC-coupled system indicates a DC offset in the digital correlator, probably in the input A/D converters. If the system is DC-coupled, then the presence of "red noise," which tends to be environmentally sensitive and therefore time-variable, makes the DC power unstable. In either case it is usually scientifically worthless.

Nevertheless, DC power *must be saved* so that one can retrieve the original Whole Correlation Function (WCF) values by taking the inverse Fourier transform. If the correct DC power value is not available, then the calculated WCF values will be displaced from zero by an offset (recall that the DC power value is proportional to the sum over all correlation function values). The ability to recover the original correlation function is particularly important for people needing to fix lag dropouts on their own.

Then comes the question: we wish to present the astronomer with N power values, not $(N+1)$, so in the astronomer's spectrum we must discard one of the channel values. The only two logical choices are channel 0 or channel N. From the standpoint of mathematical elegance and technical purity, we would prefer to retain channel 0 in the N-point astronomer's spectrum because, when taking the inverse transform to obtain the WCF, its value has far more influence than the channel N value. However, given that the power for channel 0 is scientifically meaningless, and that the astronomer's spectrum is, after all, made for pure scientific utility,[3] we should forego elegance and go for practicality.

Thus, we recommend that the N-point astronomer's spectrum consist of the power values at frequencies Δf to $N\Delta f$. It is very important to make sure that the `SDFITS` header parameters provide the correct r.f. frequency—it's easy to make a mistake and be one channel off! The zero-frequency power should be saved in `SDFITS` in the ZEROCHAN value and it should be made extremely clear in all supporting documentation that this is the zero-frequency power channel and *not* the zero-lag correlator output.

5. Producing Spectra from the Measured Autocorrelation and Cross-Correlation Functions: IDL

5.1. Producing the WCF from the Measured Correlation Functions

As an example of how to implement the method of § 4, we outline how we obtain power spectra in IDL. To head off any confusion, we note that IDL's `FFT` function requires that the input correlation function be ordered such that the negative lags follow the positive lags. Therefore, our general prescription for the WCF in equation (1) would be ordered as:

$$WCF_{\rm IDL} = \left[\frac{POS[0]+NEG[0]}{2},\ POS[1:N-1],\ \frac{POS[N-1]+NEG[N-1]}{2},\ NEG[N-1:1]\right] \quad (5)$$

where the notation $NEG[N-1:1]$ denotes the reverse of the array NEG excluding the zeroth channel. Note that the *invented* value, discussed above, is $\frac{POS[N-1]+NEG[N-1]}{2}$.

Let the measured correlation functions, each with N positive lags ranging from 0 to $(N-1)\Delta\tau$, be stored in variables names `XX_lags`, `YY_lags`, `XY_lags`, and `YX_lags`. The first two are the ACFs and the latter two the CCFs. After re-ordering, we can translate equations (2) & (3) into the IDL language and the whole autocorrelation functions can be represented as:

```
WACFXX = [ XX_lags[0:N-1], XX_lags[N-1], REVERSE(XX_lags[1:N-1]) ]
WACFYY = [ YY_lags[0:N-1], YY_lags[N-1], REVERSE(YY_lags[1:N-1]) ]
```

And the whole cross-correlation function would be:

```
WCCF = [ 0.5*(XY_lags[0] + YX_lags[0]), XY_lags[1:N-1], $
         0.5*(XY_lags[N-1] + YX_lags[N-1]), REVERSE(YX_lags[1:N-1]) ]
```

---

[3] Moreover, if channel 0 is included in the astronomer's spectrum, newbies will plot this value, which is often the largest number in the spectrum, thus causing the spectral powers to be compressed into a useless squiggle.

5.2. Taking the FFT of the WCFs and Obtaining the Power Spectra

First, we define the Fourier transform of the WCF:

```
FFT_WACFXX = FFT( WACFXX )
FFT_WACFYY = FFT( WACFYY )
FFT_WCCF = FFT( WCCF )
```

Here, each of the FFTs has $2N$ channels and is Hermitian; the zeroth channel has zero frequency (DC).

For the autocorrelation power spectra derived from the WACF, the numbers in the power spectrum are all real, but the computed imaginary parts will be nonzero because of roundoff error; we want to get rid of these, of course, so the power spectra are obtained by taking the real part:

```
PSXX[0]= REAL_PART( FFT_WACFXX[0] )
PSXX[1:N] = REAL_PART( FFT_WACFXX[1:N] + REVERSE(FFT_WACFXX[N:2*N-1]) )

PSYY[0]= REAL_PART( FFT_WACFYY[0] )
PSYY[1:N] = REAL_PART( FFT_WACFYY[1:N] + REVERSE(FFT_WACFYY[N:2*N-1]) )
```

where we have followed the prescription of § 4.3. Note that the power spectrum contains $N + 1$ values as discussed. Operationally, `SDFITS` strips the zeroth channel and stores its value separately from the remainder of the spectrum.

For the cross-correlation power spectra, we take the real part of the WCCF to obtain the XY power spectrum:

```
PSXY[0]= REAL_PART( FFT_WCCF[0] )
PSXY[1:N] = REAL_PART( FFT_WCCF[1:N] + REVERSE(FFT_WCCF[N:2*N-1]) )
```

and the imaginary part to obtain the YX power spectrum:

```
PSYX[0]= IMAGINARY( FFT_WCCF[0] )
PSYX[1:N] = IMAGINARY( FFT_WCCF[1:N] - REVERSE(FFT_WCCF[N:2*N-1]) )
```

Note that `PSYX[0]` should be zero.

6. The Current SDFITS

In 2006 May, a dialog was begun with Joe Wolf, an undergraduate summer student working with Prof. Tom Devlin at Rutgers. They were highly suspicious of the validity of the cross-correlation spectra produced by `SDFITS`. Until this point, we did not use the SDFITS spectra; rather, we had directly manipulated the correlator lags as stored in the Spectrometer instrument `FITS` files in the manner detailed in §§ 4–5.

6.1. Problems with Cross-Correlation in the Current SDFITS

Figure 3 compares our spectra with the SDFITS spectra for an L-band observation during which the correlated cal was injected. The correlated cal provides significant cross-correlated power

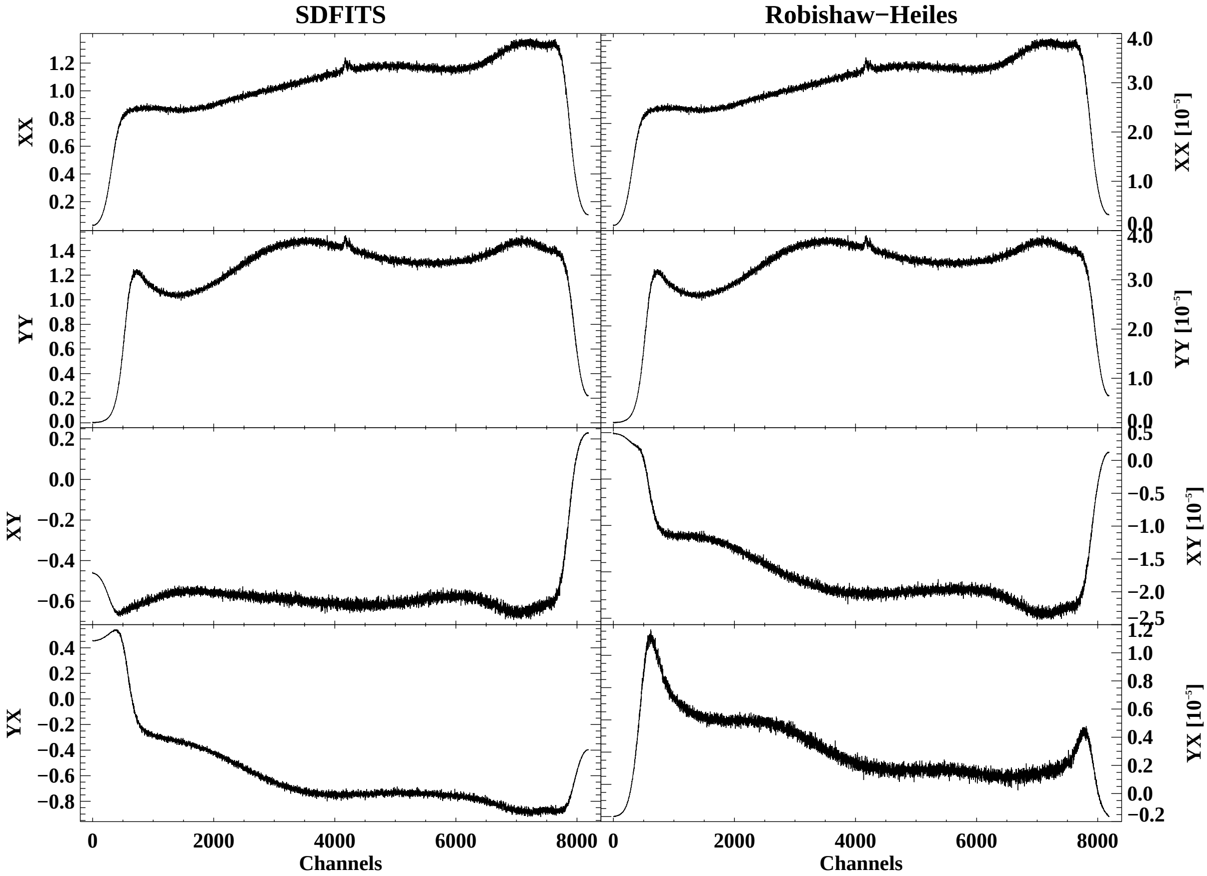


Fig. 3.— Correlation spectra with the correlated cal injected. (*Left*) `SDFITS` output. (*Right*) Assembled by us using raw lags from the Spectrometer `FITS` file. Notice that: the autocorrelation spectra are identical; the cross-correlation spectra differ; the `SDFITS` cross-correlation spectra are ill-behaved. The differing scales and the zero offsets are explained in § 6.2

that can be easily measured. It can be seen that the autocorrelation spectra look identical, but the cross-correlation spectra are completely different. Something is clearly wrong with the `SDFITS` cross-correlation spectra. We attempted to search through the Python files in `/home/sparrow` to try to parse what `SDFITS` might be doing, but we weren't able to find the source code that assembled the correlation functions or performed the FFTs.

However, thanks to Joe Wolf's discussions with Bryan Mason and Bob Garwood, he was able to parse that `SDFITS` is likely doing the following to obtain the cross-correlation spectra. First, `SDFITS` is calculating *autocorrelation functions* for each set of cross-products:

$$CCF_{XY} = [\, XY_{lags}[N-1],\ XY_{lags}[N-1:1],\ XY_{lags}[0:N-1]] \tag{6}$$

$$CCF_{YX} = [\, YX_{lags}[N-1],\ YX_{lags}[N-1:1],\ YX_{lags}[0:N-1]] \tag{7}$$

Next, these autocorrelation functions are FFTed and their real parts are taken. As a test, in Figure 4 we perform these steps to the raw Spectrometer output directly and compare with the `SDFITS` output. It's a match. *It is absolutely incorrect to calculate the cross-correlation spectra in this way!* Appendix A offers the IDL code that we used to properly fix the `SDFITS` cross-correlation spectra. Figure 5 compares the corrected `SDFITS` output with the spectra produced from the raw Spectrometer correlations via the method of §§ 4–5.

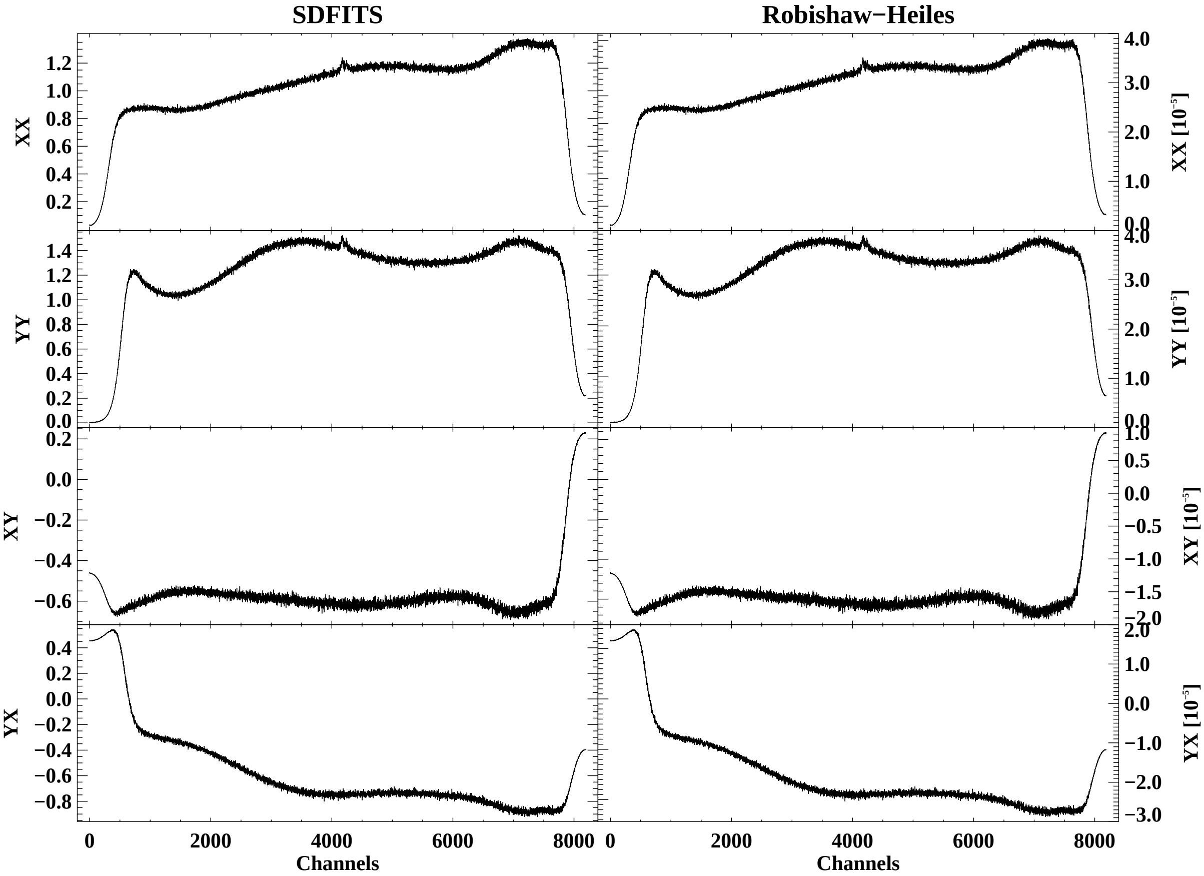


Fig. 4.— Correlation spectra with the correlated cal injected. (*Left*) `SDFITS` output. (*Right*) Assembled by us using raw lags from the Spectrometer `FITS` file, but in such a way as to reproduce the `SDFITS` output. We understand what `SDFITS` is doing, and it's wrong.

6.2. The Van-Vleck Correction and Power Scaling: Done Correctly in the Current SDFITS

When we calculated power spectra directly from the measured correlation functions according to the prescriptions of §§ 4 & 5, we were not able to obtain proper results because our spectra were offset from a proper zero point.[4] This is because we did not apply a van-Vleck clipping correction. In contrast, `SDFITS` does do a van-Vleck correction, and when we modified the current `SDFITS` output using the algorithm of Appendix A, we obtained proper zero offsets. Moreover, we obtained proper astronomical results on 3C286! This could not occur unless two conditions are satisfied:

1. The van-Vleck correction was correctly applied.
2. The correlation functions were properly scaled for total power.

E-mail conversations with Bob Garwood give us confidence that these actions are being done correctly. In particular, the `SDFITS` van-Vleck correction is that given in two technical memos written by Fred Schwab (Schwab 2001, 2002). And the scaling of the correlation functions is based on the value of the zero-lag autocorrelation channels, as it should be. We therefore conclude that the above two actions are being done correctly in the current `SDFITS`.

---

[4] You can see this offset in Figure 3, where our cross-power spectra do not fall to zero at the bandpass edges.

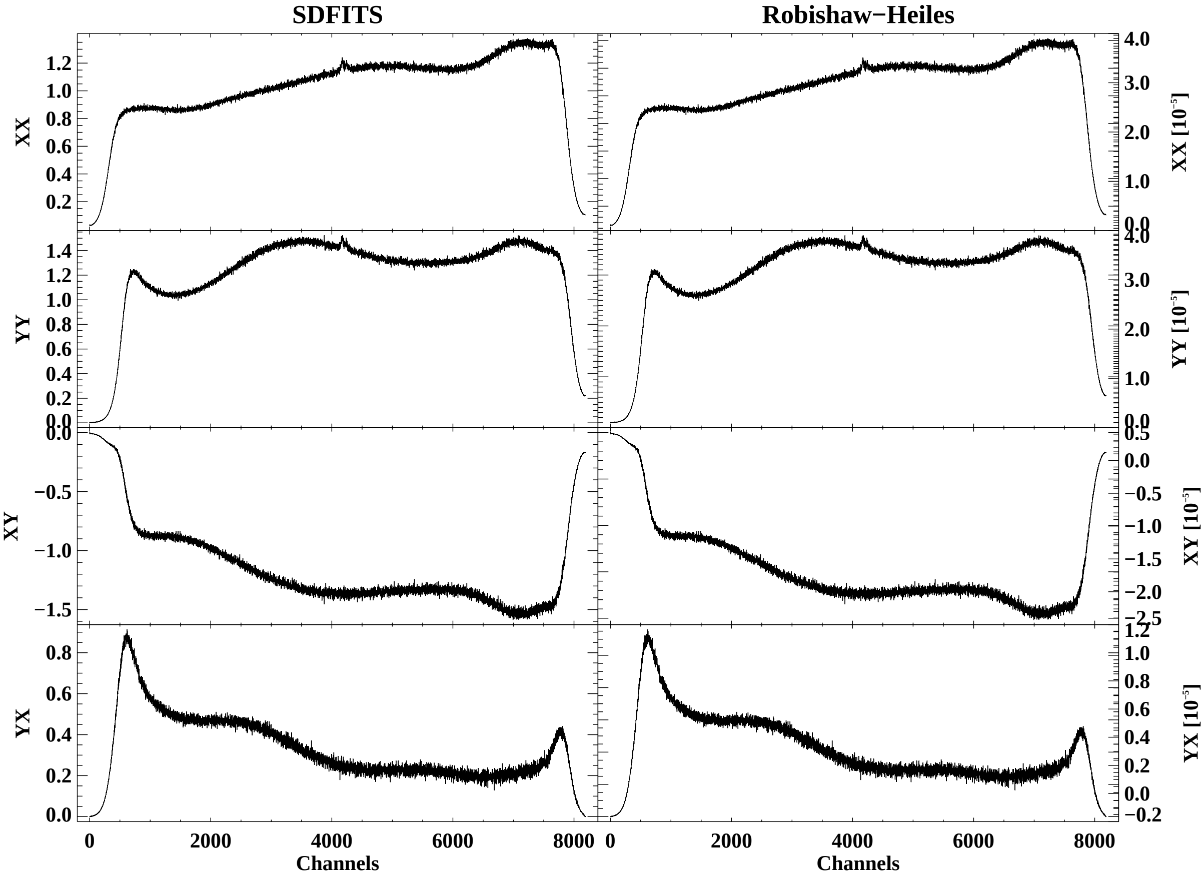


Fig. 5.— Correlation spectra with the correlated cal injected. (*Left*) Corrected `SDFITS` output. (*Right*) Assembled by us using raw lags from the Spectrometer `FITS` file. We now know how to correct the `SDFITS` output. Notice how the `SDFITS` cross-correlation spectra now properly go to zero at the edges of the bandpass.

6.3. The Current SDFITS: Recommendations for Cross-Correlation

The processing of measured correlation functions to obtain power spectra has the following sequential steps:

1. Applying the van-Vleck correction to the measured correlation function. This is being done correctly and the current Python code should be retained.

2. Scaling this corrected correlation function so that it reflects the total power. This is being done correctly and the current Python code should be retained.

3. Combining these scaled, corrected functions to obtain a proper Whole Correlation Function (WCF). This is not being done correctly. The current Python code should be scrapped and rewritten according to the prescription in § 4.

4. Fourier transforming the WCS to obtain the power spectrum. This is not being done correctly. The current Python code should be scrapped and rewritten according to the prescription in § 4.

7. Conclusion

The full-Stokes mode of the GBT Spectrometer is fully operational and should be made publicly available. The cross-correlation spectra produced by `SDFITS` are incorrect. The `SDFITS` Python code should be made to implement the method outlined in § 4. Additional astronomical information needs to be added to `SDFITS` as outlined in § 2. The authors will be happy to offer assistance to the GBT staff in their effort to ready this observing mode for use by the community.

We'd like to thank the GBT staff for their assistance. Rich Lacasse, Ray Escoffier, and Nathan Sharp worked extremely hard to get the cross-correlation mode working. Karen O'Neil, Paul Marganian, Amy Shelton and Mark Clark were extremely helpful in updating our Spectral Processor observing methods for use with the Spectrometer. We'd especially like to thank Carl Bignell for finding commissioning time to allow us to perform sky tests of the Spectrometer performance. Thanks to Joe Wolf, Tom Devlin and Brian Mason for their investigations into possible explanations for the SDFITS cross-correlation problem. Finally, we'd like to thank the Spectral Processor for all the hard work it did for us while we waited for the Spectrometer. This research was supported in part by NSF grant AST-0406987. This work was also supported in part by the NSF award GSSP06-0003 from the NRAO to TR.

REFERENCES

Schwab, F. R. 2001, Optimal Quantization Functions for Multi-Level Digital Correlators

Schwab, F. R. 2002, Van Vleck Correction for the GBT Correlator

A. An IDL Fix for the Current Cross-Correlation SDFITS Output

In order to properly calibrate our sky tests, we needed the cross-correlation spectra in `SDFITS` to be correct. The following brief IDL code was used to produce the corrected spectra. The `SDFITS` cross-correlation spectra are represented by `xy` and `yx` and their corresponding zero-channel powers by `xy_zero_channel` and `yx_zero_channel`.

```
; HOW MANY CHANNELS IN A SPECTRUM...
nchan = N_elements(xy)

; BUILD ARRAY OF SPECTRA FROM WHICH THE INCORRECTLY-DETERMINED
; SDFITS AUTOCORRELATION FUNCTIONS CAN BE RETRIEVED...
fft_acf_xy = [xy_zero_channel,xy,(reverse(xy))[1:*]]
fft_acf_yx = [yx_zero_channel,yx,(reverse(yx))[1:*]]

; TAKE THE INVERSE FFT TO GET THE AUTOCORRELATION FUNCTIONS...
acf_xy = fft(fft_acf_xy,/INVERSE,/DOUBLE)
acf_yx = fft(fft_acf_yx,/INVERSE,/DOUBLE)

; BUILD THE CORRECT CROSS-CORRELATION FUNCTION...
ccf = [0.5*(acf_xy[0] + acf_yx[0]),$
       acf_xy[1:nchan-1],$
       0.5*(acf_xy[nchan-1] + acf_yx[nchan-1]),$
       reverse(acf_yx[1:nchan-1])]

; TAKE THE FFT OF THE CROSS-CORRELATION FUNTION...
fft_ccf = fft(ccf,/DOUBLE)

; EXTRACT THE REAL AND IMAGINARY PARTS...
fft_xy = real_part(fft_ccf) ; symmetric
fft_yx = imaginary(fft_ccf) ; antisymmetric

; TAKE CHANNELS 1 THROUGH NCHAN...
fixed_xy = 2*fft_xy[1:nchan]
fixed_yx = 2*fft_yx[1:nchan]
```