The Enhanced VLBI Data Simulator *Enoise* : The Current Status and Testing Results

C. Y. Kuo[1], M. Zheng[1]

[1] *Academia Sinica Institute of Astronomy and Astrophysics, P.O. Box 23-141, Taipei 10617, Taiwan*

1. INTRODUCTION

Enoise is the enhanced version of the VLBI data simulator anoise written by Geoff Crew at MIT Haystack observatory. The main purpose of these simulators is to generate synthetic VLBI data that are sampled at different sampling rates and one can use these data to test whether DiFX can accurately correlate data from traditional VLBI stations and that from the phased ALMA, which adopts a different sampling rate than normal VLBI receivers. As a simulator for preliminary testing, anoise assumes zero baseline between simulated stations. In addition, since it is time-consuming to generate random numbers to simulate real signals, in order to save time, one usually takes a short cut and uses anoise by first simulating a short scan (e.g. 0.032 second) and extending the observation to the desired length (e.g. 4 seconds) by simple duplication of the first scan.

Generalizing anoise to simulate VLBI data with non-zero baseline is a non-trivial problem because of several complexities. First of all, in the non-zero baseline situation, due to the time-varying geometric delay, one cannot simply duplicate data generated in a short scan to a longer one in order to save time. This implies that one will have to generate a random number for every signal sample. As a result, simulating a 10 minute observation will take about 600 hours if one uses a standard linux machine and require a random number generator with the high quality. Such a long simulation time for a short observation is unrealistic, and thus we have to find a new way to produce random numbers efficiently.

The second change we need to face is to the time-varying delay. The delay of a signal arriving at a VLBI station includes not only the geometric delay, but also delays due to inaccurate Earth Orientation Parameters (EOPs), general relativity light bending around solar system bodies, and Earth tides, etc. Therefore, the delay model is sophisticated and subtle differences between the delay models used in the simulator and DiFX can introduce residual delays in the fringe-fitting results that can be difficult to interprete. In addition, since delay does not vary linearly with time, it is not easy to implement time-varying delay in the simulator with high precision. Finally, at submm/mm wavelengths, the fringe rate is significantly higher than that at the cm wavelengths for long baselines. Therefore, implementing phase rotation in the simulator with high precision is also important to ensure the residual fringe rate of the visibility data is sufficiently smaller than typical values in real VLBI observations.

A relatively easy way (suggested by Adam Deller) to deal with these challenges is to code the simulator by taking advantage of the existing modules in DiFX that are responsible for signal processing (e.g. delay compensation, fringe rotation, fractional sampling correction etc.) before cross-correlation and arranging them in the reverse order of the normal signal processing. The biggest advantage of this kind of simulator is that it can simulate VLBI data in a significant shorter timescale than the anoise-type simulator because generating a large random number to simulate pseudo-analog signal is not needed. In addition, it is easy to achieve high precision in putting delays in the signals and inserting phase rotation because the error is negligible. However, since this kind of simulator is essentially just a reversed version of DiFX, it is not easy to diagnose the problems in DiFX. Since our main goal is to examine the reliability and accuracy of DiFX to

correlate non-zero baseline VLBI data, a simulator that does not just reverse the signal processing of DiFX such as enoise is needed.

2. The Non-Zero Baseline Simulator Enoise

The non-zero baseline VLBI data simulator enoise is written based on many existing functions in anoise, but the entire code structure has been modified to incorporate the non-zero baseline effects into the simulated data. Enoise basically follows anoise to simulate signals in the time domain and assumes that the signal has a flat power spectrum (e.i. white noise). In enoise, the signal processes such as band filtering, sampling, and outputing the data into VDIF format are the same as anoise, but it also takes the two primary non-zero baseline effects into account. The first effect we consider is signal delay caused by geometric delay, inaccurate Earth Orientation Parameters (EOPs), general relativity light bending around solar system bodies, and Earth tides, etc. The second effect we include is phase rotation due to the frequency downconversion in the receiver system.

To include these non-zero baseline effects, we first allow enoise users to choose their favorite source (e.g. M87 and Sgr A* etc.) to observe, and calculate the time delay based on stations selected by the users (e.g. ALMA, SMA, GLT, SMTO, SPT, or the Iram 30-m telescope). The calculation of the total time delay due to all the effects mentioned above is not trivial because the models of EOPs and Earth tides can be sophisticated and different models can produce slightly different delays. In order to avoid confusion in interpreting the correlation results, we decide to use the same delay model that DiFX uses, and this requires us to use Calcserver in DiFX to generate the required delay model (contained in the .im file). The advantages of doing this are that we do not need to build the complicated delay model from scratch and no residual delay will arise due to differences of the delay models used by enoise and DiFX. On the other hand, we will need scripts external to enoise to run Calcserver in order to obtain the delay model. Therefore, the entire simulation will take a few steps and uses different programs.

2.1. Random Number Generation and Delay Insertion

To save the time of random number generation, instead of generating a random number for every signal sample, we generate a large number (i.e. $3.2\text{x}10^{10}$) of random numbers first and save them in an external file in text format. Whenever the simulator needs a random number, it just directly takes a number from the random number reservoir and continues to use the numbers in the file until the random numbers get exhausted. When all numbers in the external files have been used, the simulator will go back to the beginning of the file and use the numbers again. Note that "re-using" the random numbers in the external file is appropriate in the simulator because the time for a random number in the reservoir to be reused is at least a few times longer than the typical integration in DiFX.

We incorporate delays in the simulated signals by first generating empty arrays that represent the time sample before receiving the signals. Each array represents to one sec simulated data and the number of elements in the array (*sps*; sample per μsec) corresponds to the over-sampling rate we use to simulate the data. Here, oversampling means that we are trying to sample the signal in a rate that is higher than the normal sampling rate in a receiver, and the oversampled data represent pseudo-analog signals from the sky that will be sampled in the receiver. Each element in the array corresponds to a signal from the sky at a particular time stamp. The higher *sps* is, the better accuracy and better time resolution we have to simulate

the analog signal.

After the empty arrays are generated, we calculated the delay of the signal (i.e. the random numbers from the external file) and allocate the signal to the correct time stamp according to their corresponding delay. When we calculate the signal delay, we extract the delay information from the .im file (one of the outputs of VEX2DIFX in DiFX) and express the delay as a function of time in term of a polynomial for every second of the simulation. Within each second, we calculate the delay with the polynomial and include terms up to the 2nd order. The terms beyond 2nd order are usually so small that they can be ignored within one second.

The allocation of the sample to the corresponding time stamp is performed with approximation. In the simulation, we assume that the signal delay between time t and t + δt is constant, where $\delta t = 1/(\text{sps*R})$ μsec (R is the delay rate calculated at time t) and ignore delay variation within t. Note that this approxiamtion is necessary because delay is continuously varying with time and it is not possible to allocate the delay with infinite precision. The accuracy of the signal allocation is limited by the time resolution (i.e. 1/sps μsec) we have in the simulated data. After the signal allocation to the right time stamp, the simulated data is passed to a band filter which only allow the signal within certain bandwidth to pass through. This part of the code is the same as anoise.

2.2. Implementing inverse Fringe rotation

The next non-zero baseline effect we need to deal with is phase rotation due to the joint effect of frequency down-conversion and time delay. This is the effect that the fringe-rotation function in DiFX will correct for. When the sky signals pass through the front end of the receiver of an antenna, the frequency of the signal usually gets downconverted to a lower frequency. Now, considering two streams of signals that pass through a pair of antennas A and B respectively and the the signal arriving at antenna B has a time delay τ relative to antenna A. In this case, when one signal has arrived at station A at time t_p with its frequency (ν) getting downconverted to $\nu - \nu_{LO}$, the signal that has a time lag τ and will arrive at station B at time $t_p + \tau$ will still travel in the sky at the frequency ν. As a result, a phase offset $2\pi\nu\tau$ will be introduced for the delayed signal at time $t_p - \tau$ and this phase will change with time (i.e. phase rotation) depending on the delay rate (see detail explanation in Chapter 6 in Thompson, Moran, Swenson 2001 and Chapter 4 in PhD thesis of Adam Deller).

Implementing this phase rotation in the simulated signal is non-trivial because our signals are real in nature and we cannot include the phase change into the signal directly. Nor can we include the phase rotation in the frequency domain of the signal where the signal are complex in nature. This is because the phase rotation is a function of time and directly introducing the phase change in the complex signal in the frequency domain will make the signal non-Hermitian and the signal will not become real after it gets fourier transformed back to the time domain.

We solve the phase rotation problem by applying Hilbert Transform to our simulated signal (after filetering) and convert the signal to analytic signal which is the complex representation of a real signal. We then multiple the phasor $e^{i2\pi\nu\tau}$ to the analytic signal and take its real part after the phase rotation. Note that this process is performed for every time sample in the simulated signal. The signal processing after this step is basically the same as anoise and we will not elaborate the details here.

2.3. Unresolved Issue

In this α version of enoise release, there are still two minor issues that haven't been fully resolved. The first issue is that while we have designed a new way to reduce the time for random number generation, a 4 second simulation that include ALMA will still take several hours to finish and this is still too long. An apparent way to speed up the simulation is to parallelize the code and utilize the computing power of mulple CPUs in a cluster. We are currently working on this enhancement and the future release of the new version of enoise will include this functionality.

The second problem that needs to be solved is that the algorithm for delay insertion is not yet perfect and residual delay after correlation still cannot be ignored although the typical value (a few nanosecond) is small. At this moment, we are still examining the algorithm and the code to search for this little and hidden bug. We aim to find out this bug and reduce the residual delay to at lease a few times smaller than one nanosecond, the typical residual delay error in cm VLBI observations.

3. Testing Results for Enoise

Before using enoise to test the zoomband mode of DiFX for the non-zero baseline correlation with different sampling rates, we first examine whether enoise works well by simulating several datasets with only normal VLBI stations (i.e. sampling rate of 64 MHz) and correlate the data followed by fringe-fitting. In this way, we will know that any unexpected errors in the correlation will not be a result in the zoomband function of DiFX; instead, these errors will be results of either bugs in the enoise algorithm/code or differences between enoise and DiFX.

In our simulations, the main stations we use are the SMA, CARMA, and SMTO. For a preliminary testing of the zoomband function of DiFX, we also include ALMA in our simulation. Since CARMA and SMTO are closeby and SMA and CARMA/SMTO are farther away, their relative delays and rates between these two settings are quite different. Therefore, we can test whether the residual errors, if any, depends on the lengths of baselines. Since the position of the target source will affect the delay and rate of the signal, we choose two different sources (M87 and an artificial source NCP located at the North Celetial Pole) to "observe" and compare their differences. The observing frequency we choose for testing is 230 GHz, which is the primary frequency for the EHT observations. Future tests can be extended to higher frequencies such as 345 GHz and 690 GHz and see if enoise functions well at these frequencies. Note that residual fringe rate error is a function of frequency. Therefore, tests at higher frequencies can tell whether the accuracy of phase rotation performed in enoise is sufficient.

We judge the reliability and accuracy of enoise by correlating the simulated data from enoise and examining the fringe-fitting results from fourfit in the HOPS package. There are a few parameters and a feature from which we can check whether enoise produces correct datasets. These parameters include the amplitude and phase of the cross-correlation, the closure phase, the residual delay and rate. In addition, the flatness of the spectra of the correlation data can inform us the degree of decorrelation due to delay insertion error in the simulation. The expected amplitude of the correlation is 0.049. This number comes from the input correlation amplitude 0.05 minus the typical 2% correlation loss of DiFX. The expected phase and closure phase should be zero because the target source is a point source. Finally, we expect the residual delay should be less than 1 nanosecond and residual rate should be less than 1 mHz. Note that the residual delay and rate will not be zero because the accuracy we can implement delay and phase rotation is limited by the time resolution we can acheive in the simulation. In the following discussions, we will show the results from

fringe-fitting the visibility data from correlating simulated datasets from enoise. In order to save space, we will only show the result for SMA-SMATO baseline.

Figure 1 shows the results from observing NCP with SMA, SMTO, and CARMA with a single 32 MHz band. The key character of observing NCP is that the delay rate is typically small (<0.002 μsec/sec). Therefore, we can use this simple case to check enoise without the potential impacts of large fringe rate. From this test, we find that the correlation has a high quality and all of the fitting parameters reach the expected value. The zero closure phase and small residual fringe rate (0.4 mHz) show that our error control in phase-rotation is done well. The amplitude (0.0472) is also very close to the expected value 0.049, although the fitting value is 3.6% lower. This small decrease in amplitude is not well understood yet. The spectrum in the bottom-right panel of the figure is flat, indicating that our delay insertion has high enough accuracy so that the fractional delay error (after correction in DiFX) is negligible. We would like to note reader that the number of time stamps in one microsecond (i.e. *sps*) in the simulation has to be set high enough (*sps* $\geq$ 1024, corresponding time resolution better than 0.977 nsec) in order to avoid visible decorrelation (i.e. amplitude drops at higher frequency) in the specrum. We find that if *sps* is too small (e.g. 128), the delay insertion in the simulation is usually not accurate enought, leading to substantial decorrelation at higher frequency. We will show an example later.

Figure 2 shows the fringe-fitting results from observing M87 with SMA, SMTO, and CARMA with a single 32 MHz band. The delay rate ($\sim$0.8 μsec/sec) in the observation gets much higher in compasion with the observation on NCP. Thus, we can use this case to test how accurate we perfrom delay insertion and phase rotation in the high delay rate scenario. As in the previous test, we find that the correlation has a high quality and most of the fitting parameters reach the expected value. The closure phase (0.1°) is negligible and residual fringe rate is small (0.1$-$0.4 mHz). The amplitude (0.0464) is also very close to the expected value 0.049, although the fitting value is 5% lower. The spectrum in the bottom-right panel of the figure is also flat. However, the residual errors in phase and delay are not negligible: the visibility phase is 33° for the SMA-SMTO baseline and residual delay in the same baseline is 5.3 nsec. We find that these errors are not due to differences between DiFX and enoise, but come from a hidden bug in the delay insertion in the simulator. The delay offset can been seen by eye by comparing the time samples from the simulated data (before correlation). We are still looking for this little bug and fix it, and we expect that by the time the next version of enoise is released, these residual errors will be removed.

In Figure 3, we show that what will happen when the chosen *sps* is too small (i.e. poorer time resolution for delay implementation). As we can see, the most prominent effect is that the amplitude becomes 10% smaller than the expected value and signifcant decorrelation can be seen in the spectrum. This shows that when the time resolution in the simulation is poor, the delay insertion in the simulation is usually not accurate enough, and this will lead to substantial decorrelation at higher frequency. Nonetheless, it is interesting to note that while the small sps introduce errors in amplitude and correlation, the residual delay decreases from 5.3 nsec to 1.8 nsec. This happens because the true residual delay is smaller than the time resolution (7.8 nsec) and thus such error cannot be measured precisely.

Figure 4 shows the results from observing M87 with SMA, SMTO, and CARMA with four consecutive 32 MHz bands. Every thing is the same as the case of observation with a single band except that multiband delay can now be fitted. The blue curve in the top panel shows multiband delay fitting and the best-fit value is 5.741 nsec. Since no source of multiband delay is implemented in the simulator, the residual multiband delay also implies some errors in the algorithm of the simulator. We notice that the multiband delay is always nearly identical to the singleband delay, implying that they have the same origin $-$ the little bug in the delay implementation. Therefore, we can only remove the residual multiband delay when the bug is

found and fixed.

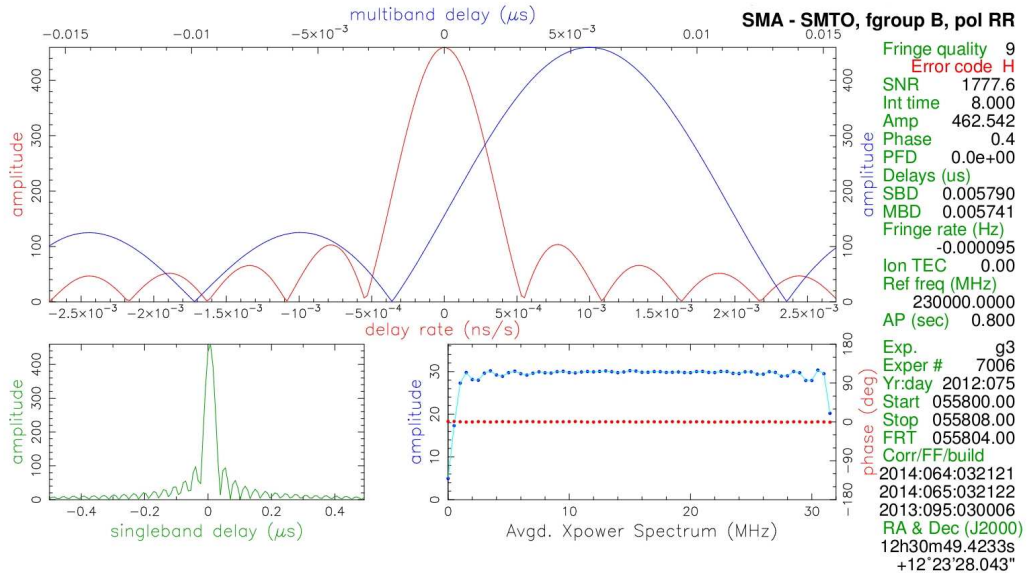

Fig. 1.— The fringe-fitting results from observing M87 with SMA, SMTO, and CARMA with four consecutive 32 MHz bands.

4. Correlating ALMA and VLBI Data

From the above discussions, we can see that enoise works well in general and the small residaul errors in phase and delay should be removed once we fix the subtle error in the delay implementation. Therefore, we can start to do preliminary testing of the reliability of the zoomband function in DiFX. As long as the remaining errors are within the expected values from correlating VLBI-only datasets, we can be confident that the zoomband function of DiFX works fine. Figure 5 shows a test that include SMA, SMTO, and ALMA. For SMA and SMTO, the observing bandwidth is 32 MHz. For ALMA, the bandwidth is 62.5 MHz. In order to correlate the ALMA data and the data from SMA and SMTO, we apply a 32 MHz zoomband in the correlation. From the figure, we can see that applying the zoomband function does not produce unexpected errors in the correlation. The amplitude, phase, and residual delay are in the expected level. The closure phase is nearly zero. The visible errors (i.e. residual phase and delay) are similar to the values found in the simulation without including ALMA. Therefore, this preliminary testing shows that the zoomband mode of DiFX does work non-zero baseline situation. Nonetheless, more simulation cases and more rigorous tesings are needed to examine the accuracy of DiFX Zoomband for correlating ALMA and VLBI data.

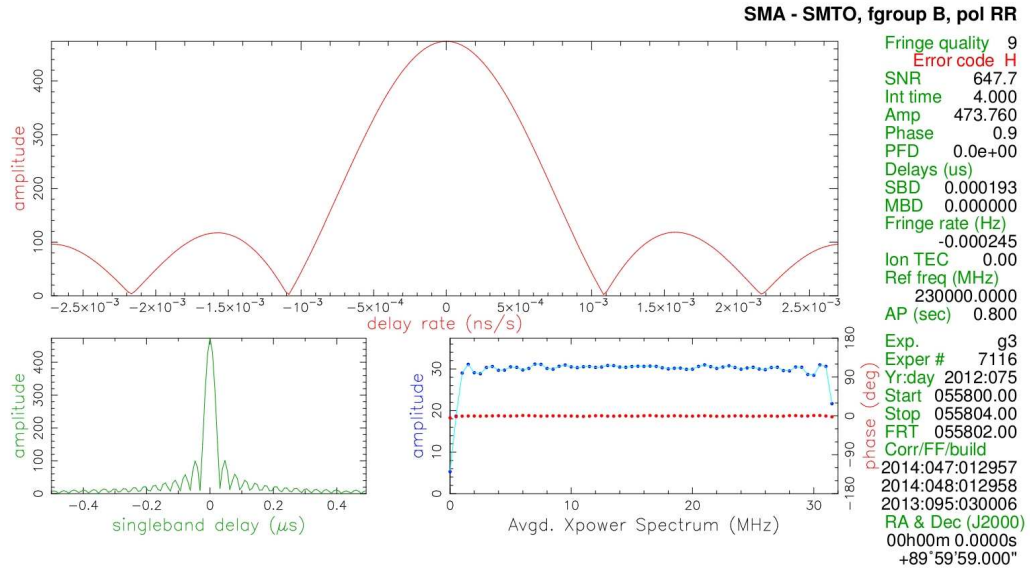

Fig. 2.— The fringe-fitting results from observing NCP with SMA, SMTO, and CARMA with a single 32 MHz band.

REFERENCES

Thompson, A. R., Moran, J. M., Swenson Jr., G. W. 2001, Interferometry and Synthesis in Radio Astronomy Wiley-Interscience Publication

Deller, A. PhD Thesis : http://www.aoc.nrao.edu/ adeller/

This preprint was prepared with the AAS LaTeX macros v5.2.

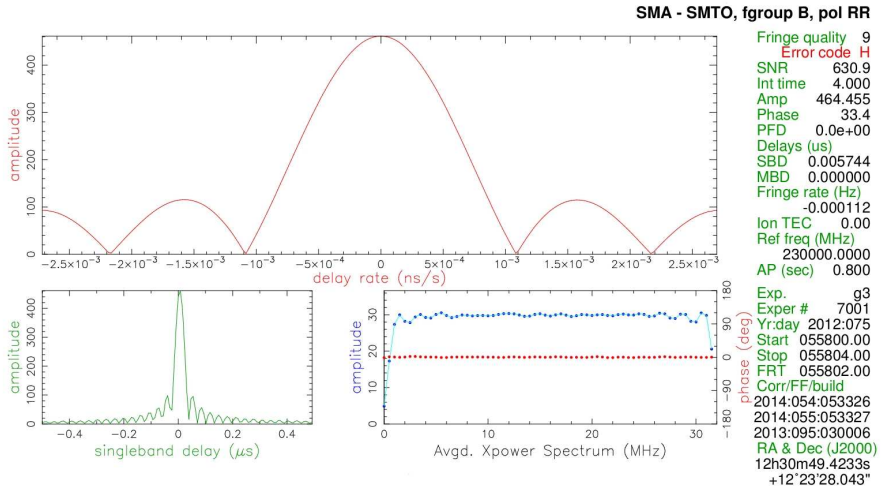

Fig. 3.— The fringe-fitting results from observing M87 with SMA, SMTO, and CARMA a single 32 MHz band.

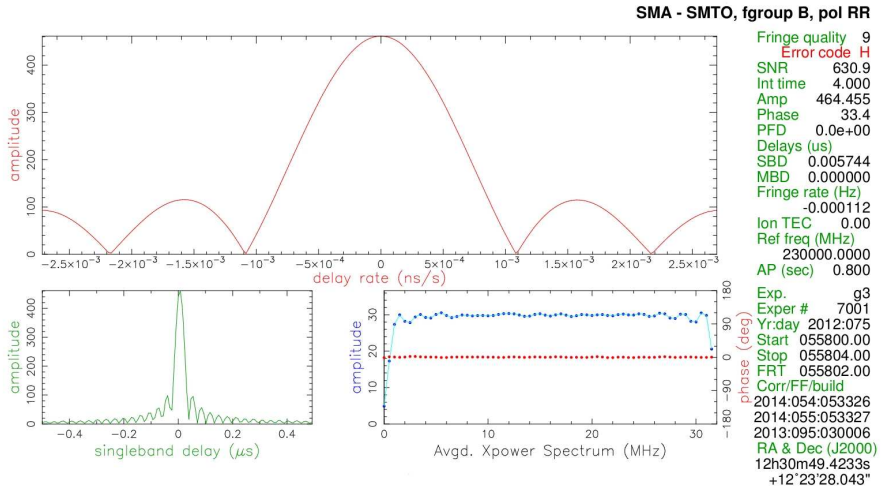

Fig. 4.— The fringe-fitting results from observing M87 with SMA, SMTO, and CARMA a single 32 MHz band.

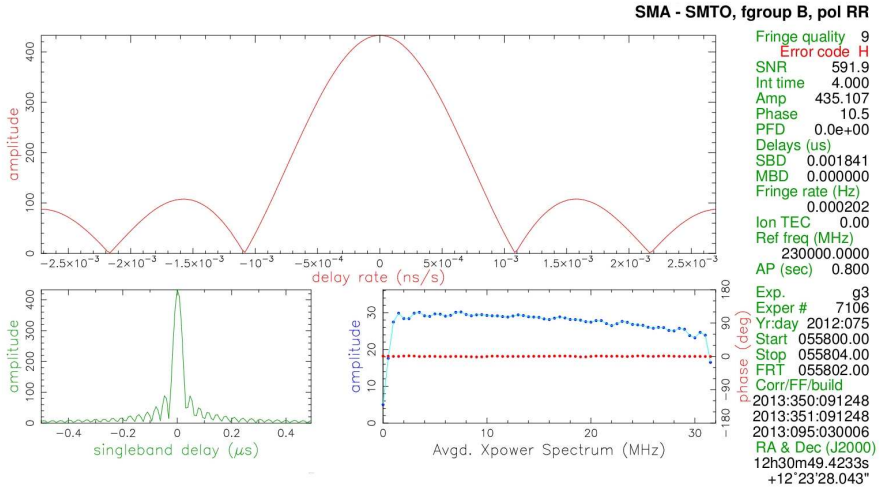

Fig. 5.— The fringe-fitting results from observing M87 with SMA, SMTO, and CARMA with a single 32 MHz band and sps equaling to 128.

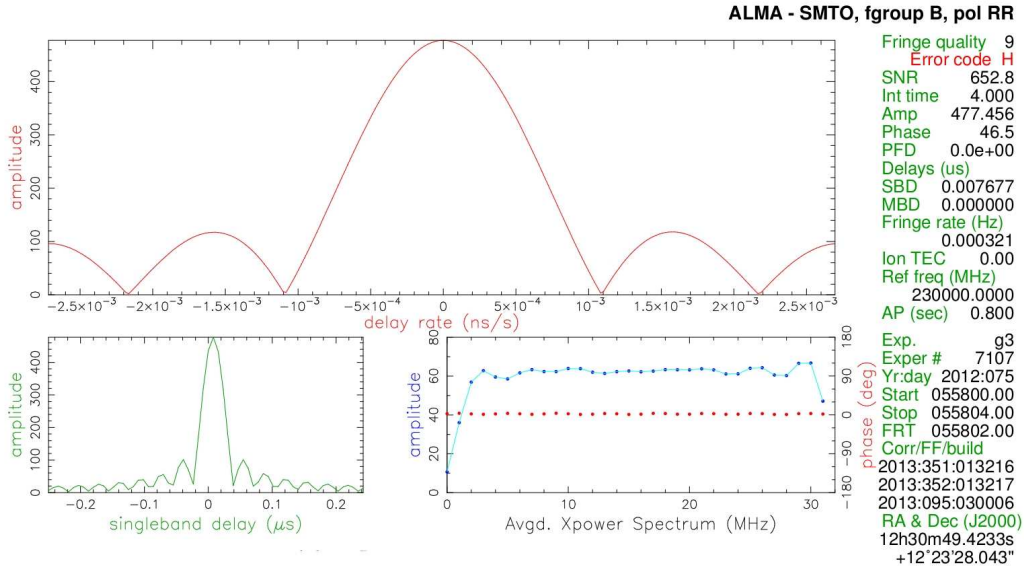

Fig. 6.— The fringe-fitting results from observing M87 with SMA, SMTO, and ALMA. For SMA and SMTO, the observing bandwidth is 23 MHz. For ALMA, the bandwidth is 62.5 MHz. A 32 MHz zoomband is applied in the correlation process to correlate ALMA data with data from SMA and SMTO.