igg

universitätbonn

DiFX Correlation & Fringe Search

Alessandra Bertarini

IGG University of Bonn & MPIfR Bonn

Max-Planck-Institut
für
Radioastronomie

DiFX

- DiFX -> Distributed FX correlator.
- DiFX is a software correlator.
- DiFX is a free downloadable software from:

http://cira.ivec.org/dokuwiki/doku.php/difx/installation

- DiFX needs IPP libraries (IPP requires licence).

igg

DiFX at Bonn

universitätbonn

RAIDs

RAIDs + fxmanager

nodes + frontend and frontend2

nodes

- 60 compute nodes
- 8 cores per node
- 20 Gbps InfinBand
- Network cards
- 10 RAIDs (220 TB)
- 1 service node (with keybord & monitor)
- 2 user interaction nodes (frontend & frontend2)

DiFX is software running on various computer clusters.
Every cluster performance is different, but...

the fundamental operations performed by the correlator are the same.

DiFX:
- receives digitized signals
- applies the correlator model
- pads the data from 2 bits to 16 bits
- aligns the data within +/- 1 sample
- performs an FFT
- performs a fractional-sample delay correction
- performs a complex multiplication & integrates
- writes the complex visibilities (in freq. domain)

What correlators need:

1) Vex file.

2) FS log files.

3) Modules or e-transferred data.

4) Mails from stations with comments for the observation.

What DiFX needs extra:

1) `v2d` file to convert the vex into DiFX-readable (ascii) files.

Vex files are used by the correlators for:

- Sky Frequency → relevant for fringe rotator
- LO tuning → relevant for fringe rotator
- Recording speed → relevant for playback speed
- Polarization → relevant for channel assignment
- No. of BBCs → relevant for channel assignment
- Sources to be observed → coordinates for corr. model
- Length of the scans → relevant for playback
- Track assignment → relevant for channel assignment
- Antenna coordinates (not required for observing)

Correlator's vex files need extra information:

- Earth orientation parameters (x-wobble, y-wobble and UT1)
- Clock information (gps-fmout from field system logs)
- Data source (Mark 5 module, files on RAID)

Correlator's vex files need (sometimes) to be changed:

Track assignment (only tape-like tracks are present in vex)

Log files are used by correlators for:

- Clocks: gps-fmout values
- vsi4 = astro / geo
- Lots of useful info in case debugging is required:

  LO tuning,
  BBC/VC frequencies,
  polarization,
  track assignment – if Mark 5A
  [….]

DiFX : Exercise of the day

In this exercise we are using R1578.

R1578 is part of a weekly experiment dedicated to measuring UT1-UTC.

It is a dual band (8 GHz and 2 GHz) experiment.

One polarization (RCP).

One bit sampling.

Observed on 25th March 2013.

24 h long, but ~ 40 s of data for these exercises.

Log in to: `portal.mpifr-bonn.mpg.de -X`
user/pwd as written on the board.

Hop to fxmanager: `ssh fxmanager -l oper -X`

Change directory to `/Exps/eratec/r1578_n`
`(where _n is the computer nr that you are using)`

In the directories there are the vex files and the FS logs.

Task 1: select the version of DiFX to use (we have lots):

Type `d21`

(d21 is the DiFX 2.1, last stable release).

Task 2: Get familiar with the vex file.

Vex is divided into sections. All sections have a $SECTION (i.e. $MODE, $STATION $SCHED, $CLOCK….)

Within a section there are definitions that point to other sections:

```
$STATION
   def Ft;
       ref $ANTENNA = FORTLEZA;
   enddef;
```

```
$MODE;
def GEOSX-SX;
ref $FREQ = GEOSX-SX01:Ft:Hh:Ke:Ny:Ts:Wz:Yg;
ref $BBC = GEOSX-SX01:Ft:Hh:Ke:Ny:Ts:Wz:Yg;
ref $IF = GEOSX-SX01:Ft:Hh:Ke:Ny:Wz:Yg;
ref $TRACKS = Mk34121-SX01:Ft:Wz;
ref $TRACKS = Mark5B:Hh:Ke:Kk:Ny:Tc:Yg;
$PHASE_CAL_DETECT = Standard:Ft:Hh:Ke:Kk:Ny:Wz;
[..]
enddef;
```

```
$STATION
def Ny;
ref $SITE = NYALES20;
ref $ANTENNA = NYALES20;
enddef;
$ANTENNA
def NYALES20;
antenna_diam = 20.00 m;
axis_type = az : el;
axis_offset = 0.51980 m;
antenna_motion = az : 120.0 deg/min : 9 sec;
antenna_motion = el : 120.0 deg/min : 9 sec;
enddef;
```

```
$FREQ
def GEOSX-SX01;
chan_def = &X : 8212.99 MHz : U : 8.000 MHz : &CH01 : &BBC01 :&U_cal;
chan_def = &X : 8212.99 MHz : L : 8.000 MHz : &CH02 : &BBC01 :&U_cal;
chan_def = &X : 8252.99 MHz : U : 8.000 MHz : &CH03 : &BBC02 :&U_cal;
chan_def = &X : 8352.99 MHz : U : 8.000 MHz : &CH04 : &BBC03 :&U_cal;
chan_def = &X : 8512.99 MHz : U : 8.000 MHz : &CH05 : &BBC04 :&U_cal;
chan_def = &X : 8732.99 MHz : U : 8.000 MHz : &CH06 : &BBC05 :&U_cal;
chan_def = &X : 8852.99 MHz : U : 8.000 MHz : &CH07 : &BBC06 :&U_cal;
chan_def = &X : 8912.99 MHz : U : 8.000 MHz : &CH08 : &BBC07 :&U_cal;
chan_def = &X : 8932.99 MHz : U : 8.000 MHz : &CH09 : &BBC08 :&U_cal;
chan_def = &X : 8932.99 MHz : L : 8.000 MHz : &CH10 : &BBC08 :&U_cal;
chan_def = &S : 2225.99 MHz : U : 8.000 MHz : &CH11 : &BBC09 :&U_cal;
chan_def = &S : 2245.99 MHz : U : 8.000 MHz : &CH12 : &BBC10 :&U_cal;
chan_def = &S : 2265.99 MHz : U : 8.000 MHz : &CH13 : &BBC11 :&U_cal;
chan_def = &S : 2295.99 MHz : U : 8.000 MHz : &CH14 : &BBC12 :&U_cal;
chan_def = &S : 2345.99 MHz : U : 8.000 MHz : &CH15 : &BBC13 :&U_cal;
chan_def = &S : 2365.99 MHz : U : 8.000 MHz : &CH16 : &BBC14 :&U_cal;
sample_rate = 16.0 Ms/sec;
enddef;
```

Task 3: Insert the EOP in the vex file.

Run the program `geteop.pl -h` (follow instructions). It creates a file called EOP.txt

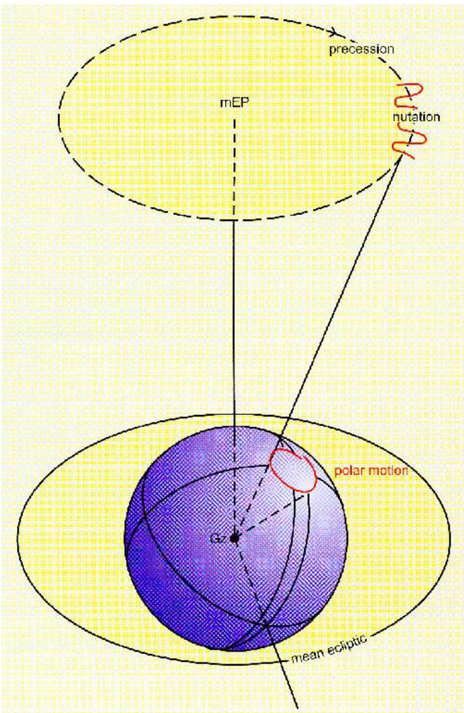
DiFX needs to have 5 values of EOP and two of them must be prior to the observation start.

Insert the file EOP.txt in the VEX (no special position as long as it does not "break a $SECTION or is within a def / enddef).

E.g. if observation starts at Day of the Year (DOY) 100, then geteop needs as start DOY 98.

DOY of r1572 is found in $SCHED section in the scan name (e.g. 100-1700 => DOY-HHMM).

Earth Orientation Parameters (x-wobble, y-wobble and UT1).



DiFX ancilliary program `geteop.pl` reads the USNO file, reformats it and creates a file called EOP.txt

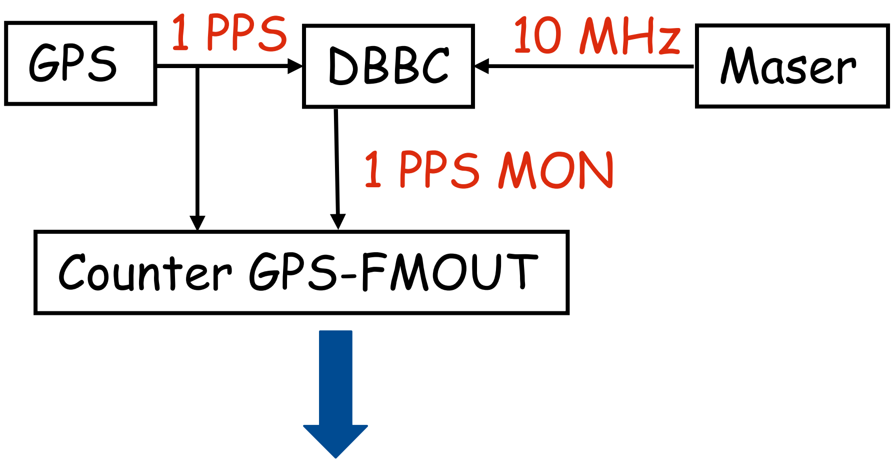
The predicted EOPs values are published from USNO:
`http://128.183.20.176/solve_save/usno_finals.erp`

EOP: VEX example for observation on DOY 035.


```
$EOP;
 def EOP0;
  TAI-UTC= 35 sec;
  A1-TAI= 0 sec;
  eop_ref_epoch=2013y033d;
  num_eop_points=1;
  eop_interval=24 hr;
  ut1-utc   = 0.237134 sec;
  x_wobble = 0.042530 asec;
  y_wobble = 0.313450 asec;
 enddef;
def EOP4;
     […]
 enddef;
```

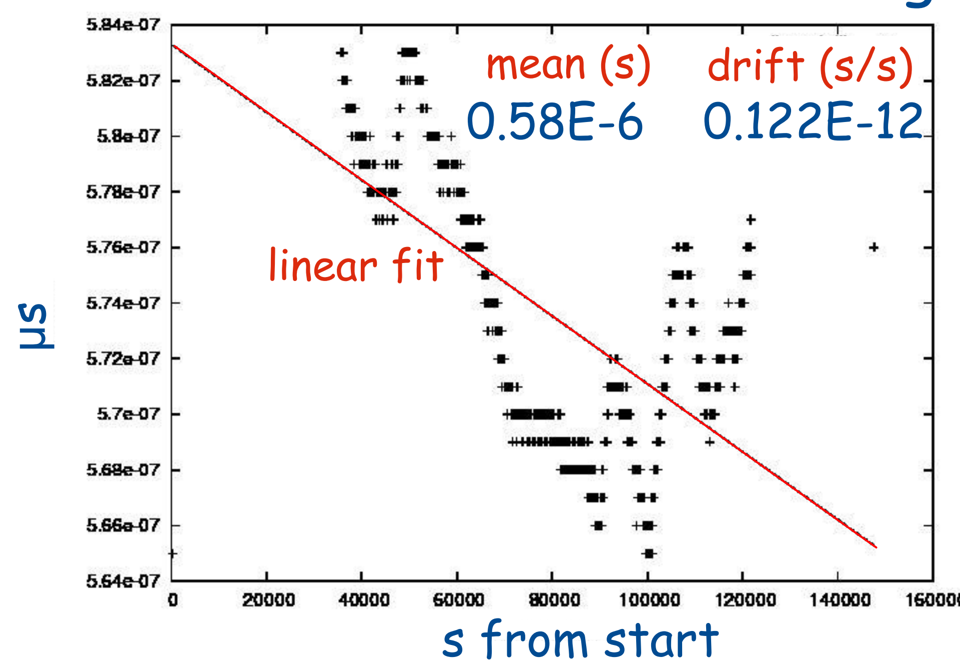
Note: DiFX needs EOPs for 5 days of which two prior to the observation !

CLOCK: estimates the time difference between the data time stamps (from formatter/M5B/ FiLa 10G) and UTC coming from GPS.



But 10 MHz drifts

GPS-FMOUT from FS log

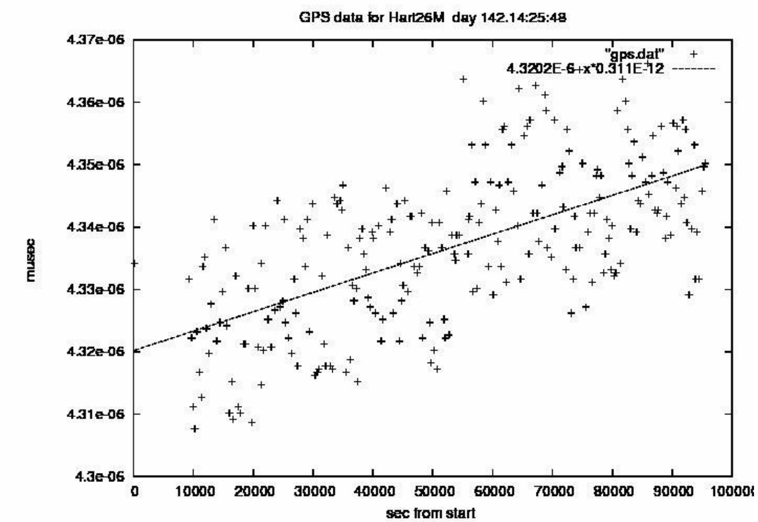


Task 4: Calculate Clock offset and rate correction

Use the program `clkm4` to calculate clock offset and clock drift from the FS logs.

The program extracts the gps – fmout values calculates the mean (clock offset) and performs a linear fit to calculate the drift.

It creates also a postscript file with the fit.



e.g. clkm4 r1578on.log

Task 5: Clock offset and rate correction in the vex

```
$CLOCK;
def Ny;
 clock_early=2012y142d17h00m : X usec :2012y142d17h00m0s : x ;
enddef;
def Wz;
clock_early=2012y142d17h00m : Y usec :2012y142d17h00m0s : y;
enddef;
[…]
```

The X and Y µs are the mean gps-fmout "*clock*" values.

The x and y s/s are the clock drifts.

DiFX needs to know whether the data are on a Mark 5 module and needs to know which module.

The program `fslog2difx.pl` does the job for the modules reading the FS log files.

Task 6: Enter the modules in the vex

cut and paste the file /Exps/eratec/TAPELOG.txt in the vex.

```
$TAPELOG_OBS;
 def Kk ;
  VSN=1 :  HOB+0025 :   2013y043d17h00m00s :2013y044d16h57m46s ;
enddef ;
 def Tc ;
  VSN=1 :  HART-014 :   2013y043d17h00m00s :2013y044d16h17m56s ;
enddef ;
```

Note: E-tranferred stations do not appear in the vex !

DiFX: Vex File - Tracks

Check "track" assignment: vex speaks (still) tape language!

Mk 4	VSI=geo	VSI=astro
1US	0	0
1UM	1	1
2US	2	2
2UM	3	3
3US	4	4
3UM	5	5
4US	6	6
4UM	7	7
5US	8	8
5UM	9	9
6US	10	10
6UM	11	11
7US	12	12
7UM	13	13
8US	14	14
8UM	15	15

Mk 4	VSI=geo	VSI=astro
1LS	16	16
1LM	17	17
2LS	-	18
2LM	-	19
3LS	-	20
3LM	-	21
4LS	-	22
4LM	-	23
5LS	-	24
5LM	-	25
6LS	-	26
6LM	-	27
7LS	-	28
7LM	-	29
8LS	18	30
8LM	19	31

DiFX: Vex File - Tracks

Check "track" assignment: vex speaks (still) tape language!

Mk 4	VSI=geo	VSI=astro
9US	21	–
9UM	22	–
10US	23	–
10UM	24	–
11US	25	–
12UM	26	–
12US	27	–
13UM	28	–
13US	29	–
14UM	30	–
14US	31	–

In vex enter VSI output + 2 !

i.e. 1US: VSI output = 0 → vex tracks = 0 + 2 = 2

TRACKS sorting in vex:

```
$FREQ;
def GEOSX-SX01;
chan_def = &X:8212.99 MHz:U:8.000 MHz:&CH01:&BBC01:&U_cal;
[..]
enddef;
[..]
$TRACK;
def Mark5B;
fanout_def = A : &CH01 : sign : 1 : 02 ;
[..]
enddef;
```

USB

From tables above:

Mk 4	VSI=geo	VSI=astro
1US (BBC01)	0	0

VSI output = 0, i.e. TRACK = 02

DiFX requires different files, based on vex .
The program `vex2difx` creates those files:

Input file
- clock,
- frequency,
- tracks,
- integration time,
- baseline to cross-correlate …

Calc file
- station coord.,
- EOPs,
- antenna mounting type,
- source coord. ….

calcif2 → Im file (Interferometer Model, i.e. correlator model)

More info:
`http://cira.ivec.org/dokuwiki/doku.php/difx/vex2difx`

vex2difx requires a v2d file.

Layout of v2d file:

vex = vex file name

antennas = two letter code of the participating stations
(e.g. antennas = WF, ON, WZ ...)

singleScan = True/False

suggested value for trial-correlation

```
SETUP r1600
{
   tInt = integration time in second (e.g. 0.2 s, 1 s ...)
   doPolar = True/False
   nChan = no. spectral channels (e.g. 128, 512, 1024.., 2^16 )
}
```

Layout of v2d file cont.:

```
RULE clock {
    scan = scan name (e.g. 222-1700)
    setup = r1600
}
ANTENNA AB
{
   filelist = ab.filelist
}
```

In trial-correlation mostly only one scan

How to create filelist:

`directory2filelist` : DiFX program to create filelists.

directory2filelist /path-to-data/ Mode > output.filelist e.g.:

directory2filelist /raid1/exp1/ Mark5B-512-8-2 > on.filelist

format Mbps bbc no. nbit

```
/raid1/exp1/No0001 56054.708345 56054.708935
/raid1/exp1/No0002 56054.709479 56054.710058
```

MJD

Task 7: Copy the filelists in the working directory

copy /Exps/eratec/*.filelist in your working directory.

Task 8: Create the v2d file

R1578 was observed using only one polarization (RCP) => doPol = False

We have only one scan => singleScan = True

Scan name → DOY-HHMM (read in the vex -> $SCHED)

File: DiFX must know whether the data are on Mark 5 module or in a file. If 'filelist =' is in v2d then data come from files.

so filelist = ab.filelist ...

Note all stations require an ANTENNA block with the respective file.

e.g. ANTENNA AB { filelist = ab.filelist}

DiFX: Create v2d File

Insert the tInt and nChan in the v2d file:

As start use tInt = 1 s and nChan = 512

Note that no of lag = no. of spectral channel (in DiFX!).

Task 9: Run the program vex2difx

e.g.: `vex2difx r1600.v2d`

vex2difx creates the files .input, .calc, ….

Task 10: Run errormon2

open a new window type `d21` to select the correlator

Type `errormon2` => error monitor

Task 11: Run the correlator

using the script `startdifx` :

```
startdifx r1600_1.input
```


Startdifx runs:

- `calcif2 -a` → to create the im file
- `genmachines <input file>` -> to create the machines and thread files
- `mpirun -np nn -machinefile <machine file> mpifxcorr <input f>`

no. of process to start
(found using wc -l machine file)

created from vd2

Run DiFX: Machine File

The machine file looks like this:

```
fxmanager 0 0
mark5fx01 0 1
[..]
Mark5fxnn 0 1
node55 7 0
node56 7 0
[..]
nodenn 7 0
```

Control the operation

Mark 5 Units available (1)

Compute nodes available (0)

no. of cores used

r1600.machines

```
fxmanager slots=1 max-slots=1
node41 slots=1 max-slots=2
node42 slots=1 max-slots=2
[..]
```

r1600.threads

```
NUMBER OF CORES:    10
7
7
[..]
```

DiFX creates a directory called `r1600_1.difx` (DiFX visibilities and pcal files) and a file `r1600_1.difxlog` (errormon output).

Task 12: create the files for fourfit

`difx2mark4 r1600_1.difx` (will create a directory 1234)

Task 13: search for fringes

run `fourfit`. > cd 1234

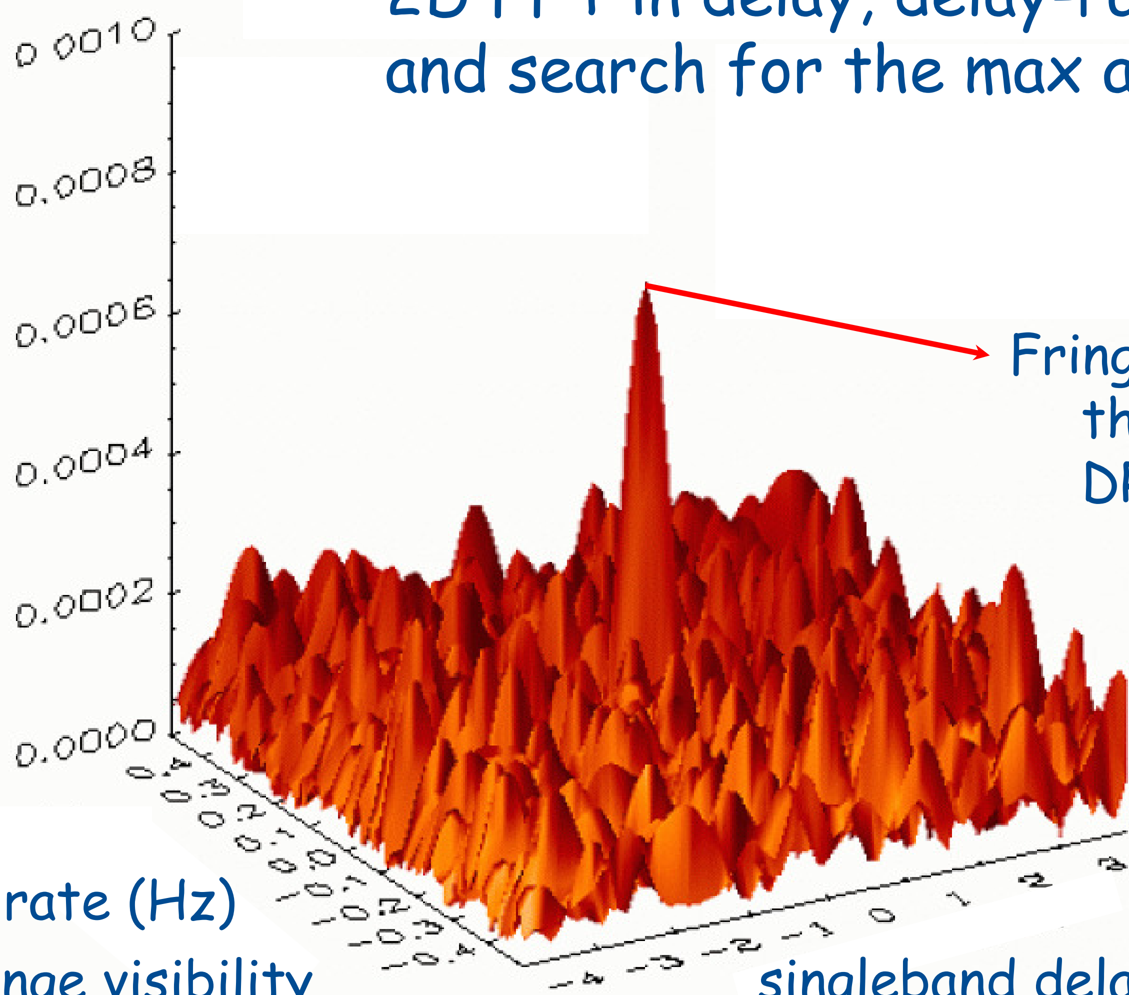
> HOPS (to set env variables for fourfit)

> fourfit -pt -c cf_1234 scan name

e.g. fourfit -pt -c ../../cf_1234 056-1700

Fringe Fit: Real Fringe Search

2D FFT in delay, delay-rate space and search for the max amplitude



Corr. amplitude

Fringes are located at this position in SBD, DR space

Fringe rate (Hz)

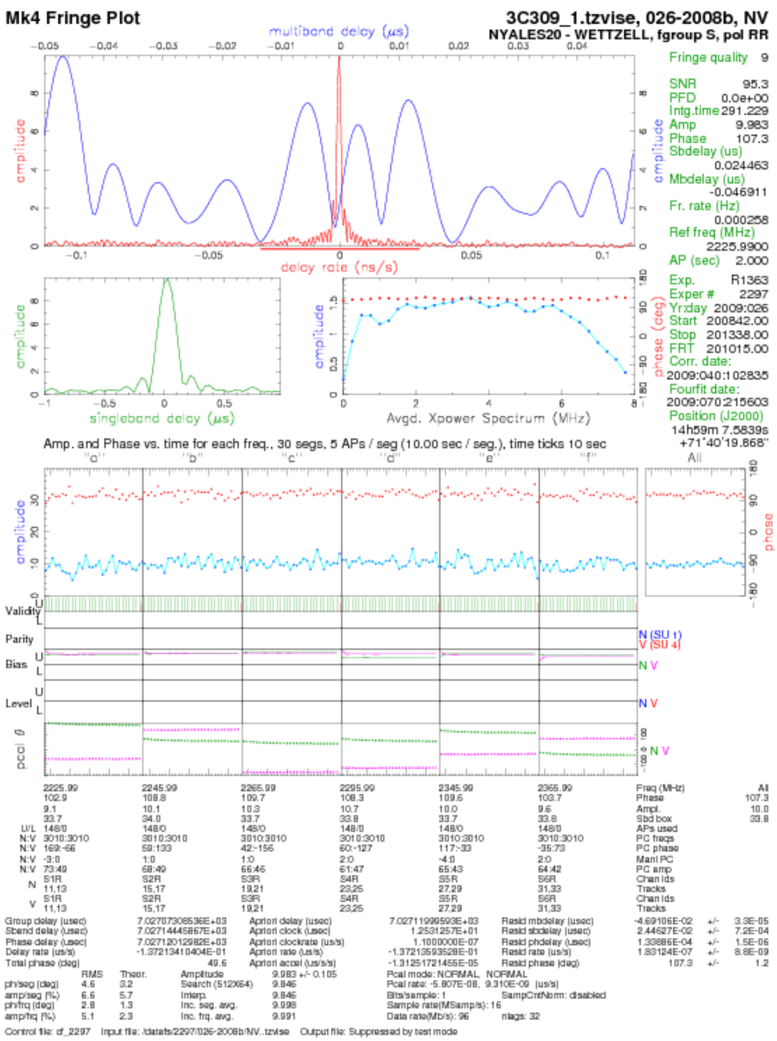
(FT fringe visibility with respect to time)

singleband delay (μs)

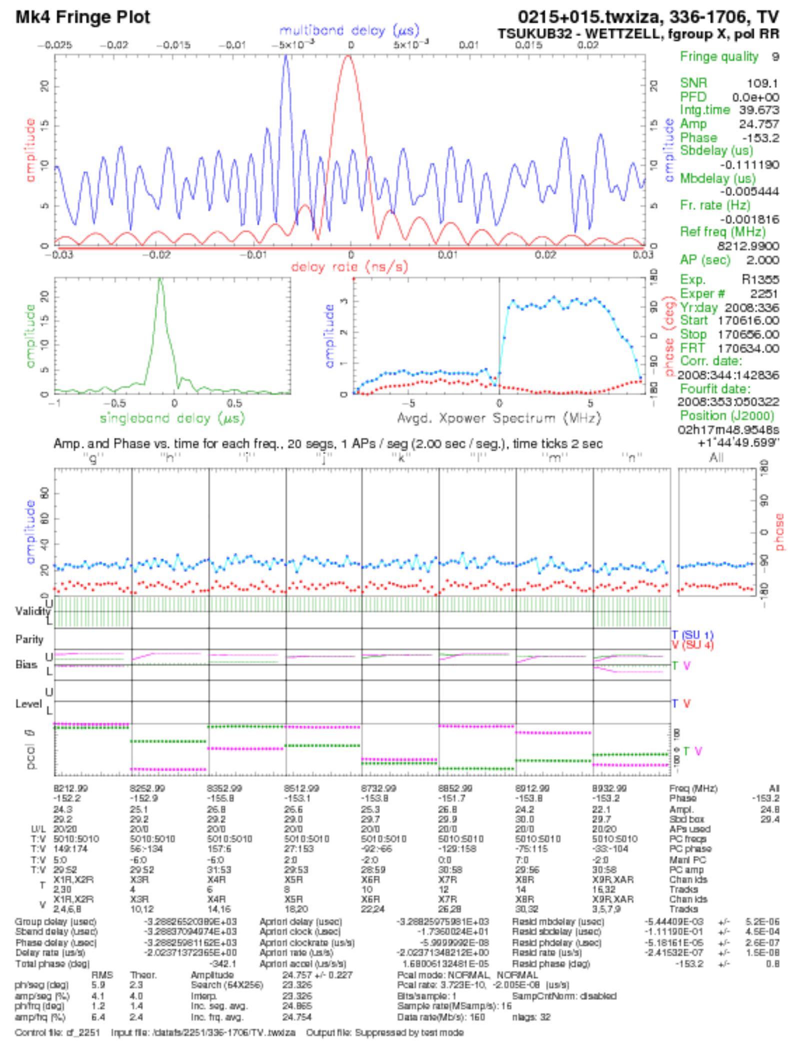
Image: Roger Cappallo

Fringe Fit: *Fourfit* Overview

S-Band:



X-Band:



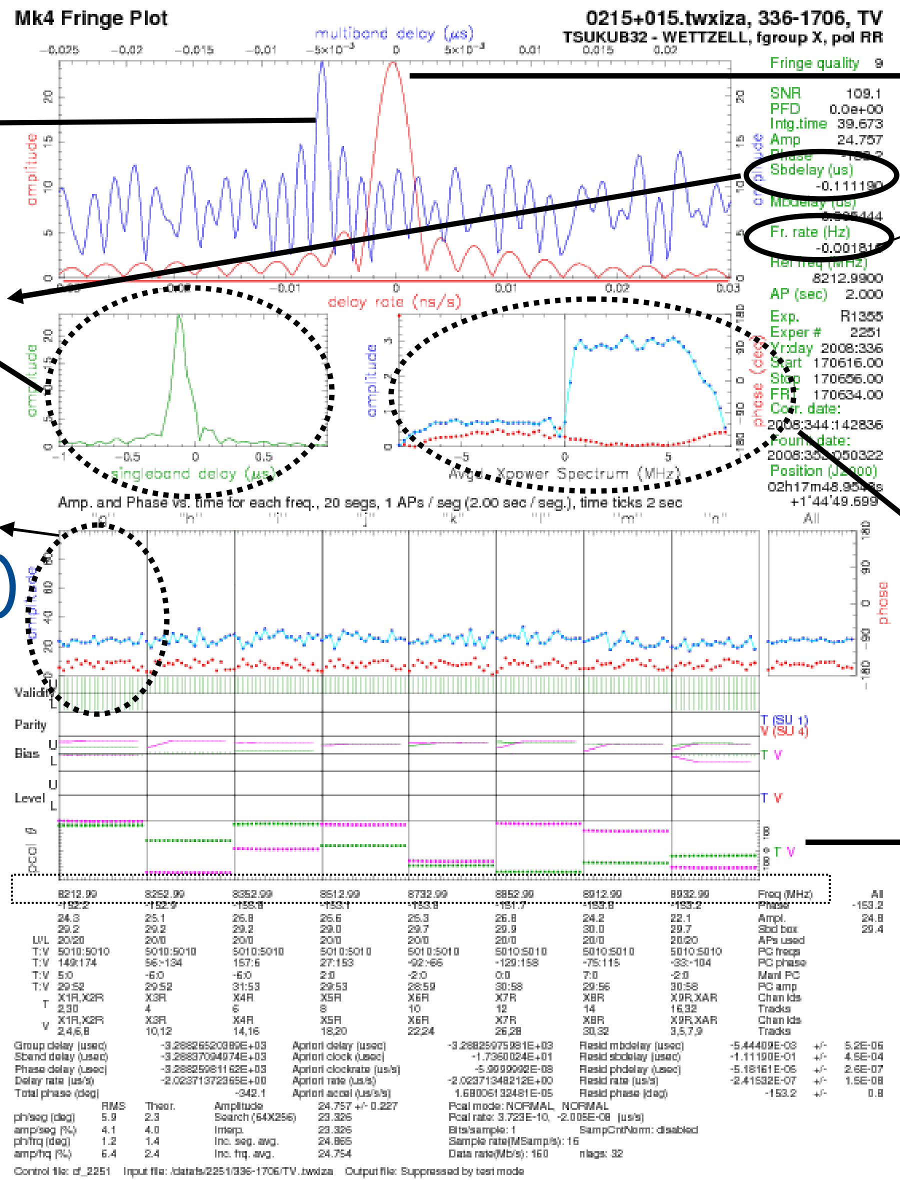
Fourfit

Multiband delay (µs)

Single band delay (µs)

Phase (red) & amp (blue) vs time for every BBC

Sky freq.



Delay rate.

Fringe rate (Hz) = Delay Rate • Sky freq.

FT of lag spectrum

Pcal phases

SBD ≠0 => non modelled station-based errors. But we can correct them!

<u>Task 14:</u> Change the clock_early entry in the vex file.

SBD is baseline-based => take one of the two station as reference and correct the other.

Suppose we have baseline AB

Suppose SBD = 0.6 μs on baseline AB.

Suppose clock_early A = 1.5 μs and clock_early B = -3.0 μs

If A is chosen to be the reference station =>

clock_early B = -3.0 + 0.6 = -2.4 μs

If B is reference => clock_early A = 1.5 - 0.6 = 0.9 μs

We have changed the clock model. Need to check it.

Remove the old *.input, calc, im, difx files

Re-run vex2difx – startdifx – difx2mark4 – fourfit.

Task 15: Is the SBD better?

Now we break things!

Task 16: Change the station position or source position or clock models. Increase/reduce tInt and nChan. Run the correlation and look at the data with fourfit. What happens?

Other Tools to Check Data

Copy raw data (~ 3 MB) onto file and check the data with linux command *od*:

`od -tx4` file name

output is like:

frame no.

time stamp:MJD & second of day

fractional second & header error check

```
0000000 abaddeed bead0001 0974ad5f f00abf01
0000016 0d645d49 57143f17 3a19c152 a0ec5b58
```

`od` byte no. in file

data

...

ABADDEED => header sync word (every 10016 bytes)

if lots of hex are zeroes -> no input to DBBC

Use `mark5access` library (part of DiFX, but should be possible to install them as stand-alone):

`m5d`: decode data (valid for all data kinds that DiFX reads).

`m5test`: decode data headers and data (valid for all data kinds that DiFX reads).

`m5bstate`: state counts summary (valid for all data kinds that DiFX reads).

`m5spec`: forms total power for each baseband channel in the file (never used by me!).

Other Tools to Check Data


```
m5d /path/file.m5b Mark5B-256-16-1 10   →
Mark5 stream: 0x89e130
stream = File-1/1=/data10/r1/nyalesund/r1538_ny_171-1212a
format = Mark5B-256-16-1 = 2
start mjd/sec = 97 43922.000000000
frame duration = 312500.00 ns
framenum = 0
sample rate = 16000000 Hz
offset = 0
framebytes = 10016 bytes
datasize = 10000 bytes
sample granularity = 1
frame granularity = 1
gframens = 312500
payload offset = 16
read position = 0
data window size = 1048576 bytes
-1   1   1   1  -1   1  -1  -1  -1  -1  1  -1  -1  -1  1  -1
 […]
10 / 10 samples unpacked
```

Other Tools to Check Data

```
m5test /path/file.m5b Mark5B-256-16-1
Mark5 stream: 0x89e130
stream = File-1/1=/data10/r1/nyalesund/r1538_ny_171-1212a
format = Mark5B-256-16-1 = 2
start mjd/sec = 97 43922.000000000
frame duration = 312500.00 ns
framenum = 0
sample rate = 16000000 Hz
offset = 0
framebytes = 10016 bytes
datasize = 10000 bytes
sample granularity = 1
frame granularity = 1
gframens = 312500
payload offset = 16
read position = 0
data window size = 1048576 bytes
frame_num=2 mjd=97 sec=43922 ns=000625000.0 n_valid=2 n_invalid=0
 [..]
frame_num=335990 mjd=97 sec=44026 ns=996875000.0 n_valid=335990
1679990000 / 1679990000 samples unpacked
```

Other Tools to Check Data

```
m5bstate /path/file.m5b Mark5B-2048-16-2
```

→

Ch	--	-	+	++	--	-	+	++	gfact
0	3937	2332	14736	3995	15.7	9.3	58.9	16.0	1.10
1	3921	8576	8552	3951	15.7	34.3	34.2	15.8	1.10
2	3968	8521	8580	3931	15.9	34.1	34.3	15.7	1.10
3	3833	8597	8651	3919	15.3	34.4	34.6	15.7	1.12
4	3857	8573	8628	3942	15.4	34.3	34.5	15.8	1.11
5	3951	8559	8518	3972	15.8	34.2	34.1	15.9	1.10
6	3947	8642	8416	3995	15.8	34.6	33.7	16.0	1.10
7	3991	8543	8525	3941	16.0	34.2	34.1	15.8	1.10
8	3961	8656	8430	3953	15.8	34.6	33.7	15.8	1.10
9	3934	8582	8531	3953	15.7	34.3	34.1	15.8	1.10
10	3896	8651	8615	3838	15.6	34.6	34.5	15.4	1.12
11	3909	8764	8458	3869	15.6	35.1	33.8	15.5	1.11
12	3971	8613	8531	3885	15.9	34.5	34.1	15.5	1.11
13	3988	8561	8370	4081	16.0	34.2	33.5	16.3	1.09
14	3844	8580	8679	3897	15.4	34.3	34.7	15.6	1.12
15	4002	8445	8581	3972	16.0	33.8	34.3	15.9	1.09

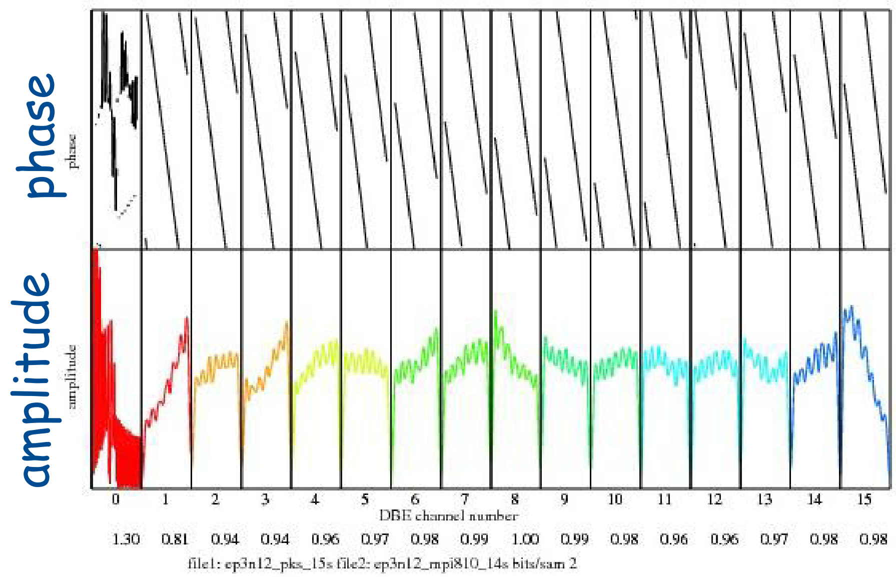
Other Tools to Check Data

Programs downloadable from Haystack:

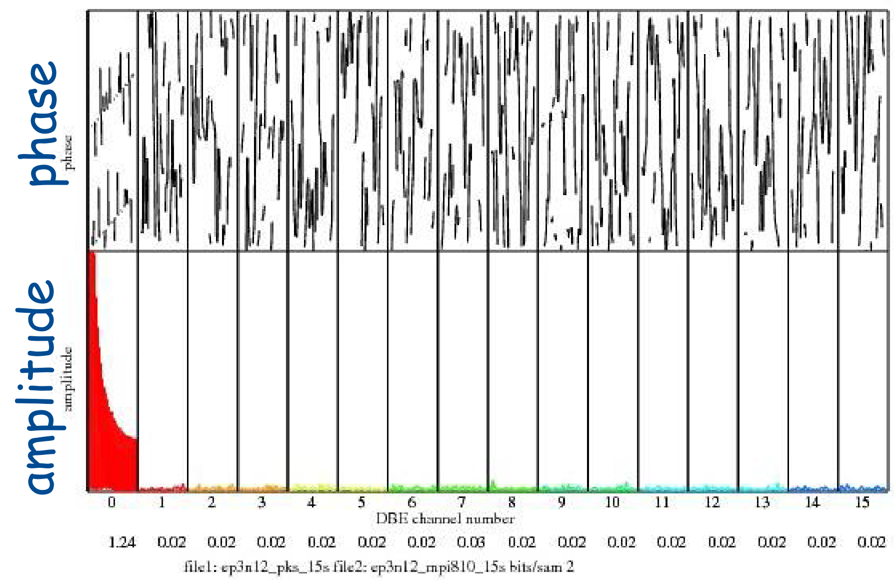
`vlbi2` only for 16-channels 2 bit sampling

`bstate` only for 16-channels 2 bit sampling

`vlbi0` only for DBE (or equivalent channel assignment)



Fringes

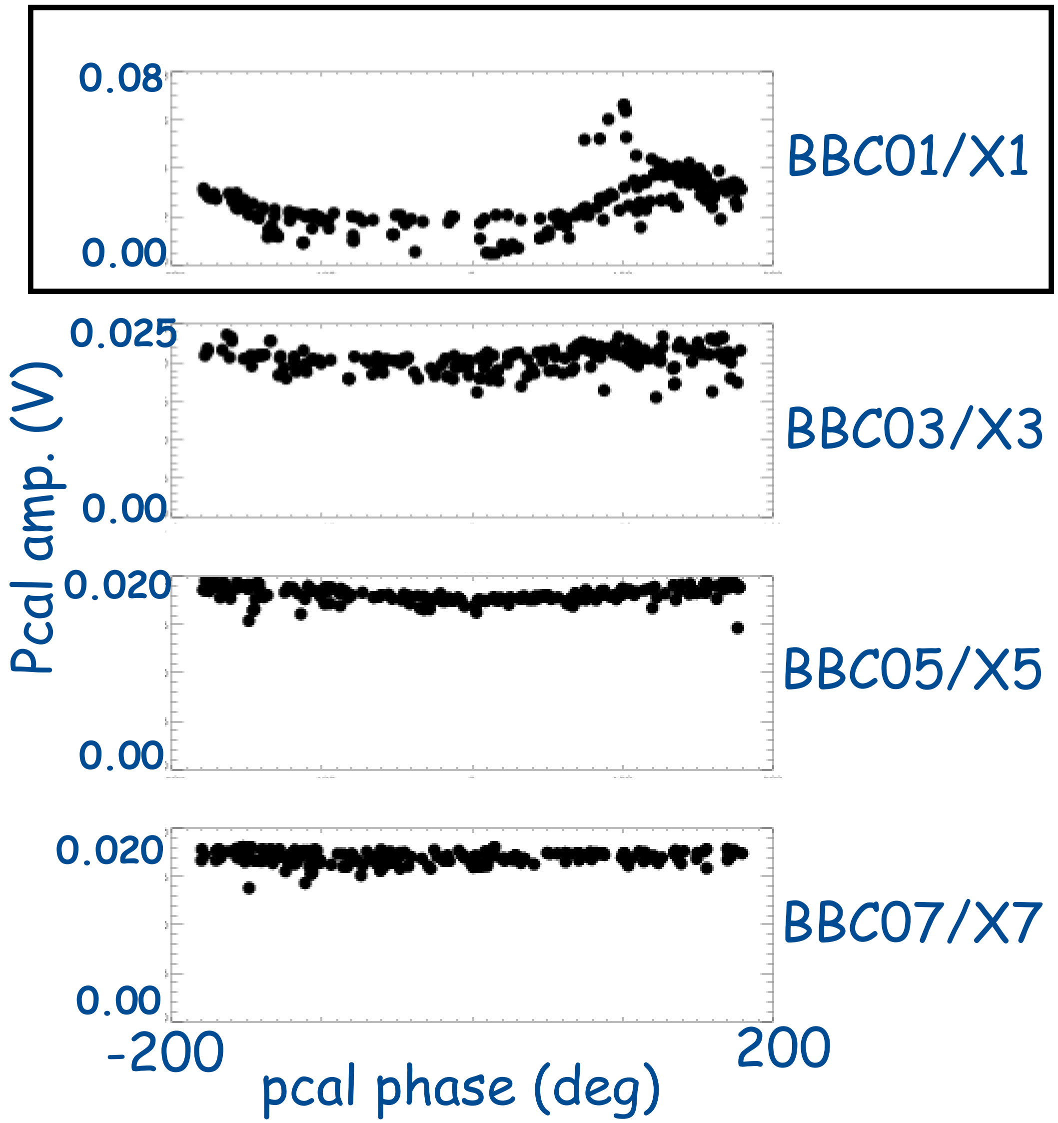


No fringes

Game Over ...

Try Again?

Aedit: Phase Cal Data – Amplitude vs Phase



- From these plots we find the spurious signals (sinusoid).
- Spur are narrowband signals coherent with the true pcal and have its same frequency.
- Corrupt the visibility phases.

Summary

FX correlator outputs are visibility (real and imaginary components) in the frequency domain.

Lag correlator outputs are correlator coefficients (real and imaginary components) in the time domain.

After correlation, correlator analysts check the data quality (e.g. using *fourfit*).

Sometimes recorrelation is required and performed.

Correlator is a very expensive spectrum analyzer => correlator analysts can help debugging problems at stations.

Correlators deliver to analysts the databases or the FITS file to the astronomers.