

Datasim

A VLBI baseband data simulator

Zheng Meyer-Zhao
ASTRON

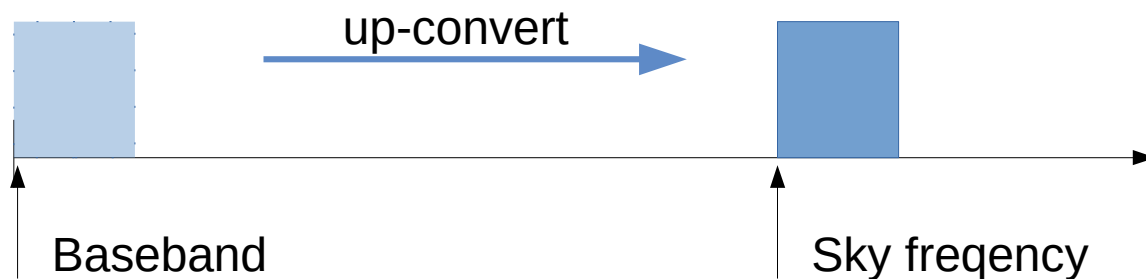
What is Datasim?

A VLBI baseband data simulator

- Part of the implementation is based on ‘anoise’ developed by Geoff Crew
- ASIAA presented ‘enoise’ at DiFX meeting in 2014 (designed by Cheng-Yu Kuo)
- ‘datasim’ (designed during DiFX meeting 2014 by Adam Deller, Walter Brisken and Chris Phillips)

Design principle

- Use functions implemented in DiFX
- Reverse engineering of how DiFX processes data
- Start from frequency domain
- Generate complex data samples at baseband
- Fourier transform the signal to time domain
- Up-convert the signal to sky frequency



How does it work?

- Generate common signal in frequency domain
 - Calculate maximum frequency coverage of all subbands of all stations
 - For example, a 3 stations simulation:
 - 62.5 MHz * 2 and 32 MHz * 4 and 64 MHz * 2
 - Max. freq coverage 128 MHz
- Each subband copy its own part of the frequency data from common signal
 - Apply source flux density
 - Add station noise
 - Apply band-pass filter
 - Transform signal to time domain

How does it work? An example

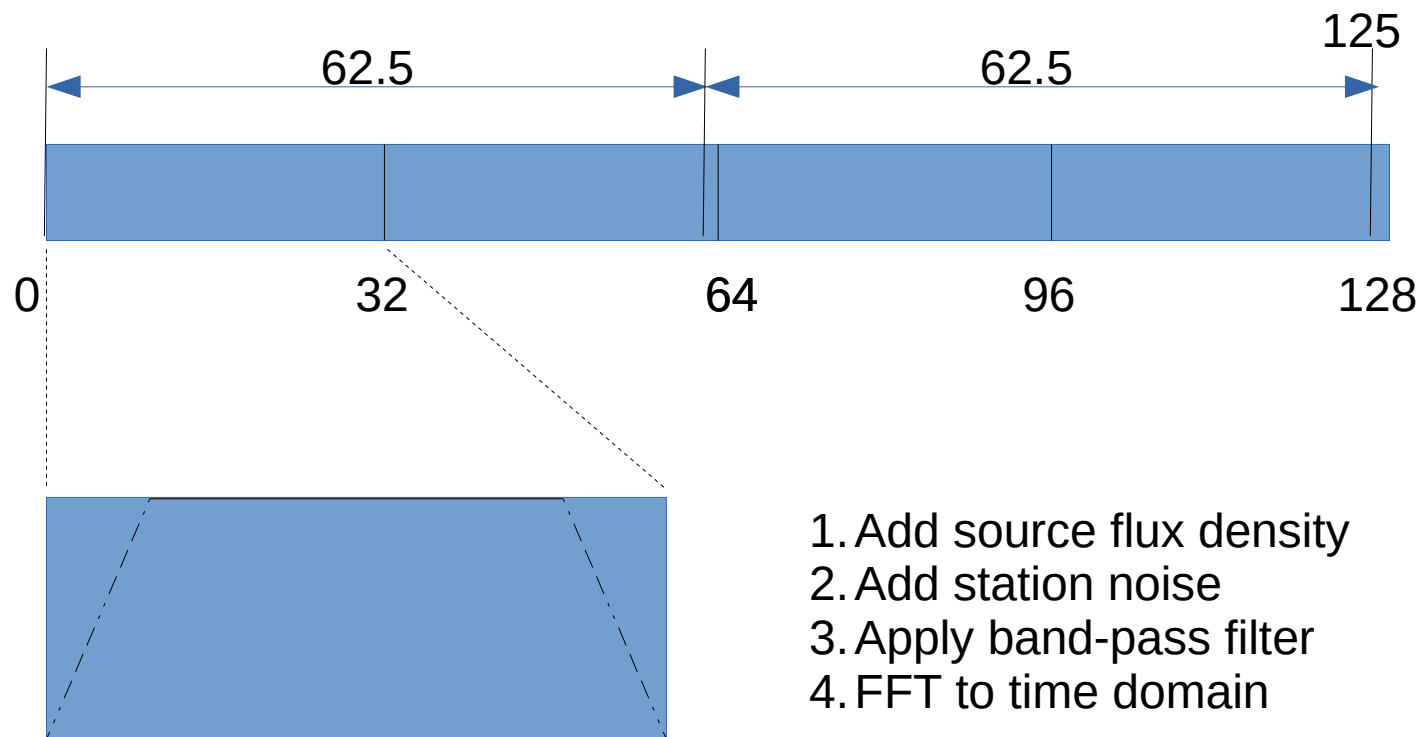
Station 1: $62.5 \text{ MHz} * 2$

Station 2: $32 \text{ MHz} * 4$

Station 3: $64 \text{ MHz} * 2$

Spectral resolution: 0.25

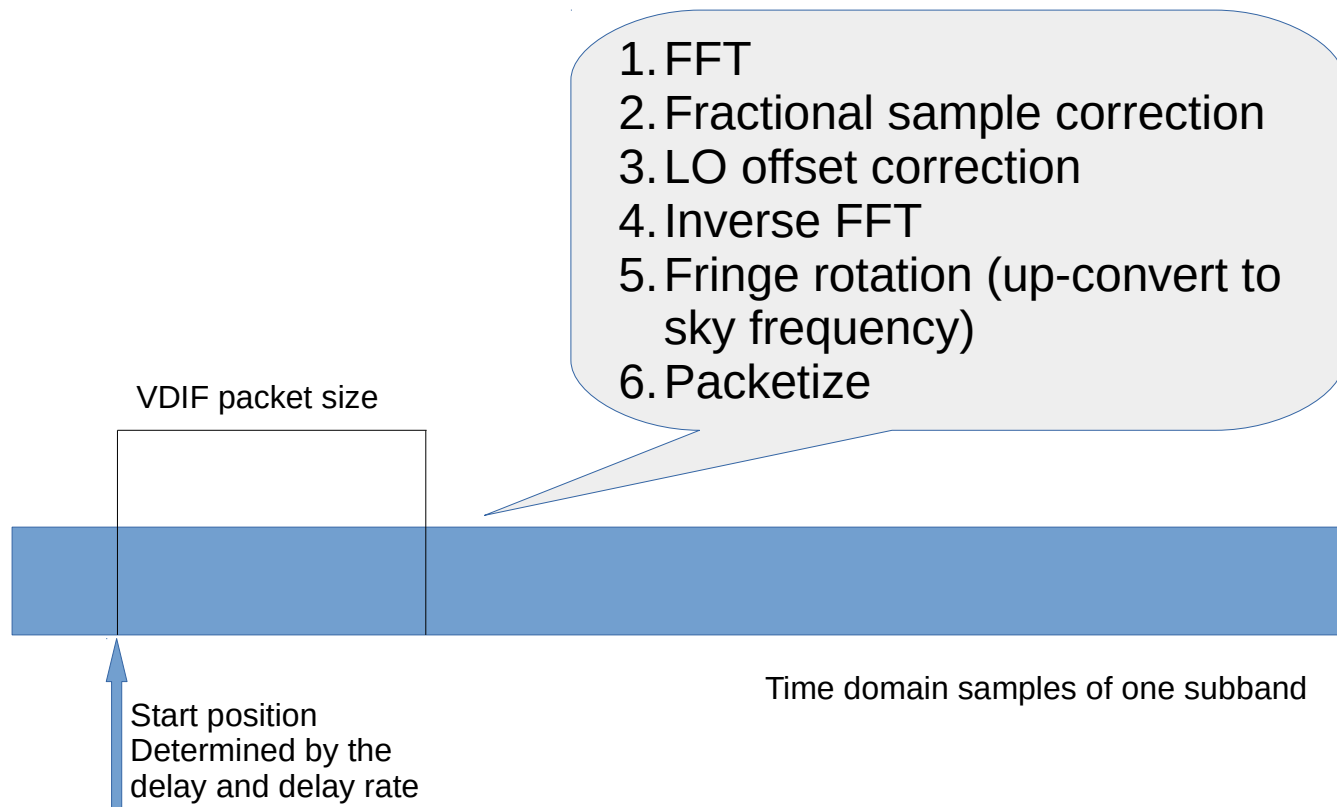
Number of samples per sec: 512 M



How does it work? Cont'd

- Calculate delay and delay rate using DiFX function calls
- Based on the delay model, select where to start processing the time domain signal
- Process data in chunks of VDIF packet size
 - FFT
 - Fractional sample correction
 - LO offset correction
 - Inverse FFT
 - Fringe rotation (up-convert to sky frequency)
 - Convert complex signal to real signal
 - Packetize

How does it work? An example



How does it work? Cont'd

- Generate data in VDIF format for each subband of each station (single thread single channel)
- Combine each subband data into one final data file (vdifzipper)

How to use it?

- What you need to have?
 - Station description file
 - Target description file
 - Simulation description file
- Generate v2d and VEX file using script `genv2dvex.sh`
- Perform `vex2difx` and `calcif2`, because `datasim` uses the `.input` file as input

How to use it? Cont'd

Station Description File

station=SMA ; id=Sp ; site_position=-5464555.493 m : -2492927.989 m : 2150797.176 m

station=ALMA ; id=Al ; site_position=2225039.530 m : -5441197.629 m : -2479303.360 m

station=SMTO ; id=St ; site_position=-1828796.200 m : -5054406.800 m : 3427865.200 m

station=CARMA ; id=Cm ; site_position=-2397378.568 m : -4482048.670 m : 3843513.202 m

station=IRAM30 ; id=Im ; site_position=5088967.900 m : -301681.600 m : 3825015.800 m

station=SC-VLBA; id=Sc ; site_position=2607848.603 m : -5488069.582 m : 1932739.671 m

How to use it? Cont'd

Target Description File

```
source_name=M87 ; IAU_name=1230+123; ra=12h30m49.4233000s ; dec=12d23'28.043000"; ref_coord_frame=J2000
source_name=Sgra ; IAU_name=1745-290; ra=17h45m40.0410000s ; dec=-29d00'28.120000"; ref_coord_frame=J2000
source_name=3C84 ; IAU_name=0319+414; ra=03h19m48.1601000s ; dec=41d30'42.103000"; ref_coord_frame=J2000
source_name=3C279 ; IAU_name=1256-057; ra=12h56m11.1665000s ; dec=-05d47'21.524000"; ref_coord_frame=J2000
source_name=NRAO530 ; IAU_name=1733-130; ra=17h33m02.7057840s ; dec=-13d04'49.548190"; ref_coord_frame=J2000
source_name=NCP ; IAU_name=1733-130; ra=00h00m00.0000000s ; dec=89d59'59.000000"; ref_coord_frame=J2000
```

How to use it? Cont'd

Simulation Description File

```
SOURCE=M87  
STATION="SMA, CARMA, SMTO"  
BAND="64.0x4, 64.0x4, 64.0x4"  
GAP="0.0, 0.0, 0.0"  
SHIFT="0.0, 0.0, 0.0"  
START_TIME=2012y075d05h58m00s  
END_TIME=2012y075d05h58m04s  
FREQ_OBS=230000.00
```

Datasim options

```
meyer@dop420 DiFX-2.5.2 ~> datasim_mpi -h
Usage:  datasim_mpi [<options>] <input file>

options can include:
-h
--help      display this information and quit.

-f
--flux      source flux density in Jansky (default is 1 Jy).

-s
--sefd      antenna SEFDs in a comma-seperated list in Jansky.
             If there are more antennas than provided SEFDs,
             SEFD of the remaining antennas is set to 1000 Jansky.
             Maximum number of antennas supported is 20.

-d
--seed      random number generator seed.

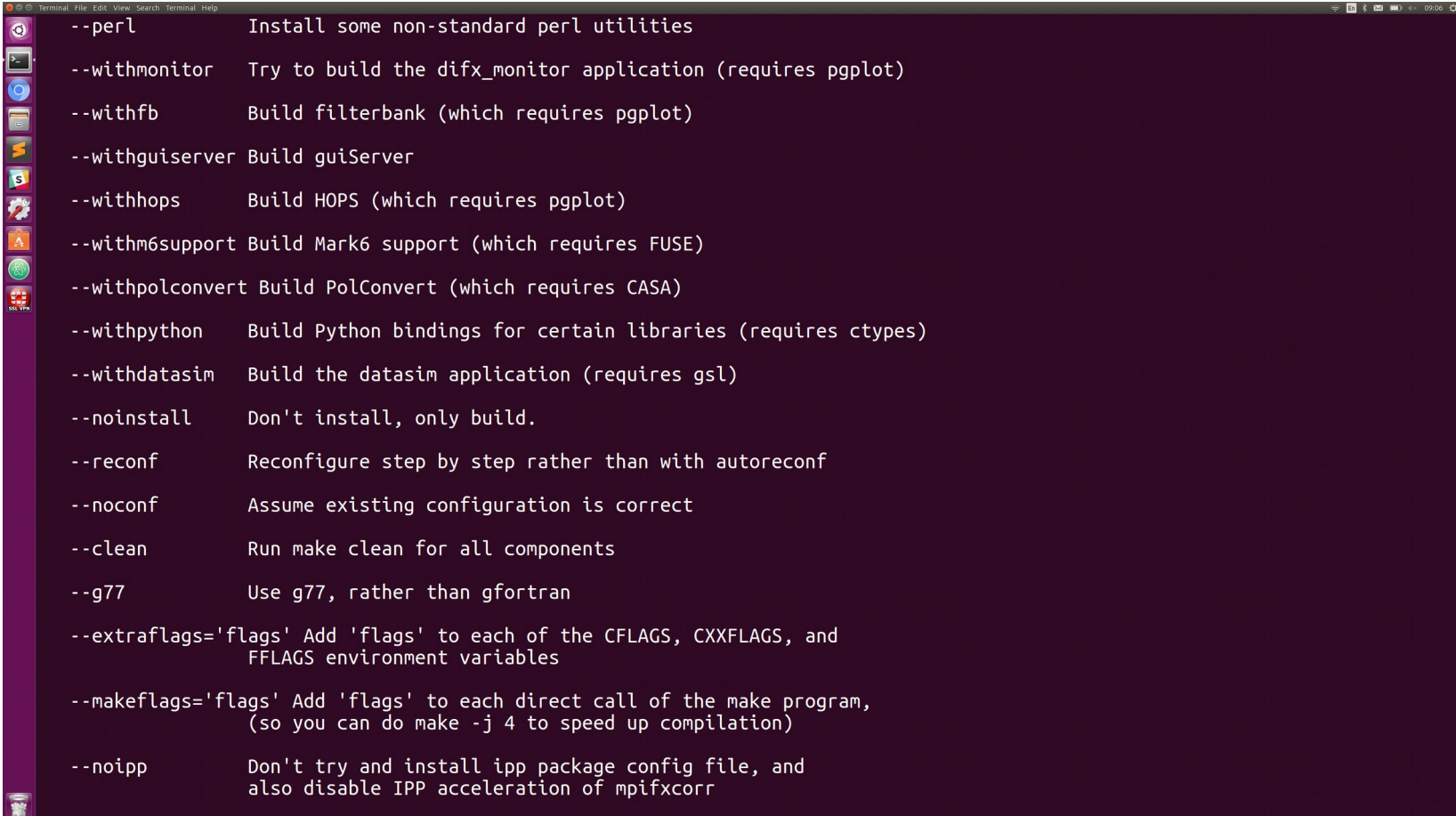
-v
--verbose   increase the verbosity of the output; -v -v for more.

-t
--test      run in test mode, generate 1 second data for each station,
             no matter what's given in the configuration file.

-l
--line      line signal in the form of frequency,amplitude.

meyer@dop420 DiFX-2.5.2 ~> █
```

Install datasim

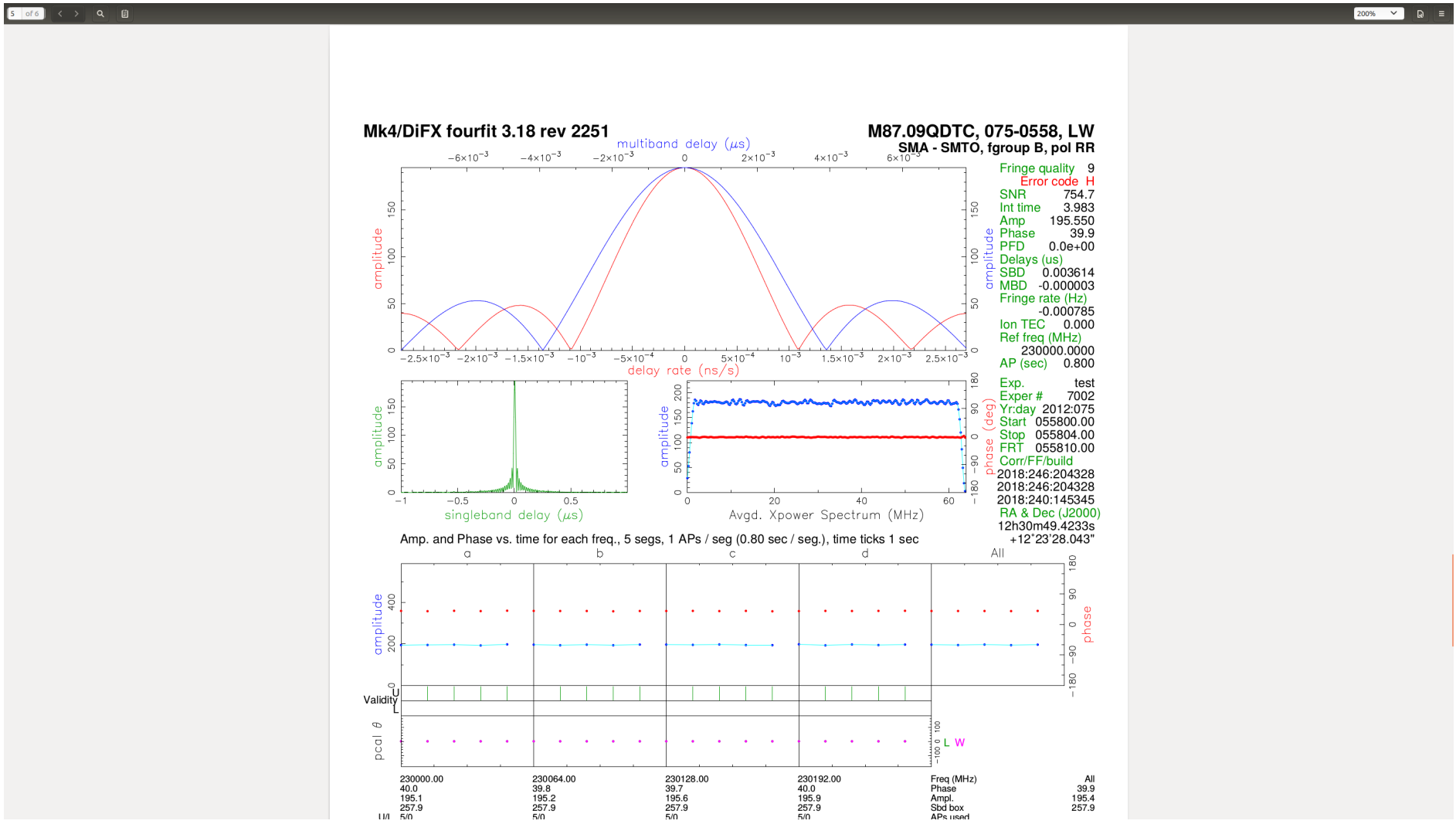
A terminal window with a dark purple background and white text. The window title is "Terminal" and it shows a list of command-line options for installing the 'datasim' application. The options are listed in a key-value format, where the option name is on the left and its description is on the right. The options include: --perl, --withmonitor, --withfb, --withguiserver, --withhops, --withm6support, --withpolconvert, --withpython, --withdatasim, --noinstall, --reconf, --noconf, --clean, --g77, --extraflags='flags', --makeflags='flags', and --noipp. The window also shows a vertical dock on the left with various application icons and a system tray on the right with the time 09:06.

```
Terminal File Edit View Search Terminal Help 09:06
--perl      Install some non-standard perl utilities
--withmonitor  Try to build the difx_monitor application (requires pgplot)
--withfb     Build filterbank (which requires pgplot)
--withguiserver Build guiServer
--withhops   Build HOPS (which requires pgplot)
--withm6support Build Mark6 support (which requires FUSE)
--withpolconvert Build PolConvert (which requires CASA)
--withpython  Build Python bindings for certain libraries (requires ctypes)
--withdatasim Build the datasim application (requires gsl)
--noinstall  Don't install, only build.
--reconf     Reconfigure step by step rather than with autoreconf
--noconf     Assume existing configuration is correct
--clean      Run make clean for all components
--g77        Use g77, rather than gfortran
--extraflags='flags' Add 'flags' to each of the CFLAGS, CXXFLAGS, and
                FFLAGS environment variables
--makeflags='flags' Add 'flags' to each direct call of the make program,
                (so you can do make -j 4 to speed up compilation)
--noipp      Don't try and install ipp package config file, and
                also disable IPP acceleration of mpifxcorr
```

Current development

- Subband-based parallelisation (done)
- Documentation (this week?)
- The following has been implemented, but there are still some issues, hopefully to be fixed this week:
 - Simulate maser source
 - Time-based parallelisation

Results



Future work

- Docker image with DiFX (without IPP optimization), datasim, hops installed
- Benchmarking
- Profiling and improve parallel implementation
- Implement parallelisation using MPI-3 shared memory