Fringe Fitting in Casa

Stephen Bourke

JIVE

Ian Stewart (JIVE, UCT), George Moellenbrock, Walter Brisken &
Jeff Kern (NRAO)

Overview

Introduction to Fringe Fitting

Existing Implementation

Casa Implementation Progress

Fringe Fitting

- Calibration of variable delay
- Due to atmosphere, geometry, clocks, frequency offsets
- Correlator corrects for predictable effects
- Residual delays require treatment

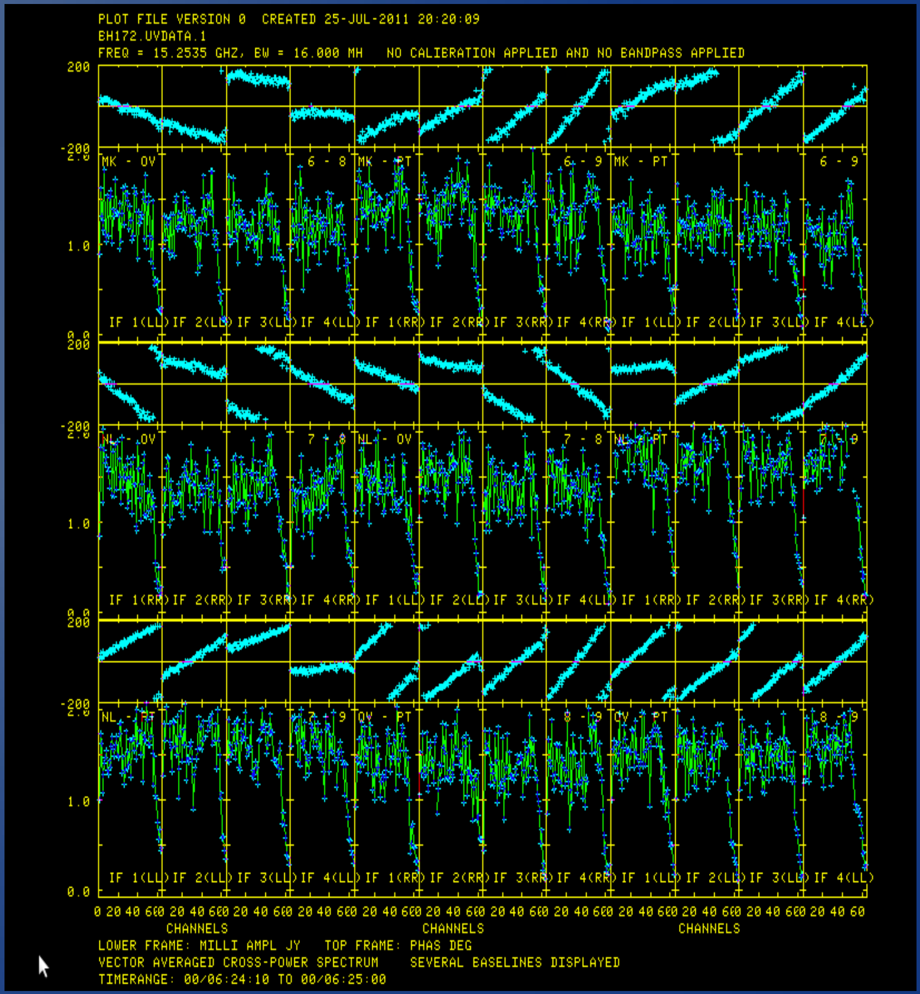
Delay

- Visibility phase:

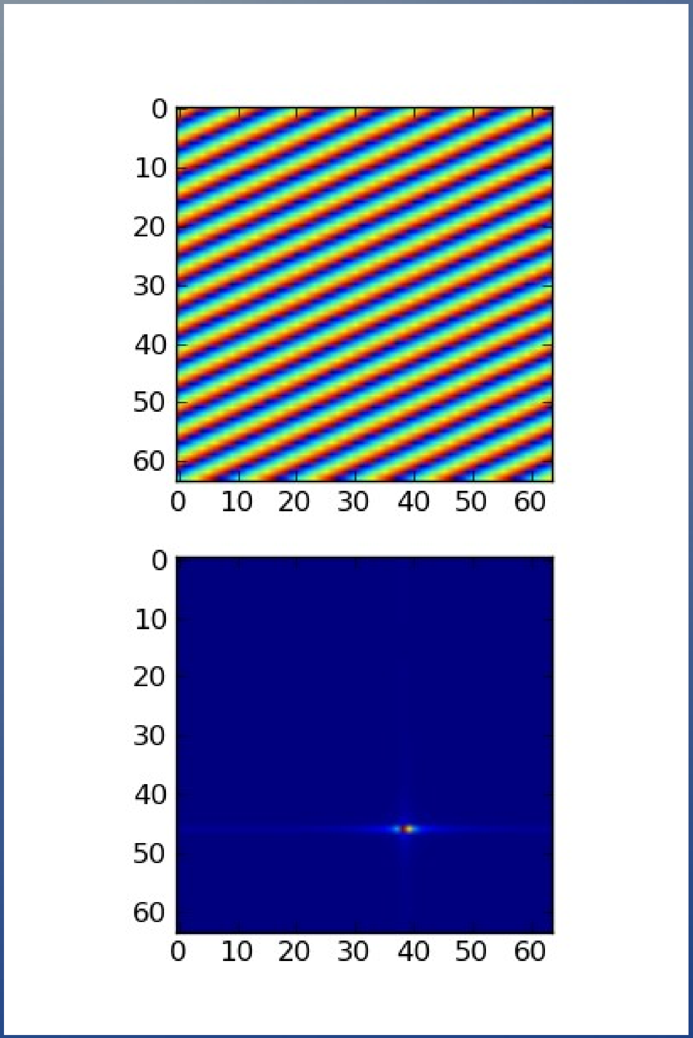
$$\phi_{t,\nu} = 2\pi\nu\tau_t \,;$$

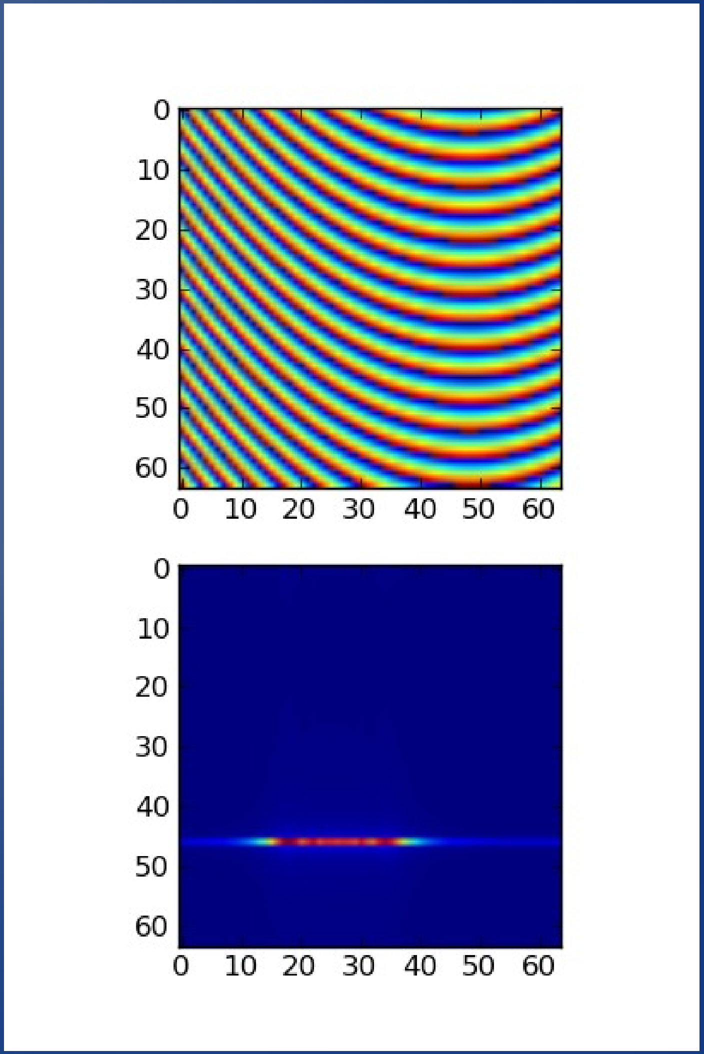
- First Order Expansion:

$$\Delta\phi_{t,\nu} = \phi_0 + \left(\frac{\partial\phi}{\partial\nu}\Delta\nu + \frac{\partial\phi}{\partial t}\Delta t\right) \,;$$



Effect on phase





Standard FF Techniques

- Baseline Based
  - FFT the visibility data
  - Locate peak in delay, fringe rate space
  - Correct phases
  - Advantages:
    - Simple
  - Disadvantages:
    - Need high S/N
    - Does not preserve closure (although it can in some cases)

Alef, Porcas Method

- Baseline with closure constraints
  - Similar to Baseline based
  - Delay, rate parameters are decomposed to antenna based quantities by a least squares fit
  - Baseline Solutions recalculated from antenna solutions and applied

Schwab, Cotton Method

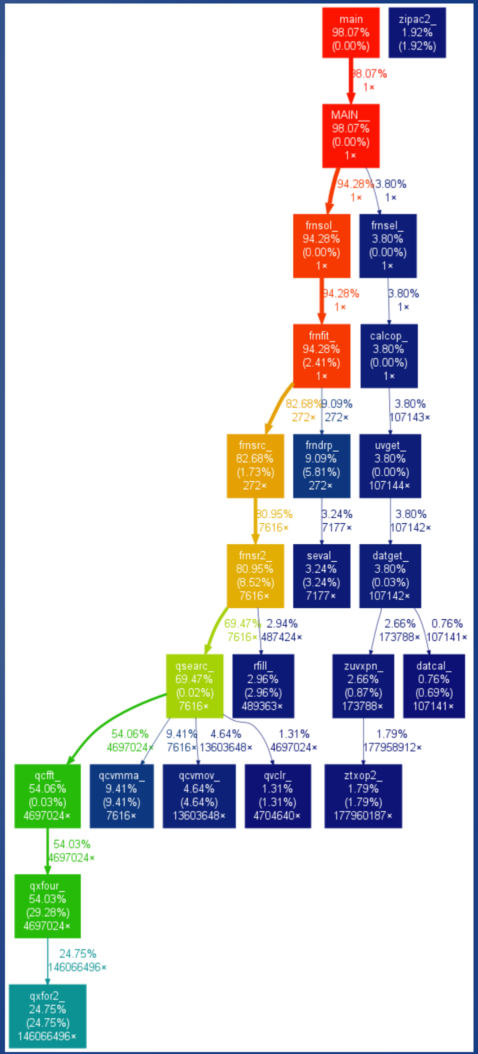
- Global Fringe Fitting
  - All data is used in calculating solutions
  - FFT with baseline stacking
    - N-1 stacked baselines
  - Then LSQ fit to paramaterisation:

$$S_3(\mathbf{x}) = \sum_{k,l}\sum_{i<j} w_{ijkl} \times |\exp\{i[\tilde{\phi}_{ij}(t_k,\nu_l) - \phi_{ij}(t_k,\nu_l)]\} - E_{ijkl}|^2$$

  - Advantages:
    - Most sensitive method
    - Copes well with homogenious arrays
    - Not much slower than other methods
    - Source model is used

AIPS Implementation

- Read Data
- Divide out model
- Optionally stack baselines
- Zero pad & FFT
- Find delay, rate peaks to reference or all antennas
- Use results to do least squares solution
- Optionally apply to data



Casa Implementation

- Calibration system is based on ME
- Highly object orientated
- Calibrator tool creates appropriate cal object
- GJones, BJones, **KJones**
- Extend existing calibration class (GJones) and modify selfSolve and applyCal
- Not entirely modular, eg. interpolation

KJones Implementation

- Initally, delay only
  - Average in time
  - Pad & FFT
  - All baselines to single referance antenna
- Currently working in the KTest class
  - Calculates Delays & Rates
  - Initial baseline stacking code
  - Very early stage of development

Design

- Abstract FringeEngine class
  - Virtual solveDelay method
  - Can be used in sequence
- Subclasses exist for delay only, delay & rate, by FFT method. LSQ planned.
- KTest object can be setup with a vector of FringeEngines and iterate over them to converge on a solution (at least that's the idea)

Summary

- Early stages of the implementation
- Casa / Casacore learning curve to overcome
- Framework in place
- Steady progress
- Hope to include a beta or alpha version in next Casa release