CHIME code overview

Richard Shaw

August 20th, 2020

UBC

CHIME code overview

- CHIME has built up an extensive set of code over the years for
    - Acquisition and telescope control
    - Simulations
    - Data analysis
- Most of our code is publicly available and open source, I'll give links for available code
- Code is primarily Python with C/C++ used for specialised systems
- Core sims/analysis pipeline $>$100k lines of code

Acquisition code

Lots of code here, but I'll briefly highlight some that is publicly available:

- Kotekan: X-engine and realtime backends. C++ `github:kotekan/kotekan`
- comet: dynamic configuration tracking. Python. `github:chime-experiment/comet`
- coco: X-engine control software. Python. `github:chime-experiment/coco`
- alpenhorn: data management and transport. `github:radiocosmology/alpenhorn`
- dias: data monitoring. `github:chime-experiment/dias`

Pipeline infrastructure: simulation and analysis

Central pipeline infrastructure is all in github:radiocosmology/**caput** (Kiyo Masui, RS. . .)

- Our pipeline code is mostly Python with the slow parts using: Numpy/Scipy, Cython or handwritten C/C++.
- In memory, MPI parallel processing
- Uses HDF5 for serialisation
- Pipelines described by YAML files connecting parameterised tasks (written as Python classes) together

Simulation code: foreground maps

All sky simulation code is in github:radiocosmology/**cora** (RS, Mateus Fandino)

- Output is full Stokes Healpix maps at requested frequencies
- Point sources
  - Bright point sources from catalogs
  - Dim point sources by random poission realisations
  - Unresolved point sources as Gaussian background
- Galactic synchrotron
  - Base is Haslam extrapolated with spectral index map
  - . . . plus fluctuations for scales not constrained by Haslam
  - . . . plus spectral fluctuations about a pure power law
  - . . . plus polarisation with differential Faraday rotation (from synthetic Faraday depth distribution)

Simulation code: signal maps

- Fast signal simulations
  - Gaussian realisations of a multi-frequency angular power spectrum
  - Includes redshift-space distortions and light-cone effects
  - . . . but linear
- (Slightly) slower signal simulations (Mateus Fandino)
  - Generate realisations of the full sky non-linear field via Zel'dovich approximation
  - 21cm generated via mapping cell mass to HI mass
  - Include lightcone effects and redshift-space distortions
  - Can also generate quick correlated catalog of arbitrary bias via importance sampling

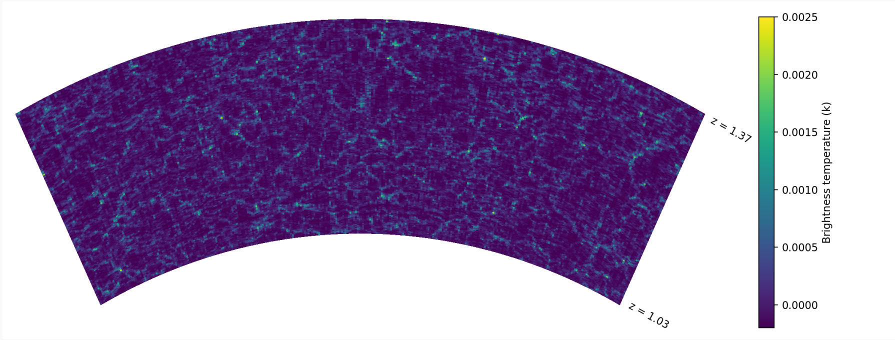

Figure 1: Equatorial slice of a full sky Zel'dovich approximation sky simulation

Instrument simulations

Main instrument simulation code is in (github:radiocosmology/**driftscan** (RS, Carolin Höfer)

- Based around the m-mode formalism
- Generates beam transfer matrices for modelling the instrument
- Needs a model of telescope (Python class) giving
  - Feed positions
  - Primary beam description
- Fast, MPI parallel
- Generate beam perturbations about idealised instruments
- Also generates the KL-filters, power spectrum estimators, and Fisher forecasts
- The bottleneck step, large products 100 TB+ for CHIME

Timestream simulations

Simulation code is primarily in github:radiocosmology/**draco** (RS, Carolin Höfer, Seth Siegel)

- Requires input maps from **cora** and instrument description from **driftscan**
- Uses m-mode formalism to produce *sidereal streams*
- Sidereal streams are then interpolated to arbitray times of observation
- Noise (both Gaussian and correctly distributed Wishart data)
- Systematics can be simulated:
  - Random correlated gain variations
  - Perturbative beam terms (using driftscan terms and random coefficients)

Analysis code

Most *analysis* code that isn't CHIME specific is public within **draco** (RS, Seth Siegel, Saurabh Singh, Carolin Höfer, Tristan Pinsonneault-Marotte, Rick Nitsche, Juan Mena-Parra. . .)

Common stuff that we often couple with simulations

- Map making
- Foreground cleaning (SVD, Delay-filtering, KL-filter)
- Power spectrum estimation (Optimal Quadratic Estimator via KL-filter)

More data based code:

- RFI cleaning
- Re-gridding timestreams onto a sidereal grid
- Stacking sidereal days together
- Source spectrum extraction via beamforming