The Journal of Open Source Software

21cmFAST v3: A Python-integrated C code for generating 3D realizations of the cosmic 21cm signal.

Steven G. Murray[1], **Bradley Greig**[2, 3], **Andrei Mesinger**[4], **Julian B. Mu~noz**[5], **Yuxiang Qin**[4], **and Jaehong Park**[4]

1 School of Earth and Space Exploration, Arizona State University, Phoenix, USA **2** ARC Centre of Excellence for All-Sky Astrophysics in 3 Dimensions (ASTRO 3D) **3** School of Physics, University of Melbourne, Parkville, VIC 3010, Australia **4** Scuola Normale Superiore, Piazza dei Cavalieri 7, 56126 Pisa, Italy **5** Department of Physics, Harvard University, 17 Oxford St., Cambridge, MA, 02138, USA

DOI: 10.21105/joss.02502

Software

- Review
- Repository
- Archive

Editor: Dan Foreman-Mackey

Submitted: 20 July 2020
Published: 28 July 2020

Summary

The field of 21-cm cosmology – in which the hyperfine spectral line of neutral hydrogen (appearing at the rest-frame wavelength of 21 cm) is mapped over large swathes of the Universe's history – has developed radically over the last decade. The promise of the field is to revolutionize our knowledge of the first stars, galaxies, and black holes through the timing and patterns they imprint on the cosmic 21-cm signal. In order to interpret the eventual observational data, a range of physical models have been developed – from simple analytic models of the global history of hydrogen reionization, through to fully hydrodynamical simulations of the 3D evolution of the brightness temperature of the spectral line. Between these extremes lies an especially versatile middle-ground: fast semi-numerical models that approximate the full 3D evolution of the relevant fields: density, velocity, temperature, ionization, and radiation. These have the advantage of being comparable to the full first-principles hydrodynamic simulations, but significantly quicker to run; so much so that they can be used to produce thousands of realizations on scales comparable to those observable by upcoming telescopes, in order to explore the very wide parameter space that still remains consistent with the data.

Amongst practitioners in the field of 21-cm cosmology, the `21cmFAST` program has become the *de facto* standard for such semi-numerical simulators. `21cmFAST` (Mesinger & Furlanetto, 2007; Mesinger, Furlanetto, & Cen, 2011) is a high-performance C code that uses the excursion set formalism (Furlanetto, Zaldarriaga, & Hernquist, 2004) to identify regions of ionized hydrogen atop a cosmological density field evolved using first- or second-order Lagrangian perturbation theory (Scoccimarro & Sheth, 2002; Zel'Dovich, 1970), tracking the thermal and ionization state of the intergalactic medium, and computing X-ray, soft UV and ionizing UV cosmic radiation fields based on parametrized galaxy models. For example, the following figure contains slices of lightcones (3D fields in which one axis corresponds to both spatial *and* temporal evolution) for the various component fields produced by `21cmFAST`.

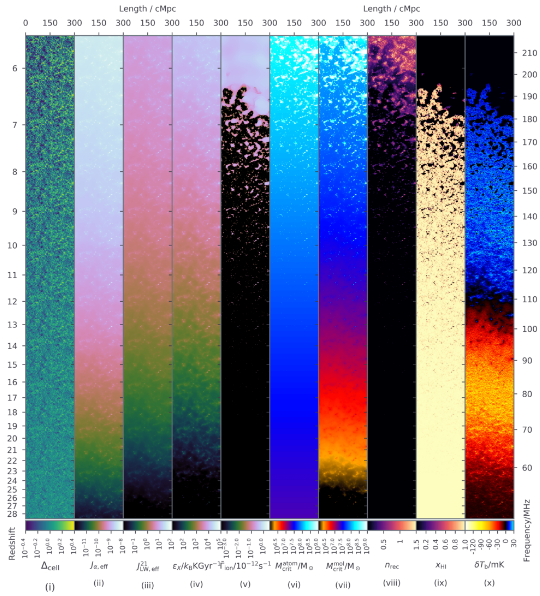

Figure 1: Sample of Component Fields output by 21cmFAST

However, `21cmFAST` is a highly specialized code, and its implementation has been quite specific and relatively inflexible. This inflexibility makes it difficult to modify the behaviour of the code without detailed knowledge of the full system, or disrupting its workings. This lack of modularity within the code has led to widespread code "branching" as researchers hack new physical features of interest into the C code; the lack of a streamlined API has led derivative codes which run multiple realizations of `21cmFAST` simulations (such as the Monte Carlo simulator, `21CMMC`) to re-write large portions of the code in order to serve their purpose. It is thus of critical importance, as the field moves forward in its understanding – and the range and scale of physical models of interest continues to increase – to reformulate the `21cmFAST` code in order to provide a fast, modular, well-documented, well-tested, stable simulator for the community.

Features of 21cmFAST v3

This paper presents `21cmFAST` v3+, which enables these essential new features. While keeping the same core functionality of previous versions of `21cmFAST`, it has been fully integrated into a Python package, with a simple and intuitive interface, and a great deal more flexibility. At a higher level, in order to maintain best practices, a community of users and developers has coalesced into a formal collaboration which maintains the project via a Github organization. This allows the code to be consistently monitored for quality, maintaining high test coverage, stylistic integrity, dependable release strategies and versioning, and peer code review. It also provides a single point-of-reference for the community to obtain the code, report bugs and request new features (or get involved in development).

A significant part of the work of moving to a Python interface has been the development of a robust series of underlying Python structures which handle the passing of data between Python and C via the `CFFI` library. This foundational work provides a platform for future versions to extend the scientific capabilities of the underlying simulation code. The primary *new* usability features of `21cmFAST` v3+ are:

- Convenient (Python) data objects which simplify access to and processing of the various fields that form the brightness temperature.

- Enhancement of modularity: the underlying C functions for each step of the simulation have been de-coupled, so that arbitrary functionality can be injected into the process.
- Conversion of most global parameters to local structs to enable this modularity, and also to obviate the requirement to re-compile in order to change parameters.
- Simple `pip`-based installation.
- Robust on-disk caching/writing of data, both for efficiency and simplified reading of previously processed data (using HDF5).
- Simple high-level API to generate either coeval cubes (purely spatial 3D fields defined at a particular time) or full lightcone data (i.e. those coeval cubes interpolated over cosmic time, mimicking actual observations).
- Improved exception handling and debugging.
- Convenient plotting routines.
- Simple configuration management, and also more intuitive management for the remaining C global variables.
- Comprehensive API documentation and tutorials.
- Comprehensive test suite (and continuous integration).
- Strict `semantic versioning <https://semver.org>_`.

While in v3 we have focused on the establishment of a stable and extendable infrastructure, we have also incorporated several new scientific features, appearing in separate papers:

- Generate transfer functions using the `CLASS` Boltzmann code.
- Simulate the effects of relative velocities between dark matter and Baryons (Muñoz, 2019a, 2019b).
- Correction for non-conservation of ionizing photons (Park, Greig et al., *in prep*).
- Include molecularly cooled galaxies with distinct properties (**???**)
- Calculate rest-frame UV luminosity functions based on parametrized galaxy models.

`21cmFAST` is still in very active development. Amongst further usability and performance improvements, future versions will see several new physical models implemented, including milli-charged dark matter models and forward-modelled CMB auxiliary data.

In addition, `21cmFAST` will be incorporated into large-scale inference codes, such as `21CMMC`, and is being used to create large data-sets for inference via machine learning. We hope that with this new framework, `21cmFAST` will remain an important component of 21-cm cosmology for years to come.

Examples

In-depth examples can be found in the official documentation. As an example of the simplicity with which a full lightcone may be produced with the new `21cmFAST` v3, the following may be run in a Python interpreter (or Jupyter notebook):

```
import py21cmfast as p21c

lightcone = p21c.run_lightcone(
    redshift=6.0,              # Minimum redshift of the lightcone
    max_redshift=30.0,
    user_params={
        "HII_DIM": 150,        # Number of cells along a side in the output cube
        "DIM": 400,            # Original high-resolution cell number
        "BOX_LEN": 300,        # Size of the simulation in Mpc
    },
```

```
    flag_options={
        "USE_TS_FLUCT": True,   # Do not assume saturated spin temperature
        "INHOMO_RECO": True,    # Use inhomogeneous recominations
    },
    lightcone_quantities=(      # Component fields to store as interpolated lightco
        "brightness_temp",
        "xH_box",
        "density"
    ),
    global_quantities=(         # Component fields to store as mean values per reds
        "xH_box",
        "brightness_temp"
    ),
)

# Save to a unique filename hashing all input parameters
lightcone.save()

# Make a lightcone sliceplot
p21c.plotting.lightcone_sliceplot(lightcone, "brightness_temp")
```

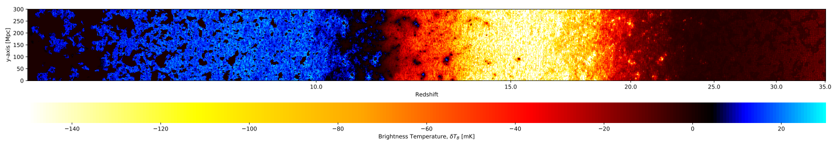

Figure 2: Brightness Temperature Lightcone

```
# Plot a global quantity
p21c.plotting.plot_global_history(lightcone, "xH")
```

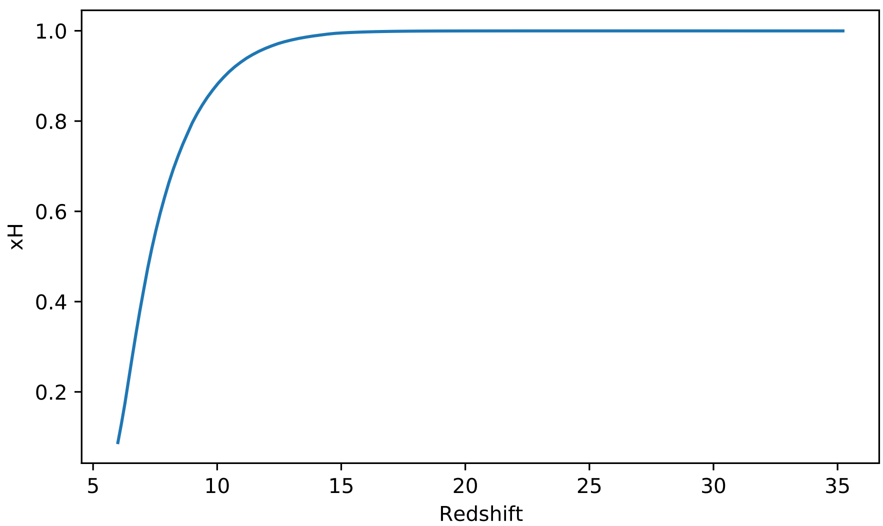

Figure 3: Global reionization history

Acknowledgements

References

Furlanetto, S. R., Zaldarriaga, M., & Hernquist, L. (2004). The Growth of H II Regions During Reionization, *613*(1), 1–15. doi:10.1086/423025

Mesinger, A., & Furlanetto, S. (2007). Efficient Simulations of Early Structure Formation and Reionization, *669*(2), 663–675. doi:10.1086/521806

Mesinger, A., Furlanetto, S., & Cen, R. (2011). 21CMFAST: a fast, seminumerical simulation of the high-redshift 21-cm signal, *411*(2), 955–972. doi:10.1111/j.1365-2966.2010.17731.x

Muñoz, J. B. (2019a). Standard Ruler at Cosmic Dawn, *123*(13), 131301. doi:10.1103/PhysRevLett.123.131301

Muñoz, J. B. (2019b). Robust velocity-induced acoustic oscillations at cosmic dawn, *100*(6), 063538. doi:10.1103/PhysRevD.100.063538

Scoccimarro, R., & Sheth, R. K. (2002). PTHALOS: a fast method for generating mock galaxy distributions, *329*(3), 629–640. doi:10.1046/j.1365-8711.2002.04999.x

Zel'Dovich, Y. B. (1970). Reprint of 1970A&A.....5...84Z. Gravitational instability: an approximate theory for large density perturbations., *500*, 13–18.