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. Muñoz**[5], **Yuxiang Qin**[4], **Jaehong Park**[4, 7], **and Catherine A. Watkinson**[6]

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 **6** School of Physics and Astronomy, Queen Mary University of London, G O Jones Building, 327 Mile End Road, London, E1 4NS, UK **7** School of Physics, Korea Institute for Advanced Study, 85 Hoegiro, Dongdaemun-gu, Seoul, 02455, Republic of Korea

DOI: 10.21105/joss.02582

Software

- Review
- Repository
- Archive

Editor: Dan Foreman-Mackey

Reviewers:

- @sambit-giri
- @sultan-hassan

Submitted: 20 July 2020
Published: 20 October 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 (Lyman-alpha, neutral hydrogen 21-cm, etc.). 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 low-frequency radio 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`.

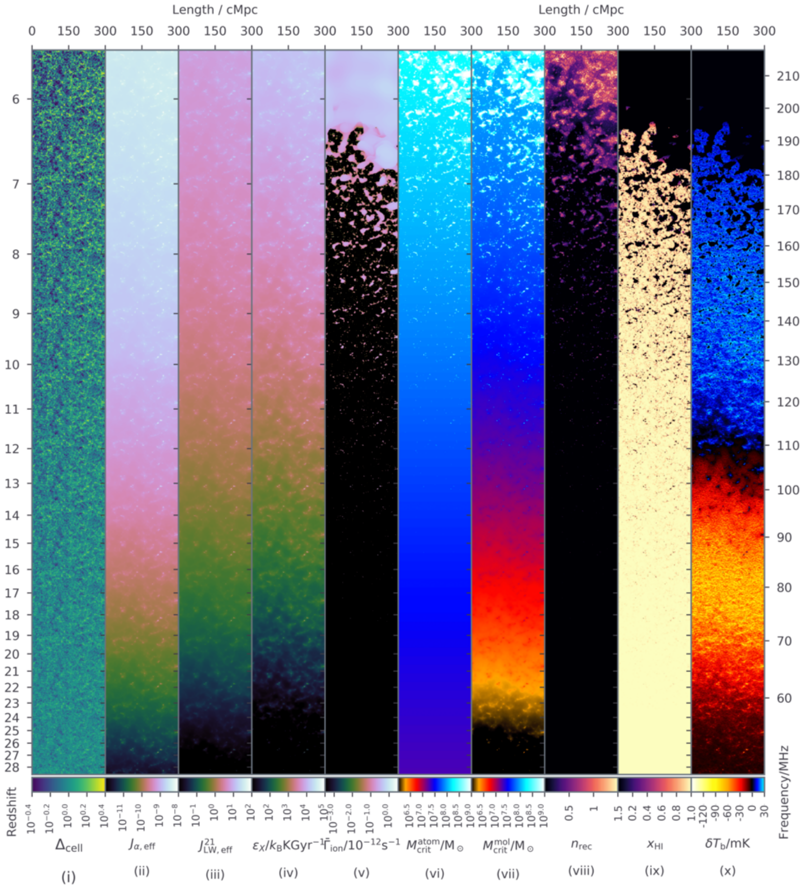

Figure 1: Sample of Component Fields output by 21cmFAST. Cosmic evolution occurs from bottom to top. From left to right, quantities shown are: (i) dark matter overdensity field; (ii) Lyman-alpha flux; (iii) Lyman-Werner flux; (iv) X-ray heating rate; (v) locally-averaged UVB; (vi) critical halo mass for star formation in Atomically Cooled Galaxies; (vii) critical halo mass for star formation in Molecularly Cooled Galaxies; (viii) cumulative number of recombinations per baryon; (ix) neutral hydrogen fraction; and (x) the 21cm brightness temperature. A high-resolution version of this figure is available at http://homepage.sns.it/mesinger/Media/lightcones_minihalo.png

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`, Greig & Mesinger, 2015) 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 is formulated to follow these essential guiding principles. 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[1].

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 (Lesgourgues, 2011).
- 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 (Qin et al., 2020)
- 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 (Muñoz, Dvorkin, & Loeb, 2018) and forward-modelled CMB auxiliary data (Qin, Poulin, et al., 2020).

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.

[1] https://semver.org

Examples

21cmFAST supports installation using conda, which means installation is as simple as typing conda install -c conda-forge 21cmFAST. The following example can then be run in a Python interpreter.

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 lightcone
    max_redshift=30.0,
    user_params={
        "HII_DIM": 150,        # N cells along side in output cube
        "DIM": 400,            # Original high-res cell number
        "BOX_LEN": 300,        # Size of the simulation in Mpc
    },
    flag_options={
        "USE_TS_FLUCT": True,  # Don't assume saturated spin temp
        "INHOMO_RECO": True,   # Use inhomogeneous recombinations
    },
    lightcone_quantities=(     # Components to store as lightcones
        "brightness_temp",
        "xH_box",
        "density"
    ),
    global_quantities=(        # Components to store as mean
        "xH_box",              # values per redshift
        "brightness_temp"
    ),
)

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

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

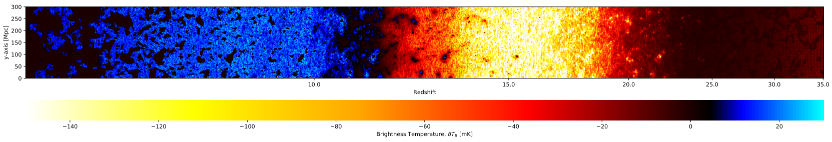

Figure 2: Brightness temperature lightcone produced by the example code in this paper.

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

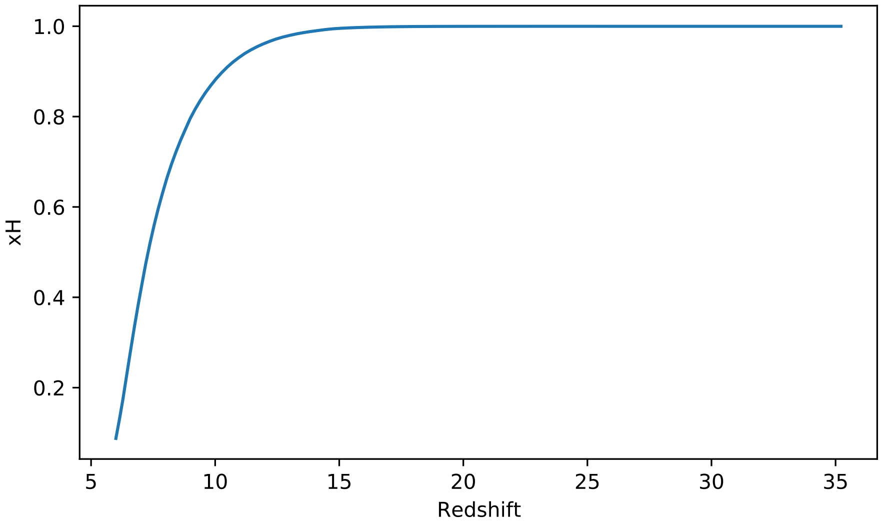

Figure 3: Globally volume-averaged hydrogen neutral fraction produced by the example code in this paper.

Performance

Despite being a Python code, `21cmFAST` v3 does not diminish the performance of previous pure-C versions. It utilises `CFFI` to provide the interface to the C-code through Python, which is managed by some custom Python classes that oversee the construction and memory allocation of each C `struct`.

OpenMP parallelization is enabled within the C-code, providing excellent speed-up for large simulations when performed on high-performance machines.

A simple performance comparison between v3 and v2.1 (the last pure-C version), running a light-cone simulation over a redshift range between 35 and 5 (92 snapshots) with spin temperature fluctuations (`USE_TS_FLUCT`), inhomogeneous recombinations (`INHOMO_RECO`), FFTW Wisdoms (`USE_FFTW_WISDOM`) and interpolation tables (`USE_INTERPOLATION_TABLES`), with a resolution of `HII_DIM=250` cells, and `DIM=1000` cells for the initial conditions, on an Intel(R) Xeon(R) CPU (E5-4657L v2 @ 2.40GHz) with 16 shared-memory cores, reveals that a clock time of 7.63(12.63) hours and a maximum RAM of 224(105) gigabytes are needed for v3(v2.1).

Note that while a full light-cone simulation can be expensive to perform, it only takes 2-3min to calculate a Coeval box (excluding the initial conditions). For instance, the aforementioned timing for v3 includes 80 minutes to generate the initial condition, which also dominates the maximum RAM required, with an additional ~4 minutes per snapshot to calculate all required fields of perturbation, ionization, spin temperature and brightness temperature.

To guide the user, we list some performance benchmarks for variations on this simulation, run with `21cmFAST` v3.0.2. Note that these benchmarks are subject to change as new minor versions are delivered; in particular, operational modes that reduce maximum memory consumption are planned for the near future.

Variation	Time (hr)	Memory (GB)
Reference	7.63	224

Variation	Time (hr)	Memory (GB)
Single Core	14.77	224
4 Shared-memory Cores	7.42	224
64 Shared-memory Cores	9.60	224
Higher Resolution (HII_DIM=500, DIM=2000)	68.37	1790
Lower Resolution (HII_DIM=125, DIM=500)	0.68	28
No Spin Temperature	4.50	224
Use Mini-Halos	11.57	233
No FFTW Wisdoms	7.33	224

At this time, the `21cmFAST` team suggests using 4 or fewer shared-memory cores. However, it is worth noting that as performance does vary on different machines, users are recommended to calculate their own scalability.

Acknowledgements

This work was supported in part by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (AIDA – #638809). The results presented here reflect the authors' views; the ERC is not responsible for their use. JBM was partially supported by NSF grant AST-1813694. Parts of this research were supported by the European Research Council under ERC grant number 638743-FIRSTDAWN. Parts of this research were supported by the Australian Research Council Centre of Excellence for All Sky Astrophysics in 3 Dimensions (ASTRO 3D), through project number CE170100013. JP was supported in part by a KIAS individual Grant (PG078701) at Korea Institute for Advanced Study.

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

Greig, B., & Mesinger, A. (2015). 21CMMC: an MCMC analysis tool enabling astrophysical parameter studies of the cosmic 21 cm signal, *449*(4), 4246–4263. doi:10.1093/mnras/stv571

Lesgourgues, J. (2011). The Cosmic Linear Anisotropy Solving System (CLASS) I: Overview. *arXiv e-prints*, arXiv:1104.2932. Retrieved from http://arxiv.org/abs/1104.2932

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

Muñoz, J. B., Dvorkin, C., & Loeb, A. (2018). 21-cm Fluctuations from Charged Dark Matter, *121*(12), 121301. doi:10.1103/PhysRevLett.121.121301

Qin, Y., Mesinger, A., Park, J., Greig, B., & Muñoz, J. B. (2020). A tale of two sites - I. Inferring the properties of minihalo-hosted galaxies from current observations, *495*(1), 123–140. doi:10.1093/mnras/staa1131

Qin, Y., Poulin, V., Mesinger, A., Greig, B., Murray, S., & Park, J. (2020). Reionization inference from the CMB optical depth and E-mode polarization power spectra, *499*(1), 550–558. doi:10.1093/mnras/staa2797

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). Gravitational instability: an approximate theory for large density perturbations., *500*, 13–18.