




1 lintsampler: Easy random sampling via linear 2 interpolation

3 **Aneesh P. Naik** ¹¶ and **Michael S. Petersen** ¹

4 ¹ Institute for Astronomy, University of Edinburgh, UK ¶ Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Mehmet Hakan Satman](#) 

Reviewers:

- [@matt-graham](#)
- [@vankesteren](#)

Submitted: 14 June 2024

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

5 Summary

6 lintsampler provides a Python implementation of a technique we term 'linear interpolant
7 sampling': an algorithm to efficiently draw pseudo-random samples from an arbitrary PDF.
8 First, the PDF is evaluated on a grid-like structure. Then, it is assumed that the PDF can be
9 approximated between grid vertices by the (multidimensional) linear interpolant. With this
10 assumption, random samples can be efficiently drawn via inverse transform sampling ([Devroye,
11 1986](#)).

12 lintsampler is primarily written with numpy, drawing some additional functionality from
13 scipy. Under the most basic usage of lintsampler, the user provides a PDF function and
14 some parameters describing a grid-like structure to the LintSampler class, and is then able
15 to draw samples via the sample method. Additionally, there is functionality for the user to
16 set the random seed, employ quasi-Monte Carlo sampling, or sample within a premade grid
17 (DensityGrid) or tree (DensityTree) structure.

Statement of need

19 For a small number of well-studied probability distributions, optimised algorithms exist to draw
20 samples cheaply. However, one often wishes to draw samples from an arbitrary PDF for which
21 no such algorithm is available. In such situations, the method of choice is typically some flavour
22 of Markov Chain Monte Carlo (MCMC), a powerful class of methods with many excellent
23 Python implementations ([Coullon & Nemeth, 2022](#); [Fonnesbeck et al., 2015](#); [Foreman-Mackey
24 et al., 2019](#); [Marignier, 2023](#)). One drawback of MCMC techniques is that they typically
25 require a degree of tuning during the setup (e.g. choice of proposal distribution, initial walker
26 positions, etc.), and a degree of inspection afterward to check for convergence. This additional
27 work is a price worth paying for many use cases, but can feel excessive in scenarios where the
28 user is less concerned with strict sampling accuracy or minimising PDF evaluations, and would
29 prefer a simpler means to generate an approximate sample.

30 lintsampler was designed with such situations in mind. In the simplest use case, the user
31 need only provide a PDF function and some one-dimensional arrays representing a grid, and a
32 set of samples will be generated. Compared with MCMC, there is rather less work involved
33 on the part of the user, but there compensating disadvantages. First, some care needs to be
34 taken to ensure the grid has sufficient resolution for the use case. Second, in high dimensional
35 scenarios with finely resolved grids, the PDF might well be evaluated many more times than
36 with MCMC.

37 We anticipate lintsampler finding use in many applications in scientific research and other
38 areas underpinned by statistics. In such fields, pseudo-random sampling fulfils a myriad of
39 purposes, such as Monte Carlo integration, Bayesian inference, or the generation of initial
40 conditions for numerical simulations. The linear interpolant sampling algorithm underpinning

41 `lintsampler` is a simple and effective alternative to existing techniques, and has no publicly
42 available implementation at present.

43 Features

44 Although `lintsampler` is written in pure Python, making the code highly readable, the
45 methods make extensive use of numpy functionality to provide rapid sampling. After the
46 structure spanning the domain has been constructed, sampling proceeds with computational
47 effort scaling linearly with number of sample points.

48 We provide two methods to define the domain, both optimised with numpy functionality for
49 efficient construction. The `DensityGrid` class takes highly flexible inputs for defining a grid.
50 In particular, the grid need not be evenly spaced (or even continuous) in any dimension; the
51 user can preferentially place grid elements near high-density regions. The `DensityTree` class
52 takes error tolerance parameters and constructs an adaptive structure to achieve the specified
53 tolerance. We also provide a base class (`DensityStructure`) such that the user could extend
54 the methods for spanning the domain.

55 Documentation for `lintsampler`, including example notebooks demonstrating a range of
56 problems, is available via a [readthedocs page](#). The documentation also has an extensive
57 explanation of the interfaces, including optimisation parameters for increasing the efficiency in
58 sampling.

59 Acknowledgements

60 We would like to thank Sergey Koposov for useful discussions. APN acknowledges funding
61 support from an Early Career Fellowship from the Leverhulme Trust. MSP acknowledges
62 funding support from a UKRI Stephen Hawking Fellowship.

63 References

- 64 Coullon, J., & Nemeth, C. (2022). SGCMCJax: A lightweight JAX library for stochastic
65 gradient markov chain monte carlo algorithms. *Journal of Open Source Software*, 7(72),
66 4113. <https://doi.org/10.21105/joss.04113>
- 67 Devroye, L. (1986). *Non-uniform random variate generation*. Springer-Verlag.
- 68 Fonnesbeck, C., Patil, A., Huard, D., & Salvatier, J. (2015). *PyMC: Bayesian Stochastic
69 Modelling in Python*. Astrophysics Source Code Library, record ascl:1506.005.
- 70 Foreman-Mackey, D., Farr, W., Sinha, M., Archibald, A., Hogg, D., Sanders, J., Zuntz, J.,
71 Williams, P., Nelson, A., de Val-Borro, M., Erhardt, T., Pashchenko, I., & Pla, O. (2019).
72 emcee v3: A Python ensemble sampling toolkit for affine-invariant MCMC. *The Journal of
73 Open Source Software*, 4(43), 1864. <https://doi.org/10.21105/joss.01864>
- 74 Marignier, A. (2023). PxMCMC: A Python package for proximal Markov Chain Monte Carlo.
75 *The Journal of Open Source Software*, 8(87), 5582. <https://doi.org/10.21105/joss.05582>