IHS Working Paper 60
August 2025

A flexible distribution family for testing MCMC implementations

Tamás K. Papp

INSTITUTE FOR
ADVANCED STUDIES
VIENNA

Author(s)

Tamás K. Papp

Editor(s)

Robert M. Kunst

Title

A flexible distribution family for testing MCMC implementations

Institut für Höhere Studien - Institute for Advanced Studies (IHS)
Josefstädter Straße 39, A-1080 Wien
T +43 1 59991-0
www.ihs.ac.at
ZVR: 066207973

A flexible distribution family for testing MCMC implementations[1]

Tamás K. Papp (tkpapp@gmail.com)
Institute for Advanced Studies, Vienna

August 2, 2025

Abstract

We propose a flexible, extensible family of distributions for testing Markov Chain Monte Carlo implementations. Distributions are created by nesting simple transformations, which allow various shapes, including multiple modes and fat tails. The resulting distributions can be sampled with high precision using quasi-random sequences, and have closed form (log) density and gradient at each point, making it possible to test gradient-based samplers without automatic differentiation.

1 Introduction

Bayesian inference results in a *posterior distribution* over the parameter space of the model. Except for special cases, such posterior distributions do not conform to any well-known family, and their analysis requires *sampling*, which is a computationally intensive task that requires specialized algorithms and software developed for this purpose. The history of Bayesian computation is a procession of increasingly complex and sophisticated algorithms (Robert and Casella 2011), with constant new developments (Bou-Rabee, Carpenter, and Marsden 2024; Bou-Rabee, Carpenter, Liu, et al. 2025).

Testing software implementations of these algorithms is challenging for three reasons. First, their output is *stochastic*, which requires a statistical approach, as opposed to simply comparing that the implementation maps known inputs to known outputs (Whittaker 1997; Ševčíková et al. 2006). Second, as emphasized by Grosse and Duvenaud (2014), it is difficult to separate the problems with the *mixing* of the Markov chains from the flaws in the *implementation*. Third, the very quantities we want to compare are, in general cases, unknown: as Talts et al. (2020) put it,

> The most straightforward way to validate a computed posterior distribution is to compare computed expectations with the exact values. An immediate problem with this, however, is that we know the true posterior expectation values for only the simplest models.

This paper addresses the last point, by providing a very general distribution family which allows

1. accurate sampling using low-discrepancy sequences, which can be compared to MCMC output,
2. closed form calculations for the gradient of the log posterior, allowing the use of gradient-based methods (Neal 2011; Hoffman and Gelman 2014) without automatic differentiation, making the software implementation easier.

We review the related literature in Section 2. Section 3 introduces the model family, while Section 4 demonstrates one possible approach to unit testing MCMC software. Section 5 concludes.

2 Related literature

Geweke (2004) relies on posterior distributions being (unscaled) products of a prior and a likelihood. Samples are obtained in two ways:

1. sample $p(\theta)$ from the prior, then sample $p(x \mid \theta)$ from the generative model,
2. given a (θ, x) from the previous step, update θ using an MCMC transition, then generate data from $p(x \mid \theta)$. The combination of these steps should preserve the joint distribution $p(\theta,x)$.

[1]This paper documents the internals and rationale of the Julia software library Papp and contributors (2025b). Tamás K. Papp acknowledges the support of the Austrian National Bank Jubileumsfonds Projekt 18847. I would like to thank Robert Kunst for comments.

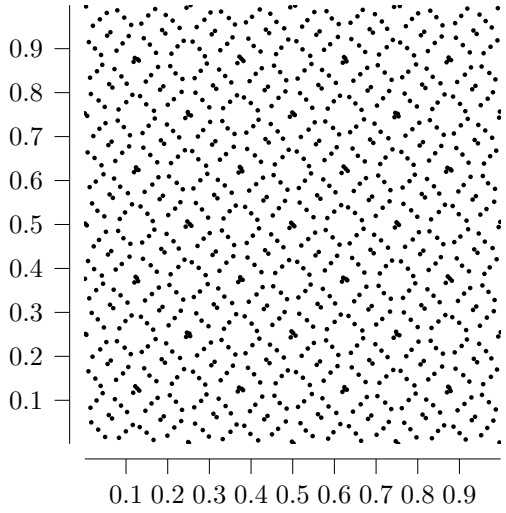

Figure 1: Sobol sequence in 2D, 1024 points.

Both methods should yield samples from exactly the same distribution, and they can be compared using various statistics.

Cook, Gelman, and Rubin (2006) suggest the following procedure. Use the prior $p(\theta)$ to generate a parameters θ_0, then the generative model $p(x \mid \theta_0)$ to generate a data sample x. Finally, obtain posterior samples θ_i, for $i = 1,2,...,L$, and calculate the quantile $\hat{q}(\theta_0) = \frac{1}{L}\sum_{i=1}^{L} 1_{\theta_0 > \theta_i}$. The authors show that as $L \to \infty$, $\hat{q}(\theta_0)$ converges to the uniform distribution. This can be evaluated graphically or using formal statistics.

Talts et al. (2020) suggest comparing the *data averaged posterior*

$$p_D(\theta) = \int p(\theta \mid y) p(y \mid \theta) \mathrm{d}(y) p(\theta)$$

to the prior distribution, where the integral is evaluated numerically, direct sampling fromt the prior $p(\theta)$ and the data generating model $p(y \mid \theta)$, and MCMC for $p(\theta \mid y)$. The comparison is performed using one-dimensional *rank statistics* $f : \Theta \to \mathbb{R}$, using empirical quantiles similarly to the $\hat{q}$ of Cook, Gelman, and Rubin (2006). The preferred method of comparison is visual, and the proposed simulation strategy also allows for error bands.

3 The distribution family

It is assumed that the user has access to an implementation of low-discrepancy sequences[2] to generate quasi-random numbers $u_{(i)}, i = 1, \ldots, N$ in $[0,1]^m$.[3] Figure 1 shows samples from a commonly used low-discrepancy sequence, the Sobol sequence, which we also use in our implementation.

Each distribution implements the *hypercube transform* $h : [0,1]^m \to \mathbb{R}^n$ and a *density function* $p : \mathbb{R}^n \to \mathbb{R}_+$ that have the property

$$\int_{\mathbb{R}^n} g(x) p(x) \mathrm{d}x \approx \frac{1}{N} \sum_{i=1}^{N} g\big(h(u_{(i)})\big) \ell\big(h(u_{(i)})\big)$$

where the properties of the approximation depend on the variation of g as described in Morokoff and Caflisch (1995) and Caflisch (1998).

Provided that a distribution has this property, we are able to

[2] See Niederreiter (1988).

[3] Lacking that, we assume that u_i are obtained from a suitable random number generator, but that is less than ideal for accuracy so we suggest that it is avoided.

1. obtain low-discrepancy samples using h, and
2. compare them to the the result of MCMC methods sampling from f.

We describe the construction of a family of functions that have this property below. For numerical accuracy, we construct the *log density function*

$$\ell(x) = \log(p(x))$$

and its gradient $\nabla\ell(x)$.

Notice the two dimensions associated with a distribution in this family: n is the dimension of the domain, while $m \geq n$ is the *hypercube dimension*. The two may be different because we need extra values to generate mixtures, as explained in Section 3.3.

3.1 Primitives

Assume that a univariate distribtion D is available with *log density* $\ell : \mathbb{R} \to \mathbb{R}_+$ and *inverse cumulative distribution function* $I : [0,1] \to \mathbb{R}$. Then, for a given dimension $n \geq 1$, we define

$$\begin{aligned}
\ell_D(x) &= \sum_{i=1}^{N} \ell(x_i) \\
\nabla\ell_D(x) &= \sum_{i=1}^{N} \ell(x_i) \\
h_D(u) &= \begin{pmatrix} I(u_1) \\ \dots \\ I(u_n) \end{pmatrix}
\end{aligned}$$

A very convenient primitive family is the standard normal. We introduce *StdNormal*(n) using the above notation[4] with

$$\begin{aligned}
\ell(x) &= -\frac{x^2/2 + \log(2\pi)}{2} \\
I(u) &= \sqrt{2}\,\mathrm{erf}^{-1}(2u-1)
\end{aligned}$$

It follows that

$$\begin{aligned}
\ell_{\mathit{StdNormal}}(x) &= -\frac{\|x\|_2^2 + n\log(2\pi)}{2} \\
\nabla\ell_{\mathit{StdNormal}}(x) &= x \\
h_{\mathit{StdNormal}}(u) &= \sqrt{2} \begin{pmatrix} \mathrm{erf}^{-1}(2u_1-1) \\ \dots \\ \mathrm{erf}^{-1}(2u_n-1) \end{pmatrix}
\end{aligned}$$

Figure 2 shows the contour plot of the *highest density regions* of the *StdNormal*(2) distribution, primarily for establishing the convention of illustrating distributions in $\mathbb{R}^2$ with 90%, …, 10% highest density contours in the paper.[5]

3.2 Transformations

All transformations are defined using a *differentiable bijection* $g : \mathbb{R}^n \to \mathbb{R}^n$ where n is the dimension of the distribution. In this section we use the convention $y = g(x)$.

[4] Notation in this paper closely follows the software package Papp and contributors (2025b), with names exported from that package typeset in *slanted* font. Upper/lowercase and minor spelling conventions may differ.

[5] Given the exact sampling using Sobol sequences, such contours are trivial to obtain computationally.

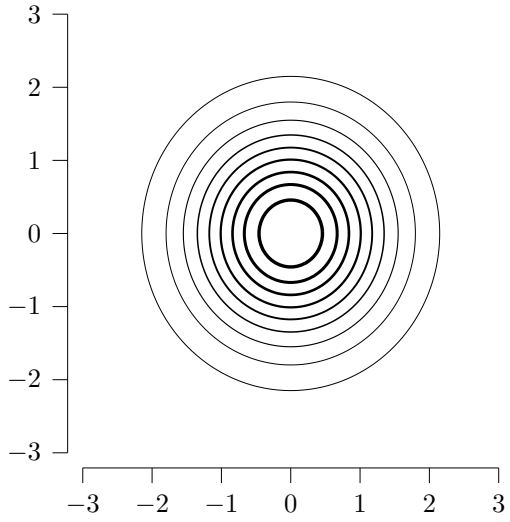

Figure 2: The standard normal distribution *StdNormal*(2). Contour lines are highest density regions containing 90%, …, 10% of the probability mass.

Given a *source* distribution S, characterized by ℓ_S,h_S, the *destination* distribution D has

$$\begin{aligned} h_D(u) &= g(h_S(u)) \\ \ell_D(y) &= \ell_S\big(g^{-1}(y)\big) - \underbrace{\log\left(\left|\det\left(\frac{\partial}{\partial x}\big(g^{-1}(y)\big)\right)\right|\right)}_{\equiv c} \\ \nabla\ell_D(y) &= \nabla\ell_S\big(g^{-1}(y)\big)\underbrace{\frac{\partial}{\partial y}\big(g^{-1}(y)\big)}_{\frac{\partial x}{\partial y}} - \frac{\partial c}{\partial y} \end{aligned}$$

where c is the correction required for transformation of the variables. For all transformations, we provide analyitical characterizations of g^{-1}, c, $\frac{\partial x}{\partial y}$, and $\frac{\partial c}{\partial y}$, which makes calculations possible without automatic differentiation.

All transformations preserve the hypercube dimension of the source distribution.

3.2.1 Linear map

The linear map *Linear*(A) uses the mapping $y = g(x) = Ax$, for an $n \times n$ *invertible* matrix A. Then

$$x = A^{-1}y \qquad \frac{\partial x}{\partial y} = A^{-1} \qquad c = \log(|\det(A)|) \qquad \frac{\partial c}{\partial y} = 0$$

Figure 3 shows an example.

3.2.2 Translation

Translation *Shift*(b) takes a vector b of length n, and is defined using $y = g(x) = x + b$. Then

$$x = y - b \qquad \frac{\partial x}{\partial y} = 0 \qquad c = 0 \qquad \frac{\partial c}{\partial y} = 0$$

Shift may seem like a trivial transformation, but it is very useful since it allows omitting a location parameter from the other transformations: a general transformation τ centered around the origin can be recentered around an arbitrary location b as *Shift*$(b) \circ \tau$*Shift*$(-b)$. We use this extensively below.

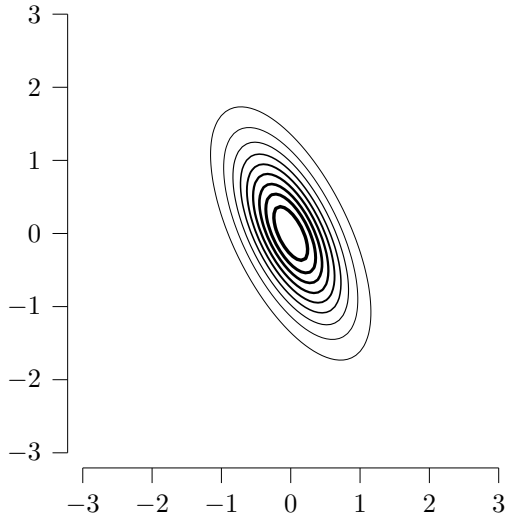

Figure 3: *Linear*(A)(*StdNormal*(2)) with $A = \left(\begin{smallmatrix} 0.2 & 0.5 \\ 0.4 & -0.7 \end{smallmatrix}\right)$. Contour lines are highest density regions containing 90%, …, 10% of the probability mass.

3.2.3 Elongate

The *Elongate*(k) transformation stretches (or shrinks) the tails around the origin. For a given parameter $k \in \mathbb{R}$, it is defined by

$$y = g(x) = x \cdot \left(1 + \|x\|_2^2\right)^k$$

In order to characterize g^{-1}, we define $\xi(Y,k)$ as the $X > 0$ that

$$Y = X \cdot (1 + X^2)^k \tag{1}$$

for all $Y > 0$. The solution always exists since the right hand side of (1) is increasing in X. Calculating $\xi(Y,k)$ requires a univariate numerical solver, such as bisection or Newton's method.[6]

Then, for a given y, let

$$\kappa = \xi(\|y\|_2, k)^2 \qquad D = (1+\kappa)^{-k} \qquad x = D \cdot y$$
$$c = kn\log(1+\kappa)^2 + \log(1 + k\kappa/(1+\kappa)) \qquad A = 1+\kappa \qquad B = 1 + (1+2k)\kappa$$

Using these, we can calculate

$$\frac{\partial x}{\partial y} = (I_n - (2k/B)xx')D \qquad \frac{\partial c}{\partial y} = \frac{k(2+Bn)}{A^{2k} + B^2} \cdot y$$

Figure 4 shows an example.

3.2.4 Funnel

The transformation for *Funnel*, inspired by the well-known example of Neal (2003), uses the mapping

$$y = g(x) = \begin{pmatrix} x_1 \\ x_2 \exp(x_1) \\ \dots \\ x_n \exp(x_1) \end{pmatrix}$$

[6] Newton's method can be set up in a way that it is always convergent in a few steps. See the related source code.

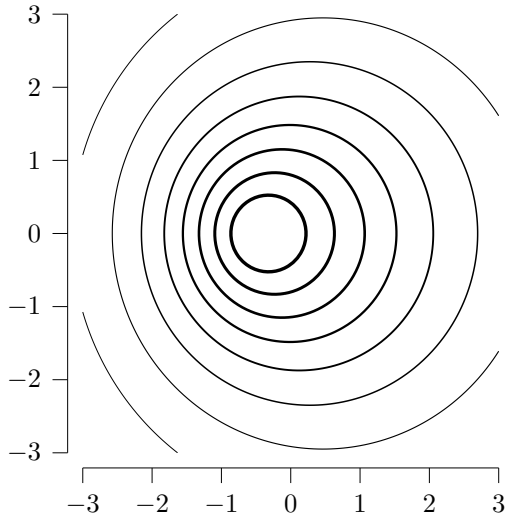

Figure 4: *Elongate*(0.5)(*StdNormal*(2)). Contour lines are highest density regions containing 90%, …, 10% of the probability mass.

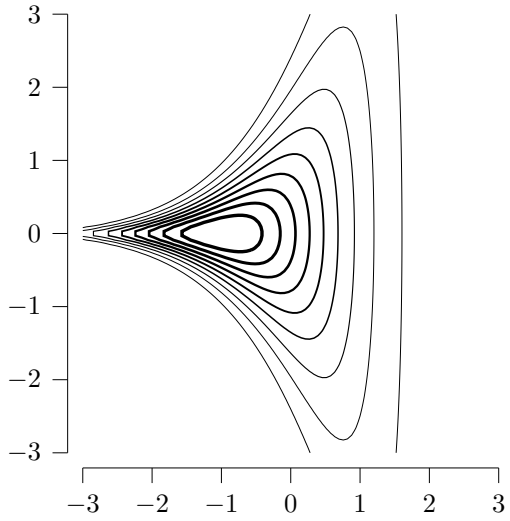

Figure 5: *Funnel*(*StdNormal*(2)). Contour lines are highest density regions containing 90%, …, 10% of the probability mass.

which has

$$\frac{\partial x}{\partial y}_{i,j} = \begin{cases} 1 & \text{if} \quad i=j=1 \\ \exp(-y_1) & \text{if} \quad i=j\neq 1 \\ -y_i\exp(-y_1) & \text{if} \quad i\neq 1, j=1 \\ 0 & \text{otherwise} \end{cases} \qquad c=(n-1)y_1 \qquad \frac{\partial c}{\partial y} = \begin{pmatrix} n-1 \\ 0 \\ ... \\ 0 \end{pmatrix}$$

Figure 5 shows an example.

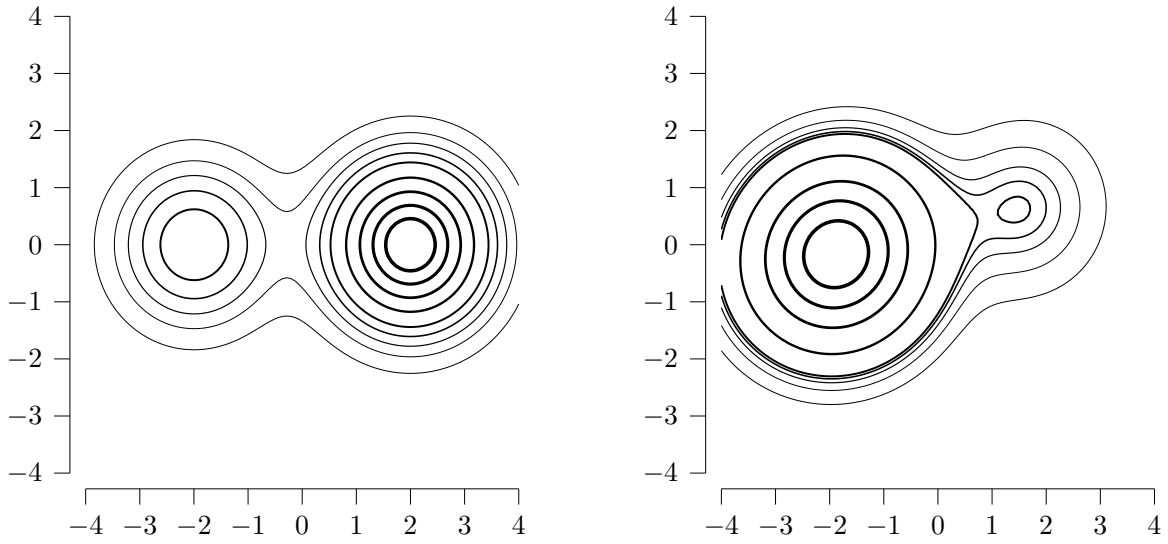

Figure 6: Let $\ell_N = \textit{StdNormal}(2)$, $b = \binom{2}{0}$, $\ell_1 = \textit{Shift}(b)(\ell_N)$, $\ell_2 = \textit{Shift}(3b)(\ell_N)$. Define two mixtures: a *constant* mixture $\textit{Shift}(-2b)(\textit{Mix}(0.3,\ \ell_1,\ \ell_2))$, (left panel) and a *directional* mixture $\textit{Shift}(-2b)(\textit{Mix}(\alpha_D(\cdot;\binom{0.5}{-0.7}),\ell_1,\ell_2))$ (right panel). Contour lines are highest density regions containing 90%, …, 10% of the probability mass.

3.3 Mixtures

Consider two n-dimensional distributions, characterized by h_A,ℓ_A,$\nabla\ell_A$ and h_B,ℓ_B,$\nabla\ell_B$, and a weight function $\alpha : \mathbb{R}^n \to [0,1]$. Define and $n+1$ dimensional distribution with

$$h(u) = \begin{cases} h_A(u[1{:}n]) & \text{if} \quad u[\text{end}] < \alpha(u[1{:}n]), \\ h_B(u[1{:}n]) & \text{if} \quad u[\text{end}] \geq \alpha(u[1{:}n]) \end{cases}$$

where $x[1{:}n]$ are the first n elements of x, and $x[\text{end}]$ is the last element: we use the last coordinate of the hypercube to mix the distributions. The *hypercube dimension* of the mixture distribution is

$$m = \max(m_A, m_B) + 1$$

The log density of the mixture distribution is[7]

$$\ell(x) = \log(\alpha(x)\exp(\ell_A(x)) + (1-\alpha(x))\exp(\ell_B(x)))$$

and its gradient is

$$\nabla\ell(x) = (\nabla\alpha(x) + \alpha(x)\nabla\ell_A(x))\exp(\ell_A(x) - \ell(x)) + (-\nabla\alpha(x) + (1-\alpha(x))\nabla\ell_B(x))\exp(\ell_B(x) - \ell(x))$$

Note that a special case of a *constant* α is what is usually called a *mixture distribution*, but the concept can be generalized readily. For example, let

$$\alpha_D(x; d) = \text{logistic}(x \cdot d)$$

define a *directional weight*, which varies between 0 and 1 in the direction of d. Combined with *Shift* and *Linear*, the 0.5 mixture hyperplane can be relocated from the origin. Figure 6 shows an example.

[7] Care should be taken to prevent over- and underflows; this is addressed in the accompanying package but not detailed here.

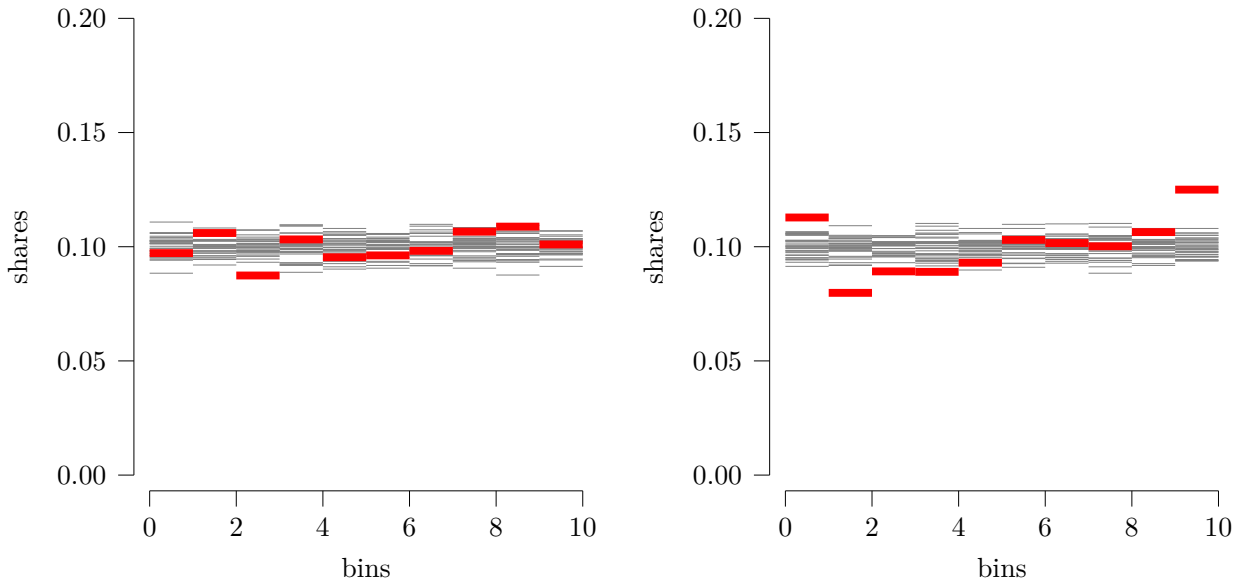

Figure 7: Bin statistics for the correct implementation (left) and an erroneous implementation (right). Thin grey: bin shares obtained using low-discrepancy sequences and a hypercube transform, thick red: MCMC sampling.

4 Use in testing MCMC implementations

Let ℓ be a (log) density constructed using the transformations described in Section 3. An MCMC implementation would provide a *sample* from the posterior ℓ, possibly pooled from multiple chains, after convergence assessment (Vehtari et al. 2021). Let L denote the total (pooled) sample size. At this point, one could generate a sample using the hypercube transform related to ℓ, and compare the two distributions using formal tests, such as the two-sample Kolmogorov–Smirnov test (Berger and Zhou 2014).

Here, we outline a more visual approach that accounts for sampling variation, with the understanding that similar *ad hoc* methods can be invented depending on the intended use. Draw $K \cdot L$ samples using the hypercube transform and a low-discrepancy sequence, where K is a small integer, eg 20–50. Similarly to Cook, Gelman, and Rubin (2006), we use a test quantity f that maps posterior samples to a real number, calculate Q quantiles $q_1, \ldots, q_Q$ of the test quantity using the sample. Then divide the sample into K subsamples and count the proportions of the test quantity in each bin. The idea is to establish sampling variation for the bin ratios using the same sample size.

In order to compare to the MCMC draws, apply f to the posterior, and bin using the same $q_1,\ldots,q_Q$ as above. One can then compare either visually, plotting the low-discrepancy ratios vs the MCMC draws, or using rank statistics.

We demonstrate using the distribution

$$\ell_T = \mathit{Mix}(0.5, \mathit{Funnel}(\ell_{N10}), \ell_{N10}) \quad \text{where} \quad \ell_{N10} = \mathit{StdNormal}(10)$$

which is Neal's funnel (Neal 2003) "tamed" by mixing it with a normal distribution. We test Papp and contributors (2025a), which is an implementation of the No-U-turn sampler as described in Betancourt (2017). The left panel of Figure 7 shows the bin statistics using the first coordinate for the correct implementation, while the right panel demonstrates the effect of introducing a typo in the probability calculations.[8] Observe how the erroneous implementation shows bin ratios outside the variation obtained from the low-discrepancy sample.

Note that the binning is just an example. In a similar manner, one can compare moments or other generalized statistics of distributions. The choice of test statistics is outside the scope of this paper, see Modrák et al. (2025) and Säilynoja, Bürkner, and Vehtari (2022).

[8] While this typo is artificial for the purpose of demonstration, similar mistakes were caught using the approach described while developing Papp and contributors (2025a).

5 Conclusion

By combining simple transformations, we have obtained a very flexible family of distributions in $\mathbb{R}^n$ from which we can obtain accurate samples using low-discrepancy sequences. By comparing these samples to the output of MCMC algorithms and implementations, we can use these to test the latter, and also devise challenging distributions to show how an MCMC algorithm improves on another.

Compared to other approaches, this one just generates a log density, which can be used a "posterior", with the understanding that there is no prior, model, or data involved. Also, the distribution is continuous, so it cannot be used to test MCMC implementations that work on discrete distributions without some appropriate transformation. Furthermore, the user still has to pick test statistics or moments to compare, and which requires understanding of the weak points of each sampler. However, even with these limitations, we believe that this family is a useful addition for unit testing MCMC implementations.

References

Berger, Vance W and YanYan Zhou (2014). "Kolmogorov–smirnov test: Overview". In: *Wiley statsref: Statistics reference online*.

Betancourt, Michael (2017). "A Conceptual Introduction to Hamiltonian Monte Carlo". In: arXiv: 1701.02434v1 [stat.ME].

Bou-Rabee, Nawaf, Bob Carpenter, Sifan Liu, et al. (Feb. 25, 2025). *The No-Underrun Sampler: A Locally-Adaptive, Gradient-Free MCMC Method*. DOI: 10.48550/arXiv.2501.18548. arXiv: 2501.18548[math]. URL: http://arxiv.org/abs/2501.18548 (visited on 03/14/2025).

Bou-Rabee, Nawaf, Bob Carpenter, and Milo Marsden (July 13, 2024). *GIST: Gibbs self-tuning for locally adaptive Hamiltonian Monte Carlo*. DOI: 10.48550/arXiv.2404.15253. arXiv: 2404.15253[math,stat]. URL: http://arxiv.org/abs/2404.15253 (visited on 09/25/2024).

Caflisch, Russel E (1998). "Monte carlo and quasi-monte carlo methods". In: *Acta numerica* 7, pp. 1–49.

Cook, Samantha R, Andrew Gelman, and Donald B Rubin (2006). "Validation of software for Bayesian models using posterior quantiles". In: *Journal of Computational and Graphical Statistics* 15.3, pp. 675–692.

Geweke, John (2004). "Getting it right: Joint distribution tests of posterior simulators". In: *Journal of the American Statistical Association* 99.467, pp. 799–804.

Grosse, Roger B and David K Duvenaud (2014). "Testing MCMC code". In: *arXiv preprint arXiv:1412.5218*.

Hoffman, Matthew D and Andrew Gelman (2014). "The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." In: *Journal of Machine Learning Research* 15.1, pp. 1593–1623.

Modrák, Martin et al. (2025). "Simulation-based calibration checking for Bayesian computation: The choice of test quantities shapes sensitivity". In: *Bayesian Analysis* 20.2, pp. 461–488.

Morokoff, William J and Russel E Caflisch (1995). "Quasi-monte carlo integration". In: *Journal of Computational Physics* 122.2, pp. 218–230.

Neal, Radford M (2003). "Slice sampling". In: *The annals of statistics* 31.3, pp. 705–767.

— (2011). "MCMC using Hamiltonian dynamics". In: *Handbook of Markov Chain Monte Carlo* 2, pp. 113–162.

Niederreiter, Harald (1988). "Low-discrepancy and low-dispersion sequences". In: *Journal of number theory* 30.1, pp. 51–70.

Papp, Tamás K. and contributors (May 2025a). *tpapp/DynamicHMC.jl: v3.5.1*. Version v3.5.1. DOI: 10.5281/zenodo.15358065. URL: https://doi.org/10.5281/zenodo.15358065.

— (June 2025b). *tpapp/LogDensityTestSuite.jl: v0.7.0*. Version v0.7.0. DOI: 10.5281/zenodo.15584840. URL: https://doi.org/10.5281/zenodo.15584840.

Robert, Christian and George Casella (2011). "A short history of MCMC: Subjective recollections from incomplete data". In: *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, pp. 49–61.

Säilynoja, Teemu, Paul-Christian Bürkner, and Aki Vehtari (2022). "Graphical test for discrete uniformity and its applications in goodness-of-fit evaluation and multiple sample comparison". In: *Statistics and Computing* 32.2, p. 32.

Ševčíková, Hana et al. (2006). "Automated testing of stochastic systems: A statistically grounded approach". In: *Proceedings of the 2006 international symposium on Software testing and analysis*. ACM, pp. 215–224.

Talts, Sean et al. (2020). "Validating Bayesian inference algorithms with simulation-based calibration". In: *arXiv preprint arXiv:1804.06788*.

Vehtari, Aki et al. (2021). "Rank-normalization, folding, and localization: An improved R for assessing convergence of MCMC". In: *Bayesian Analysis* 1.1, pp. 1–28.

Whittaker, James A (1997). "Stochastic software testing". In: *Annals of Software Engineering* 4.1, pp. 115–131.