Fast inverse transform sampling in one and two dimensions

Sheehan Olver · Alex Townsend

Abstract We develop a computationally efficient and robust algorithm for generating pseudo-random samples from a broad class of smooth probability distributions in one and two dimensions. The algorithm is based on inverse transform sampling with a polynomial approximation scheme using Chebyshev polynomials, Chebyshev grids, and low rank function approximation. Numerical experiments demonstrate that our algorithm outperforms existing approaches.

1 Introduction

Generating pseudo-random samples from a prescribed probability distribution is an important task in applied probability, statistics, computing, physics, and elsewhere. A classical approach is *inverse transform sampling*, in which pseudo-random samples $U_1, \ldots, U_N$ are generated from a uniform distribution U on $[0, 1]$ and then transformed by $F_X^{-1}(U_1), \ldots, F_X^{-1}(U_N)$, where F_X is the cumulative distribution function (CDF) of a prescribed probability distribution.

Conventional wisdom is that the practical application of inverse transform sampling is limited. To quote [6, pp. 103]:

> [Inverse transform sampling] requires a complete approximation to [the CDF] regardless of the desired sample size, it does not generalize to multiple dimensions and it is less efficient than other approaches.

Similar comments are present in other computational statistics texts, for example, [22, pp. 124], [9, pp. 155], and [17, pp. 122]. Instead, other methods are advocated for low dimensions, such as slice sampling [13] or rejection sampling [6, 9, 17, 22]. In higher dimensions, algorithms concentrate on avoiding the curse of dimensionality, such as Metropolis–Hastings sampling [12]. Moreover, there are many other

School of Mathematics and Statistics, The University of Sydney, NSW 2006, Australia. (sheehan.olver@sydney.edu.au, http://www.maths.usyd.edu.au/u/olver/). Supported by the ARC grant DE130100333 · Mathematical Institute, 24-29 St Giles', Oxford, OX1 3LB, UK. (townsend@maths.ox.ac.uk, http://people.maths.ox.ac.uk/townsend/). Supported by the EPSRC grant EP/P505666/1 and the ERC grant FP7/2007-2013 to Nick Trefethen

methods that are designed to exploit certain special properties: e.g., ziggurat algorithm [10] (requires monotonicity) and adaptive rejection sampling [7,8] (requires log-convexity).

In contradiction to conventional wisdom, we show that inverse transform sampling is computationally efficient and robust when combined with an adaptive Chebyshev polynomial approximation scheme. Approximation by Chebyshev polynomials converges superalgebraically fast for smooth functions, and hence the CDF can be numerically computed to very high accuracy, i.e., close to machine precision. Rather than forming the inverse CDF — which is often nearly singular — we invert the CDF pointwise using a simple bijection method. This approach is robust — it is guaranteed to converge to a high accuracy — and fast, due to the simple representation of the CDF.

Furthermore, we extend our approach to probability distributions of two variables by exploiting recent developments in low rank function approximation to achieve a highly efficient inverse transform sampling scheme in 2D. Numerical experiments confirm that this approach outperforms existing sampling routines for many black box distributions, see Section 4.

Remark 1 We use *black box distribution* to mean a continuous probability distribution that can only be evaluated pointwise[1]. The only other piece of information we assume is an interval $[a,b]$, or a rectangular domain $[a,b]\times[c,d]$, containing the numerical support of the distribution.

A MATLAB implementation is publicly available from [14]. As an example, the following syntax generates 2,000 samples from a 2D distribution:

```
[X,Y] = sample(@(x,y) exp(-x.^2-2.*y.^2).*(x-y).^2,[-3,3],[-3,3],2000);
```

The algorithm scales the input function to integrate to one, and generates pseudo-random samples from the resulting probability distribution.

2 Inverse transform sampling with polynomial approximation

Let $f(x)$ be a prescribed non-negative function supported on the interval $[a,b]$. Otherwise, if the support of f is the whole real line and f is rapidly decaying, an interval $[a,b]$ can be selected outside of which f is negligible. The algorithm will draw samples that match the probability distribution

$$f_X(x) = \frac{f(x)}{\int_a^b f(s)\,\mathrm{d}s}$$

up to a desired tolerance ϵ (typically close to machine precision such as 2.2×10^{-16}).

First, we replace f by a polynomial approximant $\tilde f$ on $[a,b]$, which we numerically compute such that $\|f-\tilde f\|_\infty \le \epsilon\|f\|_\infty$. That is, we construct a polynomial approximant $\tilde f$ of f in the form

$$\tilde f(x) = \sum_{k=0}^{n} \alpha_k T_k\left(\frac{2(x-a)}{b-a}-1\right), \quad \alpha_k\in\mathbb{R}, \qquad x\in[a,b], \tag{1}$$

[1] Strictly, we require the distribution to be continuous with bounded variation [20], and in practice this is almost always satisfied by continuous functions. We do not require the distribution to be prescaled to integrate to one, nor that the distribution decays smoothly to zero.

```
Algorithm: Inverse transform sampling with Chebyshev approximation

Input: A non-negative function $f(x)$, a finite interval $[a,b]$, and an integer $N$.
Output: Random samples $X_1,\dots,X_N$ drawn from $f_X = f/\int_a^b f(s)\,\mathrm{d}s$.

Construct an approximant $\tilde{f}$ of $f$ on $[a,b]$.
Compute $\tilde{F}(x) = \int_a^x \tilde{f}(s)\,\mathrm{d}s$ using (2).
Scale to a CDF, $\tilde{F}_X(x) = \tilde{F}(x)/\tilde{F}(b)$, if necessary.
Generate random samples $U_1,\dots,U_N$ from the uniform distribution on $[0,1]$.
for $k = 1,\dots,N$
    Solve $F_X(X_k) = U_k$ for $X_k$.
end
```

Fig. 1: Pseudocode for the inverse transform sampling algorithm with polynomial approximation using Chebyshev polynomials and Chebyshev grids.

where $T_k(x) = \cos(k\cos^{-1} x)$ is the degree k Chebyshev polynomial. While the polynomial approximants (1) could be represented in other polynomial bases — such as the monomial basis — the Chebyshev basis is a particularly good choice for numerical computation as the expansion coefficients $\alpha_0,\dots,\alpha_n$ can be stably calculated in $\mathcal{O}(n\log n)$ operations [11, 20]. This is accomplished by applying the fast cosine transform to function evaluations of f taken from a Chebyshev grid in $[a,b]$ of size $n = 2^k + 1$, i.e., at the set of points

$$\{(a+b)/2 + (b-a)/2\cos(j\pi/n)\}_{0\le j\le n}\,.$$

We can adaptively select n by sampling f at Chebyshev grids of size $2^3+1\,(=9)$, $2^4+1\,(=17)$, $2^5+1\,(=33)$, and so on, until the tail of the coefficients $\{\alpha_j\}_{0\le j\le n}$ has decayed to below machine precision relative to the absolute maximum of f. In addition to the fast transform, Chebyshev expansions are numerically stable [20], in contrast to the well-known numerical instability associated to interpolation at equispaced points [16].

Instead of symbolically manipulating f, which is not possible if f is a black box distribution, we numerically manipulate $\tilde{f}$ to apply the inverse transform sampling approach. This rests on fast and well-established algorithms for evaluating and integrating polynomial interpolants of the form (1). (A convenient implementation of many of these algorithms is in the Chebfun software system [21].) Figure 1 summarizes the approach we are about to describe based on the inverse transform sampling and Chebyshev approximation.

With the polynomial approximation $\tilde{f}$ in hand, we construct the corresponding cumulative function, denoted by $\tilde{F}$, by calculating the indefinite integral

$$\begin{aligned}\tilde{F}(x) = \int_a^x \tilde{f}(s)\,\mathrm{d}s &= \sum_{k=0}^{n}\alpha_k\int_a^x T_k\left(\frac{2(s-a)}{b-a}-1\right)\mathrm{d}s\\ &= \sum_{k=0}^{n}\alpha_k\frac{b-a}{2}\int_{-1}^{\frac{2(x-a)}{b-a}-1} T_k(t)\,\mathrm{d}t,\end{aligned}$$

where the last equality comes from a change of variables with $t = \frac{2(s-a)}{b-a} - 1$. Fortunately, we have the following relation satisfied by Chebyshev polynomials [11, pp. 32–33]:

$$\int^{s} T_k(t)\,\mathrm{d}t = \begin{cases} \frac{1}{2}\left(\frac{T_{k+1}(s)}{k+1} - \frac{T_{|k-1|}(s)}{k-1}\right), & k \neq 1, \\ \frac{1}{4}T_2(s), & k = 1, \end{cases} \tag{2}$$

where $s \in [-1, 1]$. Therefore, using (2), we can compute coefficients $\beta_0, \ldots, \beta_n$ in $\mathcal{O}(n)$ operations (see, for example, [11, pp. 32–33]) such that

$$\tilde{F}(x) = \sum_{k=0}^{n} \beta_k T_k\left(2\frac{x-a}{b-a} - 1\right), \quad \beta_k \in \mathbb{R}, \qquad x \in [a, b]. \tag{3}$$

At this stage we rescale $\tilde{F}$ so that it is a CDF: $\tilde{F}_X(x) = \tilde{F}(x)/\tilde{F}(b)$.

Once we have the approximant $\tilde{F}_X$, given in (3), we generate N pseudo-random samples $U_1, \ldots, U_N$ from the uniform distribution on $[0, 1]$ using a standard inbuilt pseudo-random number generator, and then solve the following rootfinding problems for $X_1, \ldots, X_N$:

$$\tilde{F}_X(X_k) = U_k, \qquad k = 1, \ldots, N. \tag{4}$$

The rootfinding problems in (4) can be solved by various standard methods such as the eigenvalues of a certain matrix [20], Newton's method, the bisection method, or Brent's method [2, Chapter 4].

To achieve robustness — guaranteed convergence, computation cost, and accuracy — we use the bisection method. The bisection method approximates the (unique) solution X_k to $F_X(X_k) = U_k$ by a sequence of intervals $[a_j, b_j]$ that contain X_k, satisfying $F_X(a_j) < U_k < F_X(b_j)$. The initial interval is the whole domain $[a, b]$. Each stage finds the midpoint m_j of the interval $[a_j, b_j]$, choosing the new interval based on whether $F_X(m_j)$ is greater or less than U_k. Convergence occurs when $b_j - a_j < \mathit{tol}(b-a)$, where $b_j - a_j = 2^{-j}(b-a)$. For example, it takes precisely 47 iterations to converge when $\mathit{tol} = 10^{-14}$ and a further five iterations to converge to machine precision. Since the CDF is represented as a Chebyshev series (3) it can be efficiently evaluated using Clenshaw's method, which is an extension of Horner's scheme to Chebyshev series [3].

The inverse transform sampling with Chebyshev approximation is very efficient, as demonstrated in the numerical experiments of Section 4 using the MATLAB implementation [14].

3 Inverse transform sampling in two variables

We now extend the inverse transform sampling approach with Chebyshev approximation to probability distributions of two variables. Figure 2 summarizes our algorithm, which generates N pseudo-random samples $(X_1, Y_1), \ldots, (X_N, Y_N)$ drawn from the probability distribution $f_{X,Y}(x, y) = f(x, y)/\int_a^b \int_c^d f(s, t)\,\mathrm{d}s\,\mathrm{d}t$. The essential idea is to replace f by a low rank approximant $\tilde{f}$ that can be manipulated in a computationally efficient manner. Again, for convenience, if f is a non-negative function that does not integrate to one then we automatically scale the function to a probability distribution.

```
Algorithm: Inverse transform sampling for $f_{X,Y}$

Input: A non-negative function $f(x,y)$, a finite domain $[a,b] \times [c,d]$, and an integer $N$.
Output: Random samples $(X_1,Y_1),\ldots,(X_N,Y_N)$ from $f_{X,Y} = f/\int_c^d \int_a^b f(s,t)\,\mathrm{d}s\,\mathrm{d}t$.
Construct a low rank approximant $\tilde{f}$ of $f$ on $[a,b] \times [c,d]$.
Compute $\tilde{f}_1(x) = \int_c^d \tilde{f}(x,y)\,\mathrm{d}y$.
Generate random samples $X_1,\ldots,X_N$ from $f_X(x) = \tilde{f}_1(x)/\tilde{f}_1(b)$.
for $i = 1,\ldots,N$
    Generate a random sample $Y_k$ from $\tilde{f}_{Y|X=X_i}(y) = \tilde{f}(X_k,y)/\tilde{f}_1(X_k)$.
end
```

Fig. 2: Pseudocode for the inverse transform sampling algorithm for a probability distribution of two variables denoted by $f_{X,Y}$.

We use a *low rank approximation* of $f(x,y)$. A rank-1 function is a function of two variables that can be written as the product of two univariate functions, i.e., $r_1(x)c_1(y)$. Moreover, a rank-k function is a sum of k rank-1 functions. We efficiently approximate a probability distribution f by a rank-k function by using Gaussian elimination [18]:

$$f(x,y) \approx \tilde{f}(x,y) = \sum_{j=1}^{k} \sigma_j r_j(x) c_j(y), \tag{5}$$

where k is adaptively chosen and c_j and r_j are polynomials of one variable represented in Chebyshev expansions of the form (1). For simplicity we assume that c_j and r_j are polynomials of degree m and n, respectively, and then a function is considered to be of low rank if $k \ll \min(m,n)$.

While most functions are mathematically of infinite rank — such as $\cos(xy)$ — they can typically be approximated by low rank functions to high accuracy, especially if the function is smooth [19]. It is well-known that the singular value decomposition can be used to compute optimal low rank approximations of matrices, and it can easily be extended for the approximation of functions. However, the Gaussian elimination approach from [18] is significantly faster and constructs near-optimal low rank function approximations [1, 18]. This algorithm is conveniently implemented in the recent software package CHEBFUN2 [19], which we utilize to construct $\tilde{f}$. Furthermore, we emphasize that this low rank approximation process only requires pointwise evaluations of f.

With the low rank approximation $\tilde{f}$, given in (5), we can efficiently perform the steps of the 2D inverse transform sampling algorithm. First, we approximate the marginal distribution of X by integrating $\tilde{f}$ over the second variable:

$$\tilde{f}_1(x) = \int_c^d \tilde{f}(x,y)\,\mathrm{d}y = \sum_{j=1}^{k} \sigma_j r_j(x) \int_c^d c_j(y)\,\mathrm{d}y.$$

Therefore, $\tilde{f}_1$ can be computed by applying a 1D quadrature rule, such as the Clenshaw–Curtis quadrature rule [4], to k polynomials of one variable (rather than anything inherently 2D). The resulting sum is a Chebyshev expansion (in x), and we use the algorithm described in Section 2 to generate N pseudo-random samples $X_1, \ldots, X_N$ from $\tilde{f}_X(x) = \tilde{f}_1(x)/\tilde{f}_1(b)$.

Afterwards, we generate the corresponding $Y_1, \ldots, Y_N$ by using a numerical approximation to the conditional distribution denoted by $\tilde{f}_{Y|X}$. To construct $\tilde{f}_{Y|X}$ for each $X_1, \ldots, X_N$ we require evaluation, i.e.,

$$\tilde{f}_{Y|X=X_i}(y) = \frac{\tilde{f}(X_i, y)}{\tilde{f}_1(X_i)}, \qquad i = 1, 2, \ldots, N,$$

where $\tilde{f}_1(X_i) = \int_c^d \tilde{f}(X_k, t)\,\mathrm{d}t$ is precisely the normalization constant. Fortunately, evaluation of $\tilde{f}(X_i, y)$ is relatively cheap because of the low rank representation of $\tilde{f}$. That is,

$$\tilde{f}(X_i, y) = \sum_{j=1}^{k} \sigma_j r_j(X_i) c_j(y),$$

and thus we only require the evaluation of k polynomials of one variable (again, nothing inherently 2D), which, as before, is accomplished using Clenshaw's algorithm [3]. The final task of generating the sample Y_i has been reduced to the 1D problem of drawing a sample from $\tilde{f}_{Y|X=X_i}$, which was solved in Section 2.

In total we have generated N pseudo-random samples $(X_1, Y_1), \ldots, (X_N, Y_N)$ from the probability distribution $f_{X,Y}$ without an explicit expression for the CDF or its inverse. Low rank approximation was important for computational efficiency, and the total computational cost for N samples, including the low rank approximation of f, is

$$\begin{aligned} &\overbrace{\mathcal{O}(k^3 + k^2(m+n))}^{\text{low rank approximation}} + \overbrace{\mathcal{O}(k(m+n))}^{\text{marginal distribution}} + \overbrace{\mathcal{O}(Nn)}^{\text{sample } X_i} \\ &\qquad + \overbrace{\mathcal{O}(N(m+kn))}^{\text{construct } \tilde{f}_{Y|X=X_i}(y)} + \overbrace{\mathcal{O}(Nm)}^{\text{sample } Y_i} \\ &= \mathcal{O}(k^3 + k^2(m+n) + N(n+km)). \end{aligned}$$

Therefore, provided the probability distribution can be well-approximated by a low rank function, the inverse transform sampling approach in 2D has the same order of complexity — i.e., linear in N, m and n — as the 1D approach described in Section 2. Many probability distribution can be approximated by functions of low rank, as we see below.

4 Numerical experiments

In Table 1, we compare the sampling times[2] of our algorithms based on inverse transform sampling, the slice sampling (`slicesample`) routine in MATLAB, and rejection sampling with a rectangular hat function bounded above by the exact maximum of the distribution. We consider the following univariate distributions:

[2] On a 2011 2.7 Ghz Intel Core i5 iMac.

PDF	Rank	ITS	SS	RS
$e^{-\frac{x^2}{2}}(1+\sin^2 3x)(1+\cos^2(5x))$	—	0.29	3.01	0.55
$e^{-4x^2}(9+72x^2-192x^4+512x^6)$	—	0.15	2.87	0.24
$2+\cos(100x)$	—	0.21	—	0.11
$\mathrm{sech}(200x)$	—	0.67	6.76	6.89
$e^{-100(x-1)^2}+e^{-100(y+1)^2}(1+\cos(20x))$	2	1.34	—	2.46
$e^{-\frac{x^4}{2}-\frac{y^4}{2}}(x-y)^2$	3	0.54	—	6.41
$e^{-x^2-2y^2}\mathrm{sech}(10xy)$	16	10.90	—	9.43
$e^{-x^2-2y^2}(y-x)^2\mathrm{sech}(10xy)$	51	19.03	—	7.12

Table 1: Sampling times (in seconds) for our inverse transform sampling (ITS) compared to `slicesample` in MATLAB (SS), and rejection sampling (RS) using the function's maximum for 10,000 samples. The 2D inverse transform sampling approach described in Section 3 is remarkably efficient when the probability distribution is approximated by a low rank function. (Note: `slicesample` only works on decaying PDFs, hence fails on the third example.)

1. Multimodal density: $e^{-\frac{x^2}{2}}(1+\sin^2(3x))(1+\cos^2(5x))$ on $[-8,8]$,
2. The spectral density of a 4×4 Gaussian Unitary Ensemble [5]: $e^{-4x^2}(9+72x^2-192x^4+512x^6)$ on $[-4,4]$,
3. Compactly supported oscillatory density: $2+\cos(100x)$ on $[-1,1]$,
4. Concentrated sech density: $\mathrm{sech}(200x)$ on $[-1,1]$,

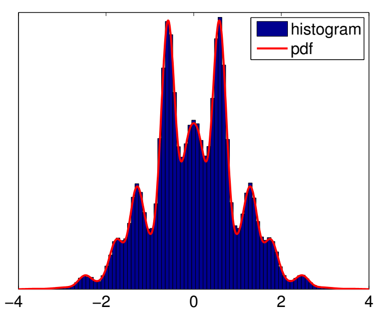
and the following bivariate distributions:

1. Bimodal distribution: $e^{-100(x-1)^2}+e^{-100(y+1)^2}(1+\cos(20x))$ on $[-2,2]\times[-2,2]$,
2. Quartic unitary ensemble eigenvalues [5]: $e^{-\frac{x^4}{2}-\frac{y^4}{2}}(x-y)^2$ on $[-7,7]\times[-7,7]$,
3. Concentrated 2D sech density: $e^{-x^2-2y^2}\mathrm{sech}(10xy)$ on $[-5,5]\times[-4,4]$,
4. Butterfly density: $e^{-x^2-2y^2}\mathrm{sech}(10xy)(x-y)^2$ on $[-3,3]\times[-3,3]$.

Figure 3 demonstrates our algorithm. Figure 3 (left) shows a histogram of one hundred thousand pseudo-random samples generated from the multimodal density, and Figure 3 (right) shows ten thousand pseudo-random samples from the butterfly density.

We observe that the inverse transform sampling approach described in Section 2 significantly outperforms the MATLAB implementation of slice sampling in 1D. In the worst case, the computational cost is comparable to rejection sampling, and in two of the examples it outperforms rejection sampling by a factor of 10.

Rejection sampling has practical limitations that inverse sampling avoids. Finding the exact maximum of the distribution for the rejection sampling is computationally expensive, and replacing the exact maximum with a less accurate upper bound causes more samples to be rejected, which reduces its efficiency. Furthermore, the computational cost of each sample is not bounded, and to emphasize this we consider generating pseudo-random samples from $\mathrm{sech}(\omega x)$ for varying ω on the interval $[-8,8]$. For large ω the percentage of the rectangular hat functions area that is under the probability distribution is small, and the rejection sampling approach rejects a significant proportion of its samples, while for our approach the dominating computational cost stays roughly constant. In Figure 4 (left) we



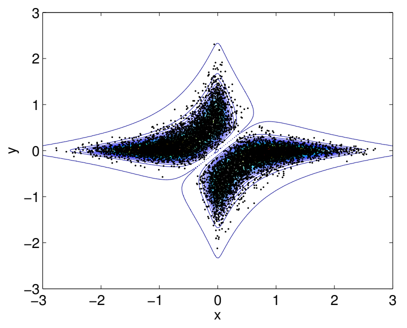
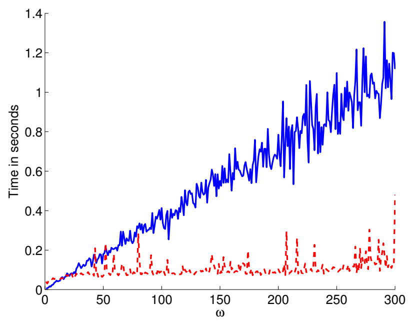


Fig. 3: Left: histogram of one hundred thousand samples from the multimodal density computed in 2.28 seconds, and it can be seen that the generated pseudo-random samples have the correct distribution. Right: ten thousand pseudo-random samples from the butterfly density: $f(x,y) = \mathrm{e}^{-x^2-2y^2}\mathrm{sech}(10xy)(x-y)^2$. Only one sample lies out of the contour lines (the last contour is at the level curve 0.0001), and this is consistent with the number of generated samples.



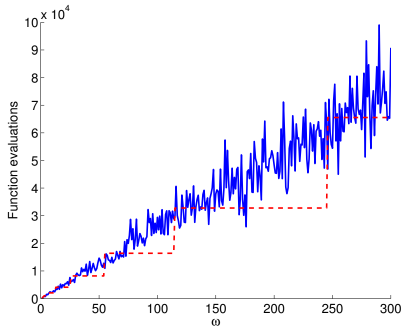


Fig. 4: Left: time (in seconds) of rejection sampling (solid) versus inverse transform method (dashed) for 100 samples of $\mathrm{sech}(kx)$. Right: the number of function evaluations for 50 pseudo-random sample generated by rejection sampling (solid) versus the number of function evaluations for any number of inverse transform samples (dashed).

compare the computational cost of rejection sampling and the approach described in Section 2, and we observe that the Chebyshev expansion outperforms rejection sampling for $\omega \geq 30$.

Remark 2 Of course, for a particular probability distribution a better hat function may be known, which would improve the efficiency of rejection sampling. However, in general accurate hat functions are not available and a rectangular hat function with a known upper bound is the practical choice.

In many applications, the actual evaluation of the probability distribution is expensive. In such circumstances, another benefit of using polynomial approximation in the inverse transform sampling method is that once an approximation has been calculated the original distribution can be discarded, whereas rejection sampling evaluates the original distribution several times per sample. Thus, in Figure 4, we compare the number of function evaluations required to generate 50 samples using rejection sampling versus the number of evaluations for our inverse transform sampling approach. The number of function evaluations are comparable at 50 samples, and hence if 500 pseudo-random samples were required then the rejection sampling approach would require about ten times the number of function evaluations.

Parallelization of the algorithm is important for extremely large sample sizes. This is easily accomplished using a polynomial approximation since the parameters used to represent the CDF can be stored in local memory and the task of generating N pseudo-random samples can be divided up among an arbitrary number of processors or machines. However, the rejection sampling approach cannot be parallelized with a black box distribution because evaluation would cause memory locking.

5 Conclusions

We have shown that inverse transform sampling is efficient when employed with Chebyshev polynomial approximation in one dimension, and with low rank function approximation in two dimensions. This allows for an automated and robust algorithm for generating pseudo-random samples from a large class of probability distributions. Furthermore, the approach that we have described can be generalized to higher dimensions by exploiting low rank approximations of tensors.

In fact, the inverse transform sampling approach is efficient for any class of probability distributions that can be numerically approximated, evaluated, and integrated. Therefore, there are straightforward extensions to several other classes of distributions of one variable:

1. Piecewise smooth probability distributions, utilizing piecewise polynomial approximation schemes and automatic edge detection [15],
2. Probability distribution functions with algebraic endpoint singularities,
3. Probability distributions with only algebraic decay at $\pm\infty$, by mapping to a bounded interval via $\frac{1-x}{1+x}$.

The software package CHEBFUN [21] implements each of these special cases.

In two dimensions and higher, approximating general piecewise smooth functions is a more difficult problem, and many fundamental algorithmic issues remain. However, if these constructive approximation questions are resolved the inverse transform sampling approach will be immediately applicable.

References

1. M. BEBENDORF, *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, Volume 63 of Lecture Notes in Computational Science and Engineering (LNCSE). Springer–Verlag, 2008.
2. R. P. BRENT, *Algorithms for Minimization without Derivatives*, Englewood Cliffs, NJ: Prentice–Hall, 1973.
3. C. W. CLENSHAW, A note on the summation of Chebyshev series, *Maths Comp.*, 9 (1955), pp. 118–110.

4. C. W. Clenshaw and A. R. Curtis, A method for numerical integration on an automatic computer, *Numerische Mathematik*, 2 (1960), pp. 197–205.
5. P. Deift, *Orthogonal Polynomials and Random Matrices: a Riemann–Hilbert Approach*, AMS, 2000.
6. J. E. Gentle, *Random Number Generation and Monte Carlo Methods*, Springer, 2003.
7. W. R. Gilks, Derivative-free adaptive rejection sampling for Gibbs sampling, In *Bayesian Statistics* (J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, eds.), Oxford Univ. Press, (1992), pp. 641–649.
8. W. R. Gilks and P. Wild, Adaptive rejection sampling for Gibbs sampling, *Appl. Statist.* 41 (1992) pp. 337–348.
9. G. H. Givens and J. A. Hoeting, *Computational Statistics, 2nd Edition*, Wiley, 2013.
10. G. Marsaglia and W. W. Tsang, The ziggurat method for generating random variables, *J. Stat. Soft.*, 5 (2000) pp. 1–7.
11. J. C. Mason and D. C. Handscomb, *The Chebyshev Polynomials*, CRC Press, 2003.
12. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, Equations of state calculations by fast computing machines, *J. Chem. Phys.* 21 (1953), pp. 1087–1092.
13. R. M. Neal, Slice Sampling, *Annals Stats*, 31 (2003), pp. 705–741.
14. S. Olver and A. Townsend, Inverse Transform Sampling Matlab Implementation, `https://github.com/dlfivefifty/InverseTransformSampling`.
15. R. Pachón, R. B. Platte, and L. N. Trefethen, Piecewise-smooth chebfuns, *IMA J. Numer. Anal.*, 30 (2010), pp. 898–916.
16. R. B. Platte, L. N. Trefethen, and A. B. J. Kuijlaars, Impossibility of fast stable approximation of analytic functions from equispaced samples, *SIAM Review*, 53 (2011) pp. 308–318.
17. H. Tanizaki, *Computational Methods in Statistics and Econometrics*, CRC PressI Llc, 2004.
18. A. Townsend and L. N. Trefethen, Gaussian elimination as an iterative algorithm, *SIAM News*, March 2013.
19. A. Townsend and L. N. Trefethen, An extension of Chebfun to two dimensions, (2013), submitted.
20. L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, 2013.
21. L. N. Trefethen et al., Chebfun Version 4.2, The Chebfun Development Team, (2011), `http://www.maths.ox.ac.uk/chebfun/`.
22. D. S. Wilks, *Statistical Methods in the Atmospheric Sciences*, Academic Press, 2011.