Continuous Random Variate Generation by Fast Numerical Inversion

WOLFGANG HÖRMANN
Institut für Statistik, WU Wien and IE Department, Boğaziçi University Istanbul
and
JOSEF LEYDOLD
Institut für Statistik, WU Wien

The inversion method for generating non-uniform random variates has some advantages compared to other generation methods, since it monotonically transforms uniform random numbers into non-uniform random variates. Hence it is the method of choice in the simulation literature. However, except for some simple cases where the inverse of the cumulative distribution function is a simple function we need numerical methods. Often inversion by "brute force" is used, applying either very slow iterative methods or linear interpolation of the CDF and huge tables. But then the user has to accept unnecessarily large errors or excessive memory requirements, that slow down the algorithm. In this paper we demonstrate that with Hermite interpolation of the inverse CDF we can obtain very small error bounds close to machine precision. Using our adaptive interval splitting method this accuracy is reached with moderately sized tables that allow for a fast and simple generation procedure.

Categories and Subject Descriptors: G.3 [**Probability and Statistics**]: Random number generation

General Terms: Algorithms

Additional Key Words and Phrases: non-uniform random variates, universal method, inversion method, spline approximation

1. INTRODUCTION

The inversion method for generating non-uniform random variates is based on the simpel observation that a random variate X with cumulative distribution function (CDF) F can be generated by

$$X = F^{-1}(U),$$

where U denotes a uniform $U(0,1)$ random variate. It is the method of choice in many books on simulation (see e.g. Bratley et al. [1983] or Law and Kelton [2000]), since it has some advantages compared to other generation methods. This is due to the fact that it transforms a stream of uniform random numbers one-to-

This work was supported by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung*, Proj. 12805-MAT.

one and monotonically into a stream of non-uniform random variates. Therefore it preservers the structural properties of the underlying uniform (pseudo-) random number generator. Using the inversion method it is easy to sample from truncated distribution, or to generate common or antithetic variates. Moreover, the quality of the generated random variates depends only on the quality of the underlying uniform random number generator, a fact that can hardly be shown for any other generation method (see Leydold et al. [2000]). Furthermore, the inversion method is a true black-box algorithm, since only the cumulative distribution function (CDF) or its inverse must be given.

However, there is also a big drawback for the inversion method: Except in rare cases where the inverse of the CDF can be expressed by simple functions that are available in some computing libraries we need numerical algorithms. Often inversion by "brute force" is used. This means that either slow iterative methods like Newton's method are applied, or the CDF is evaluated at many points and linear interpolation of the CDF is used. In the latter case one either has to accept unnecessarily large errors or huge tables that slow down the algorithm. Consequently, numerical inversion methods may be considered to be of limited practical use.

If speed is an issue the method of Ahrens and Kohrt [1981] is often recommended in the literature. There the inverse CDF is approximated by means of polynomials using numerical integration of the density. Ahrens and Kohrt suggest a fixed decomposition in u-direction into 99 subintervals which have constant probability 1/64 in the center and smaller probabilities in the tails. Inside these intervals the rescaled inverse CDF is approximated by Lagrange interpolation on nine points which is then converted into Chebyshev polynomials. If the precision is higher than required these polynomials are truncated and the resulting Chebyshev expansion is reconverted into a common polynomial which is used as an approximate inverse CDF. Hence this setup is slow and so complicated that, although recommended in the literature, numerical inversion is not included in any of the public domain or commercial scientific libraries we have heard of. In fact in a literature search we were not able to find any recent paper on numerical inversion. On the other hand, after their suggestion by Devroye [1986], fast universal random variate generation algorithms based on rejection were developed and successfully implemented in the last years, see e.g. Gilks and Wild [1992], Ahrens [1995], Hörmann [1995], and Leydold [2000].

Considering the fact that inversion is considered to be "the best" method by many authors (see e.g. Bratley et al. [1983]) we felt that a paper on numerical inversion together with an implementation of the algorithms may have considerable value for simulation practitioners. Therefore we tried to develop a numerical inversion algorithm. It is quite obvious that we can somehow turn the idea of numerical inversion into an working algorithm. But can such an algorithm be *universal* in the sense that a single program can be used to sample from very different distributions? Can it generate from any distribution with a "well behaved" density, using only subroutines to evaluate the CDF and the density of the desired distribution? What is the "optimal" order of the approximating polynomials? We had no idea about answers to these questions and in the literature we could not find them either. Thus we have written this paper to demonstrate that numerical inversion can be realized in a simple, accurate and fast algorithm utilizing tables of moderate size. It has fairly short code and is implemented in our freely available UNU.RAN library (see Leydold and Hörmann [2002]).

2. DEVELOPMENT OF THE ALGORITHM

2.1 General considerations

Most standard distributions are defined by its density $f(x)$. The first practical difficulty of numerical inversion lies thus in the fact that we need an exact routine for evaluating the cumulative distribution function (CDF). For most densities it should be possible to obtain that routine using numerical integration. So in the following we assume that we are given a distribution with CDF F, density f and bounded domain $[b_l, b_r]$. For distributions with unbounded domain we have to chop off its tails at respective points b_l and b_r such that $F(b_l)$ and $1 - F(b_r)$ are small compared to the maximal tolerated approximation error. But this is no big problem in practice as we are given the CDF anyway; see the end of Section 2.3 for a possible simple solution. We will keep in mind throughout this paper that the evaluation of the CDF is expensive or very expensive for most standard distributions. If a very big effort is necessary to obtain the CDF it is probably better to use an automatic algorithm based on the rejection method (see [Hörmann et al. 2003] for a detailed discussion).

To realize inversion we have to find a solution to the equation

$$F(x) = u \qquad \text{for arbitrary } u \text{ with } 0 < u < 1. \tag{1}$$

As iterative methods like Newton's algorithm or bisectioning require typically several evaluations of the CDF to arrive at a solution, we cannot expect them to result in a fast inversion algorithm. So we have to find an approximation to F or F^{-1}. Clearly an approximation to F^{-1} is more convenient as it allows to compute x directly from u. Unfortunately the inverse CDF of distributions with tails are known to be very difficult to approximate. This should not come as a surprise as for such distributions F^{-1} has a very steep slope for u-values close to 0 or 1. Thus a small change of u may lead to a (very) large change in x. But for random variate generation this does not matter too much since almost no points fall in such regions. For example for the standard normal distribution $F^{-1}(10^{-11}) = -6.71$ and $F^{-1}(2 \cdot 10^{-11}) = -6.60$. Now let us take into account that we have a source of uniform (pseudo-)random numbers of limited resolution. Today the set of all possible random numbers typically consists of 2^{32} equally spaced points. Then the resolution (i.e. the distance between two neighboring uniform numbers) is equal to $2^{-32} \approx 2.5 \cdot 10^{-10}$; this means that the minimal probability between two neighboring points generated by exact inversion is 2^{-32} as well. So for generating the normal distribution with exact inversion we can expect at most one point for the interval $(-6.71, -6.60)$ as the "u-distance" between these two points is only 10^{-11}. This is a first indication that in the tails of a distribution the u-error is much smaller than the x-error.

In the literature on Monte-Carlo methods and on the quality of uniform random numbers the *discrepancy* plays an important role. It is also possible to define the discrepancy for non-uniform distributions for a point set $\mathcal{P} = \{x_1, \ldots, x_n\}$ as

$$D_F(n, \mathcal{P}) = \sup_{x \in \mathbb{R}} |F_n(x) - F(x)| \tag{2}$$

where $F_n(x)$ denotes the empirical distribution function of $x_1, \ldots, x_n$ [Fang and Wang 1994].

A simple consideration shows that the one-dimensional discrepancy of the set of all numbers that can be returned by a numerical inversion algorithm can be easily estimated if we know the maximal u-error together with the one-dimensional discrepancy of the used uniform generator [Niederreiter 1992, Lemma 2.5]. This is one reason why we are going to consider mainly the u-error of the approximation. We think it is justified to call a numerical inversion method *almost exact* if the u-error is smaller than the one-dimensional discrepancy of the used uniform random number generator which is in most cases approximately 2^{-32}. The u-error has also the practical advantage that it can be computed without evaluating F^{-1}.

To design our algorithm we also have to decide about the interpolation method we could apply to approximate F^{-1}. We considered (normal) spline interpolation first. (Moore [1983] suggests quadratic splines, but did not provide any details.) However, there are several arguments that favor Hermite interpolation: It is a local approximation, i.e., we can easily improve the level of approximation by inserting new points in regions where the accuracy goal was not reached. The recalculation of the approximation is then only necessary in the newly inserted intervals. In contrast to using spline interpolation where all splines have to be recomputed as this is a global approximation method. Moreover, (normal) splines do not approximate the density in opposition to Hermite interpolation. It has also been reported that Hermite interpolation has slightly better error bounds than splines of the same order and computing the necessary constants is easier.

The simplicity of the Hermite interpolation is certainly another important advantage. Especially the evaluation of the interpolating function is fast which is important for generating random variates. A third point is the fact that there exist easily applicable results on the monotonicity properties of Hermite interpolation (see Section 2.4) but no other literature on simple interpolation methods for monotone functions.

2.2 Hermite Interpolation of F^{-1}

Hermite interpolation of order one is simple *linear interpolation*, the simplest and often used solution to our approximation problem (suggested for example by Bratley et al. [1983]). It requires a (large) table of pairs $(u_i = F(p_i), p_i)$, with $b_l = p_0 < p_1 < \ldots < p_N = b_r$. For an interval $[u_i, u_{i+1}]$ we can then use the approximation

$$H_i^1(u) = L_i(u) = p_i + \tilde{u}\,(p_{i+1} - p_i) \quad \text{where} \quad \tilde{u} = (u - u_i)/(u_{i+1} - u_i). \tag{3}$$

Hermite interpolation of order three (*cubic Hermite interpolation*) uses $f(p_i)$ and $f(p_{i+1})$ together with $F(p_i)$ and $F(p_{i+1})$ to approximate F^{-1}. Note that this is no problem in practice as for all standard distributions the density f is known. In addition the evaluation of the density is much cheaper than the CDF for most distributions. We have to make a table of tuples $(u_i = F(p_i), p_i, f(p_i))$,

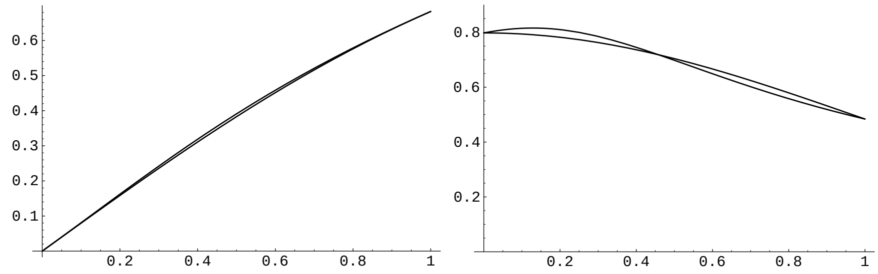

Fig. 1. Approximation of the CDF (l.h.s.) and of the density (l.h.s.) for the normal distribution on subinterval $[0, 1]$ using cubic Hermite interpolation of the inverse CDF.

for $b_l = p_0 < p_1 < \ldots < p_N = b_r$ and use cubic interpolants in each of the intervals $[u_i, u_{i+1}]$, i.e. we use polynomials $H_i^3(u) = C_i(u)$ of degree 3, such that

$$\begin{aligned} C_i(u_i) &= F^{-1}(u_i) = p_i, & C_i'(u_i) &= (F^{-1})'(u_i) = 1/f(p_i), \\ C_i(u_{i+1}) &= F^{-1}(u_{i+1}) = p_{i+1}, & C_i'(u_{i+1}) &= (F^{-1})'(u_{i+1}) = 1/f(p_{i+1}). \end{aligned} \tag{4}$$

Such a polynomial exists and is uniquely determined. It can be easily computed using the usual cubic Hermite basis functions. Thus using (4) we arrive at (see e.g. [Churchhouse 1981, §6.5.3])

$$C_i(u) = (1-\tilde{u})^2\,(1+2\tilde{u})\,p_i + \tilde{u}^2\,(3-2\tilde{u})\,p_{i+1} + (u_{i+1}-u_i)\left(\frac{(1-\tilde{u})^2\,\tilde{u}}{f(p_i)} + \frac{\tilde{u}^2\,(\tilde{u}-1)}{f(p_{i+1})}\right) \tag{5}$$

where

$$\tilde{u} = \frac{u - u_i}{u_{i+1} - u_i} \tag{6}$$

maps $u \in [u_i, u_{i+1}]$ to $[0, 1]$. This can be reduced to

$$C_i(u) = a_{i0} + a_{i1}\,\tilde{u} + a_{i2}\,\tilde{u}^2 + a_{i3}\,\tilde{u}^3 \tag{7}$$

with

$$\begin{aligned} a_{i0} &= p_i, \\ a_{i1} &= (u_{i+1} - u_i)/f(p_i), \\ a_{i2} &= 3\,(p_{i+1} - p_i) - (u_{i+1} - u_i)\ (2/f(p_i) + 1/f(p_{i+1}))\,, \quad \text{and} \\ a_{i3} &= 2\,(p_i - p_{i+1}) + (u_{i+1} - u_i)\ (1/f(p_i) + 1/f(p_{i+1}))\,. \end{aligned} \tag{8}$$

If we consider the density $g(x) = (C^{-1})'(u)$ of the distribution that we have constructed by approximating $F^{-1}(u)$ by $C(u)$ we clearly have $f(p_i) = g(p_i)$ for all p_i and $\int_{p_i}^{p_{i+1}} f(x)\,dx = \int_{p_i}^{p_{i+1}} g(x)\,dx$ for all intervals. Figure 1 illustrates the CDF and the density for the normal distribution and the subinterval $[0, 1]$. As we can see using cubic Hermite interpolation approximates the given inverse CDF much better than linear interpolation, where the corresponding density is constant on the whole interval.

Hermite interpolation of order five (*quintic Hermite interpolation*) uses $f'(p_i)$ and $f'(p_{i+1})$ together with $f(p_i)$, $f(p_{i+1})$, $F(p_i)$ and $F(p_{i+1})$ to approximate F^{-1}. The formulas for the coefficients of the polynomial can be obtained easily by solving a linear system of six equations with six unknown. Using $(F^{-1}(u_i))'' = -f'(p_i)/f(p_i)^3$ we arrive at (see e.g. Dougherty et al. [1989])

$$H_i^5(u) = Q_i(u) = a_{i0} + a_{i1}\,u + a_{i2}\,u^2 + a_{i3}\,u^3 + a_{i4}\,u^4 + a_{i5}\,u^5 \tag{9}$$

with

$$\begin{aligned} a_{i0} &= p_i, \qquad a_{i1} = 1/f_i, \qquad a_{i2} = -f_i'/2f_i^3 \\ a_{i3} &= \tfrac{3f_i'/f_i^3 - f_{i+1}'/f_{i+1}^3}{2\Delta p_i} + 2\,\tfrac{5S_i - 3/f_i - 2/f_{i+1}}{\Delta p_i^2}, \\ a_{i4} &= \tfrac{2f_{i+1}'/f_{i+1}^3 - 3f_i'/f_i^3}{2\Delta p_i^2} + \tfrac{8/f_i + 7/f_{i+1} - 15S_i}{\Delta p_i^3}, \qquad \text{and} \\ a_{i5} &= \tfrac{f_i'/f_i^3 - f_{i+1}'/f_{i+1}^3}{2\Delta p_i^3} + 3\,\tfrac{2S_i - 1/f_i - 1/f_{i+1}}{\Delta p_i^4}, \end{aligned} \tag{10}$$

where $f_i = f(p_i)$, $f_i' = f'(p_i)$, $\Delta p_i = p_{i+1} - p_i$, and $S_i = (u_{i+1} - u_i)/(p_{i+1} - p_i)$.

2.3 Choice of the intervals

A simple method for finding appropriate design points p_i is crucial for our algorithm. Choosing the p_i such that $|u_{i+1} - u_i|$ is constant is a possible solution. It is suggested in the literature for linear interpolation of the inverse CDF as this choice leads to the simplest possible algorithm for randomly choosing the interval. However, we then have to pay the price that we have to compute $p_i = F^{-1}(u_i)$ rather accurately by numerical inversion using slow iterative methods (which makes the setup much more expensive). Even more important is the disadvantage that for this choice of the design points the error of the approximation can be very different in different subintervals. For most distributions the error in the tails will be much larger than for the center of the distribution.

A user of a numerical inversion algorithm is certainly interested to have in some way control over the maximal u-error

$$\epsilon_u = \max_{u \in [u_i, u_{i+1}]} |F(H_i(u)) - u|. \tag{11}$$

Let $\bar{\epsilon}_u$ denote the maximal u-error one is willing to accept. If we want to find design points that guarantee an error bound we certainly need a way to bound the u-error in a single subinterval. We will discuss mathematical error-bounds in Section 3. Unfortunately they require the knowledge of maxima of high-order derivatives and are therefore of little practical use for a universal algorithm. But it applies to common sense and is supported by our extensive numerical experiments with standard distributions that the approximation error will be largest for $\tilde{u}$ close to 1/2 (i.e. for u in the middle of the interval $[u_i, u_{i+1}]$) if the CDF is smooth and has no point of inflection for that interval. Thus we get a good idea of the possible maximal error ϵ_u if we compute the error for $\tilde{u} = 1/2$ which is given by

$$\hat{\epsilon}_u = |F(H_i(\bar{u})) - \bar{u}|\,, \quad \text{where } \bar{u} = (u_i + u_{i+1})/2. \tag{12}$$

We can use this error estimate for recursively calculating design points p_i. We start with a single interval. We then halve this interval recursively until $|u_{i+1} - u_i| = |F(p_{i+1}) - F(p_i)|$ is smaller than some threshold value, e.g. 0.05. Now we continue with checking the error estimate $\hat{\epsilon}_u$ in each of the intervals and continue with splitting them until $\hat{\epsilon}_u$ is smaller than a given error bound. As for smooth densities the error rapidly converges to zero for an increasing number of intervals (see Section 3) the interval splitting terminates. It is clear from the construction of our error estimate that it may be very wrong for the case that the density has a local extremum (as this means an inflection point of the CDF) or is not smooth. We can overcome this problem if we include all local extrema and points of discontinuity of the density or of its derivative as construction points before we start halving the intervals. All our experiments have shown that after the construction of the intervals the average error is clearly smaller than the desired $\bar{\epsilon}_u$ and in all our simulations we only encountered very few cases where ϵ_u was slightly greater than desired. In Section 3 we will discuss error bounds. As they require the maximum of the fourth derivative of the inverse CDF it is of course not possible to use them in the setup. Thus we have no guarantee that the error is smaller than desired as we have only checked it for $\tilde{u} = 1/2$. We could use an optimization procedure to replace the error estimate $\hat{\epsilon}_u$. But for very "bad" densities the optimization program could be fooled as well. So it is important to make an empirical check if the approximation works with the desired precision by computing the u-error for a large sample of $U(0,1)$ variates. Table II reports our results for some standard distributions. Similar experiments can be easily conducted for any other CDF.

We have used the terms halving or splitting the intervals $[p_i, p_{i+1}]$ above. We have still the freedom of choice for the split point. The simplest solution $(p_i + p_{i+1})/2$ turns out to be most stable. It can also be argued that we should use $H(\bar{u})$ as this saves one evaluation of the CDF; but we have observed disadvantages of this approach for cubic Hermite interpolation when the subintervals are still quite long. Note that for linear interpolation both approaches result in the same split point.

Our splitting procedure cannot start without an initial interval. For distributions with unbounded domain we can use by far too large intervals, e.g. $[-100, 100]$ for the normal distribution if we add the following simple step in the procedure to find the design points: If $u_1 = F(p_1) < 0.1\,\bar{\epsilon}_u$ discard p_0; and if $u_i = F(p_i) > 1 - 0.1\,\bar{\epsilon}_u$ discard p_{i+1}. This simple cut off is also important if $f(b_l) = 0$ or $f(b_r) = 0$. Then the coefficients for cubic and quintic Hermite interpolation produce an overflow and the leftmost and/or rightmost intervals are split till the tail can be chopped off.

2.4 Monotonicity

As F^{-1} is always monotonically increasing we also want an approximation that preserves this property. For linear interpolation this property is clearly fulfilled, for spline interpolation (see e.g. Manni [1996]) it is not a trivial task. For cubic Hermite interpolation we can split intervals if we encounter an interval where C_i is not monotone. Fritsch and Carlson [1980] give an exact condition when C_i is monotone which only uses the data points. We use a much simpler sufficient condition which is due to de Boor and Swartz [1977] (see also Huynh [1993]): C_i is monotonically increasing, if both $(F^{-1})'(u_i)$ and $(F^{-1})'(u_{i+1})$ do not exceed $3\,(F^{-1}(u_{i+1}) - F^{-1}(u_i))/(u_{i+1} - u_i)$, or equivalently if

$$\frac{1}{f(p_i)} \leq 3\,\frac{p_{i+1} - p_i}{u_{i+1} - u_i} \quad \text{and} \quad \frac{1}{f(p_{i+1})} \leq 3\,\frac{p_{i+1} - p_i}{u_{i+1} - u_i}. \tag{13}$$

Thus we can find a monotone increasing approximation C of F^{-1} if we split all intervals $[u_i, u_{i+1}]$ where this condition is violated. If the density is continuous $(F^{-1})'$ is continuously differentiable. So we easily get

$$\frac{1}{f(p_i)} = (F^{-1})'(u_i) = \frac{p_{i+1} - p_i}{u_{i+1} - u_i} + O(u_{i+1} - u_i)$$

which clearly implies that the splitting always terminates.

For quintic Hermite interpolation the situation is much more complicated. Dougherty et al. [1989]) show the following sufficient condition. Let

$$\begin{aligned} L_0(a,b) &= -7.9a - 0.26ab, \\ U_0(a,b) &= 20 - 8a - 2b - 0.48ab, \\ L_1(a,b) &= -U_0(b,a), \qquad \text{and} \\ U_1(a,b) &= -L_0(b,a). \end{aligned} \tag{14}$$

Then (using $(F^{-1}(u_i))'' = -f'(p_i)/f(p_i)^3$) Q_i is monotone in $[p_i, p_{i+1}]$ if

$$\begin{aligned} &\tfrac{-f_i'/f_i^3}{S_i/\Delta p_i} \in [L_0(\tfrac{1}{f_i\,S_i}, \tfrac{1}{f_{i+1}\,S_i}), U_0(\tfrac{1}{f_i\,S_i}, \tfrac{1}{f_{i+1}\,S_i})] \qquad \text{and} \\ &\tfrac{-f_i'/f_i^3}{S_{i-1}/\Delta p_{i-1}} \in [L_1(\tfrac{1}{f_{i-1}\,S_{i-1}}, \tfrac{1}{f_i\,S_{i-1}}), U_1(\tfrac{1}{f_{i-1}\,S_{i-1}}, \tfrac{1}{f_i\,S_{i-1}})]. \end{aligned} \tag{15}$$

where again $f_i = f(p_i)$, $f_i' = f'(p_i)$, $\Delta p_i = p_{i+1} - p_i$, and $S_i = (u_{i+1} - u_i)/(p_{i+1} - p_i)$. We have no formal proof that the splitting procedure for quintic Hermite interpolation will terminate but did never encounter problems in practice.

2.5 Vanishing density

Cubic and quintic Hermite interpolation does not work if the density vanishes in some points of the domain, i.e. if $f(x) = 0$ for some points x. We also have a problem if its first derivative $|f'(x)|$ is unbounded. The easiest way out of this problem is to reduce the order of the interpolation: If for some interval $[p_i, p_{i+1}]$ we find $f(p_{i+1}) = 0$ then set $a_{i2} = \ldots = a_{i5} = 0$, i.e., use linear interpolation which always works.

2.6 The sampling algorithm

After we have computed the design points and the approximations of the inverse CDF in all subintervals sampling from this approximate distribution is easy: Sample a $(0,1)$-uniform random number U, use indexed search [Chen and Asau 1974] to select one of these (short) intervals $[u_i, u_{i+i}]$, and apply H_i. Notice that the expected number of comparisons does not depend on the number N of intervals if we increase the size of the index table (also called *guide-table*) linearly with N. Figure 2 collects the necessary details of fast numerical inversion. The optional data about extrema and discontinuities overcome the problem that the error estimate might be too small for special cases.

To keep the presentation simple we have omitted some details. This basic algorithm can be easily be extended such that it takes care about some special cases: Other design points can be used for the starting linked list to circumvent the problems that might occur if the density f of the given distribution is not differentiable or has some inflection points (see Section 4). We can also take care about the case where the density vanishes in some design points (Section 2.5).

Require: CDF $F(x)$; density $f(x)$ (cubic and quintic), $f'(x)$ (quintic);
order of approximation (1, 3, or 5), maximal error $\bar{\epsilon}_u$ (e.g. $\bar{\epsilon}_u \leq 10^{-10}$);
starting interval $[b_l, b_r]$ with $\mathrm{P}(X \notin (b_l, b_r)) < 0.1\,\bar{\epsilon}_u$.
Optional: All local extrema and points of discontinuity of the density or its derivatives.
Ensure: Random variate X with (approximate) CDF F.

/* Setup */
Initialize linked list for design points p_i. Start with $p_0 = b_l$ and $p_1 = b_r$.
Split the interval by inserting the x-coordinates of the local extrema and of points of discontinuity of the density or its derivatives.
Repeat
  Split intervals $[p_i, p_{i+1}]$ recursively into subintervals; inserting point $\tilde{p} = (p_i + p_{i+1})/2$ between p_i and p_{i+1}.
Until $F(p_{i+1}) - F(p_i) < 0.05$ for all points in the list.
Repeat
  Compute $F(p_i)$ and respective polynomial $H_i(u) = L_i(u)$, $C_i(u)$, or $Q_i(u)$ (depending on approx. order; use formulas given in Section 2.2).
  Compute $\bar{u} = (u_{i-1} + u_i)/2$ and $\hat{\epsilon}_u = |F(H_i(\bar{u})) - \bar{u}|$.
  Make monotonicity check (see Section 2.4).
  If $\hat{\epsilon}_u < \bar{\epsilon}_u$ and H_i is monotone **then**
    Proceed to next interval in linked list.
  Else
    Split interval $[p_i, p_{i+1}]$ at point $\tilde{p} = (p_i + p_{i+1})/2$.
Until end of linked list reached.
Cut off the tail-points such that $F(p_1) > 0.1\,\bar{\epsilon}_u$ and $F(p_{N-1}) < 1 - 0.1\,\bar{\epsilon}_u$ (N denotes index of the rightmost point).

/* Generator */
Generate $U \sim U(0,1)$.
$J \leftarrow \max\{J \colon F_J < U\}$. /* use *Indexed Search* */
Compute $X = H_J(U)$.
Return X.

Fig. 2. Inversion by Hermite interpolation

3. THEORETICAL ERROR BOUNDS

Ciarlet et al. [1967] (see also Birkhoff and Priver [1967]) have shown optimal bounds for the approximation error when using Hermite interpolation. For linear interpolation and a two times differentiable CDF we have the following error bound on the interval $[p_i, p_{i+1}]$

$$\epsilon_x \leq \max_{u \in [u_i, u_{i+1}]} |F^{-1}(u) - L(u)| \leq \frac{1}{32} \left| (u_{i+1} - u_i)^2 \max_{u \in [u_i, u_{i+1}]} (F^{-1})''(u) \right|. \tag{16}$$

Thus for equidistributed points $u_0, \ldots, u_N$ we have

$$\epsilon_x = O(1/N^2). \tag{17}$$

For a smooth CDF on the interval $[p_i, p_{i+1}]$ we have a similar error bound for cubic Hermite interpolation

$$\epsilon_x \leq \frac{1}{384} \left| (u_{i+1} - u_i)^4 \max_{u \in [u_i, u_{i+1}]} (F^{-1})''''(u) \right|. \tag{18}$$

Thus for a proper choice of design points p_i we have

$$\epsilon_x = O(1/N^4). \tag{19}$$

However, for a CDF which is not so smooth, the convergence is slower, i.e. it is of order 3 if the CDF is only three times differentiable and of order two, if it is only two times differentiable. This bound corresponds to our observation we have made in many experiments. We also remark that the error bound for the error $\epsilon_x(u)$ for a particular $u \in [u_i, u_{i+1}]$ is maximized for the center $\bar{u} = (u_i + u_{i+1})/2$ [Birkhoff and Priver 1967]. This is another argument for using the estimate $\hat{\epsilon}_u$ in (12). For quintic Hermite interpolation a similar result shows that the approximation error is of order 6.

Unluckily these error bounds are not applicable in practice, since usually the maximum of the second, forth, or sixth derivative of the inverse CDF is hardly available in a black-box algorithm.

It remains to demonstrate that the bounds for ϵ_x can be converted into bounds for ϵ_u as we designed our algorithm considering ϵ_u. For smooth distribution functions we have an obvious relationship between these two errors. If $x = C(u)$ for some particular u then $\epsilon_u(u) = |u - F(x)|$ and $\epsilon_x(u) = |x - F^{-1}(u)|$ denote the respective errors in u- and x-direction for the given u. Hence approximating the inverse CDF F^{-1} at $F(x)$ we find

$$\epsilon_x(u) = \epsilon_u(u)/f(x) + O(\epsilon_u(u)^2) \tag{20}$$

which is guaranteed to be valid for random variates with continuously differentiable densities.

Consequently, this is the mathematical expression of our observation in Section 2.1: The error $\epsilon_u(u)$ is smaller than $\epsilon_x(u)$ if $f(x)$ is small. We can use the above formula also as an argument to see why we can observe the same asymptotic behavior for ϵ_u as for ϵ_x as long as the density is bounded . So under regularity conditions we have

$$\epsilon_u = O(1/N^{d+1})$$

where d denotes the order of the Hermite interpolation.

4. EMPIRICAL RESULTS

In some sense this section is the most important part of the paper, because we need empirical experiments to see if our simple error estimate is really sufficient to achieve the desired accuracy. Here we can also try to answer the question we have raised in the introduction, which order of Hermite interpolation is best for practice. To do this we have coded Algorithm 2 in ANSI-C and included it in our UNU.RAN software library. We tested the algorithm with a variety of standard and non-standard distributions. The results for the normal, Cauchy, gamma, and beta distribution are presented here.

To start our assessment of the algorithms we will first compare the number N of design points that are necessary to reach the maximal u-error $\bar{\epsilon}_u$. Table I contains some of our results.

First we observe that these figures nicely follow the asymptotic error behavior which is known to be $O(1/N^{d+1})$, where d denotes the order of approximation as has the respective values 1, 3, and 5. Thus to reduce the error bound by a factor of 1/100 we need 10 times more intervals for linear interpolation. For cubic interpolation this factor reduces to $\sqrt{10} \approx 3.16$ and for quintic even to $\sqrt[3]{10} \approx 2.15$.

distribution	$\bar{\epsilon}_u = 10^{-6}$	$\bar{\epsilon}_u = 10^{-8}$	$\bar{\epsilon}_u = 10^{-10}$	$\bar{\epsilon}_u = 10^{-12}$
	Linear Interpolation			
Normal	1063	11533	117875	-
Cauchy	1849	17491	185335	-
Exponential	1012	10406	101959	-
Gamma(5)	1072	11225	109336	-
Gamma($\frac{1}{2}$)	1546	15432	154291	-
Beta(2,2)	823	8009	88179	-
Beta(0.3,3)	1884	18783	187786	-
	Cubic Interpolation			
Normal	109	335	941	3091
Cauchy	179	481	1491	4741
Exponential	71	207	661	2016
Gamma(5)	105	308	954	3060
Gamma($\frac{1}{2}$)	131	277	760	2306
Beta(2,2)	91	251	787	2477
Beta(0.3,3)	167	328	944	2740
	Quintic Interpolation			
Normal	73	127	245	513
Cauchy	107	175	345	743
Exponential	49	78	148	316
Gamma(5)	69	119	251	538
Gamma($\frac{1}{2}$)	108	137	218	409
Beta(2,2)	65	103	207	451
Beta(0.3,3)	146	169	255	484

Table I. Necessary number N of intervals for some distributions to obtain the required maximal u-error in our experiments. (No test for monotonicity of quintic interpolation)

The setup time for our algorithm depends heavily on the number of evaluations of the CDF which is a bit larger than two times the number of intervals. Thus the results of Table I can be used to estimate the setup time. Using the fact that we have to store $d + 2$ floating point numbers in every interval ($d + 1$ coefficients for the interpolating polynomial and $u_i = F(p_i)$) the table size is given by $(d + 2)\,N$.

Of course Table I gives us a first important help to decide about the optimal order that we should use for our approximation. If we consider e.g. a table up to a size of approximately 10^5 floating point numbers (or roughly 100kB of memory space) as acceptable, then we can see that linear interpolation can only reach maximal u-errors of about 10^{-7}. With cubic interpolation and that table size we can reach about 10^{-11} whereas quintic interpolation reaches maximal u-errors of 10^{-12} with considerably smaller tables. (We do not report results for even smaller maximal u-errors as they are too close to machine precision.) Summarizing we can say that – with the exception of linear interpolation – it is possible to reach "almost exact" numerical inversion for uniform random number sources with resolution 2^{-32}. The required table sizes remain moderate if we use cubic or quintic Hermite interpolation. For quintic interpolation we can estimate that even a maximal u-error of around 10^{-18} could be reached with approximately 5000 intervals. This could be useful for future computer architectures with higher machine precision or uniform random number sources with higher resolution.

To test the correctness of an inversion algorithm it is not necessary to make

distribution	$\bar{\epsilon}_u = 10^{-6}$		$\bar{\epsilon}_u = 10^{-8}$		$\bar{\epsilon}_u = 10^{-10}$		$\bar{\epsilon}_u = 10^{-12}$	
	max	MAE	max	MAE	max	MAE	max	MAE
	Linear Interpolation							
Normal	0.99	0.38	0.99	0.36	-	-	-	-
Cauchy	0.99	0.34	0.99	0.40	-	-	-	-
Exponential	0.99	0.38	0.99	0.33	-	-	-	-
Gamma(5)	3.45	0.45	0.99	0.33	-	-	-	-
Gamma($\frac{1}{2}$)	0.99	0.36	0.99	0.36	-	-	-	-
Beta(2,2)	0.99	0.34	0.99	0.38	-	-	-	-
Beta(0.3,3)	1.27	0.36	0.99	0.36	-	-	-	-
	Cubic Interpolation							
Normal	0.83	0.11	0.99	0.14	0.99	0.23	0.99	0.15
Cauchy	0.88	0.14	0.98	0.21	0.99	0.15	0.99	0.17
Exponential	0.89	0.13	0.93	0.17	0.99	0.13	0.99	0.22
Gamma(5)	0.98	0.13	0.99	0.17	0.99	0.21	0.99	0.14
Gamma($\frac{1}{2}$)	0.98	0.15	0.98	0.18	0.99	0.18	0.99	0.17
Beta(2,2)	0.98	0.17	0.99	0.17	0.98	0.17	0.99	0.17
Beta(0.3,3)	0.86	0.18	0.89	0.20	0.99	0.18	0.99	0.19
	Quintic Interpolation							
Normal	0.66	0.01	0.68	0.03	0.90	0.09	0.98	0.10
Cauchy	0.78	0.05	0.82	0.12	0.97	0.17	0.97	0.15
Exponential	0.87	0.02	0.84	0.03	0.99	0.09	0.94	0.13
Gamma(5)	0.96	0.18	0.98	0.04	0.94	0.08	0.98	0.08
Gamma($\frac{1}{2}$)	0.66	0.14	0.81	0.05	0.91	0.12	0.97	0.09
Beta(2,2)	0.75	0.02	0.78	0.08	0.96	0.13	0.99	0.10
Beta(0.3,3)	0.65	0.03	0.86	0.06	0.99	0.12	1.00	0.10

Table II. $|u - F(x)|/\bar{\epsilon}_u$, maximal (left column) and mean absolute (MAE, right column) value of the u-error observed for samples of size 10^6 in terms of $\bar{\epsilon}_u$

statistical tests. It is enough to compute the maximal and the average u-error $|u - F(x)|$ obtained in a large sample of uniformly distributed u values. As the u-error could increase in the tails we made extra checks for tail errors, but we did not observe any special problems in the tails. So Table II reports the results of mean absolute u-errors (MAE) and maximal u-error we observed in our simulation study.

The main message of our simulation results is that our algorithm works correctly. A value below 1 indicates that the observed error is smaller than the desired error bound $\bar{\epsilon}_u$. The only value that is clearly bigger than one comes from the fact that the mode of the distribution was not included as design point in this experiment with linear interpolation. We have made the same experiments with lots of different parameter settings and obtained very similar results. So we can conclude that our algorithm works with the tolerated maximal u-error for smooth distributions.

To get an idea what happens, if we have densities that are not smooth we tried densities with discontinuities and others with discontinuities of the first derivative. For both of them the main findings remained the same, but the necessary number of intervals tended to be a bit larger. (Nevertheless, using these points of discontinuity as design points eliminates this behavior.) For the error estimate we could certainly encounter problems for such distributions if we do not include the "bad" points as design points.

distribution	$\bar{\epsilon}_u = 10^{-6}$	$\bar{\epsilon}_u = 10^{-8}$	$\bar{\epsilon}_u = 10^{-10}$	$\bar{\epsilon}_u = 10^{-12}$
	Linear Interpolation			
Normal	0.92	1.24	3.15	-
Cauchy	0.93	1.50	4.53	-
Exponential	0.92	1.13	2.88	-
Gamma(5)	0.93	1.37	4.21	-
Gamma($\frac{1}{2}$)	0.98	1.76	6.86	-
Beta(2,2)	0.91	1.13	3.15	-
Beta(0.3,3)	0.96	1.83	6.32	-
	Cubic Interpolation			
Normal	0.88	0.91	0.94	0.99
Cauchy	0.87	0.92	0.95	1.02
Exponential	0.89	0.92	0.92	0.95
Gamma(5)	0.89	0.92	0.94	1.01
Gamma($\frac{1}{2}$)	0.88	0.92	0.95	1.03
Beta(2,2)	0.91	0.93	0.93	0.98
Beta(0.3,3)	0.88	0.92	0.95	1.04
	Quintic Interpolation			
Normal	0.97	0.97	0.97	0.98
Cauchy	0.96	0.96	0.96	0.99
Exponential	0.96	0.95	0.96	0.98
Gamma(5)	0.96	0.97	0.98	0.99
Gamma($\frac{1}{2}$)	0.97	0.96	0.97	0.95
Beta(2,2)	0.97	0.99	0.97	0.97
Beta(0.3,3)	0.97	0.96	0.99	0.98
	Newton's Method			
Normal	11.1	11.5	11.5	11.5
Cauchy	7.8	8.1	8.2	8.0
Exponential	10.4	10.4	10.4	10.4
Gamma(5)	36.7	36.1	36.0	35.1
Gamma($\frac{1}{2}$)	57.3	57.4	57.0	57.4
Beta(2,2)	24.9	24.2	25.8	25.7
Beta(0.3,3)	47.4	48.9	49.5	48.7

Table III. Timing results for generating a sample of 10^6 random points. Timings include setup and are given relative to generating exponentially distributed random variate using exact inversion via $-\log(1-U)$; Newton's method with a table of 100 starting points to improve performance and with termination condition: u-error smaller than $\bar{\epsilon}_u$. Timings environment: Pentium III, Linux, gcc 2.96.

As the third assessment we timed the different variants of our algorithm. The results in Table III show – as expected – that our method is really fast; with cubic and quintic interpolation it is about as fast as generating the exponential distribution with inversion and is not influenced by the distribution we generate. Only linear interpolation is astonishingly slow for acceptable precisions. This is due to the necessary large tables that lead to a very slow setup and to slower memory access caused by cache-effects. Thus this "brute force" method only should be realized for (rough!) approximate inversion.

Table III also includes numerical inversion using iterative methods with a small table for comparison (Newton's method). It gives the mean generation times for a sample of 10^6 random points including setup. To make the numbers less dependent from the computing environment (hardware, OS, compiler) we give all numbers relative to the generation times for sampling from the exponential distribution using exact inversion via $-\log(1-U)$. We do not give timings for the setup since it is obvious that a table method is slow when we only need a few random points.

What is the advantage of our method compared to the numeric inversion methods in the literature? The method suggested in [Bratley et al. 1983] is simply linear interpolation and we have seen above that for linear interpolation the table sizes explode if we want to reach small u-errors. Table III indicates that these large tables also have negative influence on the speed of the linear interpolation algorithm. Compared with the numerical inversion algorithm of Ahrens and Kohrt [1981] the main advantage of our new method is its simplicity and flexibility. Their paper does not even provide the details of the algorithm as it is too long and complicated.

5. CONCLUSIONS

We have demonstrated that numerical inversion with Hermite interpolation leads to simple and very fast random variate generation algorithms. With cubic or quintic interpolation we can easily reach error bounds in the order of 10^{-12} with moderate sized tables. As the inversion method is known to have advantages over other generation methods we are convinced that this version of numerical inversion is useful to generate from all distributions with computable CDF. The generation speed is for all distributions (much) faster than using iterative methods and about the same as generating an exponential random variate.

ACKNOWLEDGMENTS

The authors thank Günter Tirler and Yeliz Kayaoğlu for useful discussions, and Margit Feibel for her valuable helps in preparing this work. This work was supported by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung*, Proj. 12805-MAT.

REFERENCES

Ahrens, J. H. 1995. A one-table method for sampling from continuous and discrete distributions. *Computing 54,* 2, 127–146.

Ahrens, J. H. and Kohrt, K. D. 1981. Computer methods for efficient sampling from largely arbitrary statistical distributions. *Computing 26*, 19–31.

Birkhoff, G. and Priver, A. 1967. Hermite interpolation errors for derivatives. *J. Math. and Phys. 46*, 440–447.

Bratley, P., Fox, B. L., and Schrage, E. L. 1983. *A Guide to Simulation*. Springer-Verlag, New York.

Chen, H. C. and Asau, Y. 1974. On generating random variates from an empirical distribution. *AIIE Trans. 6*, 163–166.

Churchhouse, R. F., Ed. 1981. *Numerical Methods*. Handbook of Applicable Mathematics, vol. III. Wiley-Intersience, Chichester.

Ciarlet, P. G., Schultz, M. H., and Varga, R. S. 1967. Numerical methods of high-order accuracy for nonlinear boundary value problems. *Numer. Math. 9*, 394–430.

de Boor, C. and Swartz, B. 1977. Piecewise monotone interpolation. *J. Approx. Theory 21*, 411–416.

Devroye, L. 1986. *Non-Uniform Random Variate Generation*. Springer-Verlag, New-York.

Dougherty, R. L., Edelman, A., and Hyman, J. M. 1989. Nonnegativity-, monotonicity-, or convexity-preserving cubic and quintic Hermite interpolation. *Math. Comp. 52,* 186, 471–494.

FANG, K.-T. AND WANG, Y. 1994. *Number-theoretic Methods in Statistics.* Monographs on Statistics and Applied Probability, vol. 51. Chapman and Hall, London.

FRITSCH, F. N. AND CARLSON, R. E. 1980. Monotone piecewise cubic interpolation. *SIAM J. Numer. Anal. 17,* 2, 238–246.

GILKS, W. R. AND WILD, P. 1992. Adaptive rejection sampling for Gibbs sampling. *Applied Statistics 41,* 2, 337–348.

HÖRMANN, W. 1995. A rejection technique for sampling from T-concave distributions. *ACM Trans. Math. Software 21,* 2, 182–193.

HÖRMANN, W., LEYDOLD, J., AND DERFLINGER, G. 2003. *Automatic Non-Uniform Random Variate Generation.* Springer-Verlag, Berlin Heidelberg. accepted for publication.

HUYNH, H. T. 1993. Accurate monotone cubic interpolation. *SIAM J. Numer. Anal. 30,* 1, 57–100.

LAW, A. M. AND KELTON, W. D. 2000. *Simulation Modeling and Analysis*, 3 ed. McGraw-Hill.

LEYDOLD, J. 2000. Automatic sampling with the ratio-of-uniforms method. *ACM Trans. Math. Software 26,* 1, 78–98.

LEYDOLD, J. AND HÖRMANN, W. 2002. *UNU.RAN – A Library for Non-Uniform Universal Random Variate Generation.* Institut für Statistik, WU Wien, A-1090 Wien, Austria. available at `http://statistik.wu-wien.ac.at/unuran/`.

LEYDOLD, J., LEEB, H., AND HÖRMANN, W. 2000. Higher dimensional properties of non-uniform pseudo-random variates. In *Monte Carlo and Quasi-Monte Carlo Methods 1998*, H. Niederreiter and J. Spanier, Eds. Springer-Verlag, Berlin, Heidelberg, 341–355.

MANNI, C. 1996. C^1 comonotone Hermite interpolation via parametric splines. *J. Comp. Appl. Math. 69*, 143–157.

MOORE, L. R. 1983. Monotonic generation of positive random variables. In *Proc. 1983 Winter Simulation Conference*, S. Roberts, J. Banks, and B. Schmeiser, Eds. IEEE, 189–190.

NIEDERREITER, H. 1992. *Random number generation and quasi-Monte Carlo methods.* SIAM CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 63. SIAM, Philadelphia.