Dipl.–Ing. Heinz Zechner

Efficient Sampling from Continuous and Discrete Distributions

Dissertation

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften

eingereicht an der Technischen Universität Graz
Technisch–Naturwissenschaftliche Fakultät

Erstbegutachter: Univ. Doz. Dr. Ernst Stadlober
Zweitbegutachter: Univ. Prof. Dr. Ulrich Dieter

Institut für Statistik, Technische Universität Graz
Lessingstraße 27, A–8010 Graz

Graz, im Februar 1994

Ich versichere, diese Arbeit selbständig verfaßt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfmittel bedient zu haben.

Vorwort

Zu besonderem Dank bin ich Herrn Prof. Ulrich Dieter verpflichtet, der in mir nicht nur das Interesse für dieses Forschungsgebiet wecken konnte, sondern mich auch immer wieder großzügig unterstützte.

Danken möchte ich auch Herrn Dr. Herwig Friedl für seine Unterstützung bezüglich Software und in bezug auf diese für seine zahlreichen nützlichen Tips.

Besonders möchte ich mich aber bei Herrn Doz. Ernst Stadlober für seine anregenden Ideen und Vorschläge, sowie für die ausgezeichnete fachliche Betreuung meiner Dissertation bedanken.

Ebenso herzlich danke ich Herrn Prof. Jo Ahrens, der die Verständlichkeit dieser Arbeit durch seine kritischen Bemerkungen und technischen Vorschläge nicht unwesentlich erhöhte.

Zum Schluß möchte ich noch bemerken, daß der Großteil dieser Arbeit vom Fonds zur Förderung der wissenschaftlichen Forschung Nr. P8257–PHY *Computergerechte Erzeugung von Zufallszahlen* unterstützt wurde. Nach Ablauf dieser Förderung wurde mir noch ein Forschungsstipendium für die Dauer von vier Monaten bewilligt. Für dessen Zustandekommen möchte ich mich noch einmal recht herzlich bei meinem Betreuer Herrn Doz. Ernst Stadlober, bei Herrn Prof. Ulrich Dieter und dem Rektor der Technischen Universität bedanken.

Zusammenfassung

Die vorliegende Arbeit befaßt sich mit der Erzeugung von Zufallszahlen aus der Betaverteilung und unimodalen diskreten Verteilungen mittels der "Patchwork" Verwerfungstechnik. Das Hauptprinzip dieser Methode besteht darin, die Fläche unter der standardisierten Dichte– bzw. Histogrammfunktion $f(x)$ im Zentrum durch zwei Punktreflexionen so umzugruppieren, daß Verwerfung mit Hutfunktionen proportional zur Gleichverteilung auf einem sehr großen Bereich sehr effizient durchgeführt werden kann. Das eigentliche Problem dabei liegt in der Bestimmung gültiger Reflexionszentren — Zentren, für die das Abbild der Gipfelregion von $f(x)$ im Definitionsbereich von $f(x)$ liegt und disjunkt zur Fläche unter $f(x)$ ist — der Form $P_l = (x_2,\, f(x_2))$ und $P_r = (x_4,\, f(x_4))$.

Die verbleibenden "Tails" von $f(x)$ werden entweder durch Exponential– oder Potenzfunktionen majorisiert. Die resultierende dreiteilige Hutfunktion $h(x)$ liefert in Verbindung mit sofortigen rechteckigen Annahmebereichen eine Verwerfungsprozedur, die durchschnittlich mit sehr wenig gleichverteilten Zufallszahlen pro Stichprobe aus der Zielverteilung auskommt.

Die drei Teile der Arbeit gliedern sich wie folgt:

In **Part I** und **Part II** werden zwei neue Algorithmen zur Erzeugung von Stichproben aus Beta$(p,\, q)$–Verteilungen mit Parametern $p > 1$ *und* $q > 1$, bzw. $p > 1$ *oder* $q > 1$ vorgestellt. In beiden Algorithmen muß bei der Initialisierung der Reflexionszentren zwischen drei Parameterbereichen unterschieden werden. Für jeden dieser Bereiche existiert eine nicht unbedingt optimale, aber geschlossene Darstellung der Abszissen x_2, x_4 der Zentren P_l und P_r.

Allgemeine Bedingungen für die Zentren der Punktreflexionen im diskreten Falle werden in **Part III** hergeleitet. Anschließend daran wird ein allgemeiner "Patchwork" Algorithmus für die Erzeugung von Stichproben aus unimodalen diskreten Verteilungen entwickelt und formuliert. Danach sind spezielle Lösungen für die Poisson–, Binomial– und hypergeometrische Verteilung angeführt, die sich zu den aus der Literatur bekannten Verfahren als konkurrenzfähig erweisen. Im hypergeometrischen Fall reduziert sich die Erzeugungszeit sogar auf ungefähr die Hälfte der bisher schnellsten Verwerfungsverfahren.

Die neuen Algorithmen und eine große Anzahl von existierenden Verfahren wurden in C implementiert und bezüglich Initialisierungs– und Generierungszeiten, sowie anderer Effizienzkriterien miteinander verglichen. Die Experimente zeigen, daß die neu entwickelten Verfahren zwar relativ viel Zeit für die Initialisierungsschritte aufwenden, die Erzeugungszeiten pro Stichprobe jedoch grundsätzlich unter jenen der herkömmlichen Verfahren liegen. Die hier vorgestellten Algorithmen sind daher vor allem für die Erzeugung einer größeren Anzahl von Stichproben aus einer festen Verteilung zu empfehlen.

Contents

List of Tables

Part I

Part II

Part III

Part I

Generating Beta Variates with Both Parameters Greater than One Via Patchwork Rejection on the Beta Prime Distribution

Abstract

Generating Beta Variates Via Patchwork Rejection on the Beta Prime Distribution. A new algorithm for sampling from beta(p, q) distributions with parameters $p > 1$, $q > 1$ is developed. It is based on a method by Minh [9] which improves acceptance–rejection sampling in the main part of the distributions. Additionally, transformed uniform deviates can very often be accepted immediately, so that much fewer than two uniforms are needed for one beta variate, on the average. The remaining tests for acceptance are enhanced by 'squeezes'. Experiments covering a wide range of pairs (p, q) showed improvements in speed over competing algorithms in most cases.

AMS Subject Classification: 65C10, 65C05.

Keywords and Phrases: Random number generation, beta distribution, patchwork rejection.

1 Introduction

The beta prime distribution is an important tool in stochastic modelling and simulation; it can have a number of different shapes. When parameters p and q are both less than 1, the density is U–shaped, if exactly one parameter is less than 1, the density is J–shaped, and when both parameters are greater than 1, the density is unimodal. This variety makes the development of sampling methods covering all possible shapes difficult. A survey of existing beta generators is given in the book of Devroye [6]. The algorithms discussed in this part of the book deal with unimodal densities only. For the other cases $p < 1$ and/or $q < 1$ we refer to the papers of Atkinson [2], Atkinson & Whittaker [3], [4], Cheng [5], Jöhnk [8] and Sakasegawa [10].

Competitive methods for $p > 1$ and $q > 1$ are based on acceptance–rejection. The first method of this type is due to Ahrens & Dieter [1], [7]. They constructed a normal envelope which is inefficient in case of skew and nearly uniform beta distributions. Uniformly fast algorithms (i. e. algorithms with bounded computation times for all $p > 1,\ q > 1$) have been established later on in [5], [10], and [11]. Cheng [5] applied a log–logistic envelope to the beta distribution of the second kind. Samples from this distribution can be transformed easily to the desired beta prime variates. The resulting algorithm BB is rather simple, has low set–up costs and reasonable speed. Schmeiser & Babu [11] and Sakasegawa [10] utilized piecewise linear envelopes in the center, and exponentials in the tails. Their sampling routines B4PE and B11 are involved, but they are fast at the expense of a more time–consuming set–up.

Our new sampling method improves acceptance–rejection in the center of the distribution by the patchwork technique of Minh [9]. A certain modification of the beta density (reflection at one point $(x_2,\ f(x_2))$ to the left of the mode and at one point $(x_4,\ f(x_4))$ to the right of the mode) leads to a more efficient sampling process which uses rectangular majorizing and minorizing functions (see Figure 1). The expected number of uniforms needed per variate is bounded by $2\,(2-e^{-1}) = 3.2642\ldots$ $(p \searrow 2,\ q \to 1$ or $p \to 1,\ q \searrow 2)$, it decreases to $6/\sqrt{2\pi e} = 1.4518\ldots$ as $p,\ q\ \to \infty$. This is one of the great advantages of our method. The experimental results in Section 7 indicate that in terms of execution time the resulting algorithm B_1PRS compares well with the algorithms mentioned above.

The next two sections contain a general description of the patchwork technique and its application to the beta prime distribution. The basic patchwork–rejection sampling method is described in Section 4. Formal statements of the two new algorithms B_1PRB and B_1PRS are given in Section 5. Algorithms B4PE and B_1PRS are compared with respect to some performance measures in Section 6, and computational experience is reported in the final section. In the appendix the proof of the crucial Lemma 1 of Section 3 is established.

2 The Patchwork Technique

Let $f(x)$ be proportional to a probability density function which is bounded and unimodal with the mode at m. The main idea of the method is to transform the left part

$$T_l \;=\; \{(x,\,y) \,|\, x_2 < x < m,\; f(x_2) < y \le f(x)\}$$

of the top region under $f(x)$ into the congruent area $T_l{}'$ (in Figure 1) by using the point $P_l = (x_2,\ f(x_2))$ as the center of a reflection σ_l. Similarly, the right part

$$T_r \;=\; \{(x,\,y) \,|\, m < x < x_4,\; f(x_4) < y \le f(x)\}$$

of the top region under $f(x)$ is reflected by σ_r at $P_r = (x_4,\ f(x_4))$ into $T_r{}'$ (in Figure 1). The points P_l and P_r, that is x_2 and x_4, must be determined carefully, such that the transformed regions T_l' and T_r' contain no points below $f(x)$. Formally the reflections σ_l and σ_r read

$$\sigma_l: \ (x,\, y) \mapsto (v,\, w) = (2x_2 - x,\, 2f(x_2) - y), \qquad x_2 < x < m \tag{1}$$

$$\sigma_r: \ (x,\, y) \mapsto (v,\, w) = (2x_4 - x,\, 2f(x_4) - y), \qquad m < x < x_4 \,. \tag{2}$$

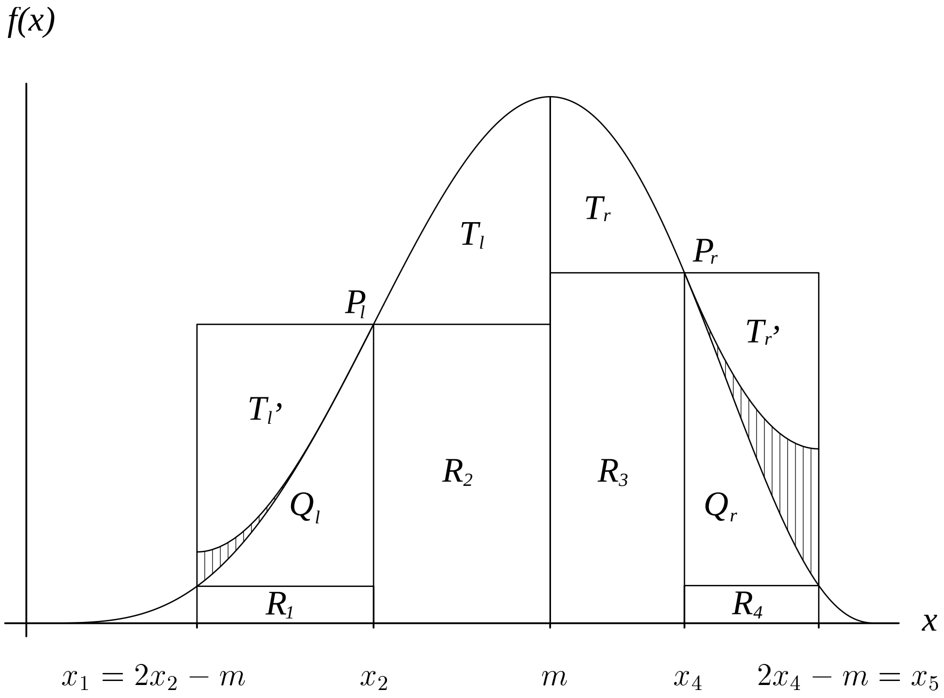

Figure 1: Patchwork (Example $\text{beta}_1(5,\, 3.5)$–distribution).

There is no overlapping of the transformed region to the left of x_2 (to the right of x_4) with the original area below $f(x)$ if, and only if condition (3) (condition (4)) holds:

$$f(x_2 + z) - f(x_2) \,\le\, f(x_2) - f(x_2 - z)\,, \qquad 0 \le z \le m - x_2 \tag{3}$$

$$f(x_4 - z) - f(x_4) \,\le\, f(x_4) - f(x_4 + z)\,, \qquad 0 \le z \le x_4 - m\,. \tag{4}$$

Differentiating (3) and (4) with respect to z and using that the equal sign holds at $z = 0$ yields the sufficient conditions

$$\begin{aligned} f'(x_2 + z) \,&\le\, f'(x_2 - z)\,, \qquad 0 \le z \le m - x_2 \\ -f'(x_4 - z) \,&\le\, -f'(x_4 + z)\,, \qquad 0 \le z \le x_4 - m \end{aligned}$$

in which first derivatives instead of differences are compared. Since $f'(x)$ is positive for $x < m$ and negative for $x > m$, both inequalities contain positive terms only. Taking logarithms, differentiating and using again that the equal sign holds at $z = 0$ leads to the simple conditions

$$\frac{d}{dx} \ln |f'(x)| \Big|_{x_2 + z} + \frac{d}{dx} \ln |f'(x)| \Big|_{x_2 - z} \leq 0\,, \qquad 0 \leq z \leq m - x_2 \tag{5}$$

$$\frac{d}{dx} \ln |f'(x)| \Big|_{x_4 - z} + \frac{d}{dx} \ln |f'(x)| \Big|_{x_4 + z} \geq 0\,, \qquad 0 \leq z \leq x_4 - m \tag{6}$$

which will be the basis for the determination of valid 'reflection points' $P_l = (x_2,\, f(x_2))$ and $P_r = (x_4,\, f(x_4))$ in the next section.

3 Application to the Beta Prime Distribution

Let

$$f_b(x) = x^{p-1}(1-x)^{q-1}/B(p,\, q)\,, \qquad 0 < x < 1, \quad p > 0,\; q > 0, \tag{7}$$

be the density of the beta prime distribution, where the normalizing constant

$$B(p,\, q) = \int_0^1 x^{p-1}(1-x)^{q-1}\, dx \;=\; \frac{\Gamma(p)\,\Gamma(q)}{\Gamma(p+q)}$$

is the complete beta function. We confine the shape parameters p, q to $p > 1$ and $q > 1$. Then $f_b(x)$ is bounded and it has one mode at $m = (p-1)/(p+q-2)$, $0 < m < 1$.
In order to simplify the following calculations we set $a = p - 1$, $b = q - 1$, and define

$$f_B(x) = x^a(1-x)^b\,, \qquad 0 < x < 1, \quad a > 0,\; b > 0\,. \tag{8}$$

Differentiating $\ln |f_B{}'(x)| = (a-1)\ln(x) + (b-1)\ln(1-x) + \ln |a - (a+b)x|$ with respect to x leads to

$$\frac{d}{dx} \ln |f_B{}'(x)| \;=\; \frac{(a+b)(a+b-1)x^2 - 2a(a+b-1)x + a(a-1)}{x(1-x)(a-(a+b)x)}\,, \qquad x \neq \frac{a}{a+b}$$

which may be written as

$$\frac{d}{dx} \ln |f_B{}'(x)| \;=\; \frac{(s-1)x^2 - 2m(s-1)x + m(ms-1)}{x(1-x)(m-x)}\,, \qquad x \neq m = \frac{a}{a+b}\,, \tag{9}$$

where $s = a + b = a/m = b/(1-m)$.

The reflection point P_l. Inserting (9) into (5) yields, after some straightforward calculations, the following sufficient condition for x_2,

$$A(x_2)z^4 - B(x_2)z^2 + C(x_2) \leq 0\,, \qquad 0 \leq z \leq m - x_2\,, \tag{10}$$

where

$$A(x) = (s-1)(x - (1-m))\,, \tag{11}$$

$$B(x) = 2(s-1)x^3 - 2(m+1)(s-1)x^2 + \\ + m((3-m)s-2m)x - m(1-m)(ms+1)\,, \tag{12}$$

and

$$C(x) = x(1-x)(m-x)((s-1)x^2 - 2m(s-1)x + m(ms-1))\,. \tag{13}$$

After an analytical and numerical examination of the inequality (10) we arrived at the following result which will be proved in the appendix.

Lemma 1 *Let x_{wl} be the left point of inflection of $f_B(x) = x^a(1-x)^b$ in (8):*

$$x_{wl} = \frac{1}{a+b}\left(a - \sqrt{\frac{ab}{a+b-1}}\right) = m - \sqrt{\frac{m(1-m)}{s-1}}\,, \quad a = ms \geq 1, \tag{14}$$

where $s = a+b$ and $m = a/(a+b)$. Furthermore let

$$z_2 = x_{wl} - \frac{x_{wl}(1-x_{wl})}{s(m-x_{wl})} \tag{15}$$

be the point at which the tangent of $f_B(x)$ to x_{wl} crosses the x–axis. Then candidates for abscissas $x_2 < m$ of reflection points P_l which fulfil the inequality (10) are

(i) $x_2 \geq \frac{m}{2}$, *if* $x_{wl} \leq \frac{m}{2}$ *and* $a \leq 2$ *or, equivalently,*

if $a \leq \min\left(\frac{1-b+\sqrt{b^2+14b+1}}{2},\, 2\right)$.

(ii) $x_2 \geq x_{wl}$, *if* $x_{wl} \geq \frac{m}{2}$ *and* $(s-6)x_{wl} - ms + 3 \leq 0$ *or, equivalently,*

if $a \geq \max\left(\frac{1-b+\sqrt{b^2+14b+1}}{2},\, \frac{-(b+3)^2+(b-3)\sqrt{b^2+54b+9}}{2\,(b-9)}\right)$.

(iii) $x_2 \geq \frac{z_2+m}{2}$, *if* $a > 2$, $x_{wl} < \frac{m}{2}$ *or* $(s-6)x_{wl} - ms + 3 > 0$ *or, equivalently,*

if $2 < a < \frac{-(b+3)^2+(b-3)\sqrt{b^2+54b+9}}{2\,(b-9)}$.

The two sketches in Figure 2 display the three regions of Lemma 1. For $a \geq 9$ the condition in (ii) is always satisfied. Therefore $x_2 = x_{wl}$ (and $x_4 = x_{wr}$) is feasible if $a \geq 9$ (and $b \geq 9$). The remark about x_4 (in parentheses) follows from the statements under 'The reflection point P_r'. Finally, note that case (i) complemented by the left end of the region belonging to case (ii) covers the stripe $0 < a \leq 2,\ b > 0$.

Remark. Let ${x_2}^{(min)}$ be the smallest x_2–value which is feasible in respect to the inequality (10). It is plausible that ${x_2}^{(min)}$ is a good choice for the abscissa of P_l. In fact, it was established numerically that the sum $A(x_2) = A_T(x_2) + A_R(x_2)$ of the tail area $A_T(x_2) = \int_0^{2x_2-m} f(x)\,dx$ and the (shaded) rejection area $A_R(x_2) = \int_{2x_2-m}^{x_2}(2f(x_2) -$

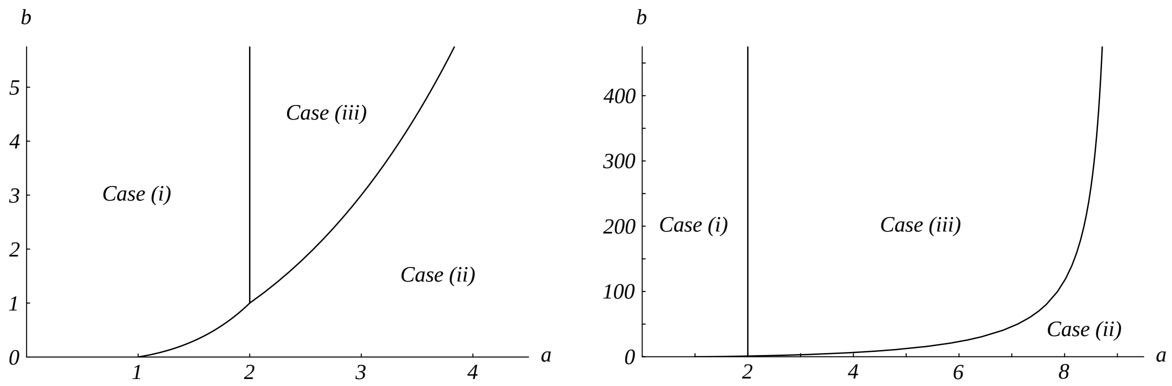

Figure 2: The three cases of Lemma 1 in the (a, b) plane.

$f(2x_2 - x) - f(x))\,dx$ increases monotonically between ${x_2}^{(min)}$ and m except for small a below 2.5.

The proof of Lemma 1 in the appendix contains the additional information that ${x_2}^{(min)} = \frac{m}{2}$ in the case (i), and ${x_2}^{(min)} = x_{wl}$ in the case (ii). However, inside the region (iii) the safe choice $x_2 = (z_2 + m)/2$ suggested in Lemma 1 is only an upper bound on ${x_2}^{(min)}$. We have no tighter bounds which would be simple enough for the use in the sampling routine.

The reflection point P_r. Interchanging the shape parameters p and q in (7) causes a reflection of the beta density $f_b(x)$ at $x = \frac{1}{2}$, i. e. the random variate X from $\text{beta}_1(p, q)$ is changed to $Y = 1 - X$ from $\text{beta}_1(q, p)$. Consequently the bounds in Lemma 1 for $x_2 < m$ can be easily modified to equivalent expressions for abscissas $x_4 > m$ of feasible reflection points P_r: interchange a and b (s remains unchanged), replace m by $1 - m$, x_{wl} by $1 - x_{wr}$, z_2 by $1 - z_4$, and x_2 by $1 - x_4$. The transformed bounds yield analogous suggestions for safe, but not always optimal, choices of x_4 which are spelt out under *Initialization* of algorithm B_1PRS in Section 5.

4 Sampling Method

From now on the rescaled function

$$f(x) = \frac{f_b(x)}{f_b(m)} = \left(\frac{x}{m}\right)^{p-1} \left(\frac{1-x}{1-m}\right)^{q-1}, \qquad 0 < x < 1, \quad p > 1,\ q > 1, \tag{16}$$

is employed instead of the original density $f_b(x)$. $f(x)$ needs no evaluation of the normalizing constant $B(p, q)$, and $f(x) \leq f(m) = 1$ holds for all x.

From Section 2 we know that the mode m and the abscissas x_2 and x_4 are important ingredients of the sampling routine as well as $x_1 = 2x_2 - m$ and $x_5 = 2x_4 - m$. x_1 and x_5 which enclose the top regions T_l, T_r and their images T_l', T_r' split up the area under $f(x)$ into three parts: the *left tail*, the *center*, and the *right tail*. This suggests a sampling routine based on acceptance–rejection with a three part envelope $h(x)$. The graph of $h(x)$ in the center is the upper boundary of T_l', R_2, R_3 and T_r' in Figure 1; the remodeled center of $f(x)$ is majorized by a two step staircase function. The left tail of $f(x)$ over $(0,\, x_1]$ and the right tail of $f(x)$ over $[x_5,\, 1)$ are both majorized by exponential functions. The details are given in subsections 4.1 and 4.2.

4.1 Sampling from the Center

Let $f_i = f(x_i)$, $i = 1, 2, 4, 5$. The part $h^{(c)}(x)$ of the hat function $h(x)$ which covers the center region — after the reflections at P_l and P_r (Figure 1) — is defined as

$$h^{(c)}(x) = \begin{cases} f_2 = f(x_2) & x_1 < x < m \\ f_4 = f(x_4) & m \le x < x_5\,. \end{cases}$$

We put stress on the fact that $f(x)$ will be evaluated not only at x_2 and x_4, but also at x_1 and x_5, since $f_1 = f(x_1)$ and $f_5 = f(x_5)$ allow for the definition of tight and efficient tail envelopes. Therefore f_2, f_4, f_1 and f_5 are in hand without additional costs. We make use of these quantities as follows.

Immediate acceptance. Under the graph of $f(x)$ in Figure 1 we find four rectangles (with heights f_i) which are the 'immediate acceptance regions':

$$R_1 = \{(x,\, y) \,|\, x_1 < x < x_2,\; 0 \le y \le f_1\}, \qquad R_2 = \{(x,\, y) \,|\, x_2 \le x < m,\; 0 \le y \le f_2\},$$
$$R_3 = \{(x,\, y) \,|\, m \le x \le x_4,\; 0 \le y \le f_4\}, \qquad R_4 = \{(x,\, y) \,|\, x_4 < x < x_5,\; 0 \le y \le f_5\}.$$

For sampling from $\bigcup R_i$ only one uniform deviate U is needed. This uniform is used at first for selecting one of the four rectangles with the correct probability, and is then transformed to a sample X from the chosen rectangle which can be accepted immediately, without carrying out any acceptance test, as a beta variate (see algorithm B$_1$PRS, Step 2 and Step 4).

Acceptance–Rejection. Let $(X,\, Y)$ be a point under $h^{(c)}(x)$ outside $\bigcup R_i$. Then $X \in (x_1,\, x_2)$ or $X \in (x_4,\, x_5)$. If $Y \le f(X)$ then the point $(X,\, Y)$ is in either Q_l or Q_r in Figure 1, and X can be accepted as a beta variate. Otherwise $(X,\, Y)$ may belong to T_l' or T_r' in which case the test applies to the corresponding point $(V,\, W)$ in T_l or T_r. Since ${\sigma_l}^{-1} = \sigma_l$ and ${\sigma_r}^{-1} = \sigma_r$, the back–transformation can be copied from (1) and (2):

$$(X,\, Y) \;\mapsto\; (V,\, W) = (2x_2 - X,\; 2f_2 - Y), \qquad x_1 < X < x_2 \tag{17}$$
$$(X,\, Y) \;\mapsto\; (V,\, W) = (2x_4 - X,\; 2f_4 - Y), \qquad x_4 < X < x_5\,. \tag{18}$$

Now V is accepted if $W \le f(V)$. Otherwise the point $(X,\, Y)$ is in the shaded area of Figure 1, rejection is indicated, and a new sample from $h(x)$ is required.

4.2 Sampling from the Tails

The hat function $h(x)$ for the tails is to touch $f(x)$ at x_1 and x_5. This means

$$h(x_1) = f_1\,,\; h'(x_1) = f'(x_1) \quad \text{and} \quad h(x_5) = f_5\,,\; h'(x_5) = f'(x_5)\,. \tag{19}$$

With this one takes full advantage of the quantities f_1 and f_5 : firstly, by using them to determine two additional immediate acceptance regions, and secondly, by using them to define tight tail envelopes.

Both tails can be covered by exponential functions which meet the requirement (19) :

$$\begin{aligned} h^{(l)}(x) &= f_1\, e^{\,(x\,-\,x_1)/\lambda_l}\,, \quad -\infty < x \le x_1\,, \quad \lambda_l = x_1(1-x_1)/(s\,(m-x_1)) \\ h^{(r)}(x) &= f_5\, e^{\,(x_5\,-\,x)/\lambda_r}\,, \quad x_5 \le x < \infty\,, \quad \lambda_r = x_5(1-x_5)/(s\,(x_5-m))\,. \end{aligned}$$

Their areas are

$$H^{(l)} = \int_{-\infty}^{x_1} h^{(l)}(x)\,dx = f_1\,\lambda_l\,, \qquad H^{(r)} = \int_{x_5}^{\infty} h^{(r)}(x)\,dx = f_5\,\lambda_r\,.$$

Samples X from $h^{(l)}(x)$ and $h^{(r)}(x)$ are obtained by inversion of the corresponding distribution functions as

$$X = x_1 + \lambda_l \log(U) \quad \text{and} \quad X = x_5 - \lambda_r \log(1-U)\,, \qquad U \sim U(0,\,1)\,.$$

Samples from the tails of $f(x)$ are generated by

Acceptance–Rejection. Let $(X,\,Y)$ be uniformly distributed under $h^{(l)}(x)$ or $h^{(r)}(x)$. If $Y \le f(X)$ then the point $(X,\,Y)$ is below $f(x)$, and X can be accepted as a beta variate. Otherwise X is rejected, and a new sample from $h(x)$ is required.

4.3 Squeeze Functions

Since the acceptance tests $W \le f(V)$ and $Y \le f(X)$ use the complicated function $f(x)$, fast pre–tests based on simple lower bounds for $f(x)$ (called squeezes) will be added to our basic sampling routine. We employ linear squeezes which are obtained as follows.

Let $x_1 \le x_{wl},\; x_5 \ge x_{wr}$. Furthermore, let $z = t(\xi) = \xi - \xi(1-\xi)/(s\,(m-\xi))$ be the point at which the tangent of $f(x)$ to ξ crosses the x–axis, and define

$$z_2 = \begin{cases} 0, & \text{if } a \le 1 \\ t(x_{wl}), & \text{if } a > 1\,, \end{cases} \qquad z_1 = t(x_1) = \begin{cases} 0, & \text{if } a \le 1 \\ x_1 - \lambda_l, & \text{if } a > 1\,, \end{cases}$$

$$z_4 = \begin{cases} 1, & \text{if } b \le 1 \\ t(x_{wr}), & \text{if } b > 1\,, \end{cases} \qquad z_5 = t(x_5) = \begin{cases} 1, & \text{if } b \le 1 \\ x_5 + \lambda_r, & \text{if } b > 1\,. \end{cases}$$

Then the line segments with end points $(z_i,\,0)$, $(x_i,\,f_i)$, $i = 1,\,2,\,4,\,5,$ are below $f(x)$. This is obviously valid for $i = 1$ and $i = 5$. For $i = 2$ and $i = 4$ the statement follows from $x_{wl} \le x_2$, $x_4 \le x_{wr}$ (one part of the fundamental condition x_2 and x_4 must satisfy; compare the inequality (24) in the appendix) and from the concavity of $f(x)$ in $[x_{wl}, x_{wr}]$. A consequence of the latter two remarks is that the segments with end points (x_2, f_2), $(m, 1)$ and $(m, 1)$, (x_4, f_4) are also below $f(x)$, and the complete piecewise linear squeeze function $L(x) \le f(x)$ reads

$$L(x) = \begin{cases} f_1(x - z_1)/(x_1 - z_1), & 0 < x \le x_1 \\ f_2(x - z_2)/(x_2 - z_2), & x_1 < x < x_2 \\ f_2 + (1 - f_2)(x - x_2)/(m - x_2), & x_2 \le x < m \\ f_4 + (1 - f_4)(x_4 - x)/(x_4 - m), & m \le x \le x_4 \\ f_4(z_4 - x)/(z_4 - x_4), & x_4 < x < x_5 \\ f_5(z_5 - x)/(z_5 - x_5), & x_5 \le x < 1\,. \end{cases}$$

Acceptance–rejection of samples $X \in (x_1, x_2)$ and $X \in (x_4, x_5)$ is modified by $L(x)$ as follows. First of all, the fast pre–test $Y \le L(X)$ is executed. Whenever it fails, (X, Y) is back–transformed to (V, W). If $W \ge 1$ then V is rejected, since $f(x) \le 1$. Otherwise the fast pre–test $W \le L(V)$ and, if necessary, the slow acceptance test $W \le f(V)$ is carried out (Step 3 and Step 5 of the algorithm B₁PRS). If both tests fail, i. e. V is rejected, the sample (X, Y) is subjected to the test $Y \le f(X)$ which finally decides on acceptance or rejection of X.

Acceptance–rejection of samples $X \in (0, x_1]$ and $X \in [x_5, 1)$ is also complemented by the fast pre–test $Y \le L(X)$ (Step 6 and Step 7 of the algorithm B$_1$PRS).

5 Algorithms B$_1$PRS and B$_1$PRB

The algorithm B$_1$PRS (**B**eta prime **P**atchwork **R**ejection method with **S**queezes) results directly from the theory in Section 3 and Section 4. Nevertheless, for a better understanding we still add some additional remarks to its formal description.

The shorter version B$_1$PRB (**B**asic method) without lower squeezes is obtained from B$_1$PRS as follows. Remove all lines that are marked with a '--' sign, and change the lines which carry combinations of '∘' and '•':

∘∘ `Set` $D_l = m/2$, $x_2 = D_l$, $x_1 = 0$;

∘• `Set` $x_1 = x_2 * (1 - (1 - x_2)/(s * (m - x_2)))$;

•∘ `Set` $D_r = (1 - m)/2$, $x_4 = 1 - D_r$, $x_5 = 1$;

•• `Set` $x_5 = x_4 * (1 + (1 - x_4)/(s * (x_4 - m)))$;

Algorithm B_1PRS $(p > 1 \text{ and } q > 1)$

$f(x, a, b, m) = \exp(a * \log(x/m) + b * \log((1-x)/(1-m)))$

Initialization:

```
     Set a = p − 1, b = q − 1, s = a + b, m = a/s;
     If (a > 1 or b > 1) then Set D = sqrt(m * (1 − m) / (s − 1));
     If (a ≤ 1) then
○ ○     Set D_l = m/2, x_2 = D_l, x_1 = z_2 = 0;                  Lemma 1, (i)
        Set f_1 = λ_l = 0;
     else
        Set D_l = D, x_2 = m − D_l, x_1 = x_2 − D_l;              Lemma 1, (ii)
--      Set z_2 = x_2 * (1 − (1 − x_2)/(s * (m − x_2)));
        If (x_1 ≤ 0 or (s − 6) * x_2 − a + 3 > 0) then
○ ●        Set x_1 = z_2;
           Set x_2 = (x_1 + m)/2, D_l = m − x_2;                  Lemma 1, (i), (iii)
        Endif
        Set f_1 = f(x_1, a, b, m);
        Set λ_l = x_1 * (1 − x_1) / (s * (m − x_1));
     Endif                                                        z_1 = x_1 − λ_l
     Set f_2 = f(x_2, a, b, m);
     If (b ≤ 1) then
● ○     Set D_r = (1 − m)/2, x_4 = 1 − D_r, x_5 = z_4 = 1;        Lemma 1, (i)
        Set f_5 = λ_r = 0;
     else
        Set D_r = D, x_4 = m + D_r, x_5 = x_4 + D_r;              Lemma 1, (ii)
--      Set z_4 = x_4 * (1 + (1 − x_4)/(s * (x_4 − m)));
        If (x_5 ≥ 1 or (s − 6) * x_4 − a + 3 < 0) then
● ●        Set x_5 = z_4;
           Set x_4 = (m + x_5)/2, D_r = x_4 − m;                  Lemma 1, (i), (iii)
        Endif
        Set f_5 = f(x_5, a, b, m);
        Set λ_r = x_5 * (1 − x_5) / (s * (x_5 − m));
     Endif                                                        z_5 = x_5 + λ_r
     Set f_4 = f(x_4, a, b, m);
     Set p_1 = f_2 * (m − x_1), p_2 = f_4 * (x_5 − m) + p_1;
     Set p_3 = f_1 * λ_l + p_2,  p_4 = f_5 * λ_r + p_3;
```

Generation:

1. Case Distinction:

```
   Generate U ~ U(0, 1) and Set U = U * p_4;
   If (U ≤ p_1) then Goto 2. [Left Center Region];
                else
   If (U ≤ p_2) then Goto 4. [Right Center Region];
                else
```

`If` $(U \le p_3)$ `then Goto 6.` [Left Tail];
`else Goto 7.` [Right Tail];
`Endif`

2. Left Center Region — Immediate acceptance of $X \in (x_1, m)$:

`Set` $Y = U/D_l - f_2$;	$Y \sim U(-f_2, +f_2)$
`If` $(Y \le 0)$ `then Deliver` $X = m - U/f_2$;	$X \in (x_2, m)$
`If` $(Y \le f_1)$ `then Deliver` $X = x_2 - Y/f_1 * D_l$;	$X \in (x_1, x_2)$

3. Left Center Region — Squeeze acceptance of $X \in (x_1, x_2)$ or Acceptance–Rejection of its counterpart $V \in (x_2, m)$:

`Generate` $U \sim U(0, 1)$ `and Set` $W = U * D_l$;	$W \sim U(0, D_l)$
`Set` $X = x_2 - W$, $V = x_2 + W$;	
`--` `If` $(Y \le f_2 * (X - z_2) / (x_2 - z_2))$ `then Deliver` X;	$Y \le L(X)$
`Set` $W = f_2 + f_2 - Y$;	$W \sim U(f_2, 2f_2 - f_1)$
`If` $(W < 1)$ `then`	
`--` `If` $(W \le f_2 + (1 - f_2) * U)$ `then Deliver` V;	$W \le L(V)$
`If` $(W \le f(V, a, b, m))$ `then Deliver` V;	$W \le f(V)$
`Endif`	
`Goto 8.` [Acceptance Test];	Rejection of V

4. Right Center Region — Immediate acceptance of $X \in (m, x_5)$:

`Set` $Y = (U - p_1)/D_r - f_4$;	$Y \sim U(-f_4, +f_4)$
`If` $(Y \le 0)$ `then Deliver` $X = m + (U - p_1)/f_4$;	$X \in (m, x_4)$
`If` $(Y \le f_5)$ `then Deliver` $X = x_4 + Y/f_5 * D_r$;	$X \in (x_4, x_5)$

5. Right Center Region — Squeeze acceptance of $X \in (x_4, x_5)$ or Acceptance–Rejection of its counterpart $V \in (m, x_4)$:

`Generate` $U \sim U(0, 1)$ `and Set` $W = U * D_r$;	$W \sim U(0, D_r)$
`Set` $X = x_4 + W$, $V = x_4 - W$;	
`--` `If` $(Y \le f_4 * (z_4 - X) / (z_4 - x_4))$ `then Deliver` X;	$Y \le L(X)$
`Set` $W = f_4 + f_4 - Y$;	$W \sim U(f_4, 2f_4 - f_5)$
`If` $(W < 1)$ `then`	
`--` `If` $(W \le f_4 + (1 - f_4) * U)$ `then Deliver` V;	$W \le L(V)$
`If` $(W \le f(V, a, b, m))$ `then Deliver` V;	$W \le f(V)$
`Endif`	
`Goto 8.` [Acceptance Test];	Rejection of V

6. Left Tail — Squeeze acceptance of $X \in (-\infty, x_1]$, computed by inversion:

`Set` $U = (U - p_2)/(p_3 - p_2)$;	$U \sim U(0, 1)$
`Set` $V = \log(U)$, $X = x_1 + \lambda_l * V$;	
`If` $(X \le 0)$ `then Goto 1.` [Continue];	Fast rejection of X
`Generate` $Y \sim U(0, 1)$ `and Set` $Y = Y * U$;	$Y \sim U(0, h(X)/f_1)$
`--` `If` $(Y \le 1 + V)$ `then Deliver` X;	$Y \le L(X)/f_1$
`Set` $Y = Y * f_1$ `and Goto 8.` [Acceptance Test];	$Y \sim U(0, h(X))$

7. Right Tail — Squeeze acceptance of $X \in [x_5, +\infty)$, computed by inversion:

`Set` $U = (U - p_3)/(p_4 - p_3)$;	$U \sim U(0, 1)$
`Set` $V = \log(U)$, $X = x_5 - \lambda_r * V$;	
`If` $(X \geq 1)$ `then Goto 1.` [Continue];	Fast rejection of X
`Generate` $Y \sim U(0, 1)$ `and Set` $Y = Y * U$;	$Y \sim U(0, h(X)/f_5)$
`-- If` $(Y \leq 1 + V)$ `then Deliver` X;	$Y \leq L(X)/f_5$
`Set` $Y = Y * f_5$ `and Goto 8.` [Acceptance Test];	$Y \sim U(0, h(X))$

8. Acceptance Test — Acceptance of X, whenever $Y \leq f(X)$:

`If` $(Y \leq f(X, a, b, m))$ `then Deliver` X;	
`else Goto 1.` [Continue];	Rejection of X

Remark. In *Initialization* the number of case distinctions is reduced by setting $x_2 = \frac{m}{2}$ for $a \leq 1$ only. For the remaining part of case (i) in Lemma 1 $x_2 = (z_2 + m)/2$ from case (iii) is substituted. This is legitimate, since $(z_2 + m)/2 \geq \frac{m}{2}$ (x_2 remains a valid abscissa of P_l). Equally we substitute $x_4 = (m + z_4)/2$ in case (i) whenever $b > 1$.

Fast rejection tests based on the condition $f(x) \leq 1$ occur in Step 3 and Step 5. Step 6 and Step 7 dealing with the tails of $f(x)$ also contain rejection tests. Their job is to discard instantly all samples X from the exponential envelopes outside the domain $(0, 1)$ of the beta prime distribution to avoid the otherwise disastrous evaluation of $f(x)$ in Step 8.

6 Characteristics of Algorithms B4PE and B$_1$PRS

The fastest existing procedures for generating beta$_1(p, q)$ variates with $p > 1$ and $q > 1$ are algorithms B4PE (see Schmeiser & Babu [11]) and B11, that part of algorithm BSA which deals with the case $p > 1$, $q > 1$ (see Sakasegawa [10]). Sakasegawa's method is based purely on acceptance–rejection, therefore it requires at least two uniforms per beta variate. Algorithm B4PE does contain immediate acceptance with one uniform; for some parameter combinations (p, q) it needs slightly fewer than two uniforms per variate (see Table 2) — we shall see that in this respect our algorithm B$_1$PRS is more efficient.

Algorithm B$_1$PRS improves B4PE substantially in sampling from the center by the patchwork technique which remodels the center region of $f(x)$, but the tail envelopes and squeeze functions are nearly the same. For this reason the two algorithms are compared with respect to some performance measures (see Table 1 to Table 4). The characteristics of B$_1$PRS are as follows.

First of all, we need the average number α of samples X from $h(x)$, until X is accepted:

$$\alpha = f_b(m)\,\Big(f_1\,\lambda_l + f_2(m - x_1) + f_4(x_5 - m) + f_5\,\lambda_r\Big). \tag{20}$$

The probability $P(I)$ of accepting X immediately, given that X is accepted, is the rescaled area of the four rectangles R_i:

$$P(I) = f_b(m)\,\Big((f_1 + f_2)(m - x_2) + (f_4 + f_5)(x_4 - m)\Big). \tag{21}$$

With these two quantities the expected number of uniforms N for one beta variate becomes

$$N = 2\alpha - P(I) - f_b(m)\Big(f_1\,\lambda_l\,e^{-x_1/\lambda_l} + f_5\,\lambda_r\,e^{-(1-x_5)/\lambda_r}\Big). \tag{22}$$

The third term represents the average number of samples X from $h(x)$ outside the interval $(0,\,1)$. They are quickly rejected without carrying out an acceptance–rejection test, therefore they use up only one uniform number (for the generation of X). This also applies to the samples X from the 'immediate acceptance regions'. All other deviates, however, require exactly two uniforms, one for their generation and one to decide on their acceptance or rejection.

The fourth and last characteristic is the probability $P(S)$ of accepting X by means of the lower squeeze $L(x)$, given that X is accepted. It reads

$$\begin{aligned} P(S) = \tfrac{1}{2} f_b(m)\,\Big(&f_1\,\lambda_l + (f_2 - f_1)(1 - f_1/f_2)(x_2 - z_2) + (1 - f_2)(m - x_2) \\ &+ (1 - f_4)(x_4 - m) + (f_4 - f_5)(1 - f_5/f_4)(z_4 - x_4) + f_5\,\lambda_r\Big). \end{aligned} \tag{23}$$

Table 1 to Table 4 display the characteristics of algorithms B4PE and B$_1$PRS for different parameter combinations $(p,\,q)$. Because of symmetry values are stated for $p \geq q$ only.

The figures of α, N and $P(I)$ in Table 1, Table 2 and Table 3 demonstrate a better performance of our proposal B$_1$PRS. Its higher immediate acceptance probabilities $P(I)$ (in Table 3) lead naturally to lower squeeze probabilities $P(S)$ (in Table 4), since fewer candidates X are left for acceptance.

Limiting Values

For $p \to \infty$ and $q \to \infty$ the characteristics in (20), (21) and (22) tend to

$$\begin{aligned} \lim_{p,q\to\infty} \alpha &= \frac{4}{\sqrt{2\pi}}\left(e^{-1/2} + \tfrac{1}{4}e^{-2}\right) = 1.0218\ldots, \\ \lim_{p,q\to\infty} P(I) &= \frac{2}{\sqrt{2\pi}}\left(e^{-1/2} + e^{-2}\right) = 0.5919\ldots, \\ \lim_{p,q\to\infty} N &= \frac{6}{\sqrt{2\pi}}\;e^{-1/2} = 1.4518\ldots. \end{aligned}$$

The above limiting values are the result of tedious calculations: We approximated the normalizing constant $B(p,\,q) = \Gamma(p)\,\Gamma(q)\,/\,\Gamma(p+q)$ by Stirling's formula and collected the resulting logarithms in $\log f_b(x)$, $x = x_1$, $x = x_2$, $x = x_4$ and $x = x_5$, such that we were able to replace them with the Taylor series of $\log(1 \pm z)$, $|z| < 1$. So we obtained simple approximations to $f_b(x_1)$, $f_b(x_2)$, $f_b(x_4)$ and $f_b(x_5)$ which we finally inserted into the formulas (20), (21) and (22). (Note that $x_2 = x_{wl} = m - D$ and $x_4 = x_{wr} = m + D$, since $p \to \infty$, $q \to \infty$, which further implies $x_1 = m - 2D$ and $x_5 = m + 2D$.)

Table 1: Average number α of samples X from $h(x)$, until X is accepted as a beta variate.

Algorithm B4PE

$p \backslash q$	1.1	1.5	2.5	5.0	1000
1.1	1.060				
1.5	1.294	1.273			
2.5	1.127	1.238	1.281		
5.0	1.104	1.154	1.210	1.180	
1000	1.070	1.116	1.162	1.152	1.148

Algorithm B_1PRS

$p \backslash q$	1.1	1.5	2.5	5.0	1000
1.1	1.030				
1.5	1.075	1.103			
2.5	1.053	1.071	1.100		
5.0	1.124	1.069	1.064	1.010	
1000	1.044	1.056	1.070	1.056	1.022

Table 2: Average number N of uniforms needed to generate one beta variate.

Algorithm B4PE

$p \backslash q$	1.1	1.5	2.5	5.0	1000
1.1	2.120				
1.5	2.588	2.546			
2.5	2.022	2.212	2.186		
5.0	2.034	2.069	2.047	1.939	
1000	1.970	2.019	2.009	1.926	1.892

Algorithm B_1PRS

$p \backslash q$	1.1	1.5	2.5	5.0	1000
1.1	1.545				
1.5	1.613	1.654			
2.5	1.453	1.578	1.622		
5.0	1.724	1.526	1.537	1.486	
1000	1.694	1.576	1.551	1.506	1.452

Table 3: Probability $P(I)$ of accepting X immediately, given that X is accepted.

Algorithm B4PE

$p \backslash q$	1.1	1.5	2.5	5.0	1000
1.1	0.000				
1.5	0.000	0.000			
2.5	0.495	0.347	0.439		
5.0	0.492	0.438	0.510	0.572	
1000	0.460	0.441	0.505	0.565	0.592

Algorithm B_1PRS

$p \backslash q$	1.1	1.5	2.5	5.0	1000
1.1	0.515				
1.5	0.538	0.551			
2.5	0.628	0.563	0.579		
5.0	0.511	0.608	0.591	0.533	
1000	0.393	0.535	0.589	0.606	0.592

Table 4: Probability $P(S)$ of accepting X by means of the lower squeeze $L(x)$, given that X is accepted.

Algorithm B4PE

$p \backslash q$	1.1	1.5	2.5	5.0	1000
1.1	0.530				
1.5	0.647	0.637			
2.5	0.413	0.478	0.446		
5.0	0.385	0.414	0.393	0.352	
1000	0.361	0.387	0.375	0.343	0.330

Algorithm B_1PRS

$p \backslash q$	1.1	1.5	2.5	5.0	1000
1.1	0.265				
1.5	0.323	0.318			
2.5	0.315	0.360	0.373		
5.0	0.339	0.303	0.349	0.382	
1000	0.343	0.302	0.309	0.312	0.330

For fixed p and $q \to 1$, on the other hand, we have

$$\lim_{q\to 1} \alpha = \begin{cases} p/2^{p-1} & 1 < p \leq 2 \\ p/(p-1) & 2 < p \end{cases},$$

$$\lim_{q\to 1} P(I) = \begin{cases} p/2^{p} & 1 < p \leq 2 \\ 0 & 2 < p \end{cases},$$

$$\lim_{q\to 1} N = \begin{cases} 3p/2^{p} & 1 < p \leq 2 \\ p\,(2 - e^{-(p-1)})/(p-1) & 2 < p \end{cases}.$$

The limiting values of α, $P(I)$ and N are discontinuous at $p = 2$, since the special beta density $\lim_{q\to 1} f_b(x) = p\,x^{\,p-1}$ changes its curvature in $(0, 1)$ at $p = 2$ from concave to convex.

7 Computational Results

In this section the algorithms B_1PRB and B_1PRS are compared with the uniformly fast algorithms BB (Cheng [5]), B11 (Sakasegawa [10]) and B4PE (Schmeiser & Babu [11]) in terms of execution times, initialization times and lengths of compiled codes. (There are also a lot of algorithms whose execution times are not bounded over the whole parameter region $p > 1$ and $q > 1$. We quote algorithm BN by Ahrens & Dieter [1], [7], Jöhnk's method [8] and algorithms B2P and B4P by Schmeiser & Shalaby [12] as examples. They behave badly in at least one of the following cases: both parameters are quite small, one parameter is very large and the other one is small or both parameters are very large.)

All algorithms were written as C routines, compiled and linked in Turbo C 2.0 with the instruction set of the 80286 and 80287 processors. Uniform random numbers were generated by the multiplicative congruential C–generator `krand` with modulus $m = 2^{54}$ and multiplicator $a = 2\,783\,377\,641\,436\,325$ (see Stadlober & Kremer [13]). This generator has a long period of 2^{52} and produces random numbers in double precision, the normal precision of C. On a PC/AT 486–33MHz it takes about 22 μsec to generate one uniform which is approximately the time for calculating $y = e^x$. The elementary operations addition, subtraction and multiplication need about 0.5 μsec. The square root requires about 3.5 μsec, the logarithm about 16 μsec, and the general power function about 43 μsec. Since the latter operation needs more time than the exponential power function together with a logarithm, y^c is replaced in all algorithms by $y^c = e^{c\log(y)}$.

Table 5 displays the execution times of BB, B11, B4PE and of our two proposals B_1PRB and B_1PRS for carefully chosen parameters p, q (best times are in bold face, second best in italics). The observed times are based on samples of one million variates for each parameter combination. Our algorithm B_1PRS seems to be faster than its competitors unless one parameter is very small (near 1) and the other one is large (greater than 4). In this case the times of B4PE are slightly better. The execution times of B4PE are very high for $p \leq 2$, $q \leq 2$. This is caused by the absence of immediate acceptance regions,

Table 5: Execution times [μsec/variate] of algorithms BB (Cheng), B11 (Sakasegawa), B4PE (Schmeiser & Babu), B_1PRB and B_1PRS (Zechner).

$p \backslash q$	1.1	1.2	1.5	2.0	2.5	3.0	5.0	10.0	1000
1.1	124 86 98 *83* **69**								
1.2	125 86 101 *85* **69**	125 86 102 *85* **70**					1st row: 2nd row: 3rd row: 4th row: 5th row:	BB B11 B4PE B_1PRB B_1PRS	
1.5	130 *84* 111 88 **70**	128 *85* 109 88 **71**	127 *86* 111 90 **72**						
2.0	137 *72* 129 91 **62**	133 *76* 123 91 **66**	129 *81* 119 92 **71**	129 *80* 123 94 **71**					
2.5	142 *71* 74 81 **58**	138 *73* 77 86 **59**	132 *76* 87 89 **65**	130 *77* 99 92 **68**	131 *76* 83 91 **67**				
3.0	146 *75* *75* 87 **67**	141 *74* 76 82 **61**	134 *76* 83 88 **62**	131 *76* 93 90 **66**	131 *75* 79 90 **66**	132 *74* 76 89 **65**			
5.0	154 *81* **79** 102 82	149 *79* *79* 92 **72**	140 *78* 82 85 **65**	134 *77* 89 87 **64**	132 *75* 78 87 **63**	132 *74* *74* 88 **62**	133 74 *71* 90 **60**		
10.0	161 *84* **81** 105 85	156 82 *80* 96 **77**	146 *80* 83 88 **70**	139 *79* 89 87 **67**	135 *77* 79 86 **65**	134 *76* *76* 86 **63**	132 74 *71* 85 **61**	134 74 *70* 86 **59**	
1000	169 85 **81** 103 *84*	164 83 *81* 96 **78**	154 *82* 83 90 **72**	145 *80* 89 88 **69**	141 *79* 82 87 **68**	138 *78* 79 86 **66**	133 *76* 77 84 **64**	129 *75* *75* 85 **61**	130 75 *68* 83 **59**

and the rather poor fit of the uniform hat function. In this parameter region B_1PRS is followed by B11 which dominates B4PE if one parameter is not too small and not too large. Our simpler method (without squeezes) B_1PRB is about 20% slower than B_1PRS, whereas the generation times of BB exceed the times of the fastest method by factors close to 2.

Initialization times (set–up costs) are important, if the values of p or q change frequently. All algorithms except BB contain some very time–consuming constants, such as function values of the standardized beta density which are prepared therefore in an extra step before *Generation*. The time needed to compute these constants depends primarily on the existence of the left and the right point of inflection, and secondarily on the size of the shape parameters.

The figures in the center of Table 6 indicate very low set–up costs for BB. The higher costs for the other four algorithms vary with the size of the shape parameters. Values for the three different cases (with 0, 1, and 2 points of inflection) are listed in Table 6; the symbol '$\sim$' signifies average values. Memory requirements are also displayed to show the complexity of the algorithms; it is clear that BB is much simpler than the faster methods B_1PRB, B4PE, B_1PRS and B11.

Table 6: Initialization times [μsec] and memory requirements [bytes] of BB (Cheng), B11 (Sakasegawa), B4PE (Schmeiser & Babu), B_1PRB and B_1PRS (Zechner).

Algorithm	Initialization times			Memory requirements of compiled code
	$p,\ q \le 2$	$q \le 2 < p$	$p,\ q > 2$	
BB	20	20	20	1711
B11	151	~ 270	~ 387	7930
B4PE	52	~ 185	~ 324	6811
B_1PRB	135	~ 227	~ 316	6377
B_1PRS	139	~ 232	~ 324	7181

The average generation time (which consists of the initialization and the execution time) depends on the number V of variates, produced successively from a fixed $\text{beta}_1(p,\, q)$ distribution. From Table 5 and Table 6 we conclude that our algorithm requires the least total time if V > 5, unless one parameter is very small and the other one is large. If V is almost always less than 5, we recommend to use algorithm BB, specially because of its modest length and its constant and very small set–up time over the whole parameter region. However, note that our algorithm already performs well for $V = 3$: it is the fastest method if $p \le 2$, $q \le 2$, and otherwise, under the above–mentioned restriction, at least the second fastest method, beaten at most either by B4PE or by BB.

To sum up, it can be said that for applications in which speed, a low number of uniforms and even performance are important, our proposal B_1PRS can be recommended. If simplicity and small set–up time are the dominant considerations, BB could be the first choice.

8 Appendix

The reflection (1), illustrated in Figure 1 of Section 2, demands that x_2 has to be at least $\frac{m}{2}$. This guarantees that the transformed top region T_l' to the left of x_2 is inside the range $(0,\,1)$ of the beta density. x_2 also has to be greater than or equal to the left point of inflection x_{wl}. This follows from (10), since for $z=0$ one has $C(x_2)\le 0$ which is satisfied if, and only if $x_{wl}\le x_2\le m$. Hence

$$x_2 \ge \max\left(\frac{m}{2}\,,\, x_{wl}\right) \tag{24}$$

is a general precondition for abscissas x_2 of valid reflection points P_l.

Proof of Lemma 1

Case (i) :

Let $x_{wl}\le \frac{m}{2}$. Then (24) demands that $x_2\ge\frac{m}{2}$. Below we show that $x_2=\frac{m}{2}$ is feasible for $a\le 2$.

For the proof x_2 is replaced with $\frac{m}{2}$ in (10). After some calculation we arrive at

$$4(s-1)(3m-2)z^2 - m(2-m)(ms+3m-4) \ge 0\,, \qquad 0\le z\le \frac{m}{2}\,. \tag{25}$$

Now three different cases are considered.

(a) If $s\le 1$, then no point of inflection exists, and $f_B(x)$ is concave in $(0,\,1)$. Hence, $x_2=\frac{m}{2}$ is obviously valid.

(b) If $s>1$ and $m<\frac{2}{3}$, then the left hand side of (25) is strictly decreasing. Therefore it attains its minimum at the right end point $z=\frac{m}{2}$, and condition (25) reduces to $(s-1)(3m-2)m^2 - m(2-m)(ms+3m-4) = 4m(1-m)(2-ms)\ge 0$. This shows that $x_2=\frac{m}{2}$ is valid, provided that $ms\le 2$, i. e. $a\le 2$.

(c) If $s>1$ and $m\ge\frac{2}{3}$, then the left hand side of (25) is increasing. Therefore it attains its minimum at the left end point $z=0$, and (25) reduces to the single condition $-m(2-m)(ms+3m-4)\ge 0$. It follows that $x_2=\frac{m}{2}$ is valid, provided that $ms+3m-4\le 0$. But this condition is equivalent to $x_{wl}\le\frac{m}{2}$ which is assumed.

Combining the results of (a), (b), and (c) we can state that $x_2=\frac{m}{2}$ is valid, whenever $x_{wl}\le\frac{m}{2}$ and $a\le 2$ or, using (14), whenever

$$a \le \min\left(\frac{1-b+\sqrt{b^2+14b+1}}{2}\,,\, 2\right), \qquad b>0\,. \tag{26}$$

If $x_2=\frac{m}{2}$ is valid, it is relatively easy to see that all x_2 between $\frac{m}{2}$ and m are also feasible. The formal proof is omitted.

Remark. For $0<a\le 1$ the feasibility of $x_2=\frac{m}{2}$ still follows from the concavity of $f_B(x)$ in $(0,\,m)$.

Case (ii) :

Let $x_{wl} \geq \frac{m}{2}$. Then (24) demands that $x_2 \geq x_{wl}$. Again we only show that $x_2 = x_{wl}$ is valid for the part of the parameter region which is described by condition $(s-6)x_{wl} - ms + 3 \leq 0$. The argumentation is as follows.

If x_{wl} in (14) is substituted for x_2, condition (10) reduces to

$$A(x_{wl})z^2 - B(x_{wl}) \leq 0\,, \qquad 0 \leq z \leq m - x_{wl}\,, \tag{27}$$

since x_{wl} substituted into (13) yields $C(x_{wl}) = 0$. We consider two cases.

(a) $A(x_{wl}) < 0$ or, equivalently from (11) and (14), $x_{wl} < 1 - m$:

The left hand side of (27) is strictly decreasing. Therefore it attains its maximum at the left end point $z = 0$, and (27) reduces to the single condition $-B(x_{wl}) \leq 0$. Using (14) in the form $(s-1){x_{wl}}^2 - 2m(s-1)x_{wl} + m(ms-1) = 0$ (to minimize the degree of the polynomial $B(x_{wl})$ and thereby to simplify it as much as possible) finally yields

$$-B(x_{wl}) \;=\; m(1-m)\left((s-6)x_{wl} - ms + 3\right) \;\leq\; 0. \tag{28}$$

(b) $A(x_{wl}) \geq 0$ or, equivalently, $x_{wl} \geq 1 - m$:

Now the left hand side of (27) is increasing. Therefore it attains its maximum at the right end point $z = m - x_{wl}$, and (27) reduces to the single condition

$$A(x_{wl})(m - x_{wl})^2 - B(x_{wl}) \;=\; m(1-m)\left((s-5)x_{wl} - ms + m + 2\right) \;\leq\; 0 \tag{29}$$

which holds for all $x_{wl} \geq 1 - m$ (precondition $A(x_{wl}) \geq 0$) and $x_{wl} \geq \frac{m}{2}$ (precondition of case (ii)). This can be easily checked by discussing the two cases $s \geq 5$ and $s < 5$, and by observing that there is no x_{wl} which fulfils $A(x_{wl}) \geq 0$ in the case $m \leq \frac{1}{2}$: From $m > \frac{1}{2}$ it instantly follows that

$$(s-5)x_{wl} - ms + m + 2 = -\,(s-5)(m - x_{wl}) - 2(2m-1) \leq 0\,,$$

provided that $s \geq 5$. In the case $s < 5$ the above inequality can be replaced by the equivalent condition

$$(s-5)^2(m - x_{wl})^2 - 4(2m-1)^2 = -\,\tfrac{1}{s-1}(ms + 3m - 4)(ms + 3m - s + 1) \leq 0\,.$$

Because of $x_{wl} \geq \frac{m}{2}$ which is equivalent to $ms + 3m - 4 \geq 0$ we have $s > 1$ and $ms + 3m - s + 1 \geq 5 - s > 0$, and the expression on the left side is in fact negative.

$A(x_{wl}) \geq 0$ implies $-B(x_{wl}) \leq A(x_{wl})(m - x_{wl})^2 - B(x_{wl}) \leq 0$, so (29) can be replaced by (28) which is now the only condition in both cases (a) and (b). Hence, x_{wl} is valid, whenever

$$x_{wl} \;\geq\; \frac{m}{2} \quad \text{and} \quad (s-6)x_{wl} - ms + 3 \;\leq\; 0\,. \tag{30}$$

According to (14), the first condition in (30) is equivalent to $a^2 + (b-1)a - 4b \geq 0$ (C1) which is fulfilled, whenever

$$a \geq a_1 = \frac{1 - b + \sqrt{b^2 + 14b + 1}}{2}, \qquad b > 0. \tag{31}$$

The second condition in (30) is equivalent to $(a+b-6)\sqrt{ab} + 3(a-b)\sqrt{a+b-1} \geq 0$ (C2) which is satisfied, whenever

$$a \geq a_2 = \frac{-(b+3)^2 + (b-3)\sqrt{b^2 + 54b + 9}}{2\,(b-9)}, \qquad b > 0. \tag{32}$$

This is verified by an elementary derivation, considering the four cases $a \geq b,\ b > 3$; $a \geq b,\ b \leq 3$; $a < b,\ a + b \geq 6$ and $a < b,\ a + b < 6$.

(a) If $a \geq b$ and $b > 3$, the inequality (C2) is obviously satisfied.

(b) If $a \geq b$ and $b \leq 3$, the inequality (C2) is fulfilled, provided that $a \geq 6 - b$ or, otherwise, whenever $a < 6 - b$ and the difference

$$\begin{aligned} d(a,\, b) &= ab(a+b-6)^2 - 9\,(a+b-1)(a-b)^2 \\ &= (a+b)\,((b-9)a^2 + (b+3)^2 a - 9b(b-1)) \end{aligned}$$

of the two terms in (C2) raised to the power of two is negative, i. e. $d(a,\, b) \leq 0$. Using that the roots of $d(a,\, b) = 0$ with respect to a are equal to

$$a_2 = \frac{(b+3)^2 + (3-b)\sqrt{b^2 + 54b + 9}}{2\,(9-b)}, \qquad a_3 = \frac{(b+3)^2 - (3-b)\sqrt{b^2 + 54b + 9}}{2\,(9-b)},$$

and that $a_2 \geq a_3$ (and $b \geq a_3$, implying $a \geq a_3$) for all $0 < b \leq 3$ shows that (C2) is true, whenever

$$a \geq \max(6-b,\, b) \quad \text{or} \quad 6 - b > a \geq \max(a_2,\, b), \qquad b \leq 3.$$

Since $6 - b \geq b$ and $a_2 \geq b$, this reduces to the simple restraint $a \geq a_2$ for all $b \leq 3$.

(c) If $a < b$ and $a + b \geq 6$ (which implies $b > 3$), the inequality (C2) is satisfied, whenever $d(a,\, b) \geq 0$ or, equivalently, whenever

$$\begin{aligned} &\min\,(a_2,\, a_3) = a_2 \leq\ a\ \leq a_3 = \max(a_2,\, a_3) && \text{for} \quad 3 < b < 9 \\ &\max(a_2,\, a_3) = a_2 \leq\ a && \text{for} \quad 9 < b. \end{aligned}$$

Taking into account that $a < b$, and using that $a_2 < b < a_3$ for all $3 < b < 9$ and $a_2 < b$ for all $b > 9$ finally yields the condition $a_2 \leq a < b$ for all $b > 3$.

Combining (a) and (c), and taking the result of (b) establishes our statement (32). To sum up, it can be said that $x_2 = x_{wl}$ is feasible if, and only if $a \geq \max(a_1,\, a_2)$, $b > 0$.

Case (iii) :

Let x_{wl} be the left point of inflection (14), z_2 the point at which the tangent of $f_B(x)$ to x_{wl} crosses the x–axis (15), and ${x_2}^{(hlf)} = (z_2 + m)/2$ our suggested bound on ${x_2}^{(min)}$.

It is easy to verify that $a_1 \geq a_2$ for $0 \leq b \leq 1$. Therefore, if $a \leq 2$, $a \geq a_1$, and $0 \leq b \leq 1$, we have $a \geq a_1 \geq a_2$ and this is covered by case (ii). $a \leq 2$, $a \geq a_1$, and $b > 1$ is not possible, since $a_1 > 2$ for $b > 1$. Finally, $a \leq 2$ and $a \leq a_1$ is case (i), and we have eliminated all cases $a \leq 2$. The remaining combinations of parameters a, b are bounded by $2 < a < \max(a_1, a_2)$, $b > 0$. Furthermore, if $b \leq 1$ then $a_1 \leq 2$ and $a_2 \leq 2$, and if $b > 1$ then $a_1 < a_2 \leq 9$. Therefore the remaining region is described more tightly by

$$2 \, < \, a \, < \, a_2 = \frac{-(b+3)^2 + (b-3)\sqrt{b^2+54b+9}}{2\,(b-9)} \leq 9\,, \qquad b > 1\,. \tag{33}$$

In order to prove the feasibility of our proposal $x_2 = {x_2}^{(hlf)}$, we compare it with ${x_2}^{(min)}$, the smallest x_2–value which is feasible in respect to the inequality (10). First of all, however, we establish analytically (and not numerically) that ${x_2}^{(hlf)}$ meets the general preconditon (24) in the region (33).
Obviously ${x_2}^{(hlf)} \geq \frac{m}{2}$, so we only have to show that ${x_2}^{(hlf)} \geq x_{wl}$. But this condition is equivalent to

$$\begin{aligned} {x_2}^{(hlf)} - x_{wl} &= \frac{m - x_{wl}}{2} - \frac{x_{wl}(1-x_{wl})}{2s(m-x_{wl})} \\ &= \frac{(s-1)(m-x_{wl})^2 + (m-x_{wl})^2 - x_{wl}(1-x_{wl})}{2s(m-x_{wl})} = \frac{1-2x_{wl}}{2s} \geq 0\,. \end{aligned}$$

Hence, that ${x_2}^{(hlf)}$ satisfies (24), (33) must be a subset of the region defined by $x_{wl} \leq \frac{1}{2}$. Using now that (32) is equivalent to the second condition in (30), the above requirement reads

$$a > 2 \, \wedge \, (s-6)x_{wl} - ms + 3 > 0 \quad \Rightarrow \quad x_{wl} \leq \tfrac{1}{2}\,.$$

For the proof of this statement we consider the two cases $s \geq 6$ and $s < 6$.

(a) Let $s \geq 6$. Then from

$$(s-6)x_{wl} - ms + 3 \, = \, 3(1-2m) - (s-6)(m-x_{wl}) \, > \, 0$$

it follows that $m < \frac{1}{2}$, and consequently $x_{wl} < \frac{1}{2}$.

(b) Let $s < 6$, and assume that $x_{wl} \geq \frac{1}{2}$. Then

$$(s-6)x_{wl} - ms + 3 \, \leq \, \tfrac{s}{2} - 3 - ms + 3 \, \leq \, (\tfrac{1}{2} - m)s \, \leq \, 0\,,$$

since $m > x_{wl}$. The assumption $x_{wl} \geq \frac{1}{2}$ presents a contradiction to the premise $(s-6)x_{wl} - ms + 3 > 0$, and therefore $x_{wl} < \frac{1}{2}$.

From the above it is clear that ${x_2}^{(hlf)}$ is a reasonable proposal.

Next we demonstrate that there is always a feasible x_2 which satisfies the inequality $A(x_2) \leq 0$. The proof of this statement uses that $A(x_{wl}) < 0$, i. e. $x_{wl} < 1 - m$, in the region (33) (a close look at the case distinction in (ii) reveals this restriction).

(a) Let $m \leq \frac{1}{2}$. Then $x_2 \leq \frac{1}{2}$ and $x_2 \leq 1 - m$.

(b) Let $m > \frac{1}{2}$ and $\frac{m}{2} < 1 - m$. Then $x_2 = 1 - m$ meets the general precondition (24), since $x_{wl} < 1 - m$. Inserting this particular x_2 into (10) yields

$$-\,m(1-m)(2m-1)(s-3)z^2 + C(1-m) \,\leq\, 0\,, \qquad 0 \leq z \leq 2m-1$$

which holds for all parameter combinations satisfying the prerequisite and (33). This proves the feasibility of $x_2 = 1 - m$.

(c) Let $m > \frac{1}{2}$ and $1 - m \leq \frac{m}{2}$. Evidently this case occurs only when $m \geq \frac{2}{3}$. Applying this restraint to the requirement $x_{wl} < \frac{m}{2}$ (following from $x_{wl} < 1 - m$ and the prerequisite $1 - m \leq \frac{m}{2}$) which is equivalent to $ms + 3m - 4 < 0$ shows that $a = ms < 2$ must hold. Hence, this case is not part of the region (33).

Using the polynomials in (11), (12), and (13), we define

$$P(z) = A(x_2)z^4 - B(x_2)z^2 + C(x_2) \quad \text{and} \quad D_P(x_2) = B^2(x_2) - 4A(x_2)C(x_2)$$

and assume that $A(x_2) \leq 0$ (a feasible x_2 with this property exists because of $A(x_{wl}) < 0$ in the region (33)). Furthermore we have $C(x_2) \leq 0$ from (24). We consider now three cases, dependent on the signs of $B(x_2)$ and $D_P(x_2)$.

(a) Let $B(x_2) \geq 0$. Then $P(z) \leq 0$ for all z, and x_2 is valid.

(b) Let $B(x_2) < 0$ and $D_P(x_2) > 0$. Then $P(z) = 0$ has exactly two positive roots. Hence $P(z) \leq 0$ for $0 \leq z \leq m - x_2$, whenever $P(m - x_2) \leq 0$ and $P'(m - x_2) > 0$. Further analysis shows that the minimum x_2 obtained from these two conditions (for $a > 2$) is either $\geq m$ or it leads to $D_P(x_2) < 0$ which contradicts the assumption $D_P(x_2) > 0$. Therefore this case yields no useful condition we could use for the determination of ${x_2}^{(min)}$.

(c) Let $B(x_2) < 0$ and $D_P(x_2) \leq 0$. Then $P(z)$ does not change its sign. Since $C(x_2) < 0$, we have $P(z) < 0$ for all z, i. e. x_2 is valid.

From the above we conclude that ${x_2}^{(min)}$ fulfils at least one of the two conditions

$$B(x_2) \geq 0 \qquad \text{or} \quad D_P(x_2) \leq 0\,, \qquad x_L < x_2 < m\,, \tag{34}$$

where $x_L = \max\left(\frac{m}{2}, x_{wl}\right)$.

$B(x_2) = 0$ implies $D_P(x_2) \leq 0$, due to $D_P(x_2) = B^2(x_2) - 4A(x_2)C(x_2) \leq B^2(x_2)$. Further observing that $B(x_L) < 0$ (otherwise (a) applies leading to case (i) or case (ii)), and that $D_P(x_L) > 0$ (otherwise (c) applies, since $B(x_L) < 0$ because of (a), which leads to case (i)), we end up with the following simple rule for the computation of ${x_2}^{(min)}$:

Table 7: Values c_{min} (35), corresponding to $x_2{}^{(min)}$, and values c_{hlf} (35), corresponding to our suggested bound $x_2{}^{(hlf)} = (z_2 + m)/2$. (Value 0 indicates case (ii)).

$b \setminus a$	2.0	2.1	2.2	2.5	3.0	4.0	5.0	7.0	9.0
1.0	0 0.167								
1.1	0.023 0.181	0 0.160							
1.2	0.043 0.193	0.005 0.173	0 0.154			not in the region (33)			
1.5	0.087 0.221	0.049 0.202	0.018 0.185	0 0.138		case (ii)			
2.0	0.134 0.250	0.099 0.233	0.068 0.218	0.004 0.176	0 0.118				
3.0	0.184 0.282	0.154 0.267	0.126 0.254	0.057 0.217	0.000 0.167	0 0.092			
5.0	0.225 0.309	0.200 0.297	0.177 0.285	0.115 0.254	0.040 0.210	0 0.146			
10	0.258 0.331	0.237 0.320	0.217 0.310	0.164 0.283	0.095 0.247	0.015 0.194	0 0.155		
20	0.275 0.342	0.256 0.332	0.238 0.323	0.191 0.299	0.128 0.267	0.047 0.220	0.008 0.187	0 0.141	
50	0.286 0.349	0.268 0.340	0.251 0.332	0.207 0.309	0.149 0.280	0.073 0.238	0.029 0.208	0.000 0.168	0 0.142
∞	0.293 0.354	0.276 0.345	0.260 0.337	0.218 0.316	0.164 0.289	0.092 0.250	0.049 0.224	0.009 0.189	0 0.167

Solve $B^2(x_2) - 4\,A(x_2)C(x_2) = 0$ *and return the smallest root* $x_2 > \max\left(\frac{m}{2},\, x_{wl}\right)$.

For the display of the numerical results we introduce the quantities

$$c_{min} \;=\; \frac{x_2{}^{(min)} - x_{wl}}{m - x_{wl}} \qquad \text{and} \qquad c_{hlf} \;=\; \frac{x_2{}^{(hlf)} - x_{wl}}{m - x_{wl}} \tag{35}$$

It is clear that $0 \le c_{min},\ c_{hlf} < 1$, and $c_{min} = 0$ signifies $x_2{}^{(min)} = x_{wl}$. With the results in Table 7 the case (iii) is concluded, but the figures also show that $x_2{}^{(hlf)}$ is a relatively crude (but simple) choice for the abscissas of valid reflection points P_l. To be exact, the limiting values of

$$c_{hlf} \;=\; \frac{x_2{}^{(hlf)} - x_{wl}}{m - x_{wl}} \;=\; \frac{1 - 2x_{wl}}{2s(m - x_{wl})} \;=\; \frac{1}{a+b} - \frac{(a-b)\sqrt{a+b-1}}{2(a+b)\sqrt{ab}}$$

in the bottom line of Table 7 were computed from

$$\lim_{b\to\infty} c_{hlf}(a,\, b) \;=\; \frac{1}{2\sqrt{a}}\,.$$

References

[1] Ahrens, J. H., Dieter, U.: Computer methods for sampling from gamma, beta, Poisson and binomial distributions. Computing *12*, 223 – 246 (1974).

[2] Atkinson, A. C.: A family of switching algorithms for the computer generation of beta random variables. Biometrika *66*, 141 – 145 (1979).

[3] Atkinson, A. C., Whittaker, J.: A switching algorithm for the generation of beta random variables with at least one parameter less than 1. J. Roy. Statist. Soc. A *139*, 462 – 467 (1976).

[4] Atkinson, A. C., Whittaker, J.: Algorithm AS 134: The generation of beta random variables with one parameter greater than one and one parameter less than 1. Applied Statistics *28*, 90 – 93 (1979).

[5] Cheng, R. C. H.: Generating beta variates with nonintegral shape parameters. Comm. ACM *21*, 317 – 322 (1978).

[6] Devroye, L.: Non–Uniform Random Variate Generation. Springer, New York, 1986.

[7] Dieter, U., Ahrens, J. H.: Acceptance–rejection techniques for sampling from the gamma and beta distributions. Techn. Report *83*, Dep. Statist., Stanford Univ., Stanford, California, 1974.

[8] Jöhnk, M. D.: Erzeugung von betaverteilten und gammaverteilten Zufallszahlen. Metrika *8*, 5 – 15 (1964).

[9] Minh, Do Le: Generating gamma variates. ACM Trans. on Math. Software *14*, 261 – 266 (1988).

[10] Sakasegawa, H.: Stratified rejection and squeeze method for generating beta random numbers. Ann. Inst. Statist. Math. *35*, 291 – 302 (1983).

[11] Schmeiser, B. W., Babu, A. J. G.: Beta variate generation via exponential majorizing functions. Oper. Research *28*, 917 – 926 (1980). (Errata: Oper. Research *31*, 802, 1983).

[12] Schmeiser, B. W., Shalaby, M. A.: Acceptance/rejection methods for beta variate generation. J. American Statist. Assoc. *75*, 673 – 678 (1980).

[13] Stadlober, E., Kremer, R.: Sampling from discrete and continuous distributions with C–rand. In: Simulation and Optimization (Pflug, G., Dieter, U. eds.), Springer Lecture Notes Econom. Math. Systems *371*, 154 – 162 (1992).

Part II

Generating Beta Variates with at Least One Parameter Greater than One Via Patchwork Rejection on the Beta Distribution of the Second Kind

Abstract

Generating Beta Variates Via Patchwork Rejection on the Beta Distribution of the Second Kind. A second algorithm for sampling from beta(p, q) distributions which works for parameter combinations $p > 1,\ q > 0$ (and $p > 0,\ q > 1$) is developed. It is based, as the beta algorithm B_1PRS in Part I, on the patchwork technique by Minh which improves acceptance–rejection sampling in the main part of the distributions. Additionally, transformed uniform deviates can often be accepted immediately, so that in most cases fewer than two uniforms are needed for one beta variate, on the average. The remaining tests for acceptance are enhanced by 'squeezes'. Characteristics describing the quality of the sampling method, however, show that the previous algorithm B_1PRS is superior to (or is at least as good as) the new algorithm presented below for all $p > 1$, $q > 1$ algorithm B_1PRS is valid for.

AMS Subject Classification: 65C10, 65C05.

Keywords and Phrases: Random number generation, beta distribution, patchwork rejection.

1 Introduction

In Part I we introduced the algorithm B_1PRS. It is based on patchwork–rejection which combines the patchwork technique, used for the reorganization of the center region under $f(x)$, with acceptance–rejection for the generation of samples from the modified distribution. B_1PRS produces variates from the beta prime distribution directly. In this part of the book we apply the patchwork technique to the density of the beta distribution of the second kind. Samples from this distribution can be transformed easily to the desired beta prime variates. The resulting algorithm B_2PRS is not so fast as B_1PRS, but we can fall back on it to generate beta variates with one parameter less than or equal to one, and the other parameter greater than one, in addition to the case where both parameters are greater than one.

The next section contains the application of the patchwork technique to the beta distribution of the second kind. The basic sampling method (specially with respect to the tails) is described in Section 3. Formal statements of the two new algorithms B_2PRB and B_2PRS are given in Section 4. Algorithms B_2PRS and B_1PRS are compared with respect to some performance measures in Section 5, and in the appendix the statements in Lemma 1 and Lemma 2 of Section 2 are established.

2 Application of the Patchwork Technique to the Beta Distribution of the Second Kind

Let Z be a random variate from the *beta prime distribution* whose density function is

$$f_b(z) = z^{p-1}(1-z)^{q-1}/B(p,\, q)\,, \qquad 0 < z < 1, \quad p > 0,\; q > 0\,. \tag{1}$$

The normalizing constant in (1) which is defined as

$$B(p,\, q) = \int_0^1 z^{p-1}(1-z)^{q-1}\, dz \;=\; \frac{\Gamma(p)\,\Gamma(q)}{\Gamma(p+q)}$$

is the complete beta function. Random variates X from the *beta distribution of the second kind* are obtained from beta prime variates Z by the transformation $X = Z/(1-Z)$. Their density is

$$f_t(x) = x^{p-1}(x+1)^{-(p+q)}/B(p,\, q)\,, \qquad x > 0, \quad p > 0,\; q > 0\,. \tag{2}$$

We confine the shape parameter p to $p > 1$. Then $f_t(x)$ is bounded and it has one mode at $m = (p-1)/(q+1),\; m > 0$.

In order to simplify the following calculations we set $a = p - 1,\, b = q + 1,$ and define

$$f_T(x) = x^a(x+1)^{-(a+b)}\,, \qquad x > 0, \quad a > 0,\; b > 1\,. \tag{3}$$

Differentiating $\ln|f_T{}'(x)| = (a-1)\ln(x) - (a+b+1)\ln(x+1) + \ln|a - bx|$ with respect to x leads to

$$\frac{d}{dx}\,\ln|f_T{}'(x)| \;=\; \frac{b(b+1)x^2 - 2a(b+1)x + a(a-1)}{x(x+1)(a-bx)}\,, \qquad x \neq \frac{a}{b}$$

which may be written as

$$\frac{d}{dx}\,\ln|f_T{}'(x)| \;=\; \frac{(b+1)x^2 - 2m(b+1)x + m(mb-1)}{x(x+1)(m-x)}\,, \quad x \neq m = \frac{a}{b}\,, \tag{4}$$

where $m = a/b$ is the mode of $f_T(x)$.

The reflection point P_l. Inserting (4) into formula (5) in Part I yields, after some straightforward calculations, the following sufficient condition for x_2,

$$A(x_2)z^4 - B(x_2)z^2 + C(x_2) \geq 0\,, \qquad 0 \leq z \leq m - x_2\,, \tag{5}$$

where

$$A(x) = (b+1)(x+m+1)\,, \tag{6}$$

$$\begin{aligned} B(x) \;&=\; 2(b+1)x^3 - 2(m-1)(b+1)x^2 - \\ &\quad - m((m+3)b - 2m)x + m(m+1)(mb+1)\,, \end{aligned} \tag{7}$$

and

$$C(x) = x(x+1)(x-m)((b+1)x^2 - 2m(b+1)x + m(mb-1))\,. \tag{8}$$

After an analytical and numerical examination of the inequality (5) we arrived at the following result which will be proved in the appendix.

Lemma 1 *Let x_{wl} be the left point of inflection of $f_T(x) = x^a(x+1)^{-(a+b)}$ in (3):*

$$x_{wl} \;=\; \frac{a}{b}\left(1 - \sqrt{\frac{a+b}{a(b+1)}}\right) \;=\; m - \sqrt{\frac{m(m+1)}{b+1}}\,, \qquad a = mb \geq 1, \tag{9}$$

where $m = a/b$ is the mode of $f_T(x)$. Furthermore let

$$z_2 \;=\; x_{wl} - \frac{x_{wl}\,(x_{wl}+1)}{b\,(m - x_{wl})} \tag{10}$$

be the point at which the line tangent to $f_T(x)$ at $x = x_{wl}$ crosses the x–axis. Then candidates for abscissas $x_2 < m$ of reflection points P_l which fulfil the inequality (5) are

(i) $x_2 \geq \dfrac{m}{2}\,,$ *if* $a \leq 2$ *(and* $b > 1$*).*

(ii) $x_2 \geq x_{wl}\,,$ *if* $(b+6)x_{wl} - a + 3 \leq 0$ *or, equivalently,*

if $a \geq \dfrac{1}{2}\left(-\,b + (b+6)\sqrt{\dfrac{b}{b-24}}\right)$ *and* $b > 24$

or

if $b \geq \dfrac{-(a^2 - 24a - 9) + (a-3)\sqrt{a^2+54a+9}}{2(a-9)}$ *and* $a > 9\,.$

(iii) $x_2 \geq \dfrac{z_2 + m}{2}\,,$ *if* $a > 2$ *and* $(b+6)x_{wl} - a + 3 > 0$ *or, equivalently,*

$$\textit{if } 2 < a < \begin{cases} \dfrac{1}{2}\left(-\,b + (b+6)\sqrt{\dfrac{b}{b-24}}\right) & \textit{if} \quad b > 24 \\ \infty & \textit{if} \quad b \leq 24\,. \end{cases}$$

The condition in (ii) can hold only when $a > 9$ and $b > 24$. Hence, case (iii) applies, whenever $a > 2$, and $a \leq 9$ or $b \leq 24$. Finally, note that case (i) covers completely the stripe $0 < a \leq 2$.

The proof of Lemma 1 in the appendix contains the additional information that $x_2{}^{(min)} = \frac{m}{2}$ in the case (i), and $x_2{}^{(min)} = x_{wl}$ in the case (ii), where $x_2{}^{(min)}$ is the smallest x_2–value which is feasible in respect to the inequality (5). However, inside the region (iii) (and inside the region (ii), as we shall show in the proof of the Lemma) the safe choice $x_2 = (z_2 + m)/2$ suggested in Lemma 1 is only an upper bound on $x_2{}^{(min)}$. We have no tighter bounds which would be simple enough for the use in the sampling routine.

The reflection point P_r. Inserting (4) into formula (6) in Part I yields, after some calculation, the following sufficient condition for x_4,

$$A(x_4)z^4 - B(x_4)z^2 + C(x_4) \leq 0\,, \qquad 0 \leq z \leq x_4 - m\,, \tag{11}$$

where $A(x)$, $B(x)$, and $C(x)$ are the functions we defined in (6), (7), and (8). We derived from (11) the following simple result which will also be proved in the appendix.

Lemma 2 *Let x_{wr} be the right point of inflection of $f_T(x) = x^a(x+1)^{-(a+b)}$ in (3):*

$$x_{wr} = \frac{a}{b}\left(1 + \sqrt{\frac{a+b}{a(b+1)}}\right) = m + \sqrt{\frac{m(m+1)}{b+1}}\,, \qquad a = mb > 0, \tag{12}$$

where $m = a/b$ is the mode of $f_T(x)$. Then candidates for abscissas $x_4 > m$ of reflection points P_r which fulfil the inequality (11) are

$$x_4 \leq x_{wr}\,, \qquad \textit{if } a > 0 \ \ (\textit{and } b > 1).$$

3 Sampling Method

Henceforth the standardized function

$$f(x) = \frac{f_t(x)}{f_t(m)} = \left(\frac{x}{m}\right)^{p-1}\left(\frac{x+1}{m+1}\right)^{-(p+q)}, \qquad x > 0, \quad p > 1,\ q > 0 \tag{13}$$

is employed instead of the original density $f_t(x)$. $f(x)$ needs no evaluation of the normalizing constant $B(p,q)$, and $f(x) \leq f(m) = 1$ holds for all x.

From Section 2 of Part I we know that the mode m and x_2, x_4, $x_1 = 2x_2 - m$ and $x_5 = 2x_4 - m$ are essential to the sampling routine. x_2 and x_4 are the abscissas of the reflection points P_l and P_r, respectively. x_1 and x_5, on the other hand, enclose the top region of $f(x)$ and its two–part image resulting from the reflections σ_l and σ_r; they split up the area under $f(x)$ into three parts: the *left tail*, the *center*, and the *right tail*. This suggests a sampling routine based on acceptance–rejection with a three part envelope $h(x)$. The center region of $f(x)$ — after the reflections at P_l and P_r — is majorized by a two step staircase function. The left tail of $f(x)$ over $(0, x_1]$ is majorized by an exponential function (power functions are also possible), and the right tail of $f(x)$ over $[x_5, \infty)$ is covered by a power function with the maximum exponent (which has the minimum area of all possible power functions).

3.1 Sampling from the Center

Sampling from the center is carried out as described in Subsection 4.1 of Part I.

3.2 Sampling from the Tails

Let $f_1 = f(x_1)$ and $f_5 = f(x_5)$. The hat function $h(x)$ for the tails is to touch $f(x)$ at x_1 and x_5. This means

$$h(x_1) = f_1,\ h'(x_1) = f'(x_1) \qquad \text{and} \qquad h(x_5) = f_5,\ h'(x_5) = f'(x_5)\,. \tag{14}$$

Furthermore, we demand that samples from the tails of $h(x)$ can be easily generated, for example, by inversion. The exponential and general power functions we state below comply with both conditions.

Envelopes for the left tail

$$\begin{aligned}
h_e{}^{(l)}(x) &= f_1\, e^{(x - x_1)/\lambda_l}\,, & & -\infty < x \le x_1\,, & \lambda_l &= |\varphi(x_1)|\,,\\
h_k{}^{(l)}(x) &= f_1 \left(\frac{d_l - x}{d_l - x_1}\right)^{-k}, & 1 < k < \infty, & -\infty < x \le x_1\,, & d_l &= x_1 + k\,|\varphi(x_1)|\,,
\end{aligned}$$

Envelopes for the right tail

$$h_k{}^{(r)}(x) = f_5 \left(\frac{x - d_r}{x_5 - d_r}\right)^{-k}, \quad 1 < k \le b\,, \qquad x_5 \le x < \infty\,, \qquad d_r = x_5 - k\,|\varphi(x_5)|\,,$$

where $\varphi(x) = 1/(\log f(x))' = f(x)/f'(x) = x(x+1)/(b(m-x))$. There is no exponential function which covers the right tail of $f(x)$, since e^{-x} tends faster towards 0 than $f(x)$ for $x \to \infty$.

The quality of the tail functions $h_e{}^{(l)}(x)$, $h_k{}^{(l)}(x)$ and $h_k{}^{(r)}(x)$ is described by their area. Simple integration immediately leads to

$$\begin{aligned}
H_e{}^{(l)} &= \int_{-\infty}^{x_1} h_e{}^{(l)}(x)\,dx = f_1\,\lambda_l &&= f_1\,|\varphi(x_1)|\,,\\
H_k{}^{(l)} &= \int_{-\infty}^{x_1} h_k{}^{(l)}(x)\,dx = f_1\,(d_l - x_1)/(k-1) &&= \tfrac{k}{k-1}\, f_1\,|\varphi(x_1)|\,,\\
H_k{}^{(r)} &= \int_{x_5}^{\infty} h_k{}^{(r)}(x)\,dx = f_5\,(x_5 - d_r)/(k-1) &&= \tfrac{k}{k-1}\, f_5\,|\varphi(x_5)|
\end{aligned}$$

which shows that the exponential function ${h_e}^{(l)}(x)$ for the left tail, and the power function ${h_b}^{(r)}(x)$ (the power function with maximum exponent $k = b$) for the right tail are the best choices in respect of tightness. The only bitterness of this result is that one logarithm and one general exponentiation will be necessary to generate a sample X from ${h_e}^{(l)}(x)$ and ${h_b}^{(r)}(x)$, respectively. This could be avoided by using the power functions with exponent $k = 2$, i. e. the functions ${h_2}^{(l)}(x)$ and ${h_2}^{(r)}(x)$, as envelopes for the tails, but then firstly both original shape parameters p and q have to be greater than one, and secondly the areas under ${h_2}^{(l)}(x)$ and ${h_2}^{(r)}(x)$ can differ quite a lot (especially when p and q are large) from the minimum areas of the optimal tail envelopes. Therefore we adhere to the tightest envelopes and set $h^{(l)}(x) = {h_e}^{(l)}(x)$ and $h^{(r)}(x) = {h_b}^{(r)}(x)$.

Samples X from $h^{(l)}(x)$ and $h^{(r)}(x)$ are obtained by inversion of the corresponding distribution functions

$$G^{(l)}(x) = e^{(x - x_1)/\lambda_l} \qquad \text{and} \qquad G^{(r)}(x) = 1 - \left(\frac{x - d_r}{x_5 - d_r}\right)^{-(b-1)}$$

as

$$X = x_1 + \lambda_l \log(U) \quad \text{and} \quad X = d_r + (x_5 - d_r)(1 - U)^{-1/(b-1)}\,, \qquad U \sim U(0,\,1)\,.$$

Samples from the tails of $f(x)$ are generated by pure

Acceptance–Rejection. Let $(X,\,Y)$ be uniformly distributed under $h^{(l)}(x)$ or $h^{(r)}(x)$. If $Y \le f(X)$ then the point $(X,\,Y)$ is below $f(x)$, and X can be accepted as a beta variate. Otherwise X is rejected, and a new sample from $h(x)$ is required.

3.3 Squeeze Functions

Since the acceptance tests use the complicated function $f(x)$, fast pre–tests based on simple lower bounds for $f(x)$ (called squeezes) will be added to our basic sampling routine. We employ linear squeezes similar to those in Subsection 4.3 of Part I.

Let $x_1 \le x_{wl}$, $x_5 \ge x_{wr}$. Furthermore, let $z = t(\xi) = \xi - \varphi(\xi) = \xi - \xi(\xi+1)/(b\,(m-\xi))$ be the point at which the line tangent to $f(x)$ at $x = \xi$ crosses the x–axis, and define

$$z_2 = \begin{cases} 0, & \text{if } a \le 1 \\ t(x_{wl}), & \text{if } a > 1\,, \end{cases} \qquad z_1 = t(x_1) = \begin{cases} 0, & \text{if } a \le 1 \\ x_1 - \lambda_l, & \text{if } a > 1\,, \end{cases}$$

$$z_4 = t(x_{wr}), \qquad z_5 = t(x_5) = x_5 + (x_5 - d_r)/b.$$

Then the line segments with end points $(z_i,\,0)$, $(x_i,\,f_i)$, $i = 1,\,2,\,4,\,5$, and the segments with end points $(x_2,\,f_2)$, $(m,\,1)$ and $(m,\,1)$, $(x_4,\,f_4)$ are below $f(x)$, and the complete piecewise linear squeeze function $L(x) \le f(x)$ reads

$$L(x) = \begin{cases} f_1(x - z_1)/(x_1 - z_1), & 0 < x \le x_1 \\ f_2(x - z_2)/(x_2 - z_2), & x_1 < x < x_2 \\ f_2 + (1 - f_2)(x - x_2)/(m - x_2), & x_2 \le x < m \\ f_4 + (1 - f_4)(x_4 - x)/(x_4 - m), & m \le x \le x_4 \\ f_4(z_4 - x)/(z_4 - x_4), & x_4 < x < x_5 \\ f_5(z_5 - x)/(z_5 - x_5), & x_5 \le x < \infty\,. \end{cases}$$

Sampling is modified by $L(x)$ in the acceptance–rejection steps of the algorithm. For the details we refer to Subsection 4.3 of Part I.

4 Algorithms B_2PRS and B_2PRB

The algorithm B_2PRS (**B**eta second kind **P**atchwork **R**ejection method with **S**queezes) results directly from the theory in Section 2 and Section 3. The shorter version B_2PRB (**B**asic method) without lower squeezes is obtained from B_2PRS as follows. Remove all lines that are marked with a '--' sign, and replace the lines which carry '∘∘' and '∘•' with the corresponding lines below:

```
∘∘  Set D_l = m/2, x_2 = D_l, x_1 = 0;
∘•  Set x_1 = x_2 * (1 - (x_2 + 1)/(b * (m - x_2)));
```

Algorithm B_2PRS $(p > 1 \text{ or } q > 1)$

$f(x, a, b, m) = \exp(a * \log(x/m) - (a+b) * \log((x+1)/(m+1)))$

Initialization:

A. Set–up of the parameters a and b (conditional on p and q):

```
    If (p ≤ 1 or (p > q and q > 1)) then
       Set a = q - 1, b = p + 1;
    else
       Set a = p - 1, b = q + 1;
    Endif
```

B. Set–up of the constants which depend on a and b only:

```
    Set m = a/b, D_r = sqrt(m * (m + 1) / (b + 1));
    If (a ≤ 1) then
∘∘     Set D_l = m/2, x_2 = D_l, x_1 = z_2 = 0;          Lemma 1, (i)
       Set f_1 = λ_l = 0;
    else
       Set x_2 = m - D_r;                                Lemma 1, (ii)
--     Set z_2 = x_2 * (1 - (x_2 + 1)/(b * (m - x_2)));
       If ((b + 6) * x_2 - a + 3 ≤ 0) then
          Set x_1 = x_2 - D_r, D_l = D_r;
       else
∘•        Set x_1 = z_2;
          Set x_2 = (x_1 + m)/2, D_l = m - x_2;          Lemma 1, (i), (iii)
       Endif
       Set f_1 = f(x_1, a, b, m);
       Set λ_l = x_1 * (x_1 + 1) / (b * (m - x_1));
    Endif                                                z_1 = x_1 - λ_l
    Set f_2 = f(x_2, a, b, m);
```

```
   Set x4 = m + Dr ,  x5 = x4 + Dr ;                          Lemma 2
-- Set z4 = x4 * (1 + (x4 + 1)/(b * (x4 − m)));
   Set f4 = f(x4, a, b, m);
   Set f5 = f(x5, a, b, m);
   Set dr = x5 * (1 − (x5 + 1)/(x5 − m));                     z5 = x5 + (x5 − dr)/b
   Set p1 = f2 * (m − x1),  p2 = f4 * (x5 − m) + p1;
   Set p3 = f1 * λl + p2,   p4 = f5 * (x5 − dr) / (b − 1) + p3;
```

Generation:

```
1. Case Distinction:
   Generate U ~ U(0, 1) and Set U = U * p4;
   If (U ≤ p1) then Goto 2. [Left Center Region];
               else
   If (U ≤ p2) then Goto 4. [Right Center Region];
               else
   If (U ≤ p3) then Goto 6. [Left Tail];
               else Goto 7. [Right Tail];
   Endif

2. Left Center Region — Immediate acceptance of X ∈ (x1, m):
   Set Y = U/Dl − f2;                                          Y ~ U(−f2, +f2)
   If (Y ≤ 0) then
      Set X = m − U/f2 and Goto 9. [Exit];                     X ∈ (x2, m)
   If (Y ≤ f1) then
      Set X = x2 − Y/f1 * Dl and Goto 9. [Exit];               X ∈ (x1, x2)

3. Left Center Region — Squeeze acceptance of X ∈ (x1, x2) or
   Acceptance–Rejection of its counterpart V ∈ (x2, m):
   Generate U ~ U(0, 1) and Set W = U * Dl;                    W ~ U(0, Dl)
   Set X = x2 − W , V = x2 + W ;
-- If (Y ≤ f2 * (X − z2) / (x2 − z2)) then Goto 9. [Exit];     Y ≤ L(X)
   Set W = f2 + f2 − Y ;                                       W ~ U(f2, 2f2 − f1)
   If (W < 1) then
--    If (W ≤ f2 + (1 − f2) * U) or                            W ≤ L(V)
      If (W ≤ f(V, a, b, m)) then                              W ≤ f(V)
         Set X = V and Goto 9. [Exit];
   Endif
   Goto 8. [Rejection Test];                                   Rejection of V

4. Right Center Region — Immediate acceptance of X ∈ (m, x5):
   Set Y = (U − p1)/Dr − f4;                                   Y ~ U(−f4, +f4)
   If (Y ≤ 0) then
      Set X = m + (U − p1)/f4 and Goto 9. [Exit];              X ∈ (m, x4)
   If (Y ≤ f5) then
      Set X = x4 + Y/f5 * Dr and Goto 9. [Exit];               X ∈ (x4, x5)
```

5. Right Center Region — Squeeze acceptance of $X \in (x_4, x_5)$ or Acceptance–Rejection of its counterpart $V \in (m, x_4)$:

	Generate $U \sim U(0, 1)$ and Set $W = U * D_r$;	$W \sim U(0, D_r)$
	Set $X = x_4 + W$, $V = x_4 - W$;	
--	If $(Y \le f_4 * (z_4 - X)/(z_4 - x_4))$ then Goto 9. [Exit];	$Y \le L(X)$
	Set $W = f_4 + f_4 - Y$;	$W \sim U(f_4, 2f_4 - f_5)$
	If $(W < 1)$ then	
--	If $(W \le f_4 + (1 - f_4) * U)$ or	$W \le L(V)$
	If $(W \le f(V, a, b, m))$ then	$W \le f(V)$
	Set $X = V$ and Goto 9. [Exit];	
	Endif	
	Goto 8. [Rejection Test];	Rejection of V

6. Left Tail — Squeeze acceptance of $X \in (-\infty, x_1]$, computed by inversion:

	Set $U = (U - p_2)/(p_3 - p_2)$;	$U \sim U(0, 1)$
	Set $V = \log(U)$, $X = x_1 + \lambda_l * V$;	
	If $(X \le 0)$ then Goto 1. [Continue];	Fast rejection of X
	Generate $Y \sim U(0, 1)$ and Set $Y = Y * U$;	$Y \sim U(0, h(X)/f_1)$
--	If $(Y \le 1 + V)$ then Goto 9. [Exit];	$Y \le L(X)/f_1$
	Set $Y = Y * f_1$ and Goto 8. [Rejection Test];	$Y \sim U(0, h(X))$

7. Right Tail — Squeeze acceptance of $X \in [x_5, +\infty)$, computed by inversion:

	Set $U = (U - p_3)/(p_4 - p_3)$;	$U \sim U(0, 1)$
	Set $V = U^{-1/(b-1)}$, $X = d_r + (x_5 - d_r) * V$;	
	Generate $Y \sim U(0, 1)$ and Set $Y = Y * U/V$;	$Y \sim U(0, h(X)/f_5)$
--	If $(Y \le 1 + b * (1 - V))$ then Goto 9. [Exit];	$Y \le L(X)/f_5$
	Set $Y = Y * f_5$ and Goto 8. [Rejection Test];	$Y \sim U(0, h(X))$

8. Rejection Test — Rejection of X, whenever $Y > f(X)$:

   If $(Y > f(X, a, b, m))$ then Goto 1. [Continue];

9. Exit — Return $\text{beta}_1(p, q)$ variate:

   If $(a = p - 1)$ then Deliver $Z = X/(X + 1)$, else Deliver $Z = 1/(X + 1)$;

Step A of *Initialization* guarantees that $a > 0$. That we set $a = \min(p, q) - 1$ and $b = \max(p, q) + 1$ if both input parameters p, q are greater than one (instead of setting $a = p - 1$ and $b = q + 1$ only) is justified by the better performance of the patchwork (indicated specially by smaller hat sizes α and higher immediate acceptance probabilities $P(I)$), following from the less frequent occurrence of the right tail case when b is large. We refer to the figures in Table 1, Table 2, Table 3, Table 4 and Table 5 which establish our assertion.

In Step B of *Initialization* we set $x_2 = \frac{m}{2}$ for $a \le 1$ only. For the remaining part of case (i) in Lemma 1 $x_2 = (z_2 + m)/2$ from case (iii) is substituted. This is legitimate, since $(z_2 + m)/2 \ge \frac{m}{2}$ (x_2 remains a valid abscissa of P_l). The reason for using ${x_2}^{(hlf)} = (z_2 + m)/2$ instead of ${x_2}^{(min)} = \frac{m}{2}$ in the region $1 < a \le 2$, $b > 1$ is that $x_2 = {x_2}^{(hlf)}$ leads to slightly higher total immediate acceptance probabilities $P(I)$ and to tighter hats for the left part of $f(x)$ which improves the total performance of the algorithm in this parameter region.

Fast rejection tests based on the condition $f(x) \leq 1$ occur in Step 3 and Step 5 of *Generation*. Step 6 dealing with the left tail of $f(x)$ also contains a rejection test which discards immediately all samples $X \leq 0$ from the exponential envelope to avoid the otherwise disastrous evaluation of $f(x)$ in Step 8.

We arranged the description of algorithm B$_2$PRS analogous to B$_1$PRS in Part I. In fact, B$_2$PRS corresponds completely with B$_1$PRS in Step 1 of *Generation*, and basically in Step 2 to Step 6. However, since the left tail case occurs very rarely, and the area of the left center region is always smaller than (for $a \to \infty$, $b \to \infty$ equal to) the area of the right center region, we recommend to change the order of the statements in Step 1, such that the more frequent cases are questioned first (i. e. check if 'Right Center Region' applies before you check whether 'Left Center Region' applies or not). But note that this requires some slight modifications in the definition of the p_i and of the statements containing them.

5 Characteristics of Algorithms B$_2$PRS and B$_1$PRS

Both algorithms B$_2$PRS and B$_1$PRS are based on patchwork–rejection. Basically therefore they use similar hat functions (three part envelopes $h(x)$ whose center piece is a two step staircase function) for the generation of the inital samples X which are transformed by acceptance–rejection to the desired beta variates. In both cases the center regions — after the reflections at P_l and P_r (see Figure 1 of Part I) — are majorized by two step staircase functions, and the left tails by exponential functions, as the right tail of $f_b(x)$. The right tail of $f_t(x)$, however, is so heavy that it cannot be majorized by an exponential function. Instead of it a power function with the maximum exponent is used as envelope. Both algorithms also contain immediate acceptance (based on the rectangles R_1, R_2, R_3 and R_4 below $f(x)$) and squeeze acceptance (based on the lower bound $L(x)$ for $f(x)$).

B$_2$PRS uses the same technique as B$_1$PRS to generate beta variates, except that B$_2$PRS produces beta variates of the second kind and transforms them to the desired beta prime variates. In order to find out which of the two algorithms is the better one, we compare them with respect to the performance measures α, $P(I)$, N and $P(S)$, and with respect to $P(Q) = P(I) + P(S)$ (see formulas (20), (21), (22), and (23) in Part I which contain the main characteristics of algorithm B$_1$PRS). Although the characteristics of algorithm B$_2$PRS do not differ very much from those of algorithm B$_1$PRS, we state them in full detail below.

First of all, we need the average number α of samples X from $h(x)$, until X is accepted as a beta variate of the second kind:

$$\alpha = f_t(m)\Big(f_1\,\lambda_l + f_2(m - x_1) + f_4(x_5 - m) + f_5\,(x_5 - d_r)/(b-1)\Big). \tag{15}$$

The probability $P(I)$ of accepting X immediately, given that X is accepted, is the rescaled area of the four rectangles R_i:

$$P(I) = f_t(m)\Big((f_1 + f_2)(m - x_2) + (f_4 + f_5)(x_4 - m)\Big). \tag{16}$$

With these two quantities the average number of uniforms N needed to generate one beta variate of the second kind reads

$$N = 2\alpha - P(I) - f_t(m)\Big(f_1\,\lambda_l\,e^{-x_1/\lambda_l}\Big). \tag{17}$$

The third term describes the average number of samples X from $h(x)$ less than or equal to 0 which are outside the domain $(0, \infty)$ of the beta density $f_t(x)$. They are quickly rejected without carrying out an acceptance–rejection test; therefore they use up only one uniform number (for the generation of X). The same applies to the samples X from the 'immediate acceptance regions'. All other deviates, however, require exactly two uniforms, one for their generation and one to decide on their acceptance or rejection.

The fourth and last characteristic is the probability $P(S)$ of accepting X by means of the lower squeeze $L(x)$, given that X is accepted. It reads

$$\begin{aligned} P(S) = \tfrac{1}{2}\, f_t(m)\Big(f_1\,\lambda_l + (f_2 - f_1)(1 - f_1/f_2)(x_2 - z_2) + (1 - f_2)(m - x_2) \\ + (1 - f_4)(x_4 - m) + (f_4 - f_5)(1 - f_5/f_4)(z_4 - x_4) + f_5\,(x_5 - d_r)/b\Big). \end{aligned} \tag{18}$$

Table 1 to Table 5 display the characteristics of algorithms B$_2$PRS and B$_1$PRS for different parameter combinations $(q,\, p)$. Because of symmetry the values of B$_1$PRS (which is valid for $p > 1,\ q > 1$) are stated for $q \geq p$ only.

The figures of N, $P(I)$, $P(S)$ and $P(Q) = P(I) + P(S)$ in Table 2, Table 3, Table 4 and Table 5 indicate a better performance of B$_1$PRS, with respect to all these characteristics. However, its average number α of samples X from $h(x)$ needed to produce one beta variate

Table 1: Average number α of samples X from $h(x)$, until X is accepted as a beta variate.

Algorithm B$_2$PRS

$q \setminus p$	1.1	1.5	3.0	10	1000
0.1	1.092	1.115	1.123	1.124	1.124
0.5	1.080	1.100	1.111	1.112	1.112
1.1	1.070	1.088	1.100	1.100	1.100
1.5	1.066	1.083	1.096	1.095	1.095
2.0	1.062	1.078	1.091	1.090	1.090
3	1.057	1.072	1.086	1.082	1.082
5	1.053	1.067	1.080	1.073	1.072
10	1.048	1.062	1.074	1.063	1.060
20	1.046	1.059	1.070	1.055	1.050
1000	1.043	1.056	1.067	1.045	1.022

Algorithm B$_1$PRS

$q \setminus p$	1.1	1.5	3.0	10	1000
0.1	—	—	—	—	—
0.5	—	—	—	—	—
1.1	1.030				
1.5	1.075	1.103			
2.0	1.046	1.106			
3	1.110	1.043	1.080		
5	1.124	1.069	1.055		
10	1.091	1.072	1.065	1.018	
20	1.068	1.066	1.068	1.019	
1000	1.044	1.056	1.067	1.020	1.022

Table 2: Average number N of uniforms needed to generate one beta variate.

Algorithm B$_2$PRS

$q \setminus p$	1.1	1.5	3.0	10	1000
0.1	2.154	2.186	2.195	2.196	2.196
0.5	2.045	2.032	2.027	2.023	2.023
1.1	1.953	1.903	1.883	1.873	1.872
1.5	1.914	1.849	1.824	1.810	1.808
2.0	1.880	1.802	1.771	1.753	1.751
3	1.835	1.744	1.706	1.681	1.677
5	1.789	1.685	1.643	1.609	1.603
10	1.746	1.634	1.589	1.546	1.536
20	1.721	1.606	1.562	1.513	1.499
1000	1.694	1.577	1.537	1.483	1.452

Algorithm B$_1$PRS

$q \setminus p$	1.1	1.5	3.0	10	1000
0.1	—	—	—	—	—
0.5	—	—	—	—	—
1.1	1.545				
1.5	1.613	1.654			
2.0	1.568	1.659			
3	1.564	1.526	1.573		
5	1.724	1.526	1.526		
10	1.736	1.565	1.522	1.465	
20	1.718	1.574	1.528	1.462	
1000	1.694	1.576	1.537	1.465	1.452

Table 3: Probability $P(I)$ of accepting X immediately, given that X is accepted.

Algorithm B$_2$PRS

$q \setminus p$	1.1	1.5	3.0	10	1000
0.1	0.030	0.044	0.051	0.052	0.052
0.5	0.115	0.169	0.195	0.201	0.201
1.1	0.187	0.273	0.317	0.328	0.329
1.5	0.217	0.316	0.367	0.380	0.382
2.0	0.245	0.354	0.411	0.427	0.428
3	0.280	0.401	0.465	0.484	0.486
5	0.316	0.448	0.517	0.538	0.541
10	0.351	0.489	0.558	0.580	0.584
20	0.370	0.511	0.579	0.598	0.601
1000	0.392	0.534	0.596	0.607	0.591

Algorithm B$_1$PRS

$q \setminus p$	1.1	1.5	3.0	10	1000
0.1	—	—	—	—	—
0.5	—	—	—	—	—
1.1	0.515				
1.5	0.538	0.551			
2.0	0.523	0.553			
3	0.612	0.559	0.587		
5	0.511	0.608	0.582		
10	0.445	0.579	0.609	0.572	
20	0.418	0.558	0.607	0.577	
1000	0.393	0.535	0.597	0.574	0.592

Table 4: Probability $P(S)$ of accepting X by means of the lower squeeze $L(x)$, given that X is accepted.

Algorithm B$_2$PRS

$q \setminus p$	1.1	1.5	3.0	10	1000
0.1	0.051	0.055	0.058	0.058	0.058
0.5	0.171	0.177	0.183	0.183	0.183
1.1	0.245	0.245	0.250	0.250	0.250
1.5	0.269	0.264	0.269	0.268	0.268
2.0	0.288	0.278	0.281	0.280	0.280
3	0.308	0.290	0.293	0.291	0.291
5	0.324	0.298	0.300	0.297	0.297
10	0.335	0.301	0.304	0.302	0.302
20	0.340	0.302	0.306	0.306	0.307
1000	0.343	0.302	0.310	0.315	0.330

Algorithm B$_1$PRS

$q \setminus p$	1.1	1.5	3.0	10	1000
0.1	—	—	—	—	—
0.5	—	—	—	—	—
1.1	0.265				
1.5	0.323	0.318			
2.0	0.413	0.361			
3	0.311	0.357	0.362		
5	0.339	0.303	0.356		
10	0.346	0.300	0.325	0.351	
20	0.346	0.301	0.316	0.344	
1000	0.343	0.302	0.310	0.335	0.330

Table 5: Probability $P(Q) = P(I)+P(S)$ of accepting X quickly, given that X is accepted.

Algorithm B_2PRS

$q \setminus p$	1.1	1.5	3.0	10	1000
0.1	0.081	0.100	0.108	0.110	0.110
0.5	0.285	0.346	0.378	0.384	0.385
1.1	0.432	0.518	0.567	0.578	0.579
1.5	0.487	0.581	0.636	0.649	0.650
2.0	0.533	0.632	0.693	0.707	0.709
3	0.588	0.692	0.758	0.775	0.777
5	0.641	0.746	0.816	0.835	0.838
10	0.686	0.790	0.862	0.882	0.886
20	0.710	0.813	0.885	0.904	0.908
1000	0.736	0.836	0.906	0.922	0.921

Algorithm B_1PRS

$q \setminus p$	1.1	1.5	3.0	10	1000
0.1	—	—	—	—	—
0.5	—	—	—	—	—
1.1	0.780				
1.5	0.861	0.870			
2.0	0.936	0.914			
3	0.923	0.916	0.949		
5	0.850	0.910	0.938		
10	0.792	0.879	0.934	0.923	
20	0.763	0.859	0.923	0.920	
1000	0.737	0.837	0.907	0.909	0.922

is often larger than that of algorithm B_2PRS, but only in the case that one parameter is small. Nevertheless, its mostly higher immediate acceptance probability $P(I)$ which naturally results in a smaller number N (as the α values of both algorithms do not differ very much) and the much higher quick acceptance probability $P(Q)$ lead us to assume that the execution time of B_2PRS is greater than that of the algorithm B_1PRS.

From the tables above we conclude that algorithm B_1PRS is more efficient than algorithm B_2PRS in the entire region $p > 1$, $q > 1$ B_1PRS is valid for. Hence, the only advantage of B_2PRS is that it can be used for the generation of beta variates with exactly one parameter greater than one. But just in this case the patchwork technique is not so good; consider the figures in Table 1, Table 2, Table 3, Table 4 and Table 5.

Limiting Values

For $p \to \infty$ and $q \to \infty$ the characteristics in (15), (16) and (17) tend, as for the beta prime distribution, to

$$\begin{aligned}
\lim_{p,q\to\infty} \alpha &= \frac{4}{\sqrt{2\pi}}\left(e^{-1/2} + \tfrac{1}{4}e^{-2}\right) = 1.0218\ldots,\\
\lim_{p,q\to\infty} P(I) &= \frac{2}{\sqrt{2\pi}}\left(e^{-1/2} + e^{-2}\right) = 0.5919\ldots,\\
\lim_{p,q\to\infty} N &= \frac{6}{\sqrt{2\pi}}\ e^{-1/2} = 1.4518\ldots.
\end{aligned}$$

The above limiting values are the result of tedious calculations similar to those in the beta prime case.

6 Appendix

Proof of Lemma 1

The reflection σ_l (formula (1) in Part I), illustrated in Figure 1 of Part I, demands that x_2 is at least $\frac{m}{2}$. This guarantees that the transformed top region T_l' to the left of x_2 is inside the domain $(0,\ \infty)$ of the beta density $f_t(x)$. x_2 also has to be greater than or equal to the left point of inflection x_{wl}. This follows from (5), since for $z = 0$ one has $C(x_2) \geq 0$ which is satisfied if, and only if $x_{wl} \leq x_2 \leq m$. Hence

$$x_2 \ \geq\ \max\left(\frac{m}{2}\,,\ x_{wl}\right) \tag{19}$$

is a general precondition for abscissas x_2 of valid reflection points P_l.

Case (i) :

Let $x_{wl} \leq \frac{m}{2}$. Then (19) demands that $x_2 \geq \frac{m}{2}$. Below we show that $x_2 = \frac{m}{2}$ is feasible for all $a \leq 2$, without any restriction from the prerequisite.

For the proof x_2 is replaced with $\frac{m}{2}$ in (5). After some calculation we arrive at

$$4(b+1)(3m+2)z^2 + m(m+2)(mb-3m-4) \leq\ 0\,, \qquad 0 \leq z \leq \frac{m}{2}\,. \tag{20}$$

The left hand side of (20) is strictly increasing. Therefore it attains its maximum at the right end point $z = \frac{m}{2}$, and condition (20) reduces to the single inequality $(b+1)(3m+2)m^2 + m(m+2)(mb-3m-4) = 4m(m+1)(mb-2) \leq 0$. This shows that $x_2 = \frac{m}{2}$ is valid, provided that $mb \leq 2$, i. e. $a \leq 2$.

According to (9), the precondition $x_{wl} \leq \frac{m}{2}$ is equivalent to $mb - 3m - 4 \leq 0$. It evidently holds for all $a \leq 4$ which establishes the feasibility of $x_2 = \frac{m}{2}$ for any $a \leq 2$ (and $b > 1$).

If $x_2 = \frac{m}{2}$ is valid, it is relatively easy to see that all x_2 between $\frac{m}{2}$ and m are also feasible. The formal proof is omitted.

Remark. For $0 < a \leq 1$ the feasibility of $x_2 = \frac{m}{2}$ still follows from the concavity of $f_T(x)$ in the interval $(0,\ m)$.

Case (ii) :

Let $x_{wl} \geq \frac{m}{2}$. Then (19) demands that $x_2 \geq x_{wl}$. Again we only show that $x_2 = x_{wl}$ is valid for the part of the parameter region which is described by condition $(b+6)x_{wl} - mb + 3 \leq 0$. The argumentation is as follows.

If x_{wl} in (9) is substituted for x_2, condition (5) reduces to

$$A(x_{wl})z^2 - B(x_{wl}) \geq 0\,, \qquad 0 \leq z \leq m - x_{wl}\,, \tag{21}$$

since x_{wl} substituted into (8) yields $C(x_{wl}) = 0$. From (6) it is clear that $A(x_{wl}) > 0$, and the left hand side of (21) is strictly increasing. Therefore it attains its minimum at the left end point $z = 0$, and (21) reduces to the single condition $-B(x_{wl}) \geq 0$. Using (9) in the form $(b+1){x_{wl}}^2 - 2m(b+1)x_{wl} + m(mb-1) = 0$ (to minimize the degree of the polynomial $B(x_{wl})$ and thereby to simplify it as much as possible) finally yields

$$-B(x_{wl}) \;=\; -\,m(m+1)\left((b+6)x_{wl} - mb + 3\right) \;\geq\; 0\,. \tag{22}$$

According to (9), condition (22) is equivalent to $3(2a+b)\sqrt{b+1} - (b+6)\sqrt{a(a+b)} \leq 0$ which is fulfilled, whenever $a^2b + ab^2 - 24a^2 - 9b^2 - 24ab - 9b \geq 0$. Factorizing the polynomial with respect to a and with respect to b yields the two equivalent inequalities

$$(b-24)a^2 \,+\, b(b-24)a \,-\, 9b(b+1) \,\geq\, 0 \tag{23}$$

$$(a-9)b^2 + (a^2 - 24a - 9)b \,-\, 24a^2 \,\geq\, 0\,. \tag{24}$$

They prove that condition (22) can hold only for parameter combinations $(a,\, b)$ which meet the requirement $a > 9$ and $b > 24$. In this region, however, the precondition $x_{wl} \geq \frac{m}{2}$ which is equivalent to $mb - 3m - 4 \geq 0$ is fulfilled. Consequently $x_2 = x_{wl}$ is valid, whenever

$$(b+6)x_{wl} - mb + 3 \,\leq\, 0\,. \tag{25}$$

This proves the first statement in (ii). The two other statements which define the region described by condition (25) with respect to parameter a and parameter b are established by solving the inequalities (23) and (24).

Case (iii) :

Let x_{wl} be the left point of inflection (9), z_2 the point at which the line tangent to $f_T(x)$ at $x = x_{wl}$ crosses the x–axis (10), and ${x_2}^{(hlf)} = (z_2 + m)/2$ our suggested bound on ${x_2}^{(min)}$.

First of all, we establish by an elementary derivation (without any numerical calculations) that ${x_2}^{(hlf)}$ meets the general preconditon (19), not only in the remaining region but even in the entire region $a \geq 1,\, b > 1$ x_{wl} is defined.
Obviously ${x_2}^{(hlf)} \geq \frac{m}{2}$, so we only have to show that ${x_2}^{(hlf)} \geq x_{wl}$. But this condition is equivalent to

$$\begin{aligned} {x_2}^{(hlf)} - x_{wl} &= \frac{m - x_{wl}}{2} - \frac{x_{wl}(x_{wl}+1)}{2b(m - x_{wl})} \\ &= \frac{(b+1)(m-x_{wl})^2 - (m-x_{wl})^2 - x_{wl}(x_{wl}+1)}{2b(m-x_{wl})} \;=\; \frac{2x_{wl}+1}{2b} \;\geq\; 0 \end{aligned}$$

which establishes our assertion. From the above and case (ii) it follows that ${x_2}^{(hlf)}$ is valid if a and b are large enough. The more general statement that ${x_2}^{(hlf)}$ is feasible for all $a \geq 1,\;\; b > 1$ is true as well, but this can be proved only by numerical means complemented with asymptotical considerations.

In order to show the feasibility of our proposal $x_2 = x_2{}^{(hlf)}$, we compare it with $x_2{}^{(min)}$, the smallest x_2–value which is feasible in respect to the inequality (5), which is obtained from (5) as follows.
Using the polynomials in (6), (7), and (8), we define

$$P(z) = A(x_2)z^4 - B(x_2)z^2 + C(x_2) \quad \text{and} \quad D_P(x_2) = B^2(x_2) - 4A(x_2)C(x_2)\,,$$

where $A(x_2) \geq 0$. Furthermore we have $C(x_2) \geq 0$ from (19). We consider now three cases, dependent on the signs of $B(x_2)$ and $D_P(x_2)$.

(a) Let $B(x_2) \leq 0$. Then $P(z) \geq 0$ for all z, and x_2 is valid.

(b) Let $B(x_2) > 0$ and $D_P(x_2) > 0$. Then $P(z) = 0$ has exactly two positive roots. Hence, $P(z) \geq 0$ for $0 \leq z \leq m - x_2$, whenever $P(m-x_2) \geq 0$ and $P'(m-x_2) < 0$. These conditions reduce, after some calculation, to

$$\begin{aligned} m(m+1)\,(m-x_2)(2x_2-m)(2x_2-m+1) &\geq 0\,, \\ -\,m(m+1)\,((b+4)x_2-(mb+2m-1)) &\leq 0 \end{aligned}$$

from which we obtain that

$$x_2 \geq \begin{cases} m/2 & \text{if} \quad a \leq 2 \\ m-(2m+1)/(b+4) & \text{if} \quad a > 2\,. \end{cases}$$

That $x_2 = \frac{m}{2}$ is feasible for $a \leq 2$ was established already in (i). For $a > 2$, however, the minimum x_2 from above leads either to $B(x_2) \leq 0$ or $D_P(x_2) \leq 0$ which contradicts the assumption $B(x_2) > 0$ and $D_P(x_2) > 0$, and this case yields no useful condition we could use for the determination of $x_2{}^{(min)}$.

(c) Let $B(x_2) > 0$ and $D_P(x_2) \leq 0$. Then $P(z)$ does not change its sign. Since $C(x_2) > 0$, we have $P(z) \geq 0$ for all z, i. e. x_2 is valid.

From the above we conclude that $x_2{}^{(min)}$ fulfils at least one of the two conditions

$$B(x_2) \leq 0 \qquad \text{or} \qquad D_P(x_2) \leq 0\,, \qquad x_L < x_2 < m\,, \tag{26}$$

where $x_L = \max\left(\frac{m}{2},\, x_{wl}\right)$.
$B(x_2) = 0$ implies $D_P(x_2) \leq 0$, due to $D_P(x_2) = B^2(x_2) - 4A(x_2)C(x_2) \leq B^2(x_2)$. Further observing that $B(x_L) > 0$ (otherwise (a) applies leading to case (i) or case (ii)), and that $D_P(x_L) > 0$ (otherwise (c) applies, since $B(x_L) > 0$, leading to case (i)), we end up with the following simple rule for the computation of $x_2{}^{(min)}$:

Solve $B^2(x_2) - 4\,A(x_2)C(x_2) = 0$ *and return the smallest root* $x_2 > \max\left(\frac{m}{2},\, x_{wl}\right)$.

For the display of the numerical results we introduce the quantities

$$c_{min} = \frac{x_2{}^{(min)} - x_{wl}}{m - x_{wl}} \qquad \text{and} \qquad c_{hlf} = \frac{x_2{}^{(hlf)} - x_{wl}}{m - x_{wl}}\,. \tag{27}$$

Table 6: Values c_{min} (27), corresponding to $x_2{}^{(min)}$, and values c_{hlf} (27), corresponding to our suggested bound $x_2{}^{(hlf)} = (z_2 + m)/2$. (Value 0 indicates case (ii)).

$b \setminus a$	2.0	3.0	5.0	7.0	10.0	20.0	50.0	1000
1.05	0.420 0.442	0.391 0.427	0.373 0.417	0.367 0.415	0.364 0.413	0.362 0.412	0.361 0.411	0.361 0.411
1.25	0.412 0.436	0.377 0.418	0.356 0.407	0.350 0.404	0.346 0.402	0.343 0.401	0.342 0.400	0.342 0.400
1.50	0.402 0.429	0.363 0.409	0.338 0.397	0.330 0.392	0.325 0.390	0.322 0.388	0.321 0.388	0.320 0.387
2.00	0.388 0.419	0.340 0.394	0.308 0.378	0.297 0.373	0.290 0.370	0.285 0.367	0.283 0.366	0.283 0.366
3.00	0.368 0.405	0.306 0.374	0.262 0.352	0.246 0.344	0.237 0.339	0.228 0.335	0.226 0.334	0.225 0.333
5.00	0.345 0.389	0.267 0.350	0.206 0.320	0.183 0.308	0.168 0.300	0.154 0.293	0.149 0.290	0.148 0.290
10.00	0.323 0.374	0.225 0.325	0.143 0.283	0.109 0.265	0.086 0.252	0.064 0.239	0.056 0.233	0.054 0.232
20.00	0.309 0.365	0.198 0.309	0.100 0.257	0.060 0.233	0.033 0.215	0.011 0.193	0.004 0.182	0.003 0.179
50.00	0.300 0.358	0.178 0.297	0.070 0.238	0.028 0.209	0.005 0.184	0 0.152	0 0.131	0 0.123
1000.00	0.293 0.354	0.165 0.289	0.050 0.224	0.010 0.190	0 0.160	0 0.114	0 0.075	0 0.033

It is clear that $0 \le c_{min},\ c_{hlf} < 1$, and $c_{min} = 0$ signifies $x_2{}^{(min)} = x_{wl}$.
Values of c_{min} for various parameters a, b are listed as the first entries in Table 6. A close look at them gives rise to the speculation that c_{min} is decreasing in both parameters a and b.
Using the definition (9), c_{hlf} reduces to

$$c_{hlf} = \frac{x_2{}^{(hlf)} - x_{wl}}{m - x_{wl}} = \frac{2x_{wl} + 1}{2b(m - x_{wl})} = \frac{2a + b}{2b}\sqrt{\frac{b+1}{a(a+b)}} - \frac{1}{b}\,.$$

If one parameter is fixed now and the other one tends to infinity, we obtain the following limiting values

$$\lim_{a\to\infty} c_{hlf}(a,\, b) = \frac{\sqrt{b+1} - 1}{b}\,, \qquad \lim_{b\to\infty} c_{hlf}(a,\, b) = \frac{1}{2\sqrt{a}} \tag{28}$$

which are lower bounds for c_{hlf}, since c_{hlf} is strictly decreasing in a and b. Table 6 displays values of c_{hlf} as the second entry for each parameter combination $(a,\, b)$. The corresponding limiting values are not stated, since they are very close to the values of c_{hlf} at $a = 1000$ and $b = 1000$, respectively (see second entries of last column and bottom line of Table 6).

Comparing the values of c_{min} and c_{hlf} in Table 6 and noticing that for $1 \leq a \leq 2$ we have ${x_2}^{(min)} = \frac{m}{2}$ which is undoubtedly smaller than ${x_2}^{(hlf)}$ proves the general assertion we made at the beginning of (iii).

With the results in Table 6 the case (iii) is concluded, but the figures also show that ${x_2}^{(hlf)}$ is a relatively crude (but simple) choice for the abscissas x_2 of valid reflection points P_l.

Proof of Lemma 2

Since the domain of the density $f_t(x)$ is $(0, \infty)$, the inequality (11) is the only condition x_4 must satisfy. Similar to Lemma 1, we immediately obtain that x_4 has to be at least less than or equal to the right point of inflection x_{wr}, and

$$x_4 \leq x_{wr} \tag{29}$$

is a general precondition for abscissas x_4 of valid reflection points P_r. Below we show that $x_4 = x_{wr}$ is valid for the whole parameter region.

If x_{wr} in (12) is substituted for x_4, condition (11) reduces to

$$A(x_{wr})z^2 - B(x_{wr}) \leq 0\,, \qquad 0 \leq z \leq x_{wr} - m\,, \tag{30}$$

since x_{wr} substituted into (8) yields $C(x_{wr}) = 0$. With the observation that $A(x_{wr}) > 0$ (30) further simplifies to

$$A(x_{wr})(x_{wr} - m)^2 - B(x_{wr}) \leq 0\,. \tag{31}$$

Inserting (6) and (7) into (31), and using the definition (12) in the original form and in the form $(b+1){x_{wr}}^2 - 2m(b+1)x_{wr} + m(mb-1) = 0$ (which allows us to decrease the degree of the polynomial $B(x_{wr})$ step by step; similar to case (ii) in the proof of Lemma 1 we end up with $B(x_{wr}) = m(m+1)\,((b+6)x_{wr} - mb + 3))$ finally yields

$$-\,m(m+1)\,((b+5)(x_{wr} - m) + 2(2m+1)) \leq 0 \tag{32}$$

which holds for all parameter combinations. This concludes the proof of Lemma 2.

Part III

Generating Discrete Variates Via Patchwork–Rejection, and its Application to the Poisson, Binomial and Hypergeometric Distributions

1 Introduction

Computer methods for sampling from discrete distributions are essential tools in modelling simulation input processes. General table methods like the alias method and the guide table method are ultra–fast if the distribution stays fixed, but any change of parameters requires a time consuming set–up of tables. On the other hand, special methods of the acceptance–rejection type are available for most of the well–known distributions. The competitive ones are uniformly fast, i. e. their execution time is bounded over the defined parameter range. They remain efficient even when the parameters of the distribution in hand vary all the time (see [2], [7], [8], [10], [12], [23] and [25]).

Our new sampling method is designed for unimodal discrete distributions. It extends the *patchwork–rejection* approach which proved to be successful for continuous distributions (see [20] and Part I in this book) to the discrete case. The crucial idea is to improve acceptance–rejection in the body of the distribution as follows.

In a centre interval $[k_1, k_5 + 1)$ the standardized histogram function $f(x)$ is modified by reflection at one point $P_l = (k_2 + \frac{1}{2}, f(k_2))$ to the left of the mode m and at one point $P_r = (k_4 + \frac{1}{2}, f(k_4))$ to the right of the mode m, such that the altered histogram fits well under two rectangles, representing uniform hat functions.

Minorizing rectangles allow for immediate acceptance of one transformed and truncated uniform deviate. Additionally, easy–to–calculate lower bounds on $f(x)$ (called 'squeezes') are used for fast pre–tests, thereby avoiding the slow computation of $f(x)$ most of the time. The remaining left and right tail of $f(x)$ may be handled by rejection with exponential or power functions.

Advantages of this method are an increased acceptance rate, a substantial decrease in the number of uniforms required per variate, and a faster generation process. A minor drawback is the elaborate sampling procedure with rather high set–up costs.

General conditions for *feasible reflection centres* k_2 and k_4, based on the recurrence relationship function $r(k) = f(k)/f(k-1)$, are given in Lemma 1 of Section 2. Specific solutions are presented for Poisson and binomial distributions in Section 6, and for hypergeometric distributions in Section 8.

If $m \geq 4$ the patchwork–rejection algorithms can use closed form reflection centres $k_2 \approx m - \sigma$ and $k_4 \approx m + \sigma$, where σ^2 is the variance of the distribution in hand. In this case the expected number of uniforms needed per variate turns out to be between 1.4 and 2.0. Simple modifications of k_2 and k_4 lead to algorithms which cover the whole parameter range of the distributions.

For the evaluation of $\log k!$ in $\log f(x)$ a Stirling approximation with truncation error ϵ: $|\epsilon| < 4.27 \times 10^{-17}$ in case of $k \geq 30$ is used, complemented by a table which stores the values of $\log k!$ for $k = 0, 1, \ldots, 29$.

Uniform random numbers are generated by a multiplicative congruential C–generator (modulus $m = 2^{54}$, multiplicator $a = 2\,783\,377\,640\,906\,189$) with a good lattice structure and the long period 2^{52} which needs about 22 μsec/uniform on a PC/AT 486–33 MHz. Our computational experience indicates that the proposed sampling routines are faster than the current state-of-the–art algorithms at the expense of a more time–consuming set–up. The most remarkable improvement is obtained in the hypergeometric case, where the execution time of our new generator is $65 - 80$ μsec/variate against $105 - 230$ μsec/variate for its competitors [10] and [23].

Section 2 contains the general description of the patchwork technique in the discrete case. The basic patchwork–rejection sampling method is described in Section 3. Formal statements of the new general algorithm GPRS for sampling from unimodal discrete distributions are given in Section 4, and its main performance measures are stated in Section 5. Section 6 and Section 8 contain the application of the patchwork technique to the Poisson and binomial, and hypergeometric distributions, respectively. Computational experience based on extensive timings is reported in Section 7 (Poisson and binomial case) and Section 9 (hypergeometric case).

In Appendix A the proof of the crucial Lemma 1 of Section 2 is established. Appendix B and Appendix C contain the proof of Lemma 3 (Poisson and binomial case) and Lemma 4 (hypergeometric case), respectively, and in Appendix D you find a collection of tables displaying the main characteristics of the Poisson, binomial and hypergeometric sampling methods we included into our computational investigations.

2 The Patchwork Technique

Let $p(k),\ k \in [k_{min},\, k_{max}]$ and k integer, be a unimodal probability function with its mode at m, and

$$f(x) \;=\; \begin{cases} p\left(\lfloor x \rfloor\right) & k_{min} \le x < k_{max}+1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

the corresponding histogram function. Furthermore, let k_2 and k_4, later on referred to as reflection centres, be two events from the domain of $p(k)$, i. e. $k_{min} \le k_2,\ k_4 \le k_{max}$, such that $k_2 < k_4$ and $m \in [k_2,\, k_4]$. k_2 and k_4 define a so–called 'centre interval' which contains the mode and at least one other event.

The basic idea of patchwork is to cut down the centre probabilities $p(i),\ i \in [k_2,\, k_4]$, such that the remaining parts of $p(i)$ left from m and right from m, respectively, are of the same height. The residual centre probabilities — i. e. the parts of $p(i)$ which project over the cut–off level — left from m are added in reverse order to the adjoining probabilities $p(j)$ on the left ($j < k_2$). The residual centre probabilities right from m are added, by analogy, in reverse order to the adjoining probabilities $p(j)$ on the right ($j > k_4$). For both sides we demand that the sums of the residual centre probabilities and probabilities $p(j)$ do not exceed the cut–off level. This particular rearrangement of the probabilities $p(i)$ can be described technically by two point reflections. It allows us to use a two–step staircase function as a very tight hat function in the centre of the modified histogram.

With the notations in Figure 1 which displays the patchwork technique at the histogram $f(x)$, the rearrangement of the probabilities $p(i)$ can be formulated more precisely.

The left part T_l of the top region under $f(x)$ is transformed into the congruent area $T_l{}'$ by using the point P_l as the centre of a reflection σ_l. Similarly the right part T_r of the top region under $f(x)$ is transformed into $T_r{}'$ by using P_r as the centre of a reflection σ_r. The reflection points

$$P_l = \left(k_2 + \tfrac{1}{2},\ f(k_2)\right), \qquad P_r = \left(k_4 + \tfrac{1}{2},\ f(k_4)\right) \tag{2}$$

must be determined carefully, such that the transformed regions $T_l{}'$ and $T_r{}'$ contain no points below $f(x)$.

Figure 1 represents that case of patchwork, where the mode m is part of the transformation to the right. The residual mode probability, however, can also be transformed to the left instead to the right side. There exists even a third reasonable possibility: m is neither part of the reflection to the left nor part of the reflection to the right. The mode is then an individual event one has to take care of in the patchwork algorithm. We could even split the mode m into two fractional parts and subject the left part to the reflection σ_l and the right part to the reflection σ_r. But this would lead to non–integer bounds for the x–range affected by the two reflections which would make the matching sampling routine a lot more complicated. For comparison, consider our special case in Figure 1. There the x–range affected by the centre tranformation is bounded on the left by $2k_2 - m + 1$ and on the right by $2k_4 - m + 1$; both values are integers. A further disadvantage of the fractional split is that the probability of immediate acceptance (see Section 5) on the

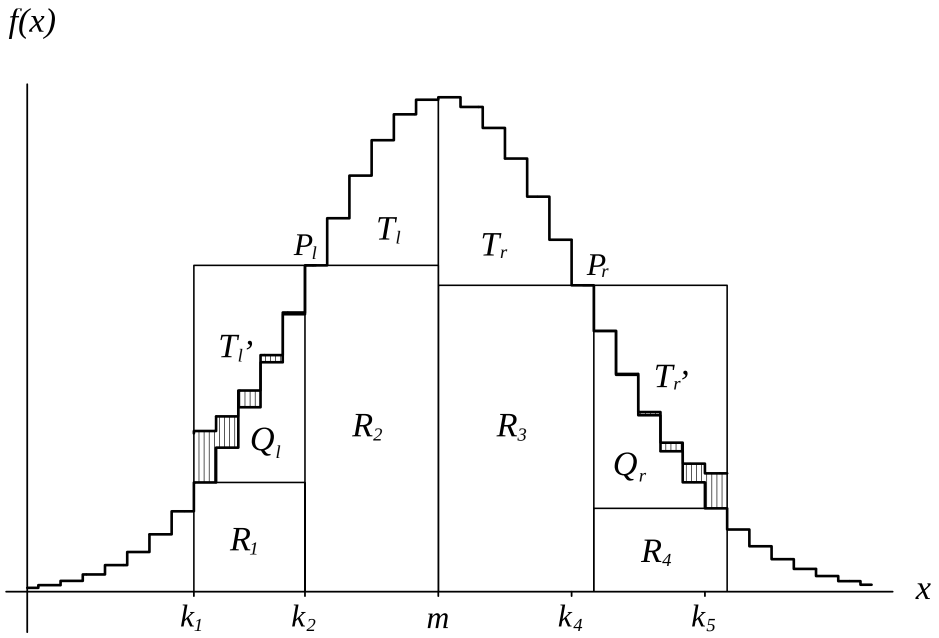

Figure 1: Transformation of the top region to the left and right hand side.

whole slightly decreases. So we decided to examine only the three cases mentioned at the beginning.

(a) Transform the residual mode to the left.

(b) Transform the residual mode to the right.

(c) Apply no reflection to the residual mode.

Introducing the two indicator quantities ${\delta_m}^{(l)}$ and ${\delta_m}^{(r)}$, we are able to discuss all three cases at once. Each of the two quantities may take the value 0, but only one of them may take the value 1. ${\delta_m}^{(l)} = 1$ indicates that the mode m is part of the reflection to the left. Analogously, ${\delta_m}^{(r)} = 1$ indicates that the mode m is part of the reflection to the right. For the three cases above we consequently have

(a) ${\delta_m}^{(l)} = 1,\ {\delta_m}^{(r)} = 0.$

(b) ${\delta_m}^{(l)} = 0,\ {\delta_m}^{(r)} = 1.$

(c) ${\delta_m}^{(l)} = 0,\ {\delta_m}^{(r)} = 0.$

Now, with ${\delta_m}^{(l)}$ and ${\delta_m}^{(r)}$, the reflections σ_l and σ_r can be stated in detail. In the following we will also deduce some necessary conditions for feasible reflection centres k_2 and k_4.

Let $k_2 \leq m - 1 + {\delta_m}^{(l)}$ and $k_4 \geq m + 1 - {\delta_m}^{(r)}$ be the centres of our reflections σ_l and σ_r. Define $k_1 = 2k_2 - (m - 1 + {\delta_m}^{(l)})$ and $k_5 = 2k_4 - (m + 1 - {\delta_m}^{(r)})$. The points above the line $y = f(k_2)$ in the left interval $[k_2 + \frac{1}{2},\, m + \delta_m{}^{(l)})$ are transformed to points below the line $y = f(k_2)$ in the adjacent interval $(k_1,\, k_2 + \frac{1}{2}]$ on the left by

$$\sigma_l : \ (x,\, y) \mapsto (v,\, w) = (2k_2 + 1 - x,\, 2f(k_2) - y)\,, \qquad k_2 + \tfrac{1}{2} \le x < m + \delta_m{}^{(l)}\,. \quad (3)$$

The points above the line $y = f(k_4)$ in the right interval $(m + 1 - \delta_m{}^{(r)},\, k_4 + \frac{1}{2}]$ are transformed to points below the line $y = f(k_4)$ in the adjacent interval $[k_4 + \frac{1}{2},\, k_5 + 1)$ on the right by

$$\sigma_r : \ (x,\, y) \mapsto (v,\, w) = (2k_4 + 1 - x,\, 2f(k_4) - y)\,, \quad m + 1 - \delta_m{}^{(r)} < x \le k_4 + \tfrac{1}{2}\,. \quad (4)$$

There is no overlapping of the transformed regions to the left of k_2 ($T_l{}'$ in Figure 1) and to the right of k_4 ($T_r{}'$ in Figure 1) — i. e. the residual centre probabilities — with the original areas below $f(x)$ — i. e. the original probabilities $p(j)$ — if and only if condition (5) and condition (6) hold:

$$p(k_2 + i) - p(k_2) \le p(k_2) - p(k_2 - i)\,, \quad 1 \le i \le m - k_2 - 1 + \delta_m{}^{(l)} \qquad (5)$$

$$p(k_4 - i) - p(k_4) \le p(k_4) - p(k_4 + i)\,, \quad 1 \le i \le k_4 - m - 1 + \delta_m{}^{(r)}\,. \qquad (6)$$

These conditions are very complex and therefore impracticable, unless there is a 'simple' relationship between $p(k-i)$, $p(k)$, and $p(k+i)$. Unfortunately, this is not the case for the most discrete distributions. Consequently we have to find some other conditions which are easy to handle. Obviously k_2 meets the condition (5) (k_4 meets the condition (6)), if the increase of the probabilities $p(i)$ right from k_2 in the interval $(k_2,\, m-1+\delta_m{}^{(l)}]$, compared to $p(i-1)$ (left from k_4 in the interval $[m+1-\delta_m{}^{(r)},\, k_4)$, compared to $p(i+1)$) is smaller than the decrease of the probabilities $p(j)$ left from k_2 related to $p(j+1)$ (right from k_4 related to $p(j-1)$). Comparing these differences of adjoining probabilities, having the same distance from the reflection centres k_2 and k_4, we obtain the so–called 'derivative conditions':

$$p(k_2 + i) - p(k_2 + i - 1) \le p(k_2 - i + 1) - p(k_2 - i)\,, \quad 1 \le i \le m - k_2 - 1 + \delta_m{}^{(l)} \quad (7)$$

$$p(k_4 - i) - p(k_4 - i + 1) \le p(k_4 + i - 1) - p(k_4 + i)\,, \quad 1 \le i \le k_4 - m - 1 + \delta_m{}^{(r)}\,. \ (8)$$

Contrary to (5) and (6), the conditions (7) and (8) are merely sufficient. Especially for $i = 1$, however, they coincide with the two necessary conditions.

Introducing $r(k) = p(k)/p(k-1)$, the so–called 'recurrence relationship' function for the probabilities $p(k)$, (7) and (8) can be further simplified to some handy conditions. Our final result which will be established in Appendix A is stated in Lemma 1 below.

Lemma 1

Let

$$p(k) = P(K = k)\,, \qquad k \in [k_{min},\, k_{max}]$$

be the probability function of a unimodal discrete random variable K with its mode at m, and let

$$r(k) = p(k)/p(k-1)\,, \quad k \in [k_{min} + 1,\, k_{max}] \qquad (9)$$

be the recurrence relationship function for the probabilities $p(k)$. Furthermore, let

$$N(k) = c_N(k)\,\Big(r(k+1)\,r(k) - 2r(k) + 1\Big) \tag{10}$$

be the 'necessary' function, and let

$$\begin{aligned} S(i,\,k) \;=\; c_S(i,\,k)\,\Big(&(r(k+i+1)-1)\,r(k+i) \\ &\times (r(k-i+1)-1)\,r(k-i) \;-\; (r(k+i)-1)\,(r(k-i)-1)\Big) \end{aligned} \tag{11}$$

be the 'sufficient' function, where $c_N(k) > 0$ and $c_S(i,\,k) > 0$ are two positive normalizing constants. Finally, let ${\delta_m}^{(l)}$ and ${\delta_m}^{(r)}$ be the two indicator quantities.

A preliminary range for the centre $k_2 \le m - 1 + {\delta_m}^{(l)}$ of the reflection σ_l is given by the interval $[{k_2}^{(min)},\, m-1+{\delta_m}^{(l)}]$, where ${k_2}^{(min)}$ is the smallest k_2 such that $N(k_2) \le 0$ for all $k_2 \ge {k_2}^{(min)}$, that is

$${k_2}^{(min)} = \min\{\,k_2\,|\,N(k) \le 0 \;\; \textit{for all} \;\; k \in [k_2,\, m-1+{\delta_m}^{(l)}]\,\}\,. \tag{12}$$

A preliminary range for the centre $k_4 \ge m + 1 - {\delta_m}^{(r)}$ of the reflection σ_r is given by the interval $[m+1-{\delta_m}^{(r)},\, {k_4}^{(max)}]$, where ${k_4}^{(max)}$ is the largest k_4 such that $N(k_4) \le 0$ for all $k_4 \le {k_4}^{(max)}$, that is

$${k_4}^{(max)} = \max\{\,k_4\,|\,N(k) \le 0 \;\; \textit{for all} \;\; k \in [m+1-{\delta_m}^{(r)},\, k_4]\,\}\,. \tag{13}$$

Candidates k_2 and k_4 from the above intervals are valid, whenever

$$k_2 \ge \frac{k_{min} + m - 1 + {\delta_m}^{(l)}}{2} \qquad \textit{and} \qquad S(i,\,k_2) \le 0, \quad 1 \le i \le m - k_2 - 2 + {\delta_m}^{(l)}\,, \tag{14}$$

$$k_4 \le \frac{k_{max} + m + 1 - {\delta_m}^{(r)}}{2} \qquad \textit{and} \qquad S(i,\,k_4) \ge 0, \quad 1 \le i \le k_4 - m - 2 + {\delta_m}^{(r)}\,. \tag{15}$$

In (12), (13), (14) and (15) only the sign of $N(k)$ and $S(i,\,k)$ is instrumental. Consequently the two constants $c_N(k)$ and $c_S(i,\,k)$ have no effect in the determination of feasible reflection centres k_2 and k_4 as long as they are positive. Nevertheless, they are not useless. Their task is to reduce $N(k)$ and $S(i,\,k)$ by such factors which do not change the sign of the two functions in the critical k– and i–range stated in (14) and (15). A good choice for $c_N(k)$ and $c_S(i,\,k)$ will lead to much simpler functions $N(k)$ and $S(i,\,k)$ as we shall see later in the application of our Lemma 1 to the Poisson/binomial and the hypergeometric distributions.

The lower bound for k_2 in (14) and the upper bound for k_4 in (15) guarantee that each transformed residual probability has a counterpart. In other words, $k_1 \ge k_{min}$ and $k_5 \le k_{max}$, and the transformed regions ${T_l}'$ and ${T_r}'$ are inside the domain of $f(x)$. Furthermore, all arguments of $p(k)$ in (5), (6), (7) and (8) (and all arguments of $r(k)$ in (10) and (11) with respect to the critical k– and i–range) are well–defined, i. e. are in the range

$[k_{min},\, k_{max}]$. Presuming that k_2 and k_4 satisfy our bounds, we need not pay attention to the special situation that $p(k)$ or $r(k)$ is undefined, due to $k \notin [k_{min},\, k_{max}]$.

It is easy to see that

$$p(k+1) - p(k) \le p(k) - p(k-1) \tag{16}$$

is equivalent to $N(k) \le 0$. Condition (16) which corresponds to the 'derivative conditions' (7) and (8), $i = 1$, is a test on the concavity of the polygon through the points $(k,\, p(k))$: If all k from an interval I meet (16), the polygon through $(k,\, p(k))$ will be concave in I. On the other hand, if each $k \in I$ does not satisfy (16), the polygon will be convex. From Lemma 1 now we know that $N(k) \le 0$ for all $k \in I_c = [{k_2}^{(min)},\, {k_4}^{(max)}]$, and the polygon through $(k,\, p(k))$ is concave in I_c. This particular property will be used later on for the determination of a simple lower bound $L(k)$ for $p(k)$. For the feasibility of $L(k)$, however, we have to presume that $N(k) \ge 0$ holds for two sufficiently large intervals adjacent to I_c.

No terms in $p(k)$, i. e. expressions with absolute probabilities, occur in (12), (13), (14) and (15). Consequently the range, taken up by the feasible reflection centres k_2 and k_4, is definitely determined by the following three factors.

- The domain of the probability function $p(k)$.
- The way how the mode m is handled.
- The recurrence relationship function $r(k)$.

We lay emphasis on the fact that ${k_2}^{(max)} = m - 1 + {\delta_m}^{(l)}$ and ${k_4}^{(min)} = m + 1 - {\delta_m}^{(r)}$ are always feasible reflection centres, provided that $k_{min} \le {k_2}^{(max)},\ {k_4}^{(min)} \le k_{max}$. Consequently any discrete and unimodal distribution with at least two mass points possesses two valid reflection centres. But notice that not a single cell of the histogram $f(x)$ is modified, i. e. no rearrangement of the probabilities $p(i)$ takes place, when $k_2 = {k_2}^{(max)}$ and $k_4 = {k_4}^{(min)}$ are the centres of the reflections σ_l and σ_r.

3 General Sampling Technique

Our sampling routine is based on the so–called hybrid rejection which is the most important rejection approach for discrete distributions. There the histogram function $f(x)$, a piecewise continuous density, must be dominated by a hat function $h(x)$, such that

$$\max\{f(x),\, \lim_{\epsilon\searrow 0} f(x-\epsilon)\} \;\le\; h(x), \qquad k_{min} \le x < k_{max}+1$$

or, for hat functions with no local minima, such that

$$f(k) \;\le\; \min\{h(k),\, h(k+1)\}, \qquad k \in [k_{min},\, k_{max}]\,.$$

Then acceptance–rejection in its general form can be carried out. This sampling principle goes back to J. v. Neumann [21]; it is based on the following fundamental theorem.

Theorem 1

Let $f(x)$ be a density on $\mathbb{R}$ and let $g(x)$ be a dominating density for $f(x)$, in other words $f(x) \le h(x) = \alpha g(x)$ for all x, $1 \le \alpha < \infty$.
Let $(X_i,\, U_i)$, $i = 1,\, 2,\, \ldots$ be a sequence of independent random vectors with independent X_i, U_i, where X_i has density $g(x)$ and U_i is uniformly distributed on $(0,\, 1)$.
Define

$$N = \min_i\{U_i\,\alpha g(X_i) \le f(X_i),\; f(X_i) > 0\}\,,$$

then

$$X = X_N \quad \textit{has density}\;\; f(x)$$

and

$$N \quad \textit{is geometrically distributed with parameter}\;\; \frac{1}{\alpha}\,,$$

i. e.

$$P(N=n) = \frac{1}{\alpha}\left(1-\frac{1}{\alpha}\right)^{n-1}, \quad n = 1,\, 2,\, \ldots\,; \qquad E(N) = \alpha, \;\; Var\,(N) = \alpha(\alpha-1)\,.$$

Proof of Theorem 1: see [23]. ■

The random variable $X = X_N$ in Theorem 1 has density $f(x)$. Therefore the truncated variable $K = \lfloor X_N \rfloor$, X_N a sample from the histogram $f(x)$, has distribution $p(k)$.

Instead of specifying a hat function for the original histogram $f(x)$, we first apply the two reflections σ_l, σ_r to $f(x)$ and determine a hat $h(x)$ for the rearranged histogram. In the sequel we will state a tight and also practicable $h(x)$. We will also outline the computation of candidates X from $h(x)$ and their acceptance or rejection. Finally we will deduce a simple lower bound for $p(k)$ which can be used for a fast pre–test on the acceptance–rejection condition. How the sampling routine works in detail will be given in the ensuing Section 4.

From Section 2 we know that the mode m and the integer abscissas k_2, k_4, k_1 and k_5 are of great importance in the sampling routine. k_2 and k_4, on the one hand, are the centres of the reflections σ_l, σ_r. k_1 and k_5+1, on the other hand, delimit the region affected by these reflections; they split up the area under $f(x)$ into three parts: the *left tail*, the *centre*, and the *right tail*. This suggests a sampling routine based on hybrid rejection with a three part envelope $h(x)$. Over $[k_1,\, k_5+1)$ the hat $h(x)$ is a two or three step staircase function, depending on whether the upper part of the mode cell is reflected or not. In the two step case the graph of $h(x)$ in the centre is the upper boundary of T_l', R_2, and R_3, T_r'. In the three step case the one step more is the upper boundary of the mode cell. If $p(k)$ is log–concave, i. e. $p^2(k) \geq p(k-1)p(k+1)$ for all k, the left tail of $f(x)$ over $[k_{min},\, k_1)$ and the right tail of $f(x)$ over $[k_5+1,\, k_{max}+1)$ can be majorized by exponential functions. For the few discrete distributions which do not fulfil the property of log–concavity, power functions could be appropriate envelopes for the tails. The details are given in subsections 3.1 and 3.2.

3.1 Sampling from the Centre

Let $f_m = f(m)$ and $f_i = f(k_i)$, $i = 1, 2, 4, 5$. The part $h^{(c)}(x)$ of the hat function $h(x)$ which covers the centre region — after the reflections at P_l and P_r (see Figure 1) — is defined as

$$h^{(c)}(x) = \begin{cases} f_2 = f(k_2) & k_1 \leq x < m + {\delta_m}^{(l)} \\ f_m = f(m) & m + {\delta_m}^{(l)} \leq x < m + 1 - {\delta_m}^{(r)} \\ f_4 = f(k_4) & m + 1 - {\delta_m}^{(r)} \leq x < k_5 + 1 \end{cases}$$

Notice that $h^{(c)}(x)$ is a three step staircase function, if and only if ${\delta_m}^{(l)} = {\delta_m}^{(r)} = 0$. Otherwise the middle step does not exist, and $h^{(c)}(x)$ reduces to a two step function. Samples from $h^{(c)}(x)$ can easily be generated by the so–called composition or probability mixing approach.

Immediate acceptance. Under the graph of $f(x)$ (and under the graph of $h^{(c)}(x)$) in Figure 1 we find four rectangles which are the 'immediate acceptance regions':

$$\begin{aligned} R_1 &= \{(x,\, y) \mid \quad k_1 \leq x < k_2, \quad 0 \leq y \leq f_1\}, \\ R_2 &= \{(x,\, y) \mid \quad k_2 \leq x < m + {\delta_m}^{(l)}, \quad 0 \leq y \leq f_2\}, \\ R_3 &= \{(x,\, y) \mid m + 1 - {\delta_m}^{(r)} \leq x < k_4 + 1, \quad 0 \leq y \leq f_4\}, \\ R_4 &= \{(x,\, y) \mid \quad k_4 + 1 \leq x < k_5 + 1, \quad 0 \leq y \leq f_5\}. \end{aligned}$$

If ${\delta_m}^{(l)} = {\delta_m}^{(r)} = 0$, we will even have a fifth immediate acceptance region which corresponds to the mode probability:

$$R_0 = \{(x,\, y) \mid m + {\delta_m}^{(l)} \leq x < m + 1 - {\delta_m}^{(r)},\; 0 \leq y \leq f_m\}.$$

The sampling from $\bigcup R_i$ requires only one uniform deviate U. This uniform is used at first for selecting one of the four/five rectangles with the correct probability, and is then transformed to a sample X from the chosen rectangle. Finally, the truncated quantity $K = \lfloor X \rfloor$ is accepted immediately (without executing any acceptance–test) as a sample from $p(k)$ (see Algorithm GPRS, Step 1, Step 2 and Step 4).

Acceptance–Rejection. Let (X, Y) be a point under $h^{(c)}(x)$ and outside $\bigcup R_i$. Then $X \in [k_1, k_2)$ or $X \in [k_4+1, k_5+1)$. If $Y \le f(X)$, then the point (X, Y) will be either in Q_l or Q_r (in Figure 1), and $K = \lfloor X \rfloor$ can be accepted as a $p(k)$ variate. Otherwise (X, Y) may belong to $T_l{}'$ or $T_r{}'$ in which case the acceptance test applies to the corresponding point (V, W) in T_l or T_r. (V, W) is obviously determined by the inverse of the reflection σ_l or σ_r. Since $\sigma_l{}^{-1} = \sigma_l$ and $\sigma_r{}^{-1} = \sigma_r$, the back–transformation can be copied from (3) and (4):

$$(X, Y) \mapsto (V, W) = (2k_2 + 1 - X,\ 2f_2 - Y), \qquad k_1 < X < k_2 \tag{17}$$

$$(X, Y) \mapsto (V, W) = (2k_4 + 1 - X,\ 2f_4 - Y), \qquad k_4 + 1 < X < k_5 + 1\,. \tag{18}$$

Now, if $W \le f(V)$, $\widehat{K} = \lfloor V \rfloor$ will be accepted as a $p(k)$ variate. Otherwise the point (X, Y) is in the shaded area of Figure 1, rejection is indicated, and a new sample X from $h(x)$ is required.

3.2 Sampling from the Tails

We suggest to use envelopes for the tails which touch the histogram $f(x)$ at k_1, $k_1 - 1$ and at $k_5 + 1$, $k_5 + 2$, as the function values $f_1 = f(k_1)$ and $f_5 = f(k_5)$ are already available. This requirement formally reads

$$h^{(l)}(k_1) = f_1,\ h^{(l)}(k_1 - 1) = f(k_1 - 1), \quad h^{(r)}(k_5 + 1) = f_5,\ h^{(r)}(k_5 + 2) = f(k_5 + 1), \tag{19}$$

where $h^{(l)}(x)$ $(h^{(r)}(x))$ is that part of $h(x)$ which covers the left (right) tail of $f(x)$. Here notice that $f(k_1 - 1) = f_1 / r(k_1)$ and $f(k_5 + 1) = f_5 * r(k_5 + 1)$; so tail envelopes satisfying the condition (19) do not burden our sampling routine with the explicite calculation of further function values. In fact, only the computation of their shape parameters (which naturally depend on the function values of $f(x)$ at the points of contact) entails some expense. Besides, the above recurrence terms apply not only to the tail envelopes, but also to the lower squeeze function $L(k) \le p(k)$ (see Subsection 3.3). The tails of a log–concave distribution $p(k)$ can always be covered by exponential functions which meet the requirement (19). A more general condition for the existence of exponential tail envelopes is stated in Lemma 2 below.

Lemma 2

Let $f(x)$ be the histogram function of the distribution $p(k)$, $k \in [k_{min}, k_{max}]$ and k integer. Furthermore, let $r(k) = p(k)/p(k-1)$.

(i) The left tail of $f(x)$ over $[k_{min}, k_1)$ will be majorized by the exponential function

$$h_e{}^{(l)}(x) = f_1\, e^{\lambda_l\,(x - k_1)}\,, \qquad x < k_1, \qquad \lambda_l = -\log(f(k_1 - 1)/f_1)\,,$$

if $r(k) \ge r(k_1)$ for all $k \in [k_{min} + 1, k_1]$.

(ii) The right tail of $f(x)$ over $[k_5+1,\, k_{max}+1)$ will be majorized by the exponential function

$$h_e{}^{(r)}(x) = f_5\, e^{\lambda_r\,(k_5+1-x)}\,, \qquad x \ge k_5+1, \qquad \lambda_r = -\log(f(k_5+1)/f_5)\,,$$

if $r(k) \le r(k_5+1)$ for all $k \in [k_5+1,\, k_{max}]$.

Both $h_e{}^{(l)}(x)$ and $h_e{}^{(r)}(x)$ meet the requirement (19).

Proof of Lemma 2 :

Since $h_e{}^{(l)}(k_1) = f_1$ and $h_e{}^{(l)}(k_1-1) = f_1\, e^{-\lambda_l} = f(k_1-1)$, $h_e{}^{(l)}(x)$ satisfies (19). That $h_e{}^{(l)}(x)$ also majorizes $f(x)$,

$$h_e{}^{(l)}(k_1-i) = f_1\Big(e^{-\lambda_l}\Big)^i = f(k_1)\Big(\frac{f(k_1-1)}{f(k_1)}\Big)^i \ge f(k_1-i)$$

or

$$\Big(\frac{f(k_1-1)}{f(k_1)}\Big)^i \ge \frac{f(k_1-i)}{f(k_1)} = \frac{f(k_1-i)}{f(k_1-i+1)}\,\frac{f(k_1-i+1)}{f(k_1-i+2)}\cdots\frac{f(k_1-1)}{f(k_1)}$$

must hold for all $1 \le i \le k_1 - k_{min}$. From the latter inequality we conclude that $h_e{}^{(l)}(x)$ will be an envelope, if

$$\frac{f(k_1-1)}{f(k_1)} \ge \frac{f(k_1-i)}{f(k_1-i+1)} \qquad \text{for all} \quad 1 \le i \le k_1 - k_{min}\,.$$

But this is equivalent to $r(k_1-i+1) \ge r(k_1)$ for all $1 \le i \le k_1 - k_{min}$. This proves our assertion in (i). In an analogous way the statement in (ii) can be established. ■

The probability function $p(k)$ is log–concave, if $p^2(k) \ge p(k-1)p(k+1)$ for all k. But this condition is equivalent to $r(k) \ge r(k+1)$ or, in other words, to $r(k)$ being a monotonically decreasing function on the integers $k \in [k_{min}+1,\, k_{max}]$. Exponential tail envelopes, however, already exist in case $r(k) \ge r(k_1)$ for all $k \le k_1$ and $r(k_5+1) \ge r(k)$ for all $k \ge k_5+1$, due to Lemma 2.

The areas of $h_e{}^{(l)}(x)$ and $h_e{}^{(r)}(x)$ are

$$H_e{}^{(l)} = \int_{-\infty}^{k_1} h_e{}^{(l)}(x)\,dx = f_1/\lambda_l\,, \qquad H_e{}^{(r)} = \int_{k_5+1}^{\infty} h_e{}^{(r)}(x)\,dx = f_5/\lambda_r\,.$$

Samples X from $h_e{}^{(l)}(x)$ and $h_e{}^{(r)}(x)$ are obtained from the corresponding distribution functions $G_e{}^{(l)}(x) = e^{\lambda_l(x-k_1)}$ and $G_e{}^{(r)}(x) = 1 - e^{\lambda_r(k_5+1-x)}$ by inversion :

$$X = k_1 + \log(U)/\lambda_l \quad \text{and} \quad X = k_5 + 1 - \log(1-U)/\lambda_r\,, \qquad U \sim U(0,\,1)\,.$$

If exponential functions are not appropriate, we could employ power functions as tail envelopes instead. Their general form is

$$h_{\rho_l}{}^{(l)}(x) = f_1(1 + c_l(k_1-x))^{-\rho_l}, \qquad 1 < \rho_l \le \rho_l{}^{(max)}\,, \quad c_l = \left(\frac{f_1}{f(k_1-1)}\right)^{1/\rho_l} - 1\,,$$

$$h_{\rho_r}{}^{(r)}(x) = f_5(1 + c_r(x-k_5-1))^{-\rho_r}, \quad 1 < \rho_r \le \rho_r{}^{(max)}\,, \quad c_r = \left(\frac{f_5}{f(k_5+1)}\right)^{1/\rho_r} - 1\,.$$

Obviously $h_{\rho_l}{}^{(l)}(x)$ and $h_{\rho_r}{}^{(r)}(x)$ meet the requirement (19).

From the above class the functions with maximum power are the tightest envelopes. But the difficulty here is to find $\rho_l{}^{(max)}$ and $\rho_r{}^{(max)}$. To avoid this problem, we confine ourselves to the two special functions

$$h_2{}^{(l)}(x) = f_1(1 + c_l(k_1 - x))^{-2}, \qquad c_l = \sqrt{f_1/f(k_1-1)} - 1\,,$$
$$h_2{}^{(r)}(x) = f_5(1 + c_r(x - k_5 - 1))^{-2}, \qquad c_r = \sqrt{f_5/f(k_5+1)} - 1\,,$$

which are feasible whenever $f(x)$ has sub–quadratic tails.

The areas of $h_2{}^{(l)}(x)$ and $h_2{}^{(r)}(x)$ are

$$H_2{}^{(l)} = \int_{-\infty}^{k_1} h_2{}^{(l)}(x)\,dx = f_1/c_l\,, \qquad H_2{}^{(r)} = \int_{k_5+1}^{\infty} h_2{}^{(r)}(x)\,dx = f_5/c_r\,.$$

Samples X from $h_2{}^{(l)}(x)$ and $h_2{}^{(r)}(x)$ are obtained as well from the corresponding distribution functions $G_2{}^{(l)}(x) = (1 + c_l(k_1 - x))^{-1}$ and $G_2{}^{(r)}(x) = 1 - (1 + c_r(x - k_5 - 1))^{-1}$ by inversion:

$$X = k_1 - \frac{1-U}{c_l\,U} \quad \text{and} \quad X = k_5 + 1 + \frac{U}{c_r(1-U)}\,, \qquad U \sim U(0,\,1)\,.$$

Samples from the tails of $p(k)$ are generated by acceptance–rejection of candidates X from $h^{(l)}(x)$ and $h^{(r)}(x)$ as follows.

First of all, let $(X,\,Y)$ be uniformly distributed under $h^{(l)}(x)$ or $h^{(r)}(x)$, i. e $X \sim h^{(l)}(x)$ and $Y \sim U(0,\,h^{(l)}(X))$ or $X \sim h^{(r)}(x)$ and $Y \sim U(0,\,h^{(r)}(X))$. Then if $Y \le f(X)$, X will be accepted as a sample from $f(x)$, and $K = \lfloor X \rfloor$ will be a sample from $p(k)$. Otherwise X is rejected and a new sample X from $h(x)$ is required.

3.3 Squeeze Functions

Since the acceptance tests $W \le f(V)$ and $Y \le f(X)$ contain the, in most cases, complicated function $f(x)$, fast pre–tests based on simple lower bounds for $f(x)$ (called squeezes) will be added to our basic sampling routine.

First of all, we will determine a piecewise linear function $\tilde{L}(x)$ which minorizes the histogram function $f(x)$. $\tilde{L}(x)$ will be very close to $f(x)$, because it profits by the known function values $f_m = f(m)$ and $f_i = f(k_i)$, $i = 1,\,2,\,4,\,5$. $\tilde{L}(x)$ and the corresponding staircase function

$$L(x) = \begin{cases} \tilde{L}(\lfloor x \rfloor + 1) & k_{min} \le x < m \\ \tilde{L}(\lfloor x \rfloor) & m \le x < k_{max} + 1\,, \end{cases}$$

which will be used in the algorithm because of its higher efficiency, are obtained as follows.

Let $I_c = [k_2{}^{(min)},\, k_4{}^{(max)}]$, where $k_2{}^{(min)}$ and $k_4{}^{(max)}$ are as defined in Lemma 1. Then $N(k) \le 0$ for all $k \in I_c$. Now assume that $N(k) \ge 0$ for all $k \notin I_c$ (that our $\tilde{L}(x)$

minorizes $f(x)$, $N(k) \geq 0$ must hold for two sufficiently large intervals adjacent to I_c only). Then the polygon through $(k, p(k))$ is concave in and convex outside I_c. (The polygon is concave even in I_c enlarged by the length 1 on both sides.)

To shorten the following considerations, we introduce the two quantities $k_l \leq k_2{}^{(min)}$ and $k_r \geq k_4{}^{(max)}$. From the above it is clear that the line through the two adjacent points $(k_l, f(k_l-1))$, $(k_l+1, f(k_l))$ is below $f(x)$ for $x < k_l$, and the line through $(k_r, f(k_r))$, $(k_r+1, f(k_r+1))$ is below $f(x)$ for $x \geq k_r+1$. In the special case that $k_2 = k_2{}^{(min)}$ and $k_4 = k_4{}^{(max)}$, $\tilde{L}(x)$ outside the interval $[k_2, k_4+1)$ can be made up consequently of four segments lying on such lines.

We can always determine feasible reflection centres k_2 and k_4, such that $k_1 \leq k_2{}^{(min)}$ and $k_5 \geq k_4{}^{(max)}$ (due to the concavity of the polygon through $(k, p(k))$ in I_c), and in this case the lines resulting from $k_l = k_1$ and $k_r = k_5$ form a lower bound for $f(x)$ on $[k_{min}, k_1)$ and $[k_5+1, k_{max}+1)$, respectively. Unfortunately we cannot presume that $k_2 = k_2{}^{(min)}$ and that $k_4 = k_4{}^{(max)}$, and the half lines suggested above (with end points $(k_l, f(k_l-1))$ and $(k_r+1, f(k_r+1)))$ need not to be entirely below $f(x)$ for $k_l = k_2$ and $k_r = k_4$. In the following, however, we will eliminate this defect by deducing a lower bound for $f(x)$ on the intervals $[k_1, k_2)$ and $[k_4+1, k_5+1)$ which is universally valid.

Define

$$z_2 = k_2{}^{(min)} - \frac{r(k_2{}^{(min)})}{r(k_2{}^{(min)})-1}\,, \qquad z_4 = k_4{}^{(max)} + \frac{1}{1-r(k_4{}^{(max)})}\,.$$

$(z_2+1, 0)$ is the point at which the line through $(k_2{}^{(min)}, f(k_2{}^{(min)}-1))$ and $(k_2{}^{(min)}+1, f(k_2{}^{(min)}))$ crosses the x–axis. Similarly, $(z_4, 0)$ is the point at which the line through $(k_4{}^{(max)}, f(k_4{}^{(max)}))$ and $(k_4{}^{(max)}+1, f(k_4{}^{(max)}+1))$ crosses the x–axis. Obviously the line segments with end points $(z_2+1, 0)$, (k_2+1, f_2) and (k_4, f_4), $(z_4, 0)$ are below $f(x)$. This immediately follows from $k_2{}^{(min)} \leq k_2$, $k_4 \leq k_4{}^{(max)}$, a necessary condition for the feasibility of k_2 and k_4, and from the concavity of the polygon through $(k, p(k))$ in $[k_2{}^{(min)}-1, k_4{}^{(max)}+1]$.

Inside $[k_2, k_4+1)$ the line segments with end points (k_2+1, f_2), $(m+1, f_m)$ and (m, f_m), (k_4, f_4) are always below $f(x)$, due to the concavity of $(k, p(k))$ in $[k_2, k_4]$.

For each of the six regions acceptance–rejection applies we have found a line segment close below $f(x)$. Putting them together yields the piecewise continuous bound $\tilde{L}(x)$. Shifting the three segments left from m by 1 to the left, and using $\lfloor x \rfloor$ instead of x as argument leads to $L(x)$. For integer arguments k the lower bound $L(x)$ now reads

$$L(k) = \begin{cases} f_1 - (k_1 - k)\,(f_1 - f(k_1-1)), & k_{min} \leq k < k_1 \\ \left\{\begin{array}{ll} f_2 - (k_2 - k)\,(f_2 - f(k_2-1)) & \text{if } k_2 = k_2{}^{(min)} \\ f_2\,(k - z_2)/(k_2 - z_2) & \text{if } k_2 > k_2{}^{(min)} \end{array}\right\} & k_1 \leq k < k_2 \\ f_2 + (k - k_2)\,(f_m - f_2)/(m - k_2), & k_2 \leq k \leq m \\ f_4 + (k_4 - k)\,(f_m - f_4)/(k_4 - m), & m \leq k < k_4 + 1 \\ \left\{\begin{array}{ll} f_4 - (k - k_4)\,(f_4 - f(k_4+1)) & \text{if } k_4 = k_4{}^{(max)} \\ f_4\,(z_4 - k)/(z_4 - k_4) & \text{if } k_4 < k_4{}^{(max)} \end{array}\right\} & k_4 + 1 \leq k < k_5 + 1 \\ f_5 - (k - k_5)\,(f_5 - f(k_5+1)), & k_5 + 1 \leq k \leq k_{max}\,. \end{cases}$$

If $N(k) \geq 0$ holds for two sufficiently large intervals adjacent to I_c, $L(x)$ will be a lower bound for $f(x)$. Otherwise we have to check whether the four segments outside $[k_2, k_4+1)$ are below $f(x)$ or not. If a segment fails to be under $f(x)$, no quick acceptance will exist for the corresponding region, and in the algorithm the squeeze test has to be omitted. Our general algorithm in Section 4, however, takes $L(x) \leq f(x)$ for granted.

Acceptance–rejection of samples $X \in [k_1, k_2)$ and $X \in [k_4+1, k_5+1)$ is modified by $L(x)$ as follows. First of all the fast pre–test $Y \leq L(X)$ is executed. Whenever it fails, (X, Y) is back–transformed to (V, W). If $W \geq f_m$, then V will be quickly rejected, since $f(x) \leq f_m$. Otherwise the fast pre–test $W \leq L(V)$ and, if necessary, the slow acceptance test $W \leq f(V)$ is carried out. If one of the two tests scores a success, $\widehat{K} = \lfloor V \rfloor$ will be accepted as a sample from $p(k)$. On the other hand, if both tests fail, i. e. if V is rejected, the sample (X, Y) will be subjected to the test $Y \leq f(X)$ which finally decides on acceptance or rejection of $K = \lfloor X \rfloor$.

Acceptance–rejection of samples $X \in [k_{min}, k_1)$ and $X \in [k_5+1, k_{max}+1)$ is also complemented by the fast pre–test $Y \leq L(X)$.

3.4 Some Additional Remarks

The continuous sample X, and also the continuous sample V, will not be generated explicitly in the final version of our algorithm. Instead of X the random distance D from $K = \lfloor X \rfloor$ to one of the four special points k_i, $i = 1, 2, 4, 5$, is calculated. The integer abscissa k_i D refers to thereby depends on the range we have in view to sample from. For the immediate acceptance regions left from m, for example, we calculate the distance D from K to their left boundary, and, proceeding in this manner, $K = k_1 + D$ or $K = k_2 + D$. For the remaining centre region left from m (or, more precisely, left from k_2) we calculate the distance D from K to the reflection centre k_2, and here $K = k_2 - D$. Since K (left from k_2) and its counterpart $\widehat{K}$ (right from k_2) are considered to be equidistant from k_2, we have to set $\widehat{K} = k_2 + D$. For the left tail we calculate the distance D from K to k_1, and here $K = k_1 - D$. In a similar way the regions right from m are handled. No continuous sample will therefore appear in the algorithm, so the histogram function can be replaced by its discrete pendant $f(k) = p(k)$. This again modifies the style of the acceptance–rejection tests which now read $Y \leq L(K)$, $Y \leq f(K)$, and $W \leq L(\widehat{K})$, $W \leq L(\widehat{K})$.

Obviously we can replace $f(k)$ by any rescaled function $\hat{f}(k) = c\, f(k)$, $c > 0$. This substitution has in principle no effect on the sampling method and on our general algorithm in Section 4. But a good choice for c could save considerable time in the initialization step. For example, setting $c = 1/p(m)$ leads to $\hat{f}(k) = p(k)/p(m) \leq \hat{f}(m) = 1$ for all $k \in [k_{min}, k_{max}]$. The constant f_m can be replaced in the algorithm by the explicite value 1, thereby lessening the number of variables by one. Furthermore, $\hat{f}(k)$ does not contain a normalizing constant, i. e. a factor which does not depend on k.

4 General Algorithm GPRS

The algorithm GPRS (**G**eneral **P**atchwork **R**ejection method, with lower **S**queeze function) results directly from the theory in Section 2 and Section 3. The shorter version GPRB (**G**eneral **P**atchwork **R**ejection algorithm, **B**asic method) without squeezes is obtained from GPRS by simply removing all the lines that are marked with a '$-$' sign.

Algorithm GPRS:

$f(k) = c * p(k), \quad r(k) = p(k)/p(k-1)$

Initialization:

Let m be the mode.
Let k_2 and k_4 be two feasible reflection centres.
Define ${\delta_m}^{(l)}$ and ${\delta_m}^{(r)}$ in an appropriate way.
Set $d_l = m - k_2 - \left(1 - {\delta_m}^{(l)}\right)$, $d_r = k_4 - m - \left(1 - {\delta_m}^{(r)}\right)$;
Set $k_1 = k_2 - d_l$, $k_5 = k_4 + d_r$;
$-$ If $\left(k_2 > {k_2}^{(min)}\right)$ then Set $z_2 = {k_2}^{(min)} - r({k_2}^{(min)})/\left(r({k_2}^{(min)}) - 1\right)$;
$-$ If $\left(k_4 < {k_4}^{(max)}\right)$ then Set $z_4 = {k_4}^{(max)} + 1/\left(1 - r({k_4}^{(max)} + 1)\right)$;
Set $f_m = f(m)$;
Set $f_2 = f(k_2)$, $f_1 = f(k_1)$, $f_4 = f(k_4)$, $f_5 = f(k_5)$;
Set $p_0 = f_m * \left(1 - ({\delta_m}^{(l)} + {\delta_m}^{(r)})\right)$;
Set $p_1 = f_2 * (d_l + 1) + p_0$, $p_2 = f_2 * d_l + p_1$;
Set $p_3 = f_4 * (d_r + 1) + p_2$, $p_4 = f_4 * d_r + p_3$;
If $(d_l > 0)$ then Set $r_2 = r(k_2)$;
If $(k_1 > k_{min})$ then
    Set $r_1 = r(k_1)$;   be sure that $1 < r_1 < \infty$
    (a) $h^{(l)}(x) = {h_e}^{(l)}(x)$:
        Set $\lambda_l = \log(r_1)$, $p_5 = f_1/\lambda_l + p_4$;
    (b) $h^{(l)}(x) = {h_2}^{(l)}(x)$:
        Set $c_l = \sqrt{r_1} - 1$, $p_5 = f_1/c_l + p_4$;
else
    Set $p_5 = p_4$;
Endif
If $(d_r > 0)$ then Set $r_4 = r(k_4 + 1)$;
If $(k_5 < k_{max})$ then
    Set $r_5 = r(k_5 + 1)$;   be sure that $0 < r_5 < 1$
    (a) $h^{(r)}(x) = {h_e}^{(r)}(x)$:
        Set $\lambda_r = -\log(r_5)$, $p_6 = f_5/\lambda_r + p_5$;
    (b) $h^{(r)}(x) = {h_2}^{(r)}(x)$:

```
        Set  c_r = 1/√r_5 − 1,  p_6 = f_5/c_r + p_5;
else
    Set  p_6 = p_5;
Endif
```

Generation:

1. Case Distinction:

```
Generate U ~ U(0, 1) and Set U = U * p_6;            0 < U < p_6
If (U < p_0) then Deliver K = m;
If (U < p_2) then Goto 2. [Left Centre Region];
              else
if (U < p_4) then Goto 4. [Right Centre Region];
              else
if (U < p_5) then Goto 6. [Left Tail];
              else Goto 7. [Right Tail];
Endif
```

2. Left Centre Region — Immediate acceptance of candidate K, $k_1 \le K \le m-1+\delta_m{}^{(l)}$:

```
Set V = U − p_1;                                 V ~ U(−f_2(d_l + 1), +f_2 d_l)
If (V < 0) then Deliver K = k_2 + ⌊U/f_2⌋;       k_2 ≤ K ≤ m − 1 + δ_m^(l)
Set Y = V/d_l;                                   Y ~ U(0, f_2)
If (Y < f_1) then Deliver K = k_1 + ⌊V/f_1⌋;     k_1 ≤ K ≤ k_2 − 1
```

3. Left Centre Region — Squeeze acceptance of candidate K, $k_1 \le K \le k_2 - 1$, or Acceptance/Rejection of its counterpart $\widehat{K}$, $k_2+1 \le \widehat{K} \le m-1+\delta_m{}^{(l)}$:

```
  Generate V ~ U(0, 1) and Set D = ⌊V * d_l⌋ + 1;    1 ≤ D ≤ d_l = k_2 − k_1
  Set K = k_2 − D;
– If (k_2 = k_2^(min)) then
–     If (Y ≤ f_2 − D * (f_2 − f_2/r_2)) then        Y ≤ L(K)
–         Deliver K                                  Quick acceptance of K
–     Endif
– else
–     If (Y ≤ f_2 * (K − z_2)/(k_2 − z_2)) then      Y ≤ L(K)
–         Deliver K                                  Quick acceptance of K
–     Endif
– Endif
  Set W = f_2 + f_2 − Y;                             W ~ U(f_2, 2f_2 − f_1)
  If (W < f_m) then
      Set K^ = k_2 + D;
–     If (W ≤ f_2 + D * (f_m − f_2)/(m − k_2)) then  W ≤ L(K^)
–         Deliver K^;                                Quick acceptance of K^
–     Endif
      If (W ≤ f(K^)) then Deliver K^;                W ≤ f(K^)
```

`Endif` — Final acceptance of $\widehat{K}$

`Goto 8.` [Acceptance Test]; — Rejection of $\widehat{K}$

4. Right Centre Region — Immediate acceptance of candidate K, $m+1-\delta_m{}^{(r)} \le K \le k_5$:

`Set` $V = U - p_3$;	$V \sim U(-f_4(d_r+1), +f_4 d_r)$
`If` $(V < 0)$ `then Deliver` $K = k_4 - \lfloor (U-p_2)/f_4 \rfloor$;	$m+1-\delta_m{}^{(r)} \le K \le k_4$
`Set` $Y = V/d_r$;	$Y \sim U(0,\, f_4)$
`If` $(Y < f_5)$ `then Deliver` $K = k_5 - \lfloor V/f_5 \rfloor$;	$k_4+1 \le K \le k_5$

5. Right Centre Region — Squeeze acceptance of candidate K, $k_4+1 \le K \le k_5$, or Acceptance/Rejection of its counterpart $\widehat{K}$, $m+1-\delta_m{}^{(r)} \le \widehat{K} \le k_4-1$:

`Generate` $V \sim U(0,1)$ `and Set` $D = \lfloor V * d_r \rfloor + 1$;	$1 \le D \le d_r = k_5 - k_4$
`Set` $K = k_4 + D$;	
– If $\left(k_4 = k_4{}^{(max)}\right)$ then	
– `If` $(Y \le f_4 - D*(f_4 - f_4 * r_4))$ `then`	$Y \le L(K)$
– `Deliver` K	Quick acceptance of K
– `Endif`	
– else	
– `If` $(Y \le f_4 * (z_4 - K)/(z_4 - k_4))$ `then`	$Y \le L(K)$
– `Deliver` K	Quick acceptance of K
– `Endif`	
– Endif	
`Set` $W = f_4 + f_4 - Y$;	$W \sim U(f_4,\, 2f_4 - f_5)$
`If` $(W < f_m)$ `then`	
`Set` $\widehat{K} = k_4 - D$;	
– `If` $(W \le f_4 + D*(f_m - f_4)/(k_4 - m))$ `then`	$W \le L(\widehat{K})$
– `Deliver` $\widehat{K}$;	Quick acceptance of $\widehat{K}$
– `Endif`	
`If` $(W \le f(\widehat{K}))$ `then Deliver` $\widehat{K}$;	$W \le f(\widehat{K})$
`Endif`	Final acceptance of $\widehat{K}$
`Goto 8.` [Acceptance Test];	Rejection of $\widehat{K}$

6. Left Tail — Squeeze acceptance of candidate K, $k_{min} \le K \le k_1 - 1$:

`Generate` $V \sim U(0,1)$;	$V \equiv G^{(l)}(X)$
(a) $h^{(l)}(x) = h_e{}^{(l)}(x)$:	
`Set` $D = \lfloor 1 - \log(V)/\lambda_l \rfloor$;	
(b) $h^{(l)}(x) = h_2{}^{(l)}(x)$:	
`Set` $D = \lfloor 1 + (1-V)/(c_l * V) \rfloor$;	
`Set` $K = k_1 - D$;	
`If` $(K < k_{min})$ `then Goto 1.` [Continue];	Fast rejection of $K < k_{min}$
(a) $h^{(l)}(x) = h_e{}^{(l)}(x)$:	$h_e{}^{(l)}(X) = f_1\left(G_e{}^{(l)}(X)\right)$

`Set` $Y = (U - p_4) * \lambda_l * V$; $Y \sim U\left(0,\, h_e{}^{(l)}(X)\right)$

(b) $h^{(l)}(x) = h_2{}^{(l)}(x)$: $h_2{}^{(l)}(X) = f_1\left(G_2{}^{(l)}(X)\right)^2$

`Set` $Y = (U - p_4) * c_l * V * V$; $Y \sim U\left(0,\, h_2{}^{(l)}(X)\right)$

- `If` $(Y \leq f_1 - D * (f_1 - f_1/r_1))$ `then Deliver` K; $Y \leq L(K)$
  `Goto 8.` [Acceptance Test];

7. Right Tail — Squeeze acceptance of candidate $K,\ k_5 + 1 \leq K \leq k_{max}$:

   `Generate` $V \sim U(0,1)$; $V \equiv 1 - G^{(r)}(X)$

   (a) $h^{(r)}(x) = h_e{}^{(r)}(x)$:
   `Set` $D = \lfloor 1 - \log(V)/\lambda_r \rfloor$;

   (b) $h^{(r)}(x) = h_2{}^{(r)}(x)$:
   `Set` $D = \lfloor 1 + (1 - V)/(c_r * V) \rfloor$;

   `Set` $K = k_5 + D$;
   `If` $(K > k_{max})$ `then Goto 1.` [Continue]; Fast rejection of $K > k_{max}$

   (a) $h^{(r)}(x) = h_e{}^{(r)}(x)$: $h_e{}^{(r)}(X) = f_5\left(1 - G_e{}^{(r)}(X)\right)$

   `Set` $Y = (U - p_5) * \lambda_r * V$; $Y \sim U\left(0,\, h_e{}^{(r)}(X)\right)$

   (b) $h^{(r)}(x) = h_2{}^{(r)}(x)$: $h_2{}^{(r)}(X) = f_5\left(1 - G_2{}^{(r)}(X)\right)^2$

   `Set` $Y = (U - p_5) * c_r * V * V$; $Y \sim U\left(0,\, h_2{}^{(r)}(X)\right)$

   - `If` $(Y \leq f_5 - D * (f_5 - f_5 * r_5))$ `then Deliver` K; $Y \leq L(K)$
     `Goto 8.` [Acceptance Test];

8. Acceptance Test — Acceptance of K, whenever $Y \leq f(K),\ Y \sim U(0, h(X))$:

   `If` $(Y \leq f(K))$ `then Deliver` K;
   `else Goto 1.` [Continue]; Rejection of K

Notice that the constant r_1, and also the constant r_5, must not take the value 1, for otherwise the areas of the tail envelopes suggested in Section 3 become infinitely large. We even recommend to use such reflection centres k_2 and k_4 and to choose that mode transformation that guarantee that r_1 and r_5 differ widely from 1. The more they differ from 1, the steeper are the corresponding envelopes and the tighter is their fit.

5 Characteristics of Algorithm GPRS

The most important performance measures of an algorithm based on the acceptance rejection method are the average number α of samples X from $h(x)$ and the average number N of uniforms U needed to generate one sample from $f(x)$. Of great importance is also the probability $P(I)$ of accepting X immediately (without carrying out any acceptance test), given that X is accepted. The probability $P(S)$ of accepting X quickly by means of a lower squeeze function $L(x)$, given that X is accepted, and the probability of rejecting X quickly by means of an upper squeeze function, given that X is rejected, on the contrary, are not so important. In the following we will specify the quantities α, N, $P(I)$ and $P(S)$ for our algorithm GPRS.

Let $p(k)$ be the probability function, and let $f(x)$ be its corresponding histogram function. Furthermore, let $f_m = f(m)$ and $f_i = f(k_i)$, $i = 1, 2, 4, 5$. The average number α of samples X from $h(x)$ needed to generate one sample K from $p(k)$ is

$$\begin{aligned} \alpha \;=\; f_m(1 - {\delta_m}^{(l)} - {\delta_m}^{(r)}) \;+\; f_2(m - k_1 + {\delta_m}^{(l)}) \;+\; H^{(l)} \\ +\; f_4(k_5 - m + {\delta_m}^{(r)}) \;+\; H^{(r)}\,, \end{aligned} \tag{20}$$

where

$$\begin{aligned} H^{(l)} \;&=\; {H_e}^{(l)} \;=\; f_1/\lambda_l \quad \text{or} \quad H^{(l)} \;=\; {H_2}^{(l)} \;=\; f_1/c_l \\ H^{(r)} \;&=\; {H_e}^{(r)} \;=\; f_5/\lambda_r \quad \text{or} \quad H^{(r)} \;=\; {H_2}^{(r)} \;=\; f_5/c_r\,, \end{aligned}$$

according to the envelopes used for the tails. The probability $P(I)$ of accepting K immediately, given that K is accepted as a sample from $p(k)$, is given by the area of the five rectangles R_i, $i = 0, 1, 2, 4, 5$:

$$\begin{aligned} P(I) \;=\; f_m(1 - {\delta_m}^{(l)} - {\delta_m}^{(r)}) \;+\; f_2(m - k_2 + {\delta_m}^{(l)}) \;+\; f_1(k_2 - k_1) \\ +\; f_4(k_4 - m + {\delta_m}^{(r)}) \;+\; f_5(k_5 - k_4)\,. \end{aligned} \tag{21}$$

A close look at algorithm GPRS reveals that the sampling from the immediate acceptance regions requires only one uniform (for the generation of the sample K), whereas the sampling from all other regions requires exactly two uniforms (one for the generation of the sample K and one for its acceptance or rejection). Consequently the average number of uniforms N needed to generate one sample K from $p(k)$ is

$$N = 2\alpha - P(I)\,. \tag{22}$$

The calculation of $P(S)$, the probability of accepting K by means of the lower squeeze $L(k)$, given that K is accepted, is a little bit complicated, due to the discreteness of $L(k)$. With the following auxiliary quantities, however, we will be able to describe $P(S)$ by a short and intelligible formula.

Let

$$
\begin{aligned}
h_1 &= \quad f_1 - f(k_1 - 1)\,, && q_1 = f_1/h_1\,,\\
h_2 &= \left\{ \begin{array}{lll} f_2 - f(k_2 - 1) & \text{if} & k_2 = k_2{}^{(min)} \\ f_2/(k_2 - z_2) & \text{if} & k_2 > k_2{}^{(min)} \end{array} \right\}, && q_2 = (f_2 - f_1)/h_2\,,\\
h_4 &= \left\{ \begin{array}{lll} f_4 - f(k_4 + 1) & \text{if} & k_4 = k_4{}^{(max)} \\ f_4/(z_4 - k_4) & \text{if} & k_4 < k_4{}^{(max)} \end{array} \right\}, && q_4 = (f_4 - f_5)/h_4\,,\\
h_5 &= \quad f_5 - f(k_5 + 1)\,, && q_5 = f_5/h_5\,.
\end{aligned}
$$

The quantities h_i describe the slope of the four outer segments of $\tilde{L}(x)$, whereas the quantities q_i describe the maximum number of cells with height h_i we can find in the vertical range taken up by the corresponding part of $f(x)$.

Strictly speaking, there are two different types of cells, cells with height h_i and cells with height $h_i(q_i - \lfloor q_i \rfloor)$ under the corresponding staircase segment. The number of cells with height h_i, reduced by the number of cells lying in the adjoining immediate acceptance region, is

$$(\lfloor q_i \rfloor - 1) \;+\; (\lfloor q_i \rfloor - 2) \;+\; \ldots \;+\; 1 \;=\; \frac{\lfloor q_i \rfloor\,(\lfloor q_i \rfloor - 1)}{2}\,.$$

The number of cells with height $h_i(q_i - \lfloor q_i \rfloor)$, under the above restriction, is $\lfloor q_i \rfloor$. Consequently the area below one of the four outer staircase segments is

$$A_i \;=\; \lfloor q_i \rfloor \left(\frac{\lfloor q_i \rfloor - 1}{2} + q_i - \lfloor q_i \rfloor \right) h_i\,, \qquad i = 1,\,2,\,4,\,5\,. \tag{23}$$

The area below the two inner staircase segments less the area of the immediate acceptance regions R_0, R_2 and R_3 is

$$A_3 \;=\; \tfrac{1}{2}(f_m - f_2)(m - k_2 - 1 + 2\delta_m{}^{(l)}) \;+\; \tfrac{1}{2}(f_m - f_4)(k_4 - m - 1 + 2\delta_m{}^{(r)}). \tag{24}$$

Hence, the total probability $P(S)$ of accepting K by means of $L(k)$, given that K is accepted, reads

$$P(S) \;=\; A_1 + A_2 + A_3 + A_4 + A_5\,. \tag{25}$$

6 Application to the Poisson and Binomial Distributions

A random variable K_n has binomial distribution with parameters n and p $(K_n \sim b(n,\, p))$ if

$$p_n(k) = P(K_n = k) = \binom{n}{k} p^k (1-p)^{n-k}\,, \quad k = 0,\, 1,\, \ldots\, n, \quad 0 \le p \le 1,\;\; n \ge 1\,. \tag{26}$$

Without loss of generality we restrict our attention to $p \le \frac{1}{2}$; it is well–known that if $K \sim b(n,\, p)$, then $L = n - K \sim b(n,\, 1-p)$. In order to obtain a sampling routine valid for all $0 < p < 1$ from a routine just valid for $0 < p \le \frac{1}{2}$, one only has to substitute $\min(p,\, 1-p)$ for p, and if $p > \frac{1}{2}$, one has to change the obtained random number K to $n - K$.
Mean and variance of a $b(n,\, p)$ distributed random variable K_n are

$$\mathrm{E}(K_n) = \mu = np, \qquad \mathrm{Var}(K_n) = \sigma^2 = np\,(1-p)\,. \tag{27}$$

The recurrence relationship for the probabilities (26) is given by

$$p_n(k) = \frac{p}{1-p}\,\frac{n+1-k}{k}\,p_n(k-1)\,, \quad k = 1,\, 2,\, \ldots\, n, \qquad p_n(0) = (1-p)^n\,. \tag{28}$$

A random variable K has Poisson distribution with parameter μ $(K \sim P(\mu))$ if

$$p(k) = P(K = k) = \lim_{\substack{n\to\infty,\; p\to 0\\ \mu = np \;\text{ fixed}}} p_n(k) = \frac{\mu^k}{k!}\,e^{-\mu}\,, \qquad k = 0,\, 1,\, 2,\, \ldots\,, \quad \mu > 0\,. \tag{29}$$

Mean and variance of a $P(\mu)$ distributed random variable K are

$$\mathrm{E}(K) = \mu, \qquad \mathrm{Var}(K) = \sigma^2 = \mu\,. \tag{30}$$

The recurrence relationship for the probabilities (29) is given by

$$p(k) = \frac{\mu}{k}\,p(k-1)\,, \quad k = 1,\, 2,\, 3,\, \ldots\,, \qquad p(0) = e^{-\mu}\,. \tag{31}$$

The $P(\mu)$ distribution is a special $b(n,\, p)$ distribution, with $n = \infty$, $p = 0$, and $\mu = np$ fixed. Introducing the quantity ν, defined as

$$\nu = \begin{cases} \mu & \text{for the } P(\mu) \text{ distribution} \\ (n+1)p & \text{for the } b(n,\, p) \text{ distribution}, \end{cases} \tag{32}$$

it appears that (28) and (31) are both covered by the single — and well–defined (contrary to (28)) — recurrence formula

$$p_n(k) = r(k)\,p_n(k-1)\,, \qquad r(k) = 1 - \frac{1}{1-p}\left(1 - \frac{\nu}{k}\right), \qquad k = 1,\, 2,\, \ldots\, n\,. \tag{33}$$

Consequently we can determine valid reflection centres simultanously for both distributions by employing the more general formula (33).

It is easy to see that $r(k)$ in (33) is a monotonically decreasing function on the integers $k \geq 1$. But this is equivalent to that the probability function $p_n(k)$ is log–concave. Therefore the tails of $p_n(k)$ can be covered, in any case, by exponential functions which meet the requirement (19). We will draw upon these functions, specified in Lemma 2, to carry through the sampling from the tails of $p_n(k)$ (that is, our Poisson and binomial algorithms will employ case (a) of the general sampling routine GPRS).

The probability functions (26) and (29) are unimodal. There is one mode at $m = \lfloor \nu \rfloor$ if the parameter ν is not an integer, and there are two adjacent modes at $m-1$ and at m if the parameter ν is an integer. Thus (26) and (29) fulfil the basic requirement of Lemma 1.

Now there is no reason why we should not apply Lemma 1 to the $b(n,\, p)$ distributions (and thereby simultanously to the $P(\mu)$ distributions). First of all, we will specify the normalizing constants $c_N(k)$ and $c_S(i,\, k)$. The definitions of $N(k)$, $S(i,\, k)$ in Lemma 1 and the general recurrence formula (33) propose to set

$$c_N(k) = (k+1)k\,(1-p)^2\,, \quad c_S(i,\, k) = (k+i+1)(k+i)(k-i+1)(k-i)(1-p)^4\,.$$

Both constants are positive in the k–range and i–range stated in (14) and (15) which is the basic requirement $c_N(k)$ and $c_S(i,\, k)$ must satisfy. Using (33) and the above definitions, $N(k)$ and $S(i,\, k)$ of Lemma 1 reduce to

$$N(k) = k^2 - (2\nu - 1)k + \nu(\nu - 2 + p) \tag{34}$$

and

$$S(i,\, k) = (2p-1)\,i^4 - B(k)\,i^2 + N(k)\,C(k)\,, \tag{35}$$

where

$$B(k) = 2\,(2p-1)k^2 - 2((3p-1)\nu - (2p-1))k + 2p\nu^2 - 2p^2\nu + 2p - 1 \tag{36}$$

and

$$C(k) = (2p-1)k^2 - (2p\nu - (2p-1))k + \nu(\nu - p)\,. \tag{37}$$

After an analytical examination of (34) (with respect to its roots) and (35) (with respect to its sign in the parameter region $n \geq 1,\ 0 \leq p \leq \frac{1}{2}$) we arrived at the following result which will be proved in Appendix B.

Lemma 3

(a) Let K_n be a $b(n,\, p)$ binomial random variable, where $n \geq 3$ and $0 < p \leq \frac{1}{2}$. Define $\nu = (n+1)p$.

(b) Let K be a $P(\mu)$ Poisson random variable, where $\mu > 0$. Define $p = 0$ and $\nu = \mu$.

Furthermore, let $m = \lfloor \nu \rfloor$ *be the mode and* $N(k)$ *(34) the 'necessary function' of* $p_n(k)$*. Finally, let*

$$x_2^{(min)} = \nu - \tfrac{1}{2} - \sqrt{(1-p)\nu + \tfrac{1}{4}}\,, \qquad k_2^{(min)} = \left\lceil x_2^{(min)} \right\rceil \tag{38}$$

$$x_4^{(max)} = \nu - \tfrac{1}{2} + \sqrt{(1-p)\nu + \tfrac{1}{4}}\,, \qquad k_4^{(max)} = \left\lfloor x_4^{(max)} \right\rfloor, \tag{39}$$

where $x_2^{(min)}$ *and* $x_4^{(max)}$ *are the roots of* $N(k) = 0$*. Then*

(i) Candidates for reflection centres $k_2 \le m - 1 + \delta_m^{(l)}$ *are all*

$$k_2 \in \left[k_2^{(min)},\; m - 1 + \delta_m^{(l)}\right] \cup \left\{m - 1 + \delta_m^{(l)}\right\},$$

provided $\nu > 2 - p$ *in the case* $\delta_m^{(l)} = 0$*, and* $\nu > 2 + \sqrt{2}$ *in the case* $\delta_m^{(l)} = 1$*.*

If $\delta_m^{(l)} = 0$*, then* $k_1 \ge 1$ *for all* $\nu > 2 + \sqrt{2}$*; in all other cases (*$\delta_m^{(l)} = 0$*,* $2 - p < \nu \le 2 + \sqrt{2}$ *and* $\delta_m^{(l)} = 1$*,* $\nu > 2 + \sqrt{2}$*)* $k_1 \ge 0$ *(*$k_1 = 2k_2 - (m - 1 + \delta_m^{(l)})$ *is the left boundary of the centre region and must meet the requirement* $k_1 \ge 0$*). Furthermore, if* $\nu \ge 3$*, then* $k_2^{(min)} \le m - 1$*, otherwise* $k_2^{(min)} \le m$*.*

(ii) Candidates for reflection centres $k_4 \ge m + 1 - \delta_m^{(r)}$ *are all*

$$k_4 \in \left[m + 1 - \delta_m^{(r)},\; k_4^{(max)}\right] \cup \left\{m + 1 - \delta_m^{(r)}\right\}.$$

In case of the binomial distribution $k_5 \le n - 1$ *for all* $\nu \ge 3$*, and* $k_5 \le n$ *for all* $\nu > 0$ *(*$k_5 = 2k_4 - (m + 1 - \delta_m^{(r)})$ *is the right boundary of the centre region and must meet the requirement* $k_5 \le n$*). If* $\nu > 2 + \sqrt{2}$*, then* $k_4^{(max)} \ge m + 1$*, otherwise* $k_4^{(max)} \ge m$*.*

The main performance measures α (the average number of samples X from the 'hat' $h(x)$ to generate one sample K from $p(k)$), $P(I)$ (the probability of accepting K immediately, given that K is accepted as a sample from $p(k)$), and N (the average number of uniforms needed to generate one sample K from $p(k)$) (see Section 5) inform us about the quality of our sampling routines resulting from the different mode transformations and reflection centres. Extensive numerical calculations lead us to assume that the extreme reflection centres possible, that are $k_2 = \min(k_2^{(min)},\, m - 1 + \delta_m^{(l)})$ and $k_4 = \max(k_4^{(max)},\, m + 1 - \delta_m^{(r)})$, minimize the measures α and N for all three kinds of mode transformations. (We calculated the characteristics of the left and of the right side separately — more precisely, we calculated the portions of the characteristics belonging to the region left from m, m included, and to the region right from m, m again included, separately — for both the Poisson and binomial distributions, and for a great number of shape parameters, thereby distinguishing between the case m remains in the centre and the case m is part of the reflection. Comparing the numerical values resulting from different feasible reflection centres k_2 and k_4 for a fixed handling of the mode always indicated the optimality of $\min(k_2^{(min)},\, m - 1 + \delta_m^{(l)})$ and $\max(k_4^{(max)},\, m + 1 - \delta_m^{(r)})$ with respect to α and

N.) This suggests to use the reflection centres $k_2 = \min(k_2^{(min)},\ m-1+\delta_m^{(l)})$ and $k_4 = \max(k_4^{(max)},\ m+1-\delta_m^{(r)})$ in our algorithms.

Additionally, if m is small ($m \leq 50$), the extreme reflection centres will lead very often to a maximum quick acceptance probability $P(Q) = P(I) + P(S)$. On the other hand, if m is large ($m \geq 500$), the extreme reflection centres will lead in nearly all cases to a maximum squeeze probability $P(S)$.

If $m \geq 50$, the characteristics α, $P(I)$, and N attain their optimal values (α and N attain a minimum, $P(I)$ attains a maximum) when m remains in the centre. The characteristics of the two other cases, however, do not differ very much; in more than 3/4 of all cases, $P(I)$ and N (and also the sum $P(I) + P(S)$) attain better numerical values when m is partially transformed to the right instead to the left.

If $m \leq 10$, α attains its minimum in the case m is part of the reflection σ_l or σ_r, but more often when m is part of σ_r. The transformation of m to the right also yields in about half the cases the minimum number N, the second best situation is no reflection. In more than half the cases $P(I)$ attains its maximum when m remains in the centre, the other both cases are optimal in nearly the same number of cases. We put stress on the fact that here the characteristics of the second best mode transformation differ quite a lot from the characteristics of the optimal mode transformation.

Furthermore, if $m \leq 100$, the quick acceptance probability $P(Q) = P(I) + P(S)$ attains its maximum in more than 3/4 of all cases at m being partially transformed to the right.

From the above we conclude that α, $P(I)$, and N, on the whole, attain slightly better numerical values in case m is neither part of σ_l nor part of σ_r (case (c) in Section 2). But we do not like to handle the mode as an individual event in the sampling routine. Furthermore, to keep the tail cases as small as possible — a main goal, since the calculation of a candidate from the tail envelopes requires one logarithm, whereas in the centre region we only need to apply a linear transformation to a uniform number — necessitates the transformation of m. From this point of view, we have to decide for one of the two other cases. Unfortunately, the behaviour of the characteristics belonging to these two cases is still erratic. In other words, for one parameter combination the transformation of m to the left, for another parameter combination the transformation of m to the right yields the better values for the characteristics. This suggests to make the transformation of m contingent on the shape parameters. However, we did not spend any time on determining a criterion which establishes the optimal mode transformation, since we aimed at designing a Poisson/binomial algorithm based on patchwork, where m is always transformed to the same side, independent of the shape parameters. With respect to the performance measures α and $P(I)$, the transformation of the upper part of m to the right (case (b) in Section 2) is the next best choice. Its mostly better values compared to the case transformation of the upper part of m to the left (case (a) in Section 2) can be explained to some extent by the existence of a second mode at $m-1$ for integer ν. Additionally, case (b) is valid for a larger parameter range ($\nu > 2-p$). It also outperforms the other two cases for a greater number of parameter combinations with the property that ν is small. All these facts suggest to transform the upper part of the mode cell to the right whenever it is possible.

Lemma 3 tells us that $N(k) \leq 0$ for all $k \in [k_2^{(min)}, k_4^{(max)}]$ and $N(k) > 0$ for all $k \notin [k_2^{(min)}, k_4^{(max)}]$. Consequently the polygon through the points $(k, p_n(k))$ is concave in and convex outside the interval $[k_2^{(min)}, k_4^{(max)}]$, and $L(k)$ is in fact a lower bound for $p_n(k)$. Furthermore, $k_2 = \min(k_2^{(min)}, m-1)$ and $k_4 = k_4^{(max)}$ are feasible reflection centres, provided $\nu > 2 - p$ and m is transformed to the right. Our general algorithm GPRS in Section 4 also recommends to use the above initialization for k_2 and k_4 whenever it is possible, for then the special $L(k)$ we can use (due to $k_2 \leq k_2^{(min)}$ and $k_4 = k_4^{(max)}$) is much simpler than $L(k)$ following from general reflection centres $k_2 > k_2^{(min)}$ and $k_4 < k_4^{(max)}$.

We finally point out that we rescale the probability functions such that $f(m) = 1$. Therewith we avoid the computation of $f_m = f(m)$; moreover, we do not even need the variable f_m, since it takes the value 1 for all parameter combinations.

Below we state the initialization steps of algorithms BPRS^r and PPRS^r (where m is always part of the transformation to the right) which are valid for $\nu > 2 + \sqrt{2}$. We also state initialization steps which lead to the general sampling routines BPRS and PPRS (where m is mostly part of the transformation to the right and rarely part of the transformation to the left), being valid for all parameter combinations we focused our attention on.

Algorithms BPRS^r and BPRS:

$f(k, n, l_p, c_m) = \exp(k * l_p - \log(k!) - \log((n-k)!) - c_m), \quad r(k, r_c, c_p) = r_c/k - c_p$

Initialization of BPRS^r: $\quad ((n+1)p > 2 + \sqrt{2}$ and $0 < p \leq \frac{1}{2})$

Set $\nu = (n+1) * p$, $m = \lfloor \nu \rfloor$, $q = 1 - p$, $D = \sqrt{\nu * q + \frac{1}{4}}$;
Set $k_2 = \left\lceil \nu - \frac{1}{2} - D \right\rceil$, $k_4 = \left\lfloor \nu - \frac{1}{2} + D \right\rfloor$; Lemma 3, (i) and (ii)
Set $d_l = m - k_2 - 1$, $d_r = k_4 - m$;
Set $k_1 = k_2 - d_l$, $k_5 = k_4 + d_r$; $k_1 \geq 1$, $k_5 \leq n - 1$
Set $r_c = \nu/q$, $c_p = p/q$;
Set $l_p = \log(c_p)$, $c_m = m * l_p - \log(m!) - \log((n-m)!)$;
Set $f_2 = f(k_2, n, l_p, c_m)$, $f_1 = f(k_1, n, l_p, c_m)$;
Set $f_4 = f(k_4, n, l_p, c_m)$, $f_5 = f(k_5, n, l_p, c_m)$;
Set $p_1 = f_2 * (d_l + 1)$, $p_2 = f_2 * d_l + p_1$;
Set $p_3 = f_4 * (d_r + 1) + p_2$, $p_4 = f_4 * d_r + p_3$;
Set $r_2 = r(k_2, r_c, c_p)$;
Set $r_1 = r(k_1, r_c, c_p)$, $\lambda_l = \log(r_1)$, $p_5 = f_1/\lambda_l + p_4$;
Set $r_4 = r(k_4 + 1, r_c, c_p)$;
Set $r_5 = r(k_5 + 1, r_c, c_p)$, $\lambda_r = -\log(r_5)$, $p_6 = f_5/\lambda_r + p_5$;

Initialization of BPRS: $\quad (n \geq 1$ and $0 < p \leq \frac{1}{2})$

Set $\nu = (n+1) * p$, $m = \lfloor \nu \rfloor$, $q = 1 - p$;
If $(\nu < 5/4)$ then

```
        Set k_2 = k_1 = 0, k_4 = k_5 = 1;                          k_4^(max) ≤ 1
        Set d_l = d_r = 0;
    else
    if (ν < 2) then
        Set k_2 = k_1 = 1, k_4 = k_5 = 2;                          k_4^(max) ≤ 2
        Set d_l = d_r = 0;
    else
        Set D = sqrt(ν * q + 1/4);
        If (ν < 3) then Set k_2 = m − 1;
                    else Set k_2 = ⌈ν − 1/2 − D⌉;                 Lemma 3, (i)
        Endif
        Set k_4 = ⌊ν − 1/2 + D⌋;                                   Lemma 3, (ii)
        Set d_l = m − k_2 − 1, d_r = k_4 − m;
        Set k_1 = k_2 − d_l, k_5 = k_4 + d_r;                      k_1 ≥ 0, k_5 ≤ n
    Endif
    Set r_c = ν/q, c_p = p/q;
    Set l_p = log(c_p), c_m = m * l_p − log(m!) − log((n − m)!);
    Set f_2 = f(k_2, n, l_p, c_m), f_1 = f(k_1, n, l_p, c_m);
    Set f_4 = f(k_4, n, l_p, c_m), f_5 = f(k_5, n, l_p, c_m);
    Set p_1 = f_2 * (d_l + 1), p_2 = f_2 * d_l + p_1;
    Set p_3 = f_4 * (d_r + 1) + p_2, p_4 = f_4 * d_r + p_3;
    If (d_l > 0) then Set r_2 = r(k_2, r_c, c_p);
    If (k_1 > 0) then
        Set r_1 = r(k_1, r_c, c_p), λ_l = log(r_1), p_5 = f_1/λ_l + p_4;
    else
        Set p_5 = p_4;
    Endif
    If (d_r > 0) then Set r_4 = r(k_4 + 1, r_c, c_p);
    If (k_5 < n) then
        Set r_5 = r(k_5 + 1, r_c, c_p), λ_r = − log(r_5), p_6 = f_5/λ_r + p_5;
    else
        Set p_6 = p_5;
    Endif
```

Algorithms PPRS^r and PPRS:

$f(k,\, l_\nu,\, c_m) = \exp(k * l_\nu - \log(k!) - c_m), \quad r(k,\, r_c) = r_c/k$

Initialization of PPRS^r: $\quad (\mu > 2 + \sqrt{2})$

```
    Set ν = μ, m = ⌊ν⌋, D = sqrt(ν + 1/4);
    Set k_2 = ⌈ν − 1/2 − D⌉, k_4 = ⌊ν − 1/2 + D⌋;                  Lemma 3, (i) and (ii)
    Set d_l = m − k_2 − 1, d_r = k_4 − m;
```

Set $k_1 = k_2 - d_l$, $k_5 = k_4 + d_r$; $\qquad k_1 \geq 1$
Set $l_\nu = \log(\nu)$, $c_m = m * l_\nu - \log(m!)$;
Set $f_2 = f(k_2,\, l_\nu,\, c_m)$, $f_1 = f(k_1,\, l_\nu,\, c_m)$;
Set $f_4 = f(k_4,\, l_\nu,\, c_m)$, $f_5 = f(k_5,\, l_\nu,\, c_m)$;
Set $p_1 = f_2 * (d_l + 1)$, $p_2 = f_2 * d_l + p_1$;
Set $p_3 = f_4 * (d_r + 1) + p_2$, $p_4 = f_4 * d_r + p_3$;
Set $r_2 = r(k_2,\, \nu)$;
Set $r_1 = r(k_1,\, \nu)$, $\lambda_l = \log(r_1)$, $p_5 = f_1/\lambda_l + p_4$;
Set $r_4 = r(k_4 + 1,\, \nu)$;
Set $r_5 = r(k_5 + 1,\, \nu)$, $\lambda_r = -\log(r_5)$, $p_6 = f_5/\lambda_r + p_5$;

Initialization of PPRS: $\quad (\mu > 0)$

Set $\nu = \mu$, $m = \lfloor \nu \rfloor$;
If $(\nu < 5/4)$ then
    Set $k_2 = k_1 = 0$, $k_4 = k_5 = 1$; $\qquad k_4^{(max)} \leq 1$
    Set $d_l = d_r = 0$;
else
    Set $D = \sqrt{\nu + \frac{1}{4}}$;
    If $(\nu < 3)$ then Set $k_2 = m - 1$;
        else Set $k_2 = \left\lceil \nu - \frac{1}{2} - D \right\rceil$; $\qquad$ Lemma 3, (i)
    Endif
    Set $k_4 = \left\lfloor \nu - \frac{1}{2} + D \right\rfloor$; $\qquad$ Lemma 3, (ii)
    Set $d_l = m - k_2 - 1$, $d_r = k_4 - m$;
    Set $k_1 = k_2 - d_l$, $k_5 = k_4 + d_r$; $\qquad k_1 \geq 0$
Endif
Set $l_\nu = \log(\nu)$, $c_m = m * l_\nu - \log(m!)$;
Set $f_2 = f(k_2,\, l_\nu,\, c_m)$, $f_1 = f(k_1,\, l_\nu,\, c_m)$;
Set $f_4 = f(k_4,\, l_\nu,\, c_m)$, $f_5 = f(k_5,\, l_\nu,\, c_m)$;
Set $p_1 = f_2 * (d_l + 1)$, $p_2 = f_2 * d_l + p_1$;
Set $p_3 = f_4 * (d_r + 1) + p_2$, $p_4 = f_4 * d_r + p_3$;
If $(d_l > 0)$ then Set $r_2 = r(k_2,\, \nu)$;
If $(k_1 > 0)$ then
    Set $r_1 = r(k_1,\, \nu)$, $\lambda_l = \log(r_1)$, $p_5 = f_1/\lambda_l + p_4$;
else
    Set $p_5 = p_4$;
Endif
Set $r_4 = r(k_4 + 1,\, \nu)$;
Set $r_5 = r(k_5 + 1,\, \nu)$, $\lambda_r = -\log(r_5)$, $p_6 = f_5/\lambda_r + p_5$;

At the end of Section 8 we will advance reasons for the correctness of the initialization steps of BPRS and PPRS. There we will also state a not so short, but also not too long routine that calculates the function values f_2, f_1, f_4, and f_5 a little bit more efficiently. The calculation is done either directly (by means of $f(k)$) or recurrently from the mode function value $f(m)$ (by using the recurrence relationship function $r(k)$ again and again). The distance between the reflection centres k_2, k_4 and m (which is described by the quantities d_l and d_r) will decide on the procedure being actually used. Formally, if $d_r \leq N_{It}$, we apply the recurrent, otherwise we apply the direct calculation (N_{It} denotes the maximum number of recurrence steps we admit in the calculation of each function value). Extensive timings on an IBM PC AT–486 showed that a value $N_{It} \approx 12$ for the Poisson distributions, and a value $N_{It} \approx 14$ for the binomial distributions, yields the best overall initialization times (with respect to our hardware configuration). Below we compare the times of the original initialization steps with those of the modified initialization steps, using the above–mentioned values for N_{It}.

Table 1: Initialization times [μsec/variate] of algorithms BPRS^r and PPRS^r, always calculating the constants f_2, f_1, f_4 and f_5 by means of $f(k)$.

Algorithm	$n \backslash \mu$	10	20	50	150	500	2500	25000
BPRS^r	20	234						
	40	317	243					
	100	371	364	505				
	300	372	372	498	505			
	1000	360	361	505	508	502		
	5000	365	364	510	510	510	508	
	50000	362	362	507	508	507	513	503
PPRS^r	∞	199	221	353	354	352	356	353

Table 2: Initialization times [μsec/variate] of algorithms BPRS$^r{}_{14}$ ($N_{It} = 14$) and PPRS$^r{}_{12}$ ($N_{It} = 12$), calculating the constants f_2, f_1, f_4 and f_5 either by means of $f(k)$ or recurrently from the mode function value $f(m)$.

Algorithm	$n \backslash \mu$	10	20	50	150	500	2500	25000
BPRS$^r{}_{14}$	20	139						
	40	192	168					
	100	193	207	265				
	300	193	215	294	332			
	1000	193	215	307	395	502		
	5000	193	215	308	417	510	508	
	50000	193	214	308	417	507	513	503
PPRS$^r{}_{12}$	∞	138	181	241	328	352	356	353

7 Computational Results, Poisson and Binomial Algorithms

In this section we give a short description of the algorithms we included into our considerations. But beforehand we present some general remarks.

There are four particular tasks in the class of programmed sampling routines which are accomplished by external functions. The first one, occuring in all algorithms, is the generation of a $U(0,\,1)$ uniform random number. The calculation of $\log k!$ is a second job which has to be done. Each time the, possibly rescaled, Poisson/binomial probability function $f(k)$ which contains at least one factorial is evaluated, an external function which returns $\log k!$ for all $k \geq 0$ in the desired precision will be called. This function therefore appears in all algorithms based on acceptance–rejection — since the test condition

$$U\,h(X) \leq f(K), \qquad U \sim U(0,\,1), \quad K = \lfloor X \rfloor \text{ and } X \text{ a sample from } h(x)$$

contains the function $f(k)$ — and in some of the algorithms based on sequential search — in those algorithms that start the search at the mode m and that do not employ a simple approximation to the modal probability. However, there are algorithms which use the Stirling approximation to $\log k!$ inline to reduce the number of logarithms in the acceptance–rejection test condition. These algorithms will call an external function to calculate the difference between $\log k!$ and the first order Stirling approximation (without the series in $\frac{1}{k}$). This transports a case distinction — which becomes necessary because of the lousy approximation of $\log k!$ by Stirling's formula when k is small — which otherwise had to occur in the sampling routines to the external function, and the structure of the algorithms becomes a little bit clearer. Besides, more than one sampling routine will call this function and take advantage of it, without having to know anything about its implementation except that the numerical value returned will meet the demanded precision. The fourth and last special task is the generation of a $N(0,\,1)$ standard normal deviate. Each algorithm which is based on acceptance–rejection containing the density of the $N(\hat{\mu},\,\hat{\sigma})$ normal distribution requires samples from that distribution. We decided to use the ACR method [9], since this method firstly is not so hard to implement in a high level programming language, secondly needs few uniforms, and thirdly is very fast due to a large immediate acceptance region and very tight but, nevertheless, simple squeezes. Below we describe, in more or less detail, the four corresponding functions `krand1`, `normal`, `logfak` and `stirlc`, and after this we present all the Poisson and binomial algorithms we implemented in C.

7.1 General Functions

krand1 generates random numbers from the $U(0,\,1)$ uniform distribution.

The function is based on the multiplicative congruential method with modulus $m = 2^{54}$ and multiplicator $a = 2\,783\,377\,640\,906\,189$. This generator has a long period of 2^{52} and produces high–class uniform random numbers in double precision as has been established by J. H. Ahrens. Our C–implementation, however, takes advantage of the internal representation of the data types long integer and double on IBM and IBM–compatible computers and is therefore not portable.

normal generates variates from the $N(0,\, 1)$ standard normal distribution.

The function is based on the acceptance–complement technique combined with patchwork–rejection and the ratio of uniforms method. Due to the high immediate acceptance and squeeze probability, and the small average number of uniforms — which is about 1.4853 — needed for one normal deviate, the sampling is very efficient and quite fast, even when the generator is implemented in a high level programming language as C. For the details we refer to the paper of W. Hörmann & G. Derflinger [9].

logfak calculates $\log k!$ in double precision.

The function combines Stirling's formula with tabulated values to guarantee that $\log k!$ is calculated quickly in the desired double precision. Whenever $k \geq 30$, `logfak` falls back upon the Stirling approximation

$$\begin{aligned} \log k! &= \log k + \log \Gamma(k) \\ &\approx (k + \tfrac{1}{2}) \log k - k + \log \sqrt{2\pi} + \tfrac{1}{12k} - \tfrac{1}{360k^3} + \tfrac{1}{1260k^5} - \tfrac{1}{1680k^7} \end{aligned}$$

which has the truncation error ϵ: $|\epsilon| < 4.2658 \times 10^{-17}$ for $k \geq 30$. It yields the correct value for $\log k!$ within the double precision we demand (which is about 16 decimal digits). Otherwise, if $0 \leq k < 30$, `logfak` will return a tabulated value which is exact down to the first 16 digits behind the decimal point.

stirlc calculates the difference between $\log k!$ and the first order Stirling approximation in double precision.

The function approximates the difference $\log k! - ((k+\frac{1}{2}) \log k - k + \log \sqrt{2\pi}\,)$ which corresponds to the polynomial series in $\frac{1}{k}$ of the Stirling approximation. Similar to `logfak`, `stirlc` returns a tabulated value which corresponds with the exact quantity in the first 16 significant decimal digits in the case $1 \leq k < 30$. If $k \geq 30$, the function simply returns the value of $\frac{1}{12k} - \frac{1}{360k^3} + \frac{1}{1260k^5} - \frac{1}{1680k^7}$.

7.2 Sampling Methods for the Poisson Distribution

Search Algorithms

PINV (**Inv**ersion, implemented as low index sequential search, valid for $\mu > 0$).

The algorithm uses sequential search starting at the lowest index $k = 0$ to generate the desired Poisson sample. It circumvents the explicit calculation of cumulative probabilities (which appear in the formal description of the inversion method for discrete distributions) by means of the chop–down technique. The initial probability

$p_0 = e^{-\mu}$ is set up in an initialization step; all other probabilities still needed afterwards are calculated in the search loop through the recurrence relationship $p(k+1) = r(k+1)\,p(k)$ (forward recurrence). The recurrence is a potential source of roundoff error which accumulates and becomes serious as k increases; therefore k is not allowed to exceed a precalculated specific bound (which is large enough to guarantee that the probabilities higher up are insignificant with respect to the given accuracy).

PMDU (**M**odal **D**own–**U**p sequential search, valid for $\mu > 0$) by C. D. Kemp [16].

The algorithm is based on sequential search starting at the mode m. Since the cumulative modal probability is not available, the search must take place alternately down and up from m. When it reaches the lowest index $k = 0$, the alternating search ends and modifies to a pure upward sequential search starting at the index $2m+1$. It is terminated, at the latest, when the current search index exceeds a particular safety bound (which guarantees that the returned samples have a significant probability, related to the postulated precision). The modal probability

$$p(m) = \frac{\mu^m}{m!}\, e^{-\mu}, \qquad m = \lfloor \mu \rfloor$$

the search starts with is calculated, in a set–up step, with the help of Stirling's formula. However, we use the Stirling approximation to $\log m! = \log \Gamma(m+1)$ (instead of $\log m! = \log m + \log \Gamma(m)$ to avoid the term $\log m$ in the approximation) inline and not the external function `logfak` — this saves one logarithm — and compute the difference between $\log m!$ and the corresponding first order Stirling approximation by `stirlc`. All other probabilities still needed are calculated by the recurrence relationship $p(k-1) = r^{-1}(k)\,p(k)$ (backward recurrence) or $p(k+1) = r(k+1)\,p(k)$ (forward recurrence). In the search loop cumulative probabilities are avoided again, as in the algorithm PINV, by the chop–down technique.

KPLO (**K**emps' **P**oisson algorithm for **Lo**w parameters μ, being valid for $0 < \mu < 30$, which combines down–up sequential search from the mode m with composition) by A. W. Kemp & C. D. Kemp [15].

The algorithm is based on sequential search starting at the mode m, and on composition. The search takes place alternately down and up from m in the composition region $[0,\, 2m+1]$; it is complemented, similar to algorithm PMDU, by a pure upward sequential search which, however, starts at $2m+2$. The composition — which depends on properties of the sum of the two adjacent Poisson probabilities $p(m)$ and $p(m+1)$ — rests basically upon the approximation of $p(k)$ by a simple lower bound $b(k)$ in the interval $[0, 2m+1]$. This bound $b(k)$ is characterized nearly completely by the recurrence relationship function $r(k)$ and the property that the sum $b(m) + b(m+1)$ assumes the same value for all $P(\mu)$ distributions with mode at m. This allows for a fast primary initialization, since the set–up of the central bounds $b(m)$ and $b(m+1)$ can fall back on tabulated values. (The algorithm contains a table with 30 entries which stores the sums $b(m) + b(m+1)$, $m = \lfloor \mu \rfloor$.

These tabulated values are identical to twice the modal probability of the Poisson distributions with parameter $\hat{\mu} = \lfloor \mu \rfloor + 1$.)

The search starts with the lower bounds $b(m)$ and $b(m+1)$ and uses the relationships $b(k-1) = r^{-1}(k)\, b(k)$ (backward recurrence) and $b(k+1) = r(k+1)\, b(k)$ (forward recurrence) to calculate the next lower bounds (which approximate the corresponding Poisson probabilities). If this search yields no result, the algorithm will consider the residual probabilities (the differences $p(k) - b(k)$) and, if necessary, the tail probabilities (the probabilities right from $2m+1$ up to a certain safety bound). In all subsequent calls however, provided that μ remains unchanged, KPLO then searches on the original Poisson probabilities, since the values $b(k)$ which are stored in a table and which are now completely available lead with little expense, that means quickly, to the desired probabilities $p(k)$. (The residual probabilities follow as well instantly from the tabulated values $b(k)$.)

KPOI (**K**emps' **Poi**sson algorithm, being valid for $\mu \geq 10$, which carries out a sequential search either downwards or upwards from the mode m) by C. D. Kemp & A. W. Kemp [19].

The algorithm uses a highly accurate approximation to the cumulative modal probability, worked out by A. W. Kemp [14], to decide on the direction of the sequential search. It in fact does not calculate very often the exact value of the relatively complex approximation, as there exist simple and close lower and upper bounds on it. The algorithm also uses a very good approximation to the modal probability that gets along without the `logfak` function. This keeps the overall initialization time again a little bit smaller. Actually, it is less than the initialization time of the algorithm PMDU. As there, all the other probabilities required for the search are calculated by the backward or forward recurrence relationship. Cumulative probabilities are avoided again, as in all other table free search algorithms, by the chop–down technique.

Acceptance–Rejection Algorithms

PRUA (**R**atio of **U**niforms method with an **A**pproximation to the slope parameter of the optimal table mountain hat $h(x)$, valid for $\mu \geq 1$) by J. H. Ahrens & U. Dieter [3], slightly modified by E. Stadlober [23].

The algorithm is based on a symmetric table mountain hat $h(x)$ proportional to the density

$$g(x) = \begin{cases} \frac{1}{4s} & a-s \leq x \leq a+s \\ \frac{s}{4(x-a)^2} & \text{elsewhere}, \end{cases}$$

where a is the location and s the slope parameter. The average number of samples from $h(x)$ needed to generate one Poisson variate — which describes the efficiency of the hat $h(x)$ — is $\alpha = 4sp(m)$. J. H. Ahrens & U. Dieter fixed parameter a at $\mu + \frac{1}{2}$, anticipating possible extensions to other discrete distributions: $a = \mu + \frac{1}{2}$

is in fact the best choice for symmetric binomial and hypergeometric distributions. Once a has been fixed, one only has to minimize the quantity s to yield the tightest hat then possible, as the hat size α increases with s. This algorithm here, however, uses an approximation $\hat{s}$ — close above the optimal slope parameter s, that is $\hat{s} \geq s$ — instead of the optimal s. Because of the high tails of $h(x)$ E. Stadlober includes precalculated safety bounds in his algorithm to guarantee that only samples with significant probability are produced.

Simple lower and upper bounds which squeeze the expression $2 \log U$ in the ultimate acceptance–rejection condition help to avoid the evaluation of this logarithm most of the time. However, when the sample K, $K = \lfloor X \rfloor$ and X a sample from $h(x)$, does not differ very much from the mode m (say $|K - m| \leq 15$), the rescaled Poisson probability $f(K)$, needed in the acceptance–rejection test, is computed recurrently from the rescaled modal probability $f(m) = 1$, since this is faster than to evaluate $f(k)$.

PRUE (**R**atio of **U**niforms method which uses the **E**xact value of the optimal slope for the definition of the table mountain hat $h(x)$, valid for $\mu \geq 1$) by E. Stadlober [23].

The algorithm is based on a symmetric table mountain hat function $h(x)$ with the, in relation to the shift $a = \mu + \frac{1}{2}$, optimal slope parameter s. Poisson deviates are generated by the principally same sampling routine that applies in the algorithm PRUA, that is the generation steps of the two ratio of uniforms algorithms are identical. The set–up step here, however, differs from that of algorithm PRUA; naturally, it is quite a bit longer, due to the slightly complicated calculation of the optimal slope s.

PAMS (**A**tkinson's Poisson algorithm, **M**odified with respect to the shape parameters of the hat $h(x)$ and with respect to the test condition which has been complemented by **S**queezes, valid for $\mu \geq 2$) by E. Stadlober [24].

The algorithm is based on a hat function $h(x)$ which is proportional to the density $g(x) = b\, e^{-b(x-a)}/(1 + e^{-b(x-a)})^2$ of the general logistic distribution. The shape parameters are fixed by E. Stadlober at $\hat{a} = \mu + \frac{1}{5}$ (location) and $\hat{b} = 1.5/\sqrt{\mu}$ (scale). These quantities are really simple and, nevertheless, close to the best a and b possible.

The algorithm also contains straightforward lower and upper squeeze tests for the acceptance–rejection condition by which the evaluation of a logarithm is circumvented in about 75 percent of the cases, on the average. Again, whenever a candidate K, $K = \lfloor X \rfloor$ and X a sample from $h(x)$, is in the neighbourhood of the mode m (say $|K - m| \leq 15$), the true Poisson probability $f(K)$ needed to decide on the acceptance or rejection of K is computed recurrently from the modal probability $f(m) = p(m)$.

PDEX (**D**ouble **Ex**ponential method with squeezes based on the Stirling approximation, valid for $\mu \geq 4$) by J. H. Ahrens & U. Dieter [1].

The algorithm is based on the double–exponential hat function

$$h(x) = \tfrac{\hat{\alpha}}{2s}\, e^{-|x-a|/s}\,, \qquad a = \mu + \tfrac{1}{6}, \quad s = \sqrt{\mu}, \quad \text{and} \quad \hat{\alpha} = \sqrt{2/\pi}\, e^{\,1/2 + O(s^{-1})}\,.$$

Our implementation, however, differs from the original algorithm in three points:

1. The acceptance test is implemented directly and only in double precision.
2. Instead of an external function for $\log k!$ (as `logfak` which combines table values with the Stirling approximation) we employ an external function that calculates the correction term of the first order Stirling approximation (like `stirlc`).
3. The quantity $\log k!$ which appears in the calculation of the logarithm of the Poisson probability $p(k)$ is approximated by Stirling's formula with argument $k+1$ instead of k ($\log k! = \log\Gamma(k+1)$ in place of $\log k! = \log k + \log\Gamma(k)$). Then the correction term of the first order Stirling approximation may be squeezed by two consecutive members of the series in $\frac{1}{k+1}$ even when $k = 0$. (We used the very simple bounds 0 and $\frac{1}{12(k+1)}$.)

PNEX (**N**ormal **Ex**ponential method which totally avoids the approximate calculation of factorials, or logarithms of factorials, valid for $\mu \geq 6$) by L. Devroye [4].

The algorithm is based, in principle, on two half normal envelopes of which the first one covers the probabilities $p(k)$ left from m, m not included, and the second one covers the probabilities $p(k)$ right from $m+1$, $m+1$ not included, up to a certain breakpoint $m+d$. In the algorithm samples X from these half normal distributions are produced, candidates which lie outside the centre interval $[0,\, m+d+1)$, however, are quickly rejected. The two central mass points at m and $m+1$ are covered by rectangles with height $p(m)$ — this implies the immediate acceptance of candidates from the mode rectangle — and the right tail (beginning at $x = m+d+1$) is covered by an exponential function. To determine the hat $h(x)$ sketched above, L. Devroye standardized the probability function $p(k)$. For convenience, he still shifted the function by m to the left so that its mode moved to $k = 0$. Both transformations — the normalization and the shift which bring out the function $f(k) = p(m+k)/p(m) \leq 1$, where $k \in [-m,\, \infty)$, in place of $p(k)$ — of course apply to the algorithm too.

The algorithm contains lower and upper squeeze tests which help to avoid the evaluation of the factorials in

$$f(k) \;=\; (\mu/m)^k\, (m!\, m^k)/(m+k)!\,, \qquad m = \lfloor \mu \rfloor$$

most of the time. Otherwise, if the squeezes lead neither to acceptance nor to rejection, $f(k)$ will be calculated directly, in accordance with the above representation, by a loop statement which is repeated about $|k|$ times. (No Stirling approximation is used for the calculation of the factorials.)

PDAC (Generating **P**oisson **D**eviates, using tabulated inversion if parameter $\mu < 10$, and an **A**cceptance **C**omplement routine based on discrete normal probabilities if parameter $\mu \geq 10$) by J. H. Ahrens & U. Dieter [2].

The tabulated inversion is based, in principle, on sequential search starting at the lowest index $k = 0$. However, compared to the other inversion routines, this algorithm stores all cumulative Poisson probabilities (up to a certain threshold) that are calculated in the search loop in an array to speed up the inversion. As it happens, the cumulative probability of $m' = \max(m, 1)$ is greater than or equal to 0.458 for all $\mu < 10$. Therefore the search may start at m' if the largest index for which the cumulative probability is available is greater than or equal to m' or, if this is not the case, the search may start just at this index (less than m'), provided the uniform number generated at the beginning of the inversion routine is greater than or equal to 0.458. Especially in subsequent calls (always with parameter μ) the tabulation of the cumulative probabilities will save quite a lot of time.

The acceptance–complement method falls back upon the discrete normal distribution to generate an initial candidate K (a truncated suitable normal deviate) which is accepted very often as the desired Poisson sample. Whenever K is rejected, the algorithm has to return a sample from the difference distribution. This sample is produced by acceptance–rejection with a double–exponential function that majorizes the histogram belonging to the difference of the two probability functions.

A great number of the initial samples K is accepted without carrying out any acceptance–rejection test, they are immediately accepted. A further part of the initial samples is accepted by means of a squeeze function $z(k)$ which bounds the test quotient tightly from below. The test quotient finally decides on acceptance or rejection of the remaining initial samples. Because of the high immediate and squeeze acceptance rate this test is carried out rarely, generating samples from the difference distribution occurs seldom, and therefore the overall generation time will be quite small. (Actually, it mainly depends on the uniform and normal random number generators used and on their implementation, either in a high level language or in Assembler.)

P2PE (**2 P**oint method using **E**xponential tail envelopes, complemented by squeezes, valid for $\mu \geq 5$) by V. Kachitvichyanukul & B. W. Schmeiser [10].

The algorithm is an adaptation of the hypergeometric algorithm H2PE which is based on a three–part envelope $h(x)$ (rectangle in the centre region, exponentials in the tails) to the Poisson distribution.

The standardized Poisson probability $f(K)$, needed in the acceptance–rejection test, is computed recurrently from the rescaled modal probability $f(m) = 1$ when the mode m or the candidate K is small — more precisely, when $m < 150$ or $K < 75$ — to circumvent its direct calculation. Under the opposite circumstances there are upper and lower bounds — which are based on the Stirling approximation, some numerical estimations and simple squeezes on $\log(1 \pm z)$ — which help to avoid the evaluation of $\log f(k)$, $k = K$, most of the time. However, as we found out, these squeeze tests save scarcely time which most likely is due to their complexity (a larger number of auxiliary quantities must be calculated).

PTPE (**T**riangle **P**arallelogram **E**xponential method with a triangular immediate acceptance region, valid for $\mu \geq 10$) by B. W. Schmeiser & V. Kachitvichyanukul [22].

The algorithm is based on a hat $h(x)$ which consists of a triangle (which covers the centre region) and two exponential functions (which cover the tails). A triangular immediate acceptance region in the centre region — with lateral sides parallel to the triangular piece of $h(x)$ — considerably speeds up the whole generation step (compare the execution times of algorithm P2PE with those of algorithm PTPE in Table 5). Moreover, contrary to the hat $h(x)$ of algorithm P2PE the hat function here does not touch the histogram $f(x)$ which is due to the use of crude approximations in place of some exact and time–consuming constants in $h(x)$. For this reason the initialization time of algorithm PTPE is evidently smaller than of algorithm P2PE (see Table 6).

If the mode m or the candidate K, $K = \lfloor X \rfloor$ and X a sample from $h(x)$, is small — more precisely, if $m < 100$ or $K \leq 50$ — the standardized Poisson probability $f(K)$, needed in the acceptance–rejection test in case K is not immediately accepted, will be computed recurrently from the normalized modal probability $f(m) = 1$ to circumvent its direct calculation. Under the opposite circumstances there are particular upper and lower bounds — which are not identical to the bounds in algorithm P2PE, though they also rest upon the Stirling approximation, some numerical estimations and simple squeezes on $\log(1 \pm z)$ — B. W. Schmeiser & V. Kachitvichyanukul included in their algorithm to circumvent the direct calculation of $\log f(K)$ very often.

PTRS (**T**ransformed **R**ejection method with **S**queezes, valid for $\mu \geq 10$) by W. Hörmann [7].

W. Hörmann decided to approximate the inverse distribution function $F^{-1}(u)$ of the Poisson histogram function $f(x)$ by

$$G^{-1}(u) = \Big(\frac{2a}{1/2 - |u|} + b\Big)u + c, \qquad -\tfrac{1}{2} < u < +\tfrac{1}{2}\,,$$

where a, b are scale parameters and c is a location parameter. He determined the optimal values of the parameters a, b and c for a large number of parameters μ by a numerical search procedure. Then he approximated them by simple functions, but only in the range $\mu \geq 10$.

A relatively large rectangle with support $[-0.43, +0.43]$ below the transformed Poisson histogram function $f(u) = f(G^{-1}(u))(G^{-1})'(u)$ is used for squeeze acceptance. Furthermore, since the fit of the hat $h(u)$ which is proportional to the density of the $U(-\frac{1}{2}, +\frac{1}{2})$ uniform distribution is bad in the tails of $f(u)$, W. Hörmann worked out close linear upper bounds on $f(u)$ in the ranges $0.487 \leq |u| < 0.5$. The resulting upper squeeze tests have a very high rejection rate, related to the total number of discarded candidates; therefore they really speed up the algorithm. What accelerates the sampling routine most, however, is the large squeeze acceptance region which helps to circumvent the acceptance–rejection test very often.

The basic sampling method here which differs slightly from the usual acceptance–rejection technique is as follows. Generate a point (U, V) uniformly distributed under $h(u) = \hat{\alpha}$, where

$$\hat{\alpha} = 1.1239 + \frac{1.1328}{b - 3.40} \quad \text{and} \quad b = 0.9310 + 2.53\sqrt{\mu}\,.$$

$\hat{\alpha}$ is a safe approximation to the minimum hat size $\alpha = \max_{-1/2<u<+1/2} f(u)$ which can be determined only numerically, in other words $\hat{\alpha} \geq \alpha$. If (U, V) is below $f(u)$ — to decide this, make use of the squeezes and, only if they fail, make use of $f(u)$ — U is accepted as a sample from $f(u)$, and $K = \lfloor G^{-1}(U) \rfloor$ is the desired Poisson variate. Otherwise U is rejected and we have to repeat the whole procedure with a new pair (U, V) uniformly distributed under $h(u)$.

PTRD (**T**ransformed **R**ejection method with **D**ecomposition or, using our notation, with immediate acceptance, valid for $\mu \geq 10$) by W. Hörmann [7].

This algorithm is based on the same technique as the algorithm PTRS but uses decomposition instead of squeezing.

Here the relatively large rectangle below the transformed Poisson histogram function $f(u) = f(G^{-1}(u))(G^{-1})'(u)$ is used as immediate acceptance region (and is no longer interpreted as a horizontal lower squeeze line). This modification to algorithm PTRS remarkably accelerates the calculation of the desired Poisson variate, since candidates from the rectangle are now accepted instantly after their generation without any further costs, like producing a second uniform number.

7.3 Sampling Methods for the Binomial Distribution

Search Algorithms

BINV (**Inv**ersion, implemented as low index sequential search).

This algorithm is based on the same logic, that is applies the same technique as the algorithm PINV. However, the initial probability here is $p_0 = (1-p)^n$. Further the sequential search must stop when the search index is on the point of exceeding n, in addition to that situation, where the search index is about to pass the safety bound (which guarantees that only samples with significant probabilities are returned by the algorithm). Since the execution time of this algorithm is proportional to the mean of the input distribution, we confine it to parameters n and $0 < p \leq \frac{1}{2}$. However, supplying the inversion with parameters n and $\min(p, 1-p) \leq \frac{1}{2}$ and returning the obtained sample K if $p \leq \frac{1}{2}$ and the transformed sample $n - K$ if $p > \frac{1}{2}$ yields again a sampling routine that is valid for all $0 < p < 1$ (see beginning of Section 6), but which is significantly faster than the pure inversion routine in case $p \gg \frac{1}{2}$.

BMDU (**M**odal **D**own–**U**p sequential search, which is valid for $n \geq 1,\ 0 < p \leq \frac{1}{2}$) by C. D. Kemp [17].

This algorithm is based on the same logic as the algorithm PMDU. Like PMDU it does not calculate the modal probability

$$p_n(m) = \frac{n!}{m!\,(n-m)!}\, p^m\,(1-p)^{n-m}\,, \qquad m = \lfloor (n+1)p \rfloor$$

with the help of the external function `logfak`. It instead employs the Stirling approximation (to $\log k! = \log \Gamma(k+1),\ k = n,\ k = m,$ and $k = n - m$) inline — this saves some logarithms — and determines the difference between $\log k!$ and the corresponding first order Stirling approximation by `stirlc`. Analogous to algorithm BINV the pure upward sequential search from index $2m+1$ — coming after an unsuccessful alternately down and upward search from the mode m — must be terminated when the current search index exceeds n or the safety bound. Here we note that if $p = \frac{1}{2}$ and $n = 2l+1$ is uneven we have to decrease m by 1 to guarantee that $2m \leq n$ (otherwise we would have $m = l + 1 > n/2$). As a consequence the down–up search loop need not check whether the search index passed n or not.

Acceptance–Rejection Algorithms

BRUA (**R**atio of **U**niforms method with an **A**pproximation to the slope parameter of the optimal table mountain hat $h(x)$, valid for $\mu = np \geq 1,\ 0 < p \leq \frac{1}{2}$) by E. Stadlober [23].

The approximations to the optimal location and slope used for the Poisson distributions proved appliable to the more general binomial distributions. Except for the essential differences — the generalization of the shape parameters a and $\hat{s}$ and the use of the standardized binomial probability function and its recurrence relationship function in place of the corresponding Poisson functions — the binomial algorithm is nearly identical to PRUA (BRUA still uses more set–up constants to allow for a faster calculation of the above–mentioned functions, and rejects instantly all candidates $K,\ K = \lfloor X \rfloor$ and X a sample from $h(x)$, greater than n, in addition to the samples greater than the safety bound and less than zero).

BRUE (**R**atio of **U**niforms method which uses the **E**xact value of the optimal slope for the definition of the table mountain hat $h(x)$, valid for $\mu = np \geq 1,\ 0 < p \leq \frac{1}{2}$) by E. Stadlober [23].

The algorithm is based on a symmetric table mountain hat function $h(x)$ with the, related to the shift $a = \mu + \frac{1}{2}$, optimal slope parameter s. Binomial deviates are generated by the sampling step used in algorithm BRUA, that is the generation steps of the two algorithms are identical. The set–up step differs from that of algorithm BRUA, though, but only in the calculation of the slope parameter of $h(x)$.

BAMS (**A**tkinson's algorithm, **M**odified with respect to the shape parameters of the hat $h(x)$ and complemented by **S**queezes, applied to the binomial distributions, valid for $\mu = np \geq 3,\ 0 < p \leq \frac{1}{2}$) by J. Thurner [26].

J. Thurner realized that it had to be possible to apply Atkinson's, strictly speaking, fundamental sampling idea — which is based on a hat $h(x)$ proportional to the density of the general logistic distribution, and which is obviously applicable to all log–concave distributions — to the little more complicated binomial distributions. He generalized the location $\hat{a}$ and the scale $\hat{b}$ of algorithm PAMS to the $b(n,\, p)$ distributions — instead of the one parameter μ in the Poisson case there are the two parameters $n,\, p$ now $\hat{a}$ and $\hat{b}$ will depend on. Through extensive calculations he established that

$$\hat{a} = np + \tfrac{1}{5} + \tfrac{3}{5}p \qquad \text{and} \qquad \hat{b} = 1.5/\sqrt{np(1-p)}\,,$$

$\hat{a}$ improved by an additional term in p, are close approximations to the optimal a and b. The resulting hat size α — presenting the efficiency of $h(x)$ — which can be determined only numerically is approximated safely by $\hat{\alpha} = 1.15 + (1.6 + 2p)/(np)$ (compare with the Poisson case, where $\hat{\alpha} = 1.15 + 1.6/\mu$) which definitely defines the scaling of the hat. That is, $h(x) = \hat{\alpha}\, g(x)$, where $\hat{\alpha} \geq \alpha$ and $g(x)$ is the density of the logistic distribution.

BDEX (**D**ouble **Ex**ponential method with squeezes based on the Stirling approximation, valid for $np(1-p) \geq 4$) by J. H. Ahrens & U. Dieter [1].

The algorithm — an extension of the corresponding Poisson method PDEX to the binomial distributions — is based on a double–exponential hat $h(x)$ with location $a = np + \frac{1}{6} + \frac{2}{3}p$ (a the mode) and scale $s = \sqrt{np(1-p)}$ (s the stretching factor). The optimal hat size α which can be determined only numerically is approximated safely by $\hat{\alpha} = \sqrt{2/\pi}\, e^{\,1/2 \,+\, 1/(2s) \,+\, 1/(10s^2)}$ ($\hat{\alpha} \geq \alpha$). Again our implementation differs from the original algorithm in the reference in three points:

1. The acceptance test is implemented directly and only in double precision.
2. Instead of an external function for $\log k!$ we employ an external function that calculates the correction term of the first order Stirling approximation, like `stirlc`.
3. The quantities $\log k!$ and $\log(n-k)!$ in the logarithm of the probability function $p_n(k)$ are approximated by Stirling's formula with argument $k+1$ instead of k and argument $n-k+1$ instead of $n-k$, respectively. This allows us to squeeze the correction term of the first order Stirling approximation by two consecutive members of the series in $\frac{1}{k+1}$ (in $\frac{1}{n-k+1}$) even when $k = 0$ (when $k = n$). The quantity $\log n!$ on the contrary is calculated by Stirling's formula with argument n, as $n \geq 16$ holds at least. We use 0 as lower and $\frac{1}{12(k+1)}$, $\frac{1}{12(n-k+1)}$ and $\frac{1}{12n}$, respectively, as upper bounds; they lead to pre–tests which circumvent the full length correction terms.

BNEX (**N**ormal **Ex**ponential method which totally avoids the approximate calculation of factorials, or logarithms of factorials, valid for $n \geq 1,\ 0 < p \leq \frac{1}{2}$) by L. Devroye & A. Naderisamani [5].

The algorithm combines the 'first waiting time' method [4] — which counts the number of random variates, except the last one, from the geometric distribution with parameter p (the current probability of success) that must be generated until their sum exceeds n (the current number of Bernoulli trials), that is the number of successes in n Bernoulli trials — with acceptance–rejection with half normal and exponential functions. The first waiting time algorithm is carried out to generate the desired binomial variate if $\mu = np$ is small (if $\mu < 7$ in our routine) or, otherwise, to generate the residual binomial number L in case of non–integral μ. In the second case actually two variates, the decisive sample $K \sim b(n,\, p_-)$ and subsequently, conditional on K, the residual sample $L \sim b(n - K,\, (p - p_-)/(1 - p_-))$, where $p_- = \lfloor np \rfloor / n$, are produced. The main algorithm which generates K is based on two half normal envelopes which cover the standardized and shifted probabilities $f(k) = p_n(m+k)/p_n(m)$ — now $k \in [-m,\, n-m]$ — of the $b(n,\, p_-)$ distribution in the centre of the corresponding histogram $f(x)$, and on two exponential functions of which the first one majorizes the left tail (below a breakpoint $-d_1$) and the second one majorizes the right tail (upward a breakpoint $+d_2$) of $f(x)$. All candidates X from the half normal distributions outside the centre intervals $[-d_1,\, 0)$ and $[0,\, +d_2)$ are rejected instantly. This also applies to all exponential samples X, generated inline by inversion, outside the domain $[-m,\, n - m + 1)$ of the histogram $f(x)$. To compensate the shift by m to the left, the algorithm has to return evidently $K = m + \lfloor X \rfloor$ instead of $\lfloor X \rfloor$ (this is truncation to the left, even for negative values of X).

The algorithm contains upper and lower squeeze tests which help to avoid the evaluation of the factorials in

$$f(k) = (p_-/(1 - p_-))^k\,(m!\,(n - m)!)/((m + k)!\,(n - m - k)!), \quad m = np_-$$

most of the time. Otherwise, if the squeezes lead neither to acceptance nor to rejection, $f(k)$ will be calculated, in accordance with the above representation, directly by a loop statement which is repeated $|k|$ times. (No Stirling approximation is used for the calculation of the factorials.)

B2PE (**2 P**oint method using **E**xponential tail envelopes, complemented by squeezes, valid for $(n + 1) \times \min(p,\, 1 - p) \geq 5$) by V. Kachitvichyanukul & B. W. Schmeiser [10].

The algorithm is a special version of the hypergeometric algorithm H2PE — which is based on a three–part envelope $h(x)$ (rectangle in the centre region, exponentials in the tails) with minimal scaling factor — made fit to the binomial distributions.

The standardized binomial probability $f(K)$, needed in the acceptance–rejection test, is computed recurrently from the rescaled modal probability $f(m) = 1$, whenever one of the two quantities m (the mode) or K (the candidate) is small (near

0; more precisely, if $m < 100$ or $K \leq 50$) or really large (near n; more precisely, if $m > n - 100$ or $K \geq n - 50$), to circumvent its direct calculation. Under the opposite circumstances there are upper and lower bounds which help to avoid the evaluation of $\log f(K)$ most of the time. However, as we found out, these squeeze tests — which contain quite a few auxiliary quantities — scarcely save time.

BTPE (**T**riangle **P**arallelogram **E**xponential method with a triangular immediate acceptance region, valid for $\mu = np \geq 10$, $0 < p \leq \frac{1}{2}$) by V. Kachitvichyanukul & B. W. Schmeiser [12], [13].

The algorithm is based on a hat function $h(x)$ made up of an upside down V (which covers the centre region) and two exponential functions (which cover the tails). It also includes a triangular immediate acceptance region which speeds up the generation of binomial variates from the centre of the distribution. Contrary to the hat $h(x)$ of the algorithm B2PE, the hat function here does not touch the histogram $f(x)$, since crude approximations in place of some exact and time–consuming constants are used in its definition. Just for this the initialization time of algorithm BTPE is considerably smaller than the set–up time of algorithm B2PE (see Table 21).

If the candidate K is near the mode m (say $|K - m| \leq 20$) or far away from m (in a distance larger than about half the variance of the distribution), the standardized probability $f(K)$, needed in the acceptance–rejection test in case K is not immediately accepted, is computed recurrently from the rescaled modal probability $f(m) = 1$. The first restraint serves the purpose of accelerating the calculation of $f(K)$. The second one guarantees that for samples K at which no particular lower and upper bounds on $\log f(k)$ — based on the De Moivre limiting theorem — exist, no incorrect squeeze tests are carried out, even $f(k)$, $k = K$, is not evaluated at all. Furthermore, the final acceptance–rejection test utilizes the Stirling approximation inline, it is implemented in the way W. Hörmann proposed in [8]. There he states the test condition with the minimum number of logarithms. The remaining differences between $\log k!$ and $\log(n-k)!$ ($k = K$ and $k = m$) and the corresponding first order Stirling approximations with argument $k+1$ and $n-k+1$, respectively, are determined again by `stirlc`.

BTRS (**T**ransformed **R**ejection method with **S**queezes, valid for $n \times \min(p, 1-p) \geq 10$) by W. Hörmann [8].

This algorithm is of the same logic as the corresponding Poisson algorithm PTRS. It uses transformed rejection — based on the same type of approximation to the inverse distribution function $F^{-1}(u)$ — combined with the squeeze technique to calculate the desired sample. However, compared to PTRS, no upper bounds which might accelerate the rejection of candidates from the tails of the hat are included. Besides, W. Hörmann fixed the location parameter c at $c = np + \frac{1}{2}$ — he did not approximate the optimal values of all three parameters by close expressions as in the Poisson case, where he found $c = \mu + 0.445$ to be a proper approximation — and determined good approximations to the corresponding optimal parameters a and b in the region $n \times \min(p, 1-p) \geq 10$.

BTRD (**T**ransformed **R**ejection method with **D**ecomposition or, using our notation, with immediate acceptance, valid for $n \times \min(p, 1-p) \geq 10$) by W. Hörmann [8].

This algorithm is of the same logic as the algorithm BTRS, except that it uses decomposition instead of squeezing. For the advantages of this modification we refer to look up the more detailed description of the algorithm PTRD.

Most of the above binomial algorithms are feasible for $0 < p \leq \frac{1}{2}$ only. However, as we already mentioned at the beginning of Section 6, they can be extended in an easy way to all $0 < p < 1$: replace the parameter p with $\min(p, 1-p)$ and return the obtained sample K if $p = \min(p, 1-p)$, that is if $p \leq \frac{1}{2}$, and the transformed sample $n - K$ otherwise.

7.4 Comparison of Algorithms

All algorithms were written as C–routines, compiled and linked in Borland C++, Version 3.0, with the standard compiler options (Model *compact*, Floating point *emulation*, Instruction set *8088/8086*, and Optimization *default*). The execution and initialization times following below were all taken on a PC/AT 486–33MHz. They are based on samples of 100 000 variates for each parameter combination.

It takes about 19 μsec to generate one uniform number with `krand1` which is approximately the time for calculating $y = e^x$. The elementary operations addition and subtraction need about 0.45 μsec, whereas a multiplication requires about 0.60 μsec and a division about unbelievable 2.5 μsec. The square root consumes about 3.6 μsec and the natural logarithm about 16 μsec. To complete the above times, below in Table 3 we specify the execution times and, additionally, the memory requirements of the four supporting routines `krand1`, `normal`, `logfak` and `stirlc`.

Table 3: Execution times [μsec/call] and memory requirements [bytes] of the supporting functions `krand1`, `normal`, `logfak`, and `stirlc`.

Supporting function	Execution time	Memory requirements
`krand1`	19.2	1008
`normal`	38.0	2829
`logfak`	28.0	1259
`stirlc`	7.7	1023

We put stress on the fact that the delivery of a tabulated value by `logfak` or `stirlc` (which happens when $k < 30$) expends about 1.9 μsec only.

Our algorithms PPRS^r and BPRS^r seem to be faster than their competitors for large means ($\mu \geq 20$) — just compare the execution times in Table 5 and in Table 8 to Table 20.

In this respect, when large means are only considered, the next best Poisson algorithms are PDAC, PTRD, PTPE, PPRB^r and PTRS. All of them, except PTRS, contain immediate acceptance. The next best binomial algorithms BTRD, BTPE, BPRB^r, and BTRS use immediate acceptance as well, except for BTRS which, on the contrary, contains a very large squeeze acceptance rectangle. For small means the search algorithms beat the acceptance–rejection methods — compare the execution times in Table 4 and in Table 7. But PPRS^r, PPRB^r, PDAC and PTRD in the Poisson case, and BPRS^r, BPRB^r and BTRD in the binomial case just succeed in competing with the simple inversion routines PINV and BINV, respectively, for means about 10. The tabulated inversion in PDAC yields the smallest execution times in the range $\mu < 10$, closely followed by KPLO and PMDU. The search from the mode is almost always faster than the search from the lowest index both in the Poisson and the binomial case. However, the set–up times of the first search method are about twice as large as of the second method. For larger means the search from the mode is preferable because of the smaller execution time (which increases with σ, σ^2 the variance of the distribution, whereas the execution time of the low index sequential search grows with μ) and, what is more important, because of the smaller number of recurrence steps carried out, on the average, as each of them is a potential source of roundoff error. For the above remarks we propose to use PINV and BINV only when $\mu < 25$, and PMDU and BMDU only when $\mu < 250$. KPLO can be recommended for its whole validity range, whereas KPOI may be used for $10 \leq \mu < 1000$ only.

To sum up it can be said that the tabulated inversion in algorithm PDAC is the most suitable method to generate Poisson variates with parameter $0 < \mu < 10$. With respect to the wider range $0 < \mu \leq 20$ KPLO seems to be the appropriate method. Nevertheless, there is no reason why we should not fall back upon the algorithms PINV and PMDU to produce Poisson variates with parameter $0 < \mu \leq 5$ and $0 < \mu \leq 10$, respectively. For larger means PPRS^r is the favourite algorithm, since it yields the smallest execution times in the range $\mu \geq 20$. The times also do not depend very much on the shape parameter. Considering speed only, we can further recommend the algorithms PDAC and PTRD. If simplicity and speed are the dominant criteria, we could recommend the combinations PINV/PTRS, PINV/PTRD, and PINV/PRUA with cut off points $\mu^* \approx 10$.

If μ changes frequently, algorithms with low set–up costs should be used. Viewed from this standpoint, the combinations KPLO/PDAC$_2$ and KPLO/PTRD are the best choices (consider the figures in Table 6). Algorithm PPRS^r — which has the highest set–up costs — combined with PDAC$_1$, on the other hand, is a proper method to produce batches of samples from a fixed Poisson distribution.

The above remarks in principle apply also to the binomial case. However, note that some of the above Poisson algorithms have no pendant. The combinations we proposed in the two preceding paragraphes therefore reduce to BINV/BTRS, BINV/BRUA (simplicity and speed are the dominant attributes; here we refuse to recommend the combination BINV/BTRD, since BTRD is a little bit too long), BINV/BTRD (parameters change frequently) and BMDU/BPRS^r (batches of samples from a fixed $b(n, p)$ distribution are generated). In the first two combinations the cut off point $\mu^* = np$, $p \leq \frac{1}{2}$, should be a little bit larger than 10 (we suggest to use $\mu^* = 12$).

In addition to the execution and initialization times the average number N of uniforms needed to generate the desired variate is a further important factor in the appraisal and classification of a sampling method. Algorithms based on sequential search need exactly one uniform — or a second one if the search yields no result because of passing the safety bound or because of rounding errors. In Appendix D we will display the average number α of samples X from $h(x)$, until the truncated candidate $K = \lfloor X \rfloor$ is accepted as the desired Poisson/binomial deviate, for all acceptance–rejection algorithms. Subsequently we will state the quantity N for those methods which violate the relationship $N = 2\alpha$. Algorithms based on pure acceptance–rejection with closed invertible hat functions satisfy this condition. They require one uniform number for the candidate K and a second one for the test condition. However, there are routines which include fast rejection — the candidate K which uses up exactly one uniform is outside the domain of the distribution, then it can be rejected immediately — or immediate acceptance, or which are based on hat functions that do not get along with one uniform to generate the hat sample.

N is stated for all Poisson algorithms, though many of them satisfy the condition $N = 2\alpha$, but only for the binomial algorithms BAMS, BNEX, BTRD, and BPRS^r which meet one of the above conditions. Algorithms PTRS, PTRD and algorithms BTRS, BTRD are based on the same hat $h(x)$, thus they require the same average number α of samples X from $h(x)$ for a Poisson/binomial variate. The first algorithms rest upon pure acceptance–rejection, that is $N = 2\alpha$, the second ones, however, still contain immediate acceptance.

For the above it suffices to present the characteristics α and N of the algorithms PTRD and BTRD only. Our basic patchwork methods (marked with the letter B) are also completely omitted, since they are identical to the final methods with squeezes, except that they contain no squeeze acceptance. That is $P(S) = 0$, where $P(S)$ is the probability of accepting K by means of a lower squeeze function, given that K is accepted. In addition to our favourite methods marked with ^{r}, where m is part of the reflection to the **r**ight, we still consider the two other fundamental methods. They are marked with ^{l} (m is part of the transformation to the **l**eft) and ^{c} (m remains in the **c**entre of the distribution and is interpreted as immediate acceptance region). Apart from the characteristics α and N, we will present $P(I)$, the probability of immediate acceptance, and $P(Q) = P(I) + P(S)$, the probability of quick acceptance, of all three fundamental patchwork methods.

Table 4: Execution times [μsec/variate] of the Poisson algorithms for small means ($0 < \mu \leq 10$).

Algorithm \ μ	0.2	0.5	1.0	2.0	3.5	5.0	10.0
PINV	27	28	32	38	47	56	87
PMDU	26	28	32	36	41	44	53
KPLO	32	33	34	36	37	38	42
KPOI	—	—	—	—	—	—	42
PRUA	—	—	127	118	117	112	112
PRUE	—	—	127	116	114	110	110
PAMS	—	—	—	151	130	122	116
PDEX	—	—	—	—	—	178	168
PNEX	—	—	—	—	—	—	144
PDAC	26	27	28	30	32	32	68
P2PE	—	—	—	—	—	88	92
PTPE	—	—	—	—	—	—	107
PTRS	—	—	—	—	—	—	96
PTRD	—	—	—	—	—	—	87
PPRB^r	—	—	—	—	77	61	56
PPRS^r	—	—	—	—	72	58	52

Table 5: Execution times [μsec/variate] of the Poisson algorithms for large means ($\mu \geq 10$).

Algorithm \ μ	10	20	50	150	500	2500	25000
PINV	87	149	334	960	$\gg$	$\gg$	$\gg$
PMDU	53	66	91	141	237	503	1540
KPLO	42	50	—	—	—	—	—
KPOI	42	48	59	83	130	255	769
PRUA	112	114	117	125	124	121	120
PRUE	110	113	117	125	124	122	120
PAMS	116	116	120	128	128	128	126
PDEX	168	163	156	152	149	148	146
PNEX	144	129	120	114	110	107	106
PDAC	68	65	64	62	61	60	60
P2PE	92	98	112	138	137	137	136
PTPE	107	99	90	71	66	65	65
PTRS	96	87	88	81	77	74	73
PTRD	87	75	74	66	60	58	56
PPRB^r	56	54	66	68	69	70	70
PPRS^r	52	48	53	53	52	52	52

Table 6: Initialization times [μsec] and memory requirements [bytes] (without the memory requirements of the supporting external functions) of the Poisson algorithms.

Algorithm	Initialization times		Memory requirements
	$m < 30$	$m \geq 50$	of compiled code
PINV	30 – 35	30 – 35	1263
PMDU	60 – 65	65 – 70	2177
KPLO	10 – 30	—	2949
KPOI	30 – 35	15 – 35	3095
PRUA	35 – 40	55 – 65	2617
PRUE	65 – 75	110 – 120	3032
PAMS	75 – 80	55 – 105	2625
PDEX	14 – 16	12 – 14	1870
PNEX	105 – 110	105 – 110	4659
PDAC_1	20 – 80	—	4659
PDAC_2	10 – 12	10 – 12	
P2PE	130 – 135	205 – 215	4366
PTPE	40 – 45	40 – 45	4559
PTRS	20 – 25	15 – 20	2251
PTRD	30 – 35	30 – 35	2661
PPRB^r	185 – 225	335 – 345	5056
PPRS^r	190 – 230	340 – 350	6056

PDAC_1 : tabulated inversion ($\mu < 10$)

PDAC_2 : acceptance–complement ($\mu \geq 10$)

KPOI and PDAC_2, although programmed in double precision, are mere single precision routines.

KPOI uses approximations which are correct in a single precision environment.

$\mathbf{PDAC_2}$ calculates the probabilities of the discrete normal distribution in single precision only.

Table 7: Execution times [μsec/variate] of the binomial algorithms for small means ($0 < \mu \leq 10$).

Algorithm	$n \setminus \mu$	0.2	0.5	1.0	2.0	3.5	5.0	10.0
BINV	20	30	32	35	40	50	58	87
	40	30	31	35	40	50	59	88
	50000	30	32	35	41	50	58	88
BMDU	20	30	32	36	41	46	47	50
	40	30	32	36	41	46	49	56
	50000	30	32	37	41	47	50	60
BRUA	20	—	—	131	121	119	116	115
	40	—	—	130	121	120	115	115
	50000	—	—	130	120	120	115	115
BRUE	20	—	—	130	114	110	106	105
	40	—	—	129	116	114	109	109
	50000	—	—	130	117	116	112	114
BAMS	20	—	—	—	—	141	133	123
	40	—	—	—	—	137	129	121
	50000	—	—	—	—	132	125	119
BDEX	20	—	—	—	—	—	—	220
	40	—	—	—	—	—	220	210
	50000	—	—	—	—	—	213	202
BNEX	20	57	69	89	130	190	250	246
	40	57	69	90	130	190	250	251
	50000	57	69	89	130	189	250	276
B2PE	20	—	—	—	—	—	89	91
	40	—	—	—	—	—	90	94
	50000	—	—	—	—	—	89	93
BTPE	20	—	—	—	—	—	—	140
	40	—	—	—	—	—	—	142
	50000	—	—	—	—	—	—	147
BTRS	20	—	—	—	—	—	—	109
	40	—	—	—	—	—	—	114
	50000	—	—	—	—	—	—	119
BTRD	20	—	—	—	—	—	—	92
	40	—	—	—	—	—	—	87
	50000	—	—	—	—	—	—	85
BPRB^r	20	—	—	—	—	77	62	59
	40	—	—	—	—	92	72	63
	50000	—	—	—	—	96	76	71
BPRS^r	20	—	—	—	—	74	60	56
	40	—	—	—	—	82	65	55
	50000	—	—	—	—	85	69	62

Table 8: Execution times [μsec/variate] of algorithms BINV and PINV.

Algorithm	$n \setminus \mu$	10	20	50	150	500	2500	25000
BINV	20	88						
	40	89	148					
	100	89	149	331				
	300	89	149	330	942			
	1000	89	149	330	941	$\gg$		
	5000	89	149	331	940	$\gg$	$\gg$	
	50000	89	149	330	940	$\gg$	$\gg$	$\gg$
PINV	∞	87	149	334	960	$\gg$	$\gg$	$\gg$

Table 9: Execution times [μsec/variate] of algorithms BMDU (C. D. Kemp, 1986) and PMDU (C. D. Kemp, 1982).

Algorithm	$n \setminus \mu$	10	20	50	150	500	2500	25000
BMDU	20	50						
	40	55	60					
	100	58	69	80				
	300	60	72	95	119			
	1000	60	74	100	148	195		
	5000	60	74	102	156	254	405	
	50000	60	74	103	158	265	550	1222
PMDU	∞	53	66	91	141	237	503	1540

Table 10: Execution times [μsec/variate] of algorithms BRUA and PRUA (Stadlober, 1989a).

Algorithm	$n \setminus \mu$	10	20	50	150	500	2500	25000
BRUA	20	115						
	40	115	116					
	100	116	119	121				
	300	115	119	125	133			
	1000	115	119	126	138	146		
	5000	115	119	127	139	149	154	
	50000	115	119	127	139	149	155	158
PRUA	∞	112	114	117	125	124	121	120

Table 11: Execution times [μsec/variate] of algorithms BRUE and PRUE (Stadlober, 1989a).

Algorithm	$n \setminus \mu$	10	20	50	150	500	2500	25000
BRUE	20	106						
	40	109	109					
	100	113	115	116				
	300	114	117	123	131			
	1000	114	118	125	137	143		
	5000	114	118	125	140	148	153	
	50000	114	118	125	139	148	155	158
PRUE	∞	110	113	117	125	124	122	120

Table 12: Execution times [μsec/variate] of algorithms BAMS (Thurner, 1990) and PAMS (Stadlober, 1989b).

Algorithm	$n \setminus \mu$	10	20	50	150	500	2500	25000
BAMS	20	122						
	40	121	117					
	100	120	118	119				
	300	119	119	124	130			
	1000	119	119	125	137	142		
	5000	119	119	126	138	147	153	
	50000	119	119	126	139	148	155	158
PAMS	∞	116	116	120	128	128	128	126

Table 13: Execution times [μsec/variate] of algorithms BDEX and PDEX (Ahrens & Dieter, 1980).

Algorithm	$n \setminus \mu$	10	20	50	150	500	2500	25000
BDEX	20	219						
	40	210	206					
	100	206	199	195				
	300	204	197	191	187			
	1000	204	196	190	185	183		
	5000	203	196	189	184	181	180	
	50000	201	194	188	183	181	179	178
PDEX	∞	168	163	156	152	149	148	146

Table 14: Execution times [μsec/variate] of algorithms BNEX (Devroye & Naderisamani, 1980) and PNEX (Devroye, 1986).

Algorithm	$n \setminus \mu$	10	20	50	150	500	2500	25000
BNEX	20	247						
	40	252	208					
	100	274	220	176				
	300	349	306	264	154			
	1000	276	224	187	239	141		
	5000	277	224	188	242	145	133	
	50000	276	224	188	161	145	134	127
PNEX	∞	144	129	120	114	110	107	106

Table 15: Execution times [μsec/variate] of algorithms B2PE and P2PE (Kachitvichyanukul & Schmeiser, 1985).

Algorithm	$n \setminus \mu$	10	20	50	150	500	2500	25000
B2PE	20	92						
	40	94	94					
	100	93	100	104				
	300	93	100	112	160			
	1000	93	100	114	159	158		
	5000	93	100	115	160	157	157	
	50000	93	100	114	160	158	156	156
P2PE	∞	92	98	112	138	137	137	136

Table 16: Execution times [μsec/variate] of algorithms BTPE (Kachitvichyanukul & Schmeiser, 1988b) and PTPE (Schmeiser & Kachitvichyanukul, 1982).

Algorithm	$n \setminus \mu$	10	20	50	150	500	2500	25000
BTPE	20	140						
	40	142	125					
	100	144	127	108				
	300	146	126	111	95			
	1000	147	126	110	94	84		
	5000	146	126	110	95	81	76	
	50000	147	126	109	95	81	74	72
PTPE	∞	107	99	90	71	66	65	65

Table 17: Execution times [μsec/variate] of algorithms BTRS (Hörmann, 1993) and PTRS (Hörmann, 1992).

Algorithm	$n \setminus \mu$	10	20	50	150	500	2500	25000
BTRS	20	110						
	40	115	100					
	100	121	112	118				
	300	120	110	110	104			
	1000	119	109	108	98	94		
	5000	120	109	108	98	91	88	
	50000	119	109	107	97	91	86	84
PTRS	∞	96	87	88	81	77	74	73

Table 18: Execution times [μsec/variate] of algorithms BTRD (Hörmann, 1993) and PTRD (Hörmann, 1992).

Algorithm	$n \setminus \mu$	10	20	50	150	500	2500	25000
BTRD	20	92						
	40	87	84					
	100	85	82	80				
	300	85	81	80	78			
	1000	85	81	81	76	72		
	5000	85	81	80	76	70	65	
	50000	85	81	80	75	70	64	60
PTRD	∞	87	75	74	66	60	58	56

Table 19: Execution times [μsec/variate] of algorithms BPRB^r and PPRB^r (Zechner, 1993).

Algorithm	$n \setminus \mu$	10	20	50	150	500	2500	25000
BPRB^r	20	59						
	40	63	58					
	100	70	71	83				
	300	70	72	83	85			
	1000	70	73	83	86	86		
	5000	71	74	83	86	87	89	
	50000	70	73	83	86	87	89	89
PPRB^r	∞	56	54	66	68	69	70	70

Table 20: Execution times [μsec/variate] of algorithms BPRS^r and PPRS^r (Zechner, 1993).

Algorithm	$n \setminus \mu$	10	20	50	150	500	2500	25000
BPRS^r	20	56						
	40	55	54					
	100	60	59	58				
	300	61	61	60	60			
	1000	61	62	60	60	60		
	5000	61	62	60	60	59	59	
	50000	61	62	60	60	59	59	59
PPRS^r	∞	52	48	53	53	52	52	52

Table 21: Initialization times [μsec] and memory requirements [bytes] (without the memory requirements of the supporting external functions) of the binomial algorithms.

Algorithm	Initialization times			Memory requirements of compiled code
	$m < 30$		$m \geq 50$	
	$n - m < 30$	$n - m \geq 50$		
BINV	55 – 60	55 – 60	55 – 60	2073
BMDU	100 – 110	110 – 115	115 – 125	3514
BRUA	40 – 50	70 – 75	90 – 100	3371
BRUE	75 – 90	125 – 140	175 – 190	3875
BAMS	105 – 135	155 – 165	110 – 185	3706
BDEX	19 – 21	19 – 21	17 – 19	2488
BNEX_1	22 – 24	20 – 23	—	6850
BNEX_2	175 – 195	170 – 190	165 – 185	
B2PE	150 – 160	205 – 235	310 – 315	5667
BTPE	45 – 50	45 – 50	45 – 50	5674
BTRS	35 – 40	45 – 55	35 – 55	2741
BTRD	30 – 32	30 – 32	30 – 35	4889
BPRB^r	215 – 240	305 – 365	485 – 500	5851
BPRS^r	220 – 245	315 – 370	490 – 510	6880

BNEX_1: first waiting time algorithm $(n \times \min(p, 1-p) < 7)$

BNEX_2: acceptance–rejection based on two half normal envelopes, combined with first waiting time algorithm $(n \times \min(p, 1-p) \geq 7)$

8 Application to the Hypergeometric Distributions

A random variable K_N has hypergeometric distribution with parameters N, M, and n $(K_N \sim H(N,\,M,\,n))$, $1 \le M \le N$, $1 \le n \le N$, if

$$p_N(k) = P(K_N = k) = \frac{\binom{M}{k}\binom{N-M}{n-k}}{\binom{N}{n}}\,, \qquad \max\,(n-N+M,\,0) \,\le\, k \,\le\, \min\,(n,\,M). \tag{40}$$

Mean and variance of a $H(N,\,M,\,n)$ distributed random variable K_N are

$$\mathrm{E}(K_N) = \mu = n\,\frac{M}{N}\,, \qquad \mathrm{Var}(K_N) = \sigma^2 = n\,\frac{M}{N}\left(1-\frac{M}{N}\right)\left(1-\frac{n-1}{N-1}\right). \tag{41}$$

The recurrence relationship for the probabilities (40) is given by

$$p_N(k) = \frac{(M-k+1)(n-k+1)}{k\,(N-M-n+k)}\,p_N(k-1)\,. \tag{42}$$

The numerator is monotonically decreasing, and the denominator is monotonically increasing on the integers $k \ge \max(n-N+M,\,0)$, and the above fraction, the recurrence relationship function for $p_N(k)$, is a monotonically decreasing function. Thus the probability function $p_N(k)$ is log–concave, and its tails can be covered by exponential functions.

The distribution remains the same if parameters M and n are interchanged, i. e.

$$p_N(k;\;N,\,M,\,n) \;=\; p_N(k;\;N,\,n,\,M)\,. \tag{43}$$

There are three other parameter transformations which do not change the shape of the probability function in principal. We state them in equation (44) below.

$$\begin{aligned} p_N(k;\;N,\,M,\,n) &= p_N(\;n-k;\;N,\,N-M,\,n) \\ &= p_N(M-k;\;N,\,M,\,N-n) \\ &= p_N(\,N-M-n+k;\;N,\,N-M,\,N-n)\,. \end{aligned} \tag{44}$$

The first two identities describe a reflection of $p_N(k)$ at $k = n/2$ and at $k = M/2$, respectively. The last identity describes a shift of $p_N(k)$ by $N-M-n$ to the right.

The four relationships above allow us to restrict our attention to designing an algorithm which produces samples only from hypergeometric distributions with shape parameters satisfying $1 \le n \le M \le N/2$. Samples from the general $H(N,\,M,\,n)$ distributions, where $1 \le M \le N$, $1 \le n \le N$, are obtained by such a routine just valid for approximate an eighth of the total parameter range as follows.

Algorithm HYUT (Utility routine for generating $H(N,\,M,\,n)$ variates)

(a) Preparation of the parameters:

`If` $(n \le N/2)$ `then Set` $n^* = n$;
`else Set` $n^* = N - n$; $\qquad\qquad n^* \le N/2$

```
    Endif
    If (M ≤ N/2) then Set M* = M;
                 else Set M* = N − M;                M* ≤ N/2
    Endif
    If (M* < n*) then
       Set t = n*, n* = M*, M* = t;
    Endif                                            n* ≤ M*
```

(b) Generate a random variate K from $H(N, M^*, n^*)$

(c) Transformation to variate K from $H(N, M, n)$:

```
    If (n ≤ N/2) then
       If (M ≤ N/2) then return K;
                    else return K = n − K;
       Endif
    else
       If (M ≤ N/2) then return K = M − K;
                    else return K = n − N + M + K;
       Endif
    Endif
```

To keep the calculations as simple as possible and the formulas as short as possible, we introduce the quantities

$$\mathcal{N} = N + 2, \quad \nu_p = n + 1, \quad p = (M+1)/(N+2), \quad \text{and} \quad \nu = \nu_p\, p\,. \tag{45}$$

ν represents the real mode of the distribution, and p represents the probability of success in the limiting binomial distributions ($N,\, M \to \infty,\; M/N$ fixed), as we will outline in more detail below.

The probability function (40) is unimodal. There is one mode at $m = \lfloor \nu \rfloor$ if ν is not an integer, and there are two adjacent modes at $m-1$ and at m if ν is an integer. Thus (40) fulfils the basic requirement of Lemma 1.

For $N \to \infty$ and p fixed, (40) tends to the $b(n,\, p)$ distribution. In other words,

$$\lim_{\substack{N,\, M \to \infty \\ p \;\text{ fixed}}} p_N(k) \;=\; \binom{n}{k} p^k (1-p)^{n-k}\,, \qquad k = 0,\, 1,\, \ldots\, n\,. \tag{46}$$

Therefore it should be possible to represent the recurrence formula (42), such that it implies formula (33). Using the quantities defined in (45), (42) may be written as follows.

$$p_N(k) = r(k)\, p_N(k-1)\,, \qquad r(k) = 1 - \left(1 - \frac{\nu}{k}\right)\Big/\left(1 - p - \frac{\nu_p - k}{\mathcal{N}}\right). \tag{47}$$

If $\mathcal{N} = \infty$ (ν_p and p fixed), (47) obviously coincides with the Poisson/binomial recurrence relationship function (33). Consequently the following considerations apply to the Poisson and binomial distributions as well.

Again, our first step is to specify the normalizing constants $c_N(k)$ and $c_S(i,\,k)$. Here we set

$$c_N(k) \,=\, k\Big(k+1\Big)\Big(1-p\,-\,\frac{\nu_p-k}{\mathcal{N}}\Big)\Big(1-p\,-\,\frac{\nu_p-(k+p)}{\mathcal{N}+1}\Big)$$

and

$$\begin{aligned} c_S(i,\,k) \,=\, &(\mathcal{N}+1)\Big(k+i+1\Big)\Big(k+i\Big)\Big(k-i+1\Big)\Big(k-i\Big)\\ &\times\Big(1-p\,-\,\frac{\nu_p-(k+i+p)}{\mathcal{N}+1}\Big)\Big(1-p\,-\,\frac{\nu_p-(k+i)}{\mathcal{N}}\Big)\\ &\times\Big(1-p\,-\,\frac{\nu_p-(k-i+p)}{\mathcal{N}+1}\Big)\Big(1-p\,-\,\frac{\nu_p-(k-i)}{\mathcal{N}}\Big). \end{aligned}$$

Both constants are positive in the k–range and i–range stated in (14) and (15). Using (47) and the above definitions, $N(k)$ and $S(i,\,k)$ of Lemma 1 reduce to

$$N(k) = k^2\,-\,\Big(2\nu-1\Big)k\,+\,\nu\Big(\nu-2+p\,+\,\frac{\nu_p-\nu+1-p}{\mathcal{N}+1}\Big) \tag{48}$$

and

$$S(i,\,k) = A(k)\,i^4\,+\,B(k)\,i^2\,+\,N(k)\,C(k)\,, \tag{49}$$

where

$$A(k) = \,-\,2k\,+\,2\nu\,-\,1\,-\,(1-2p)(\mathcal{N}-2\nu_p), \tag{50}$$

$$\begin{aligned} B(k) \,=\,\;\; &4k^3\,+\,2\Big((1-2p)\mathcal{N}-2\nu_p-2\nu+3\Big)k^2\\ &+\,2\Big(((3p-1)\nu+1-2p)\mathcal{N}+3\nu\nu_p-\nu^2\\ &\qquad -\,2\nu_p-(3p-1)\nu+2-\frac{3\nu(\nu_p-\nu+1-p)}{\mathcal{N}+1}\Big)k\\ &-\,(2p\nu^2-2p^2\nu-(1-2p))\mathcal{N}-2\nu^2\nu_p+2\nu^3+2\nu^2\\ &\qquad -\,2\nu_p\,+\,2(1-p^2)\nu\,+\,1\,+\,\frac{2\nu({\nu_p}^2-\nu^2-(1-p^2))}{\mathcal{N}+1} \end{aligned} \tag{51}$$

and

$$\begin{aligned} C(k) \,=\, &-\,2k^3\,-\,\Big((1-2p)\mathcal{N}-2\nu_p-2\nu+3\Big)k^2\\ &-\,\Big((2p\nu+1-2p)\mathcal{N}+2\nu\nu_p-2\nu_p-2\nu+1\Big)k\\ &+\,\nu(\nu-p)\mathcal{N}\,-\,\nu\nu_p\,+\,\nu\,. \end{aligned} \tag{52}$$

We need not determine valid reflection centres k_2 and k_4 for all parameter combinations but only for such which satisfy $1\le n\le M\le N/2$, due to the four relationships stated in (43) and (44). With respect to the parameters $\mathcal{N}$, p, and ν_p, this region reads

$$0\,\le\,p\,\le\,\tfrac{1}{2}\,,\qquad 2\,\le\,\nu_p\,\le\,p\mathcal{N}. \tag{53}$$

After an analytical examination of (48) (with respect to its roots) and (49) (with respect to its sign in the parameter region (53)) we arrived at the following result which will be proved in Appendix C.

Lemma 4

Let K_N be a $H(N,\,M,\,n)$ hypergeometric random variable, where $n \le M \le N/2$ and $n \ge 3$. Define $p = (M+1)/(N+2)$ $(0 < p \le \frac{1}{2})$ and $\nu = (n+1)p$.
Furthermore, let $m = \lfloor \nu \rfloor$ be the mode and $N(k)$ (48) the 'necessary function' of $p_N(k)$. Finally, let

$$x_2{}^{(min)} = \nu - \frac{1}{2} - \sqrt{\Big(1-p\Big)\Big(1-\frac{n+2}{N+3}\Big)\nu + \frac{1}{4}}\,, \qquad k_2{}^{(min)} = \left\lceil x_2{}^{(min)} \right\rceil \tag{54}$$

$$x_4{}^{(max)} = \nu - \frac{1}{2} + \sqrt{\Big(1-p\Big)\Big(1-\frac{n+2}{N+3}\Big)\nu + \frac{1}{4}}\,, \qquad k_4{}^{(max)} = \left\lfloor x_4{}^{(max)} \right\rfloor, \tag{55}$$

where $x_2{}^{(min)}$ and $x_4{}^{(max)}$ are the roots of $N(k) = 0$. Then

(i) Candidates for reflection centres $k_2 \le m - 1 + \delta_m{}^{(l)}$ are all

$$k_2 \in \left[k_2{}^{(min)},\, m - 1 + \delta_m{}^{(l)}\right] \cup \left\{m - 1 + \delta_m{}^{(l)}\right\},$$

provided $\nu > 2 - p$ in the case $\delta_m{}^{(l)} = 0$, and $\nu > 2 + \sqrt{2}$ in the case $\delta_m{}^{(l)} = 1$.

If $\delta_m{}^{(l)} = 0$, then $k_1 \ge 1$ for all $\nu > 2 + \sqrt{2}$; in all other cases ($\delta_m{}^{(l)} = 0$, $2 - p < \nu \le 2 + \sqrt{2}$ and $\delta_m{}^{(l)} = 1$, $\nu > 2 + \sqrt{2}$) $k_1 \ge 0$ ($k_1 = 2k_2 - (m - 1 + \delta_m{}^{(l)})$ is the left boundary of the centre region and must meet the requirement $k_1 \ge 0$).

(ii) Candidates for reflection centres $k_4 \ge m + 1 - \delta_m{}^{(r)}$ are all

$$k_4 \in \left[m + 1 - \delta_m{}^{(r)},\, k_4{}^{(max)}\right] \cup \left\{m + 1 - \delta_m{}^{(r)}\right\}.$$

Furthermore, $k_5 \le n - 1$ for all $\nu \ge 3$, and $k_5 \le n$ for all $\nu > 0$ ($k_5 = 2k_4 - (m + 1 - \delta_m{}^{(r)})$ is the right boundary of the centre region and must meet the requirement $k_5 \le n$).

$k_2{}^{(min)}$ (38) of the Poisson/binomial distributions meets the inequality $k_2{}^{(min)} \le m - 1$, provided $\nu \ge 3$. Similarly, $k_4{}^{(max)}$ (39) meets the inequality $k_4{}^{(max)} \ge m+1$, provided $\nu > 2 + \sqrt{2}$. In the case of the more general hypergeometric distributions this unfortunately does not hold. Therefore the initialization step of a hypergeometric sampling routine has to check explicitly that $k_2 \le m - 1$ in the case $\delta_m{}^{(l)} = 0$, and $k_4 \ge m + 1$ in the case $\delta_m{}^{(r)} = 0$. (Note that $k_2{}^{(min)} \le m$ and $k_4{}^{(max)} \ge m$ holds good for all $\nu > 0$.)

Extensive numerical calculations again lead us to assume that the extreme reflection centres possible, that are $k_2 = \min(k_2{}^{(min)},\, m - 1 + \delta_m{}^{(l)})$ and $k_4 = \max(k_4{}^{(max)},\, m + 1 - \delta_m{}^{(r)})$, minimize the measures α and N for all three kinds of mode transformations.

This suggests to use the reflection centres $k_2 = \min(k_2{}^{(min)},\ m-1+\delta_m{}^{(l)})$ and $k_4 = \max(k_4{}^{(max)},\ m+1-\delta_m{}^{(r)})$ in our algorithm. For the same reasons we mentioned in Section 6 we recommend to transform the upper part of the mode cell to the right whenever it is possible. (The overall numerical result for the hypergeometric distributions, however, differs slightly from the result for the Poisson/binomial distributions. For instance, if $m \geq 50$, in more than 3/4 of all cases α, in about 3/4 of all cases N and the sum $P(I)+P(S)$, and only in about half the cases $P(I)$ attain better numerical values when m is transformed to the right and not to the left. Furthermore, if $m \leq 10$, $P(I)$ attains its maximum at m remains in the centre in slightly less than half the cases, followed closely, with respect to the frequency, by the case m is part of the reflection to the right. Finally, if $m \leq 100$, the quick acceptance probability $P(Q) = P(I)+P(S)$ attains its maximum in about 3/4 of all cases at m being partially transformed to the right.)

Once again we have $N(k) \leq 0$ for all $k \in [k_2{}^{(min)},\ k_4{}^{(max)}]$ and $N(k) > 0$ for all $k \notin [k_2{}^{(min)},\ k_4{}^{(max)}]$. Consequently the polygon through the points $(k,\ p_N(k))$ is concave in and convex outside the interval $[k_2{}^{(min)},\ k_4{}^{(max)}]$, and $L(k)$ is in fact a lower bound for $p_N(k)$. As we know from Lemma 4, $k_2 = \min(k_2{}^{(min)},\ m-1)$ and $k_4 = k_4{}^{(max)}$ are feasible reflection centres, provided $\nu > 2-p$ and m is part of the transformation to the right. Not only the characteristics α and N, but also the general squeeze function $L(k)$ suggests to use the above initialization, for then the special $L(k)$ we can use (due to $k_2 \leq k_2{}^{(min)}$ and $k_4 = k_4{}^{(max)}$) is much simpler than $L(k)$ following from general reflection centres $k_2 > k_2{}^{(min)}$ and $k_4 < k_4{}^{(max)}$.

We still have to decide on the envelopes we will use for the tails. Thereafter our sampling routine will be complete down to the last detail, only the specification of some general constants (which are used in the general algorithm GPRS) will follow yet below. As the hypergeometric distributions are log–concave, their left and right tails are majorized, at any rate, by exponential functions which meet the requirement (19). These functions, specified in Lemma 2, achieve a smaller hat area α (and a smaller expected number of uniforms N) than the power functions of degree 2 in Section 3 (which of course are feasible too). Therefore we propose to let the left and right tail of $p_N(k)$ be covered by these exponential functions (that is, our specific algorithm will employ case (a) of the general sampling routine GPRS).

Furthermore, note that $k_{min} = 0$ and $k_{max} = n$ in the special parameter region our algorithm is valid for. Finally, note that we rescale the probability function such that $f(m) = 1$. Therewith we avoid the computation of $f_m = f(m)$; moreover, we do not even need the variable f_m, since it takes the value 1 for all parameter combinations.

Below we state the initialization step of algorithm HPRS^r (m is always part of the transformation to the right) which is valid for $\nu > 2+\sqrt{2}$. We also state an initialization which leads to the more general sampling routine HPRS (m is mostly part of the transformation to the right and rarely part of the transformation to the left) which is valid for all parameter combinations satisfying $1 \leq n \leq M \leq N/2$.

Algorithms HPRS^r and HPRS:

$f(k,\, c_N,\, M,\, n,\, c_m) = \exp(c_m - (\log(k!) + \log((c_N + k)!) + \log((M - k)!) + \log((n - k)!)))$,
$r(k,\, c_N,\, \mathcal{M},\, \nu_p) = (\nu_p/k - 1) * (\mathcal{M} - k)/(c_N + k)$

Initialization of HPRS^r: $((n+1)(M+1)/(N+2) > 2 + \sqrt{2}$ and $n \le M \le N/2)$

```
Set 𝓜 = M + 1,  ν_p = n + 1,  c_N = N − M − n;
Set p = 𝓜/(N + 2),  ν = ν_p * p,  m = ⌊ν⌋;
Set D = sqrt(ν * (1 − p) * (1 − (n + 2)/(N + 3)) + 1/4);
Set k_2 = min(⌈ν − 1/2 − D⌉, m − 1),  k_4 = ⌊ν − 1/2 + D⌋;          Lemma 4, (i) and (ii)
Set d_l = m − k_2 − 1,  d_r = k_4 − m;
Set k_1 = k_2 − d_l,  k_5 = k_4 + d_r;                               k_1 ≥ 1,  k_5 ≤ n − 1
Set c_m = log(m!) + log((c_N + m)!) + log((M − m)!) + log((n − m)!);
Set f_2 = f(k_2, c_N, M, n, c_m),  f_1 = f(k_1, c_N, M, n, c_m);
Set f_4 = f(k_4, c_N, M, n, c_m),  f_5 = f(k_5, c_N, M, n, c_m);
Set p_1 = f_2 * (d_l + 1),  p_2 = f_2 * d_l + p_1;
Set p_3 = f_4 * (d_r + 1) + p_2,  p_4 = f_4 * d_r + p_3;
Set r_2 = r(k_2, c_N, 𝓜, ν_p);
Set r_1 = r(k_1, c_N, 𝓜, ν_p),  λ_l = log(r_1),  p_5 = f_1/λ_l + p_4;
Set r_4 = r(k_4 + 1, c_N, 𝓜, ν_p);
Set r_5 = r(k_5 + 1, c_N, 𝓜, ν_p),  λ_r = −log(r_5),  p_6 = f_5/λ_r + p_5;
```

Initialization of HPRS: $(n \ge 1$ and $n \le M \le N/2)$

```
Set 𝓜 = M + 1,  ν_p = n + 1,  c_N = N − M − n;
Set p = 𝓜/(N + 2),  ν = ν_p * p,  m = ⌊ν⌋;
If (ν < 5/4) then
    Set k_2 = k_1 = 0,  k_4 = k_5 = 1;                               k_4^(max) ≤ 1
    Set d_l = d_r = 0;
else
if (ν < 2) then
    Set k_2 = k_1 = 1,  k_4 = k_5 = 2;                               k_4^(max) ≤ 2
    Set d_l = d_r = 0;
else
    Set D = sqrt(ν * (1 − p) * (1 − (n + 2)/(N + 3)) + 1/4);
    Set k_2 = min(⌈ν − 1/2 − D⌉, m − 1),  k_4 = ⌊ν − 1/2 + D⌋;      Lemma 4, (i) and (ii)
    Set d_l = m − k_2 − 1,  d_r = k_4 − m;
    Set k_1 = k_2 − d_l,  k_5 = k_4 + d_r;                           k_1 ≥ 0,  k_5 ≤ n
Endif
Set c_m = log(m!) + log((c_N + m)!) + log((M − m)!) + log((n − m)!);
Set f_2 = f(k_2, c_N, M, n, c_m),  f_1 = f(k_1, c_N, M, n, c_m);
Set f_4 = f(k_4, c_N, M, n, c_m),  f_5 = f(k_5, c_N, M, n, c_m);
```

```
Set p_1 = f_2 * (d_l + 1), p_2 = f_2 * d_l + p_1;
Set p_3 = f_4 * (d_r + 1) + p_2, p_4 = f_4 * d_r + p_3;
If (d_l > 0) then Set r_2 = r(k_2, c_N, M, ν_p);
If (k_1 > 0) then
    Set r_1 = r(k_1, c_N, M, ν_p), λ_l = log(r_1), p_5 = f_1/λ_l + p_4;
else
    Set p_5 = p_4;
Endif
If (d_r > 0) then Set r_4 = r(k_4 + 1, c_N, M, ν_p);
If (k_5 < n) then
    Set r_5 = r(k_5 + 1, c_N, M, ν_p), λ_r = − log(r_5), p_6 = f_5/λ_r + p_5;
else
    Set p_6 = p_5;
Endif
```

The generation step of algorithm HPRS^r (which agrees with the generation step of algorithm GPRS, given that $1 - \delta_m{}^{(l)} - \delta_m{}^{(r)} = 0$, $k_2 = k_2{}^{(min)}$, $k_4 = k_4{}^{(max)}$ and $h^{(l)}(x) = h_e{}^{(l)}(x)$, $h^{(r)}(x) = h_e{}^{(r)}(x)$) can be adopted by algorithm HPRS without any modifications. This is clear as far as the calculation of the candidate K is concerned. (K is calculated without employing the quantities $\delta_m{}^{(l)}$ and $\delta_m{}^{(r)}$. Specially notice that the auxiliary quantities d_l, d_r and the reflection centres k_2, k_4 or the centre boundaries k_1, k_5 are used for the calculation of a candidate K from the centre region. Due to the constants d_l, d_r, defined in the set–up step, the mid–centre bounds $m - 1 + \delta_m{}^{(l)}$ and $m + 1 - \delta_m{}^{(r)}$ need not be called in to help. Furthermore, m is always part of the reflection σ_l or σ_r. Here it does not matter to which side m is transformed, the main thing is that m is transformed at all.) The only question which may arise is whether $L(k)$ remains valid or not. The above initialization, however, guarantees that $L(k)$ is a lower bound for $f(k)$, even for parameter combinations where $\nu \leq 2 + \sqrt{2}$, where m is possibly part of the reflection to the left, and where $k_2 > k_2{}^{(min)}$ may happen which actually describes the only situation — apart from the two cases $k_2 < k_2{}^{(min)}$ and $k_4 > k_4{}^{(max)}$, but then $k_1 = k_2$, $m - k_2 = 1$ and $k_5 = k_4$, $k_4 - m = 1$, respectively, from which immediately follows the correctness of $L(k)$ — where the legitimacy of $L(k)$ could be doubted. That $L(k)$ (based on the reflection centres $k_2 = k_2{}^{(min)}$ and $k_4 = k_4{}^{(max)}$) is in fact legitimate for algorithm HPRS (which imposes no limitations on the parameter ν) follows directly from the statements below.

If $\nu < 5/4$, then $k_2 = k_1 \leq k_2{}^{(min)}$, $m - k_2 \leq 1$ and $k_4 = k_5 \geq k_4{}^{(max)}$, $k_4 - m \leq 1$. If $\nu \geq 2$, then $k_2 = k_1 \leq k_2{}^{(min)}$, $m - k_2 = 1$ or $k_2 = k_2{}^{(min)}$, $k_1 \leq k_2{}^{(min)}$, and $k_4 = k_4{}^{(max)}$, $k_5 \geq k_4{}^{(max)}$. In case of $5/4 \leq \nu < 2$ we have $k_4 = k_5 \geq k_4{}^{(max)}$, $k_4 - m = 1$, but it is possible that $k_2 > k_2{}^{(min)}$ (which happens when $k_2{}^{(min)} = 0$). However, $L(k)$ is valid also in this case, since the critical range, given by the left tail of $p_N(k)$, consists of one event only ($k = 0$), and here $L(0) = f(0)$, due to $L(k_1 - 1) = f(k_1 - 1)$.

The most time consuming step in the two algorithms is the computation of the function values f_2, f_1, f_4, and f_5. Apart from the direct calculation by means of $f(k, c_N, M, n, c_m)$, we could also calculate these constants recurrently from the mode function value $f(m)$

(which should apply in case of d_l and d_r are not too large). To accelerate the initialization step, we will modify the set–up step such that f_2, f_1, f_4, and f_5 will be calculated either recurrently from m or directly by means of $f(k)$, where the size of d_r (or the size of d_l, since both quantities are of the same order) decides on the procedure being used. In other words, whenever $d_r \le N_{It}$, where N_{It} denotes the maximum number of recurrence steps we admit in the calculation of each function value, we calculate f_2, f_1, f_4, and f_5 recurrently from m (notice that $f(m) = 1$), otherwise we calculate the function values directly. Note that we concurrently change the set–up of r_2 and r_4, since calculating f_1 recurrently from k_2 requires the quantity r_2, provided $d_l > 0$, and calculating f_5 recurrently from k_4 requires the quantity r_4, provided $d_r > 0$.

Calculation of f_2, f_1, f_4 and f_5 (and of r_2 and r_4):

```
If (d_r ≤ N_It) then
    Set f_2 = f_4 = 1;
    Set i = m;
    While (i > k_2) do
        Set f_2 = f_2 / r(i, c_N, ℳ, ν_p), i = i − 1;
    Endwhile
    Set f_1 = f_2;
    If (i > k_1) then                                   d_l > 0  (i = k_2)
        Set r_2 = r(i, c_N, ℳ, ν_p), f_1 = f_1 / r_2, i = i − 1;
        While (i > k_1) do
            Set f_1 = f_1 / r(i, c_N, ℳ, ν_p), i = i − 1;
        Endwhile
    Endif
    Set i = m;
    While (i < k_4) do
        Set i = i + 1, f_4 = f_4 * r(i, c_N, ℳ, ν_p);
    Endwhile
    Set f_5 = f_4;
    If (i < k_5) then                                   d_r > 0  (i = k_4)
        Set i = i + 1, r_4 = r(i, c_N, ℳ, ν_p), f_5 = f_5 * r_4;
        While (i < k_5) do
            Set i = i + 1, f_5 = f_5 * r(i, c_N, ℳ, ν_p);
        Endwhile
    Endif
else
    Set f_2 = f(k_2, c_N, M, n, c_m), f_1 = f(k_1, c_N, M, n, c_m);
    Set f_4 = f(k_4, c_N, M, n, c_m), f_5 = f(k_5, c_N, M, n, c_m);
    Set r_2 = r(k_2, c_N, ℳ, ν_p), r_4 = r(k_4 + 1, c_N, ℳ, ν_p);
Endif
```

Extensive timings on an IBM PC AT–486 showed that a value N_{It} about 16 yields the best overall initialization times (with respect to our hardware configuration). Therefore we propose to set $N_{It} = 16$. In the coded version of the initialization step we of course replaced the function $r(k)$ by its definition and circumvented the division in the recurrent calculation of f_2 and f_1 by multiplying the current quantities f_2 and f_1 with $r^{-1}(k)$. Below we compare the times of the original initialization step with those of the modified initialization step.

Table 22: Initialization times [μsec/variate] of algorithm HPRS^r, always calculating the constants f_2, f_1, f_4 and f_5 by means of $f(k)$.

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	208							
20	10	229	268						
40	20	228	246	245					
80	20	391	375						
	40	490	507	408	266				
200	50	508	523	589					
	100	486	505	531	513	785			
800	100	506	645	654					
	200	507	521	585	652	790			
	400	490	503	526	532	792	788		
4000	250	631	639	648					
	500	502	639	649	648	789			
	1000	503	518	597	641	789	792		
	2000	486	498	521	523	769	785	779	
40000	1250	636	646	654	647				
	2500	637	647	655	654	793			
	5000	509	647	654	654	792	794		
	10000	505	521	601	643	792	792	792	
	20000	496	509	532	533	794	792	789	800

Table 23: Initialization times [μsec/variate] of algorithm HPRS$^r{}_{16}$ ($N_{It} = 16$), calculating the constants f_2, f_1, f_4 and f_5 either by means of $f(k)$ or recurrently from the mode function value $f(m)$.

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	126							
20	10	126	143						
40	20	125	143	142					
80	20	168	168						
	40	178	196	230	177				
200	50	196	213	256					
	100	177	196	232	231	317			
800	100	194	242	275					
	200	194	213	256	292	388			
	400	177	196	230	266	355	455		
4000	250	219	239	274					
	500	194	239	274	307	423			
	1000	194	212	274	307	404	576		
	2000	176	194	229	264	352	523	727	
40000	1250	221	241	275	309				
	2500	221	241	276	308	437			
	5000	195	242	274	309	422	643		
	10000	193	213	274	308	406	611	799	
	20000	178	196	231	266	389	524	796	807

9 Computational Results, Hypergeometric Algorithms

Sampling from the hypergeometric distribution has received much less attention in the literature than generating deviates from Poisson or binomial distributions. Indeed, it is not easy to find methods which are efficient for most values of the parameters that are likely to occur in practice. Only four different methods are known at present:

1. Sampling without replacement.
2. Inversion, one time starting at the lowest index, and the other time starting at the mode.
3. Ratio of uniforms method based on symmetric table mountain hats $h(x)$, one time with an approximation to the optimal slope parameter, and the other time just with the optimal slope itself.
4. Acceptance–rejection with a uniform–exponential hat $h(x)$ (rectangle in the centre and exponentials in the tails).

Below we give a detailed description of this small number of algorithms. Basically they are all supplied with parameters N, M and n satisfying the sequence of inequalities $1 \le n \le M \le N/2$ by the utility routine HYUT, given in Section 8 — note that the domain of the corresponding $H(N,\, M,\, n)$ hypergeometric distribution is $\{\, 0,\, 1,\, \ldots,\, n \,\}$. Strictly speaking, HYUT will extend the fundamental methods to parameter combinations N, M^* and n^* — which are altered by HYUT to parameter combinations $1 \le n \le M \le N/2$ — that satisfy the main restriction in the head of the algorithms. In nearly all cases restraints are layed on the mean $\mu = nM/N$, that is a basic algorithm is valid for larger means only, due to the special method the algorithm rests upon, or may be restricted to small means when its execution time grows with μ. (There is one method whose execution time grows with parameter n. We suggest to use this method only when n is small.) Exactly one algorithm makes a demand on the mode m of the distribution.

The algorithms described below utilize, like the Poisson and binomial algorithms, the supporting functions `krand1` and `logfak`. We do not add the external functions `normal` and `stirlc` to the above list, since there is no hypergeometric algorithm which is based on normal or half normal envelopes or which uses the Stirling approximation inline.

9.1 Sampling Methods for the Hypergeometric Distribution

Simulation Algorithms

HDBU (**D**raw **B**alls from an **U**rn which simulates sampling without replacement, valid for $N \ge 2$, $1 \le n \le M \le N/2$, and $n \le c_n$ to avoid huge execution times) by G. S. Fishman [6].

Suppose an urn contains N balls of which M are red (success) and $N - M$ are black (failure). Each sample decreases the current number N' of balls in the urn ($N' = N$ at the beginning of the routine). The success probability in the current step is M'/N', where M' is the number of red balls left (initially $M' = M$). M' is decreased with every success by 1; as the number of experiments is n (the number of balls that have to be drawn) and $M \geq n$, there are red balls in the urn as long as samples are taken. This applies to the black balls as well, since $N - M \geq n$. That is, exactly n experiments are necessary to determine one hypergeometric variate which, as anybody knows, corresponds with the total number of red balls drawn, in accordance with the above rule. Hence the execution time is proportional to n which is just the number of uniforms the method will use up, since to decide whether a ball is red or black requires precisely one uniform. For this reason we recommend to apply this method only in case n is not too large; $c_n = 40$ or maybe even a smaller value seems to be a proper threshold.

Search Algorithms

HINV (**Inv**ersion, implemented as low index sequential search, valid for $N \geq 2$, $1 \leq n \leq M \leq N/2$, and $\mu = nM/N \leq c_\mu$ to avoid large execution times).

The algorithm uses sequential search starting at the extreme left mass point of the distribution which moves upward, until the uniform number U generated at the beginning of the routine indicates the present sample. As the domain of the $H(N, M, n)$ distribution is $\{0, 1, \ldots, n\}$, the index the search starts at is $k = 0$. In the algorithm the explicit calculation of cumulative probabilities — which appear in the formal description of the inversion method for discrete distributions — is circumvented by means of the chop–down technique: The current hypergeometric probability $p_N(k)$ is subtracted simply from the variable that initially stores the uniform number U. The index k at which this variable assumes for the first time a negative value becomes the sample the algorithm will return in this call.

The initial probability

$$p_0 = \frac{(N-M)!\,(N-n)!}{N!\,(N-M-n)!}$$

is set up in an initialization step with the help of the external function `logfak` which, we recall, returns the value of $\log k!$ for all $k \geq 0$ in the desired precision. All other probabilities needed afterwards are calculated in the search loop through the recurrence relationship $p_N(k+1) = r(k+1)\, p_N(k)$ (forward recurrence). The recurrence is a potential source of roundoff error which accumulates and becomes serious as k increases; therefore k is not allowed to exceed a precalculated specific bound (which is large enough to guarantee that the probabilities higher up are insignificant with respect to the given accuracy). Additionally, the sequential search must stop when the search index is about to exceed n, since n is the largest feasible variate that can be returned.

HMDU (**M**odal **D**own–**U**p sequential search, valid for $N \geq 2$, $1 \leq n \leq M \leq N/2$, and $\mu = nM/N \leq c_\mu$ to avoid large execution times) based on C. D. Kemp's modal down–up sequential search [17].

The algorithm is based on sequential search starting at the mode $m = \lfloor \nu \rfloor$, where $\nu = (n+1)(M+1)/(N+2)$. However, if ν is an integer and $N = 2M$, that is $\nu = \lfloor \nu \rfloor = (n+1)/2$, we set $m = \nu - 1$. This guarantees that $2m \leq n$ for all parameter combinations, and no case distinction is necessary to avoid the — in the contrary case, that is without this modification in the defintion of m, and under the above–mentioned conditions — illicit calculation of the non–existing hypergeometric probability $p_N(2m) = p_N(2\nu) = p_N(n+1)$ in the main search loop.

Since the cumulative modal probability is not available, the search must take place alternately down and up from m. When it reaches the lowest index $k = 0$, the alternating search ends and modifies to a pure upward sequential search starting at the index $2m+1$. This upward search — coming after an unsuccessful alternately down and upward search — is terminated, at the latest, when the current search index exceeds n (the largest hypergeometric variate that can be returned) or a particular safety bound b (which guarantees that the returned variates have a significant probability). The modal probability

$$p_N(m) = \frac{(N-M)!\,(N-n)!\,M!\,n!}{N!\,(N-M-n+m)!\,(M-m)!\,(n-m)!\,m!}\,, \qquad m = \left\lfloor (n+1)\,\frac{M+1}{N+2} \right\rfloor$$

the search starts with is calculated with the help of the external function `logfak`. However, using the Stirling approximation to $\log k!$ inline instead of the `logfak` function may avoid some logarithms and speed up the initialization step. To compute the differences between $\log k!$ and the corresponding first order Stirling approximations then we propose to use the function `stirlc`. All other probabilities needed are calculated from this modal probability by the recurrence relationship $p_N(k-1) = r^{-1}(k)\,p_N(k)$ (backward recurrence) or $p_N(k+1) = r(k+1)\,p_N(k)$ (forward recurrence). In the search loop cumulative probabilities are avoided again, as in the algorithm HINV, by the simple chop–down technique.

Acceptance–Rejection Algorithms

The following algorithms are based on hat functions $h(x)$ which majorize the histogram $f(x)$, belonging to the standardized probability function $f(k) = p_N(k)/p_N(m)$. Strictly speaking, all of them use the so–called hybrid rejection technique, where truncated samples from a continuous hat are transformed by acceptance–rejection to variates from the discrete target distribution, to generate the desired hypergeometric variates.

HRUA (**R**atio of **U**niforms method with an **A**pproximation to the slope parameter of the optimal table mountain $h(x)$, valid for $\mu = nM/N \geq 1$, $2 \leq n \leq M \leq N/2$) by E. Stadlober [23].

The algorithm is based on a symmetric table mountain hat $h(x)$ proportional to the density

$$g(x) = \begin{cases} \frac{1}{4s} & a - s \le x \le a + s \\ \frac{s}{4(x-a)^2} & \text{elsewhere}\,, \end{cases}$$

where a is the location and s the slope parameter. The average number of samples from $h(x)$ needed to generate one hypergeometric variate — which describes the efficiency of the hat $h(x)$ — is $\alpha = 4s\,p_N(m)$. J. H. Ahrens & U. Dieter fixed parameter a at $a = \mu + \frac{1}{2}$ for their Poisson algorithm, anticipating possible extensions to other discrete distributions: $a = \mu + \frac{1}{2}$ is in fact the best choice for symmetric hypergeometric distributions. Once a has been fixed, one only has to minimize the quantity s to yield the tightest hat then possible, as the hat size α increases with s. This algorithm here, however, uses an approximation $\hat{s}$ — close above the optimal slope s, since $\hat{s} \ge s$ must hold on account of the minimality of s — instead of the optimal s. The approximations E. Stadlober suggested in the Poisson and binomial case proved straightforwardly appliable to the hypergeometric distributions. Except for the essential differences — the generalization of the shape parameters a and $\hat{s}$ to

$$a = \mu + \tfrac{1}{2}\,, \quad \hat{s} = \sqrt{\tfrac{2}{e}(\sigma^2 + \tfrac{1}{2})} + \tfrac{3}{2} - \sqrt{\tfrac{3}{e}}\,,$$

μ the mean and σ^2 the variance of the $H(N,\,M,\,n)$ distribution, and the use of the standardized hypergeometric probability function and its recurrence relationship function in place of the corresponding Poisson/binomial functions — the hypergeometric algorithm is nearly identical to PRUA and BRUA. However, HRUA still uses other set–up constants and rejects instantly all candidates $K = \lfloor X \rfloor$, where X is a sample from $h(x)$, greater than n, in addition to the candidates less than zero.

Simple lower and upper bounds which squeeze the expression $2\log U$ in the ultimate acceptance–rejection condition help to avoid the evaluation of this logarithm most of the time. However, if the mode m is small (say $m \le 100$) or when the candidate K does not differ very much from m ($|K - m| \le 20$ in our implementation), we avoid logarithms in the acceptance–rejection test and compute the rescaled probability $f(K)$ recurrently from the normalized modal probability $f(m) = 1$ which is faster than to evaluate $\log f(k)$ and $\log U$.

To get an algorithm that is valid for any parameter combination, we have to embed this basic method in the utility routine HYUT (contrary to PRUA and BRUA, though the later one calls for a similar utility routine) and combine it with one of the two inversion methods (both of them proved superior to algorithm HDBU, compare simply the execution times in Table 24 to Table 26).

HRUE (**R**atio of **U**niforms method which uses the **E**xact value of the optimal slope for the definition of the table mountain $h(x)$, valid for $\mu = nM/N \ge 1$, $N \ge 16$ and $2 \le n \le M \le N/2$) by E. Stadlober [23].

The algorithm is based on a symmetric table mountain hat function $h(x) = \alpha\, g(x)$ with the minimal slope s possible on condition that $a = \mu + \frac{1}{2}$.

E. Stadlober could show that the quotient

$$q_N(k) = \frac{f(k)}{h(k)} = \frac{m!\,(M-m)!\,(n-m)!\,(N-M-n+m)!}{k!\,(M-k)!\,(n-k)!\,(N-M-n+k)!}\,\frac{(a-k)^2}{s^2}, \qquad |k-a| \geq s$$

attains its maximum only at one of the two points

$$k^* = \left\lfloor a - \sqrt{2a(1-\tfrac{M}{N})(1-\tfrac{n}{N})} \right\rfloor \quad \text{or} \quad k^* = \left\lceil a - \sqrt{2a(1-\tfrac{M}{N})(1-\tfrac{n}{N})} \right\rceil.$$

Hence the best value of s is $s = (a-k^*)\sqrt{f(k^*)}$.

Hypergeometric deviates are generated by the sampling step used in algorithm HRUA, that is the generation steps of the two algorithms are identical. The set–up step clearly differs from that of algorithm HRUA, but only in the calculation of the slope parameter.

H2PE (**2 P**oint method using **E**xponential tail envelopes, complemented by squeezes, valid for $\nu = (n+1)(M+1)/(N+2) \geq 5$) by V. Kachitvichyanukul & B. W. Schmeiser [10], [11].

The algorithm is based on a three–part envelope

$$h(x) = \begin{cases} f(k_2)\,e^{-\lambda_l\,(k_2-x)} & -\infty < x \leq k_2 \\ 1 & k_2 < x < k_4+1 \\ f(k_4)\,e^{-\lambda_r\,(x-k_4-1)} & k_4+1 \leq x < \infty\,, \end{cases}$$

where $\lambda_l = \log r(k_2)$ and $\lambda_r = -\log r(k_4+1)$, $r(k) = p_N(k)/p_N(k-1)$ the recurrence relationship function of the $H(N,\,M,\,n)$ hypergeometric distribution. $h(x)$ obviously touches the standardized histogram $f(x)$ at the points $x = k_2$, $x = k_2-1$, $x = k_4+1$, $x = k_4+2$, and at $x \in [m,\,m+1)$. To complete the definiton of $h(x)$, we still have to specify k_2 and k_4. V. Kachitvichyanukul & B. W. Schmeiser fixed them at

$$k_2 = m-d, \quad k_4 = m+d, \quad \text{where} \quad d = \left\lfloor \tfrac{3}{2}\sigma \right\rfloor = \left\lfloor \tfrac{3}{2}\sqrt{n\,\tfrac{M}{N}\,\tfrac{N-M}{N}\,\tfrac{N-n}{N-1}} \right\rfloor.$$

The areas of the three different regions under $h(x)$ — uniform centre, left exponential tail, and right exponential tail — are $a_1 = 2d+1$, $a_2 = f(k_2)/\lambda_l$ and $a_3 = f(k_4)/\lambda_r$. Hence the average number of samples from $h(x)$ needed to generate the desired hypergeometric variate — which describes the efficiency of $h(x)$ — is $\alpha = (a_1+a_2+a_3)\,p_N(m)$.

The standardized hypergeometric probability $f(K)$, where $K = \lfloor X \rfloor$ and X is a sample from $h(x)$, needed in the acceptance–rejection test is computed recurrently from the normalized modal probability $f(m) = 1$ if m (the mode) or K (the candidate) is small — more precisely, when $m < 100$ or $K \leq 50$ — or if K is large — more precisely, when $K \geq n-50$ — or, additionally, when K is near m ($|K-m| \leq 20$ in our implementation) to circumvent its direct calculation. The last restraint serves the purpose of accelerating the computation of $f(K)$, whereas the first three conditions guarantee the feasibility of squeeze tests — based on the Stirling approximation, some numerical estimations and simple bounds on $\log(1 \pm z)$ — which help to avoid the evaluation of $\log f(k)$, $k = K$, very often. However, these quick acceptance and rejection tests — which contain quite a few auxiliary quantities — do not save just a lot of time.

9.2 Comparison of Algorithms

All algorithms were written as C–routines, compiled and linked in Borland C++, Version 3.0, with the standard compiler options. The execution and initialization times following below were all taken on a PC/AT 486–33MHz. They are based on samples of 100 000 variates for each parameter combination. The execution times of the elementary operations and the supporting functions which principally determine the generation times of the algorithms and which therefore affect our opinion on the quality of the algorithms have already been specified previously in Subsection 7.4 .

Our algorithm HPRS^r seems to be faster than its competitors for large means, say $\mu \geq 20$ — compare the execution times in Table 26 to Table 31. Actually, it outperforms all the other algorithms based on hybrid rejection in the range $\mu \geq 5$. For small means, say $\mu \leq 10$, however, algorithm HPRS^r is beaten by the search algorithms, as all the other acceptance–rejection routines. Yet HPRS^r and also the basic method HPRB^r manage to compete with the simple inversion routine HINV in case of means about 5.

The search from the mode (algorithm HMDU) is almost always faster than the search from the lowest index (algorithm HINV). However, the set–up times of the first search method are almost twice as large as of the second method. For larger means the search from the mode is preferable because of the smaller execution time (which increases with σ, where σ^2 is the variance of the $H(N,\, M,\, n)$ distribution, whereas the execution time of the low index sequential search grows with mean μ) and, what is more important, because of the smaller number of recurrence steps that are required, on the average, to determine the desired hypergeometric variate, as each of them is a potential source of roundoff error. For the above remarks we propose to use HINV only when $\mu < 25$ and HMDU only when $\mu < 500$.

The execution times of HDBU are very bad, even when μ is small. Therefore we recommend to keep away from this method. We instead urge to use theinversion methods HINV and HMDU to produce hypergeometric variates with parameters satisfying $0 < \mu \leq 5$ and $0 < \mu \leq 10$, respectively. For larger means HPRS^r is the best algorithm, since it yields the smallest execution times. If simplicity and speed are the dominant attributes, we could recommend the combination HINV/HRUA with cut off point $\mu^* \approx 12$ — use this combination also in case parameters change frequently, since both algorithms are comparatively fast and have very low set–up costs. Algorithm HPRS^r which has the highest set–up costs combined with HMDU, on the other hand, is a proper method to produce batches of samples from a fixed hypergeometric distribution. Here a cut off point $\mu^* \approx 10$ seems to be a good choice.

Besides the execution and initialization times the average number N of uniforms needed to generate the desired variate is a further important factor. Algorithms based on sequential search need exactly one uniform — or a second one if the search yields no result because of passing the safety bound or because of rounding errors. In Appendix D we display the average number α of samples from $h(x)$ needed to generate the desired hypergeometric variate for all acceptance–rejection algorithms. Note that all of them, except the patchwork algorithms, meet the relationship $N = 2\alpha$. In addition to our favourite patchwork method marked with ^{r}, where m is part of the reflection to the **r**ight, we still consider the two other fundamental methods. They are marked with ^{l} (m is part of the transformation to the **l**eft) and ^{c} (m remains in the **c**entre of the distribution and is interpreted as immediate acceptance region). Apart from the characteristics α and N we still present $P(I)$, the probability of immediate acceptance, and $P(Q) = P(I) + P(S)$, the probability of quick acceptance, of the above–mentioned patchwork versions.

Table 24: Execution times [μsec/variate] of algorithm HDBU (Simulation of sampling without replacement — draw balls from an urn) (Fishman, 1973).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	172							
20	10	172	242						
40	20	172	242	475					
80	20	333	471						
	40	172	242	474	939				
200	50	333	472	934					
	100	172	242	474	939	$\gg$			
800	100	655	932	$\gg$					
	200	333	471	934	$\gg$	$\gg$			
	400	172	242	474	939	$\gg$	$\gg$		
4000	250	1298	$\gg$	$\gg$					
	500	655	932	$\gg$	$\gg$	$\gg$			
	1000	333	472	934	$\gg$	$\gg$	$\gg$		
	2000	172	242	474	939	$\gg$	$\gg$	$\gg$	
40000	1250	$\gg$	$\gg$	$\gg$	$\gg$				
	2500	1294	$\gg$	$\gg$	$\gg$	$\gg$			
	5000	653	929	$\gg$	$\gg$	$\gg$	$\gg$		
	10000	332	470	931	$\gg$	$\gg$	$\gg$	$\gg$	
	20000	171	241	473	936	$\gg$	$\gg$	$\gg$	$\gg$

Table 25: Execution times [μsec/variate] of algorithm HINV (Inversion — chop–down search starting at the lowest index).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	68							
20	10	68	83						
40	20	68	84	136					
80	20	68	84						
	40	68	84	137	242				
200	50	68	84	136					
	100	68	84	137	242	560			
800	100	68	84	137					
	200	68	84	137	242	559			
	400	68	84	136	242	560	$\gg$		
4000	250	68	84	137					
	500	68	84	137	242	559			
	1000	68	84	136	242	559	$\gg$		
	2000	68	84	136	242	559	$\gg$	$\gg$	
40000	1250	68	84	136	241				
	2500	68	84	136	241	555			
	5000	68	84	136	241	556	$\gg$		
	10000	68	84	137	242	559	$\gg$	$\gg$	
	20000	68	84	137	242	559	$\gg$	$\gg$	$\gg$

Table 26: Execution times [μsec/variate] of algorithm HMDU (Inversion — chop–down search from the mode, searching alternately down and up) (Kemp, 1986).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	47							
20	10	49	47						
40	20	51	51	54					
80	20	54	55						
	40	52	53	60	65				
200	50	55	58	69					
	100	52	53	63	75	87			
800	100	58	61	75					
	200	56	59	72	89	118			
	400	53	54	65	80	106	147		
4000	250	58	63	78					
	500	58	62	76	96	133			
	1000	56	59	74	92	128	214		
	2000	53	54	65	81	111	188	296	
40000	1250	59	64	79	100				
	2500	58	63	78	100	142			
	5000	58	62	77	98	138	246		
	10000	56	60	74	93	131	233	468	
	20000	53	54	65	81	112	196	398	877

Table 27: Execution times [μsec/variate] of algorithm HRUA (Ratio of uniforms method, approximate slope parameter) (Stadlober, 1989a).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	139							
20	10	135	150						
40	20	131	140	146					
80	20	137	141						
	40	128	136	140	150				
200	50	136	138	146					
	100	128	134	138	148	171			
800	100	143	142	148					
	200	136	136	145	161	188			
	400	127	133	136	148	178	182		
4000	250	145	141	149					
	500	143	142	149	164	198			
	1000	136	136	144	161	195	200		
	2000	127	133	136	148	180	194	217	
40000	1250	145	141	149	165				
	2500	145	141	148	165	202			
	5000	143	141	148	164	200	207		
	10000	136	136	145	162	197	205	229	
	20000	127	133	136	148	180	197	227	237

Table 28: Execution times [μsec/variate] of algorithm HRUE (Ratio of uniforms method, exact slope parameter) (Stadlober, 1989a).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	—							
20	10	115	115						
40	20	114	117	125					
80	20	120	125						
	40	113	118	120	134				
200	50	119	124	134					
	100	112	117	123	136	158			
800	100	131	130	139					
	200	120	123	133	152	179			
	400	112	117	123	136	167	175		
4000	250	135	133	143					
	500	131	132	142	156	191			
	1000	120	123	134	154	188	196		
	2000	112	117	123	136	170	188	213	
40000	1250	137	134	144	160				
	2500	136	134	144	159	198			
	5000	132	132	142	156	196	204		
	10000	120	122	134	154	189	201	227	
	20000	112	117	123	137	170	190	224	236

Table 29: Execution times [μsec/variate] of algorithm H2PE (Uniform–exponential method) (Kachitvichyanukul & Schmeiser, 1985).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	—							
20	10	—	105						
40	20	—	108	105					
80	20	—	105						
	40	—	106	108	111				
200	50	—	106	111					
	100	—	105	111	116	122			
800	100	—	110	115					
	200	—	107	116	122	137			
	400	—	105	111	117	134	157		
4000	250	—	109	115					
	500	—	109	115	127	147			
	1000	—	107	115	123	143	182		
	2000	—	105	110	118	135	176	198	
40000	1250	—	108	115	128				
	2500	—	108	115	127	152			
	5000	—	109	115	127	148	189		
	10000	—	107	116	123	147	187	202	
	20000	—	105	111	117	134	176	201	204

Table 30: Execution times [μsec/variate] of algorithm HPRB^r (Uniform–exponential method, combined with patchwork) (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	96							
20	10	105	70						
40	20	113	77	85					
80	20	87	103						
	40	160	106	72	64				
200	50	102	83	93					
	100	163	110	83	95	108			
800	100	108	96	94					
	200	104	85	97	98	111			
	400	165	111	83	85	110	115		
4000	250	125	99	94					
	500	109	96	94	98	112			
	1000	104	85	88	97	111	117		
	2000	164	111	83	85	110	115	116	
40000	1250	126	101	96	100				
	2500	125	100	95	99	114			
	5000	110	97	94	98	112	118		
	10000	104	85	88	97	112	118	120	
	20000	166	112	84	86	112	115	118	123

Table 31: Execution times [μsec/variate] of algorithm HPRS^r (Uniform–exponential method with linear squeeze, combined with patchwork) (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	91							
20	10	98	66						
40	20	106	74	80					
80	20	80	91						
	40	130	92	58	60				
200	50	90	72	74					
	100	131	94	66	80	76			
800	100	94	81	70					
	200	91	74	78	74	70			
	400	132	94	68	64	75	68		
4000	250	101	83	72					
	500	93	81	71	71	69			
	1000	91	74	66	69	71	70		
	2000	132	94	69	64	77	70	70	
40000	1250	102	84	74	75				
	2500	102	83	73	74	70			
	5000	94	81	72	72	71	70		
	10000	92	75	66	69	72	70	69	
	20000	133	96	69	64	68	71	69	68

Table 32: Initialization times [μsec] and memory requirements [bytes] of algorithms HDBU (Fishman, 1973), HINV, HMDU (Kemp, 1986), HRUA (Stadlober, 1989a), HRUE (Stadlober, 1989a), H2PE (Kachitvichyanukul & Schmeiser, 1985), HPRB^r (Zechner, 1993), and HPRS^r (Zechner, 1993). (Memory requirements without 1259 bytes for an external function that calculates $\log k!$, and without 1008 bytes for the uniform number generator.)

Algorithm	Initialization times ($M > 60$, $n \le M \le N/2$)				Memory requirements of compiled code
	$m < 30$			$m \ge 50$	
	$n - m < 30$		$n - m \ge 50$		
	$n < 30$	$n \ge 30$			
HDBU	—	—	—	—	1472
HINV	160	160	160	160	2826
HMDU	230	255	280	305	4053
HRUA	95	95	120	145	4376
HRUE	190	190	240	295	4839
H2PE	325	325	405	485	7946
HPRB^r	500	510	630	780	6599
HPRS^r	515	525	645	795	7863

Appendix A

Proof of Lemma 1

The whole proof rests upon formula (9) by which the difference of two adjoining probabilities can be reduced to one of the two probabilities, called the base probability, and a factor containing the recurrence relationship function $r(k)$.

Case (i): Reflection centre k_2

Our first goal is to deduce the lower bound for k_2, stated in (14). For that consider the general condition (7). From the judicious demand that the transformed residual probabilities are to be in the domain $[k_{min},\, k_{max}]$ of $p(k)$, we obtain

$$k_2 - i \;\geq\; k_{min} \qquad \text{for all} \qquad 1 \;\leq\; i \;\leq\; i_{max} = m - 1 + \delta_m{}^{(l)} - k_2\,.$$

The upper bound i_{max} on i, already stated in (7), is due to the fact that the indices of the centre probabilities subjected to the reflection σ_l are bounded on the right by $k_2{}^{(max)} = m - 1 + \delta_m{}^{(l)}$, the maximum possible reflection centre for σ_l. With respect to (7) this means $k_2 + i \leq k_2{}^{(max)}$, and $i_{max} = k_2{}^{(max)} - k_2$.

The inequality above is satisfied when $k_2 \;\geq\; k_{min} + i_{max} = k_{min} + k_2{}^{(max)} - k_2$ or, in other words, when

$$k_2 \;\geq\; \tfrac{1}{2}\left(k_{min} + m - 1 + \delta_m{}^{(l)}\right). \tag{56}$$

If $i_{max} = 0$, the one and only reflection centre which is possible is $k_2 = k_2{}^{(max)}$. This k_2 is feasible whenever it is in the domain of $p(k)$, i. e. whenever $k_2{}^{(max)} \geq k_{min}$. That $k_2{}^{(max)} < k_{min}$ obviously occurs only when $m = k_{min}$ and $\delta_m{}^{(l)} = 0$. Then, however, no left reflection centre does exist.

If $i_{max} \geq 1$, we will use the technique of induction to establish the statement (12) and the right part of the statement (14), all along taking for granted that k_2 fulfils (56).

Beginning: Let $i = 1$. Then (7), and also (5), reads

$$p(k_2 + 1) - p(k_2) \;\leq\; p(k_2) - p(k_2 - 1)\,.$$

Using that

$$p(k_2 + 1) = r(k_2 + 1)\; p(k_2)\,, \qquad p(k_2 - 1) = \frac{1}{r(k_2)}\; p(k_2)\,,$$

due to (9), the condition above reduces to

$$r(k_2 + 1)\, r(k_2) - 2r(k_2) + 1 \;\leq\; 0 \qquad \text{or} \qquad N(k_2) \leq 0\,.$$

Now let $k_2{}^{(min)} = \min\{k_2 |\; N(k) \leq 0 \;\text{ for all }\; k \in [k_2,\, k_2{}^{(max)}]\}$. All $k_2 \in [k_2{}^{(min)},\, k_2{}^{(max)}]$ are considered to be candidates for the centre of the reflection σ_l. That a k_2 from this interval is really feasible, we have to verify that (7) holds for all $2 \leq i \leq i_{max}$ as well.

Supposition: Let $i = j$ and $1 \leq j \leq i_{max} - 1$. Define $j_- = j - 1$ and $j_+ = j + 1$. Confining j to $j \leq i_{max} - 1$, all indices in **Conclusion** (and here) are well–defined, i. e. lie in the interval $[k_{min}, {k_2}^{(max)}]$ (specially $k_2 + j_+ \leq {k_2}^{(max)}$ and $k_2 - j_+ \geq k_{min}$).

With the definitions above (7) reads

$$p(k_2 + j) - p(k_2 + j_-) \;\leq\; p(k_2 - j_-) - p(k_2 - j)\,.$$

Because of (9) we can express $p(k_2 + j_-)$ and $p(k_2 - j_-)$ by

$$p(k_2 + j_-) = \frac{1}{r(k_2 + j)}\,p(k_2 + j)\,, \qquad p(k_2 - j_-) = r(k_2 - j_-)\,p(k_2 - j)\,,$$

and, since $r(k) > 1$ for $k \leq m - 1$, we end up with the remodeled and more useful supposition

$$p(k_2 + j) \;\leq\; \frac{r(k_2 + j)}{r(k_2 + j) - 1}\,\Big(r(k_2 - j_-) - 1\Big)\,p(k_2 - j)\,.$$

The right side is an upper bound for $p(k_2 + j)$ and will be used in **Conclusion** for an estimation of $p(k_2+j)$, thereby reducing the one base probability $p(k_2+j)$ to a product of recurrence terms and the other base probability $p(k_2-j)$. $p(k_2-j)$ will be a common factor in our new and slightly stronger condition and can therefore be neglected. The remaining expressions then will depend on recurrence terms only.

Conclusion: Let $i = j + 1$. We have to show that

$$p(k_2 + j_+) - p(k_2 + j) \;\leq\; p(k_2 - j) - p(k_2 - j_+)$$

or, equivalently, that

$$\Big(r(k_2 + j_+) - 1\Big)\,p(k_2 + j) \;-\; \Big(1 - \frac{1}{r(k_2 - j)}\Big)\,p(k_2 - j) \;\leq\; 0\,.$$

Taking **Supposition** into account, for the feasibility of k_2 it suffices to show that

$$\Big(r(k_2 + j_+) - 1\Big)\,\frac{r(k_2 + j)}{r(k_2 + j) - 1}\,\Big(r(k_2 - j_-) - 1\Big)\,p(k_2 - j) \;-\; \Big(1 - \frac{1}{r(k_2 - j)}\Big)\,p(k_2 - j) \;\leq\; 0$$

which instantly reduces to

$$\Big(r(k_2+j_+)-1\Big)\,r(k_2+j)\,\Big(r(k_2-j_-)-1\Big)\,r(k_2-j) \;-\; \Big(r(k_2-j)-1\Big)\Big(r(k_2+j)-1\Big) \;\leq\; 0.$$

But this is equivalent to $S(j,\, k_2) \leq 0$, due to (11). This new test condition slightly differs from the condition we started with; it is a little bit *stronger*. In other words, not necessarily all k_2 which satisfy (7), $i = j+1$, must fulfil our new restraint $S(j,\, k_2) \leq 0$. However, it is simpler than our initial condition, since it contains recurrence terms only and no probability terms. It can be used for the verification of candidates k_2 fulfilling **Beginning**, provided that $r(k)$ is an elementary function, such as a low degree polynomial or a low degree rational function.

The induction shows that k_2 is feasible whenever

$$N(k_2) \leq 0, \qquad \text{and} \qquad S(i,\,k_2) \leq 0, \quad 1 \leq i \leq m - k_2 - 2 + \delta_m{}^{(l)}\,, \tag{57}$$

with $N(k)$ and $S(i,\,k)$ as defined in (10) and (11). The combination of (56) and (57) concludes the proof of the first part of Lemma 1.

Case (ii): Reflection centre k_4

Analogous to case (i), we demand that the transformed residual probabilities are in the domain $[k_{min},\,k_{max}]$ of $p(k)$. With respect to (8) this means

$$k_4 + i \,\leq\, k_{max} \qquad \text{for all} \qquad 1 \,\leq\, i \,\leq\, i_{max} = k_4 - m - 1 + \delta_m{}^{(r)}\,.$$

The upper bound i_{max} on i is due to the fact that the indices of the centre probabilities subjected to σ_r are bounded on the left by $k_4{}^{(min)} = m+1-\delta_m{}^{(r)}$, the minimum possible k_4 ($k_4 - i \geq k_4{}^{(min)}$, and $i_{max} = k_4 - k_4{}^{(min)}$).
The inequality above is satisfied when $k_4 \,\leq\, k_{max} - i_{max} \,=\, k_{max} + k_4{}^{(min)} - k_4$ or, equivalently, when

$$k_4 \,\leq\, \tfrac{1}{2}\left(k_{max} + m + 1 - \delta_m{}^{(r)}\right). \tag{58}$$

If $i_{max} = 0$, the only reflection centre possible is $k_4 = k_4{}^{(min)}$. This k_4 is feasible whenever it is in the domain of $p(k)$, i. e. whenever $k_4{}^{(min)} \leq k_{max}$. If $m = k_{max}$ and $\delta_m{}^{(r)} = 0$, $k_4{}^{(min)}$ will fail to meet the inequality (58), and no right reflection centre will exist. Hence, for the existence of at least one feasible k_4 one of the two last identities must not hold.

If $i_{max} \geq 1$, we will use again the technique of induction to establish the statements belonging to k_4.

Beginning: Let $i = 1$. Then (8), and also (6), reads

$$p(k_4 - 1) - p(k_4) \,\leq\, p(k_4) - p(k_4 + 1)\,.$$

Using (9), we obtain

$$r(k_4+1)\,r(k_4) - 2r(k_4) + 1 \,\leq\, 0 \qquad \text{or} \qquad N(k_4) \leq 0\,.$$

Let $k_4{}^{(max)} = \max\{\,k_4 \,|\, N(k) \leq 0 \text{ for all } k \in [k_4{}^{(min)},\,k_4]\,\}$. All $k_4 \in [k_4{}^{(min)},\,k_4{}^{(max)}]$ are considered to be candidates for the centre of the reflection σ_r. That a k_4 from this interval is really feasible, we still have to verify that (8) holds for all $2 \leq i \leq i_{max}$.

Supposition: Let $i = j$ and $1 \leq j \leq i_{max} - 1$. Define $j_- = j - 1$ and $j_+ = j + 1$. Confining j to $j \leq i_{max} - 1$, all indices in **Conclusion** (and here) are well–defined, i. e. lie in the interval $[k_4{}^{(min)},\,k_{max}]$ (specially $k_4 - j_+ \geq k_4{}^{(min)}$ and $k_4 + j_+ \leq k_{max}$).

With the definitions above (8) reads

$$p(k_4 - j) - p(k_4 - j_-) \,\leq\, p(k_4 + j_-) - p(k_4 + j)\,.$$

Applying (9) to $p(k_4 - j_-)$ and $p(k_4 + j_-)$, and using that $r(k) < 1$ for $k \geq m + 1$ finally leads to the more suitable supposition

$$p(k_4 - j) \;\leq\; \frac{1 - r(k_4 + j)}{r(k_4 + j)\big(1 - r(k_4 - j_-)\big)}\; p(k_4 + j)\,.$$

Estimating $p(k_4 - j)$ in **Conclusion** by the right side of the inequality above will reduce our first base probability $p(k_4 - j)$ to a product of recurrence terms and our second base probability $p(k_4 + j)$. $p(k_4 + j)$ will be a common factor in our new and now slightly stronger condition and may therefore be neglected.

Conclusion: Let $i = j + 1$. We have to show that

$$p(k_4 - j_+) - p(k_4 - j) \;\leq\; p(k_4 + j) - p(k_4 + j_+)$$

or, equivalently, that

$$\Big(\frac{1}{r(k_4 - j)} - 1\Big)\, p(k_4 - j) \;-\; \Big(1 - r(k_4 + j_+)\Big)\, p(k_4 + j) \;\leq\; 0\,.$$

Taking **Supposition** into account, the condition above reduces to the a little bit *stronger* restraint

$$\Big(1 - r(k_4 - j)\Big)\,\Big(1 - r(k_4 + j)\Big) \;-\; \Big(1 - r(k_4 + j_+)\Big)\, r(k_4 + j)\,\Big(1 - r(k_4 - j_-)\Big)\, r(k_4 - j) \;\leq\; 0$$

which is equivalent to $S(j,\, k_4) \geq 0$, due to (11).

The induction shows that k_4 is feasible whenever

$$N(k_4) \leq 0, \qquad \text{and} \qquad S(i,\, k_4) \geq 0, \quad 1 \leq i \leq k_4 - m - 2 + {\delta_m}^{(r)} \tag{59}$$

which concludes the proof of the second part of Lemma 1. ■

Appendix B

Proof of Lemma 3

Case (i): Reflection centre k_2

Lemma 1 tells us that $k_2 \in [{k_2}^{(min)},\ m-1+{\delta_m}^{(l)}]$ is a possible reflection centre. In the sequel we will show that all $k_2 \in [{k_2}^{(min)},\ m-1+{\delta_m}^{(l)}]$ are feasible (meet the requirement (14)), provided $\nu > 2-p$ in the case ${\delta_m}^{(l)} = 0$ (${k_2}^{(min)} \geq (m-1)/2$ and $S(i,\,k_2) \leq 0$ for all $1 \leq i \leq m-k_2-2$ and $k_2 \in [{k_2}^{(min)},\ m-1]$) and $\nu > 2+\sqrt{2}$ in the case ${\delta_m}^{(l)} = 1$ (${k_2}^{(min)} \geq m/2$ and $S(i,\,k_2) \leq 0$ for all $1 \leq i \leq m-k_2-1$ and $k_2 \in [{k_2}^{(min)},\ m]$). The proof of this statement obviously splits up into the following two parts (consider the above remarks in parentheses).

(1) Show that ${k_1}^{(min)} \geq 0$ for $\nu > 2-p$, and that ${k_1}^{(min)} \geq 1$ for $\nu > 2+\sqrt{2}$, where ${k_1}^{(min)} = 2{k_2}^{(min)} - (m-1)$. (${k_1}^{(min)}$ refers to the case ${\delta_m}^{(l)} = 0$.)

(2) Show that $S(i,\,k_2) \leq 0$ for all $1 \leq i \leq m-k_2-1$ and $k_2 \in [{k_2}^{(min)},\ m]$. (This statement refers to the case ${\delta_m}^{(l)} = 1$ which obviously covers the case ${\delta_m}^{(l)} = 0$.)

First of all, however, we will show that ${k_2}^{(min)} \leq m-1$ for all $m \geq 3$. This allows us to set $k_2 = {k_2}^{(min)}$ in our algorithms BPRS^r and PPRS^r without the need to explicitly check that $k_2 \leq m-1$ is satisfied.

${k_2}^{(min)} \leq m-1$ holds good if, and only if

$$ {x_2}^{(min)} = \nu - \tfrac{1}{2} - \sqrt{(1-p)\nu + \tfrac{1}{4}} \ \leq\ m-1 = \lfloor \nu \rfloor - 1\,. \tag{60} $$

As the left side attains its maximum, with respect to $0 \leq p \leq \frac{1}{2}$ and ν fixed, at $p = \frac{1}{2}$, we only need to show that

$$ \nu - \lfloor \nu \rfloor + \tfrac{1}{2} \ \leq\ \sqrt{\tfrac{1}{2}\nu + \tfrac{1}{4}}\,. $$

If $\nu \geq 4$, the right side of this inequality is bounded from below by $\frac{3}{2}$ which is always greater than the left side. To establish the inequality for $3 \leq \nu < 4$, we raise both sides to the power of two and collect the squared expressions on one side. This process brings us the inequality $(\nu - \frac{5}{2})^2 - (\frac{1}{2}\nu + \frac{1}{4}) = \nu^2 - \frac{11}{2}\nu + 6 \leq 0$. The roots of the corresponding equation encase the critical range $3 \leq \nu < 4$ which concludes the proof of our assertion that ${k_2}^{(min)} \leq m-1$ for all $m \geq 3$.

(1) In the sequel we will show that ${k_1}^{(min)} \geq 1$ for all $\nu > 2+\sqrt{2}$ and ${k_1}^{(min)} \geq 0$ for all $2-p < \nu \leq 2+\sqrt{2}$.

For the proof of the first statement notice that ${k_1}^{(min)} = 2{k_2}^{(min)} - (m-1) \geq 1$ if, and only if ${k_2}^{(min)} \geq i$ for parameter $\nu \in [2i-1,\ 2i+1)$. Furthermore using that ${k_2}^{(min)} \geq i$ if, and only if ${x_2}^{(min)} > i-1$, we arrive at the inequality

$$ {x_2}^{(min)} = \nu - \tfrac{1}{2} - \sqrt{(1-p)\nu + \tfrac{1}{4}} \ >\ i-1, \qquad \nu \in [2i-1,\ 2i+1)\,. \tag{61} $$

The left side attains its minimum, with respect to $0 \le p \le \frac{1}{2}$ and ν fixed, at $p = 0$. Therefore (61) holds good, whenever

$$\nu - i + \tfrac{1}{2} - \sqrt{\nu + \tfrac{1}{4}} > 0, \qquad \nu \in [2i-1,\, 2i+1)$$

or, equivalently, whenever

$$(\nu - i + \tfrac{1}{2})^2 - (\nu + \tfrac{1}{4}) = \nu^2 - 2i\nu + i(i-1) > 0, \qquad \nu \in [2i-1,\, 2i+1)\,.$$

From the latter inequality we conclude that (61) holds good, whenever $\nu > i + \sqrt{i}$. Consequently ${k_1}^{(min)} \ge 1$ for all

$$\max{(i + \sqrt{i},\, 2i-1)} \le \nu < 2i+1\,, \qquad i \ge 0\,.$$

The largest continuous range that is covered by the above ranges is given by the interval $(i_0 + \sqrt{i_0}, \infty)$, where $i_0 = \max\{i \ge 0 \mid i + \sqrt{i} \ge 2i-1\}$. A simple calculation shows that $i_0 = 2$ which proves our assertion ${k_1}^{(min)} \ge 1$ for all $\nu > 2 + \sqrt{2}$.

For the proof of the second statement notice that ${k_1}^{(min)} = 2{k_2}^{(min)} - (m-1) \ge 0$, where $m \le 3$, holds good, if ${k_2}^{(min)} \ge 1$. Using that ${k_2}^{(min)} \ge 1$ if, and only if ${x_2}^{(min)} > 0$, we end up with the inequality

$${x_2}^{(min)} = \nu - \tfrac{1}{2} - \sqrt{(1-p)\nu + \tfrac{1}{4}} > 0\,.$$

This condition holds good for $\nu > 2 - p$ which proves our assertion ${k_1}^{(min)} \ge 0$ for all $2 - p < \nu \le 2 + \sqrt{2}$.

(2) The proof of $S(i, k_2) \le 0$ for all $1 \le i \le m - k_2 - 1$ and $k_2 \in [{k_2}^{(min)}, m]$ can be split up into the following two parts.

(a) Verify that $R(i, k_2) = (2p-1)\, i^2 - B(k_2) \le 0$.

(b) Verify that $C(k_2) \ge 0$.

Both statements together with $N(k_2) \le 0$ establish the feasibility of $k_2 \ge {k_2}^{(min)}$, for then $S(i, k_2) = R(i, k_2)\, i^2 + N(k_2) C(k_2) \le 0$. For the proof of the two statements take into account that $\nu - k_2 \ge 2$, due to $1 \le i \le m - k_2 - 1$ and $\nu \ge m$.

(a) Since $p \le \frac{1}{2}$, $R(i, k_2)$ attains its maximum, with respect to i, at $i = 1$. Hence,

$$R(i, k_2) \le R(1, k_2) = -\,2(1-2p)k_2(\nu - k_2 - 1) - 2p\nu(\nu - k_2 - p) \le 0\,.$$

(b) A practicable lower bound for $C(k_2)$ is found by estimating k_2^2 in $C(k_2)$ by the fundamental restraint $N(k_2) \le 0$, that is $k_2^2 \le (2\nu - 1)k_2 - \nu(\nu - 2 + p)$:

$$\begin{aligned} C(k_2) &= -\,(1-2p)k_2^2 - (2p\nu - 2p + 1)k_2 + \nu(\nu - p) \\ &\ge -\,(1-2p)\big((2\nu - 1)k_2 - \nu(\nu - 2 + p)\big) - (2p\nu - 2p + 1)k_2 + \nu(\nu - p) \\ &= 2(1-p)\nu(\nu - k_2 - 1 + p) \ge 0\,. \end{aligned}$$

This concludes the proof of the second statement in (14) which establishes the feasibility of all $k_2 \in [k_2{}^{(min)},\ m-1+\delta_m{}^{(l)}]$, provided $\nu > 2-p$ in the case $\delta_m{}^{(l)} = 0$ and $\nu > 2+\sqrt{2}$ in the case $\delta_m{}^{(l)} = 1$.

Case (ii): Reflection centre k_4

A preliminary range for the reflection centres k_4 is given by the interval $[m+1-\delta_m{}^{(r)},\ k_4{}^{(max)}]$, due to Lemma 1. We will show that for $\delta_m{}^{(r)} = 1$ (m is partially transformed to the right) all $k_4 \in [m, k_4{}^{(max)}]$ are feasible (meet the requirement (15)), provided $n \geq 3$. (An obvious consequence of this statement is that all $k_4 \in [m+1,\ k_4{}^{(max)}]$ are feasible in case $\delta_m{}^{(r)} = 0$.) Analogous to Case (i), the proof of this statement splits up into two parts.

(1) Show that $k_5{}^{(max)} \leq n$, where $k_5{}^{(max)} = 2\,k_4{}^{(max)} - m$.

(2) Show that $S(i,\, k_4) \geq 0$ for all $1 \leq i \leq k_4 - m - 1$ and $k_4 \in [m,\ k_4{}^{(max)}]$.

The first statement applies to the binomial distributions only. There we can even show that $k_5{}^{(max)} \leq n-1$ for all $\nu \geq 3$. In case of the Poisson distributions we have an infinite right tail, and statement (1) is irrelevant.

We can see with half an eye that $x_4{}^{(max)} \geq \nu$, that is $k_4{}^{(max)} \geq m$, for all $\nu > 0$. This allows us to set $k_4 = k_4{}^{(max)}$ in our algorithms BPRS^r and PPRS^r without the need to explicitly check that $k_4 \geq m$ is satisfied. By the way, $k_4{}^{(max)} \geq m+1$ for all $\nu > 2+\sqrt{2}$. (The proof of this statement is similar to that of the corresponding statement $k_2{}^{(min)} \leq m-1$ for all $\nu \geq 3$ in Case (i)).

(1) In the sequel we will show that $k_5{}^{(max)} \leq n-1$ for all $\nu \geq 3$ and $k_5{}^{(max)} \leq n$ for all $0 < \nu < 3$.

$k_5{}^{(max)} = 2\,k_4{}^{(max)} - m \leq n-1$ or, equivalently, $k_4{}^{(max)} \leq \lfloor(\nu+n-1)/2\rfloor$ holds good if, and only if $k_4{}^{(max)} \leq i$ for parameter $\nu \in [2i-n+1,\ 2i-n+3)$. Using that $k_4{}^{(max)} \leq i$ if, and only if $x_4{}^{(max)} < i+1$, and that $i > (\nu+n-3)/2$, due to $\nu < 2i-n+3$, we arrive at the slightly stronger inequality

$$x_4{}^{(max)} = \nu - \tfrac{1}{2} + \sqrt{(1-p)\nu + \tfrac{1}{4}} \;\leq\; (\nu+n-3)/2 \,+\, 1\,. \tag{62}$$

Arranging the square root on one side and all other expressions on the other side leads to $\sqrt{(1-p)\nu+\frac{1}{4}} \leq (n-\nu)/2$. Furthermore squaring both sides and using that $\nu = (n+1)p$ yields

$$\begin{aligned}(1-p)\nu + \tfrac{1}{4} - \left(\frac{n-\nu}{2}\right)^2 &= (n+1)p(1-p) + \tfrac{1}{4} - \tfrac{1}{4}((n+1)(1-p)-1)^2 \\ &= \tfrac{1}{4}(n+1)(1-p)\,(4p+2-(n+1)(1-p)) \;\leq\; 0\,.\end{aligned}$$

The last factor on the left side is bounded from above by $4-\nu$ which is due to $p \leq \frac{1}{2} \leq 1-p$. This, for the moment, establishes $k_5{}^{(max)} \leq n-1$ for $\nu \geq 4$.

To complete the proof of our assertion, we still have to verify that ${k_5}^{(max)} \leq n-1$ for all $3 \leq \nu < 4$. But ${k_5}^{(max)} \leq n-1$ if, and only if ${x_4}^{(max)} < \lfloor(\nu+n-1)/2\rfloor + 1$, due to the two equivalences we mentioned above. Using the identity $\nu = (n+1)p$, and taking into account that $\lfloor(\nu+n-1)/2\rfloor = \lfloor n/2\rfloor + 1$ for $3 \leq \nu < 4$, the latter inequality simplifies to

$${x_4}^{(max)} = (n+1)p - \tfrac{1}{2} + \sqrt{(n+1)p(1-p) + \tfrac{1}{4}} < \left\lfloor \tfrac{n}{2} \right\rfloor + 2\,. \tag{63}$$

The left side of this inequality attains its maximum, with respect to $0 \leq p \leq \frac{1}{2}$, at $p = \frac{1}{2}$. Further observing that the left side increases with n for fixed p and that the right side truncates $\frac{n}{2}$, we see that it suffices to verify (63) for the particular parameter values $n = 2l+1$ and $p = \frac{1}{2}$. According to these parameter restrictions, (63) reduces to

$$\tfrac{1}{2}(2l+2) - \tfrac{1}{2} + \sqrt{\tfrac{1}{2}\nu + \tfrac{1}{4}} - (l+2) \;=\; \sqrt{\tfrac{1}{2}\nu + \tfrac{1}{4}} - \tfrac{3}{2} \;<\; 0\,.$$

This inequality holds for all $\nu < 4$ which concludes the proof of ${k_5}^{(max)} \leq n-1$ for all $\nu \geq 3$.

${k_5}^{(max)} = 2\,{k_4}^{(max)} - m \leq n$ holds good if, and only if ${x_4}^{(max)} < \lfloor(\nu+n)/2\rfloor + 1$, due to the equivalences we mentioned at the beginning of (1).

In case $2 \leq \nu < 3$, the above inequality reduces to ${x_4}^{(max)} < \lfloor n/2\rfloor + 2$, already stated in (63). As we have shown above, (63) holds for all $\nu < 4$, and consequently for our special parameter range $2 \leq \nu < 3$.

In case $1 \leq \nu < 2$, the above inequality simplifies to ${x_4}^{(max)} < \lfloor(n+1)/2\rfloor + 1$. Here it suffices to show that this inequality holds good for all $n = 2l$ and $p < 2/(2l+1)$, where $l \geq 2$. (The upper bound on p follows from the restriction of ν to $\nu < 2$.) Using that $\lfloor(n+1)/2\rfloor = l$ and $\nu = (2l+1)p$, our condition becomes

$$(2l+1)p - \tfrac{1}{2} + \sqrt{(2l+1)p(1-p) + \tfrac{1}{4}} - (l+1) \;<\; 0\,.$$

This inequality holds good for all $l \geq 2$ and $p < 2/(2l+1)$ or, in other words, for all $n \geq 3$ and $\nu < 2$.

For $\nu < 1$ our condition reads ${x_4}^{(max)} < \lfloor n/2\rfloor + 1$, due to $\lfloor(\nu+n)/2\rfloor = \lfloor n/2\rfloor$. Here it suffices to show that this inequality holds good for all $n = 2l+1$ and $p < 1/(2l+2)$, where $l \geq 1$. Using that $\lfloor n/2\rfloor = l$ and $\nu = (2l+2)p$, our condition reduces to

$$(2l+2)p - \tfrac{1}{2} + \sqrt{(2l+2)p(1-p) + \tfrac{1}{4}} - (l+1) \;<\; 0\,.$$

This inequality holds good for all $l \geq 1$ and $p < 1/(2l+2)$ or, in other words, for all $n \geq 2$ and $\nu < 1$. (For the proof of the two inequalities, first of all, substitute $p_{max} = \frac{2}{5}$ and $p_{max} = \frac{1}{4}$, respectively, for p. Then arrange the square root on one side and all other terms on the other side, square both sides and collect the resulting expressions on one side. The final expression is a quadratic polynomial in l which has the desired sign.) This concludes the proof of our assertion that ${k_5}^{(max)} \leq n$ for all $\nu > 0$, provided $n \geq 3$ and $0 < p \leq \frac{1}{2}$.

(2) Analogous to Case (i), the proof of $S(i, k_4) \geq 0$ for all $1 \leq i \leq k_4 - m - 1$ and $k_4 \in [m, k_4{}^{(max)}]$ can be split up into the following two parts.

(a) Verify that $R(i, k_4) = (2p-1)\,i^2 - B(k_4) \geq 0$.

(b) Verify that $C(k_4) \leq 0$.

Both statements together with $N(k_4) \leq 0$ establish the feasibility of $k_4 \leq k_4{}^{(max)}$, for then $S(i, k_4) = R(i, k_4)\,i^2 + N(k_4)C(k_4) \geq 0$. For the proof of the two statements, however, we have to take into account that $k_4 - \nu \geq 1$, due to $1 \leq i \leq k_4 - m - 1$ and $\nu < m + 1$.

(a) $R(i, k_4)$ attains its minimum, with respect to i, at $i_{max} = k_4 - m - 1$, due to $p \leq \frac{1}{2}$. To avoid the variable m in our test condition and to keep the following calculation as simple as possible, we substitute the little bit larger value $\hat{\imath}_{max} = k_4 - \nu$ for i_{max}. Therewith

$$\begin{aligned} R(i, k_4) \geq R(i_{max}, k_4) \geq R(\hat{\imath}_{max}, k_4) &= R(k_4 - \nu, k_4) \\ &= (2p-1)((k_4-\nu)^2 - 2k_4(k_4 - \nu + 1) - 1) + 2p\nu(k_4 - \nu + p) \\ &= (1-2p)((k_4-\nu)(k_4+\nu) + 2k_4 + 1) + 2p\nu(k_4 - \nu + p) \geq 0\,. \end{aligned}$$

(b) $C(k_4)$ is negative for all $k_4 \in [m+1, k_4{}^{(max)}]$, due to

$$\begin{aligned} C(k_4) &= \quad (2p-1)k_4^2 - (2p\nu - (2p-1))k_4 + \nu(\nu - p) \\ &= -(1-2p)k_4(k_4 - \nu + 1) - \nu(k_4 - \nu + p) \leq 0\,. \end{aligned}$$

This concludes the proof of Lemma 3. ■

Appendix C

Proof of Lemma 4

Case (i): Reflection centre k_2

We will show that all $k_2 \in [{k_2}^{(min)}, m-1+{\delta_m}^{(l)}]$ are feasible (meet the requirement (14)), provided $\nu > 2-p$ in the case ${\delta_m}^{(l)} = 0$, and $\nu > 2+\sqrt{2}$ in the case ${\delta_m}^{(l)} = 1$. The proof of this statement obviously splits up into the following two parts.

(1) Show that ${k_1}^{(min)} \geq 0$ for $\nu > 2-p$, and that ${k_1}^{(min)} \geq 1$ for $\nu > 2+\sqrt{2}$, where ${k_1}^{(min)} = 2{k_2}^{(min)} - (m-1)$. (The notation ${k_1}^{(min)}$ refers to the case ${\delta_m}^{(l)} = 0$.)

(2) Show that $S(i, k_2) \leq 0$ for all $1 \leq i \leq m - k_2 - 1$ and $k_2 \in [{k_2}^{(min)}, m]$. (This statement refers to the case ${\delta_m}^{(l)} = 1$ which obviously covers the case ${\delta_m}^{(l)} = 0$.)

${k_1}^{(min)}$ refers to the case ${\delta_m}^{(l)} = 0$; it denotes the minimum k_1 which can occur here. In the case ${\delta_m}^{(l)} = 1$, the minimum k_1 is ${k_1}^{(min)} - 1$, due to the reflection of one more cell (the mode cell) to the left, compared with the case ${\delta_m}^{(l)} = 0$. Therefore, whenever ${k_1}^{(min)} \geq 1$, the minimum k_1 possible in the case ${\delta_m}^{(l)} = 1$ meets the requirement $k_1 \geq 0$.

It is easy to see that ${x_2}^{(min)} \leq \nu - 1$ for all $\nu > 0$. Rounding up both sides shows that ${k_2}^{(min)} \leq m$. The more useful restraint that ${k_2}^{(min)} \leq m-1$ unfortunately does not hold for all $\nu \geq 3$ as it happens to be in the Poisson/binomial case. Therefore, in case ${\delta_m}^{(l)} = 0$ (m is part of the transformation to the right or remains in the centre) we have to guarantee by a test that $k_2 \leq m-1$, even for parameter combinations, where $\nu \geq 3$.

(1) That ${k_1}^{(min)} \geq 0$ for all $\nu > 2-p$, and ${k_1}^{(min)} \geq 1$ for all $\nu > 2+\sqrt{2}$ follows directly from Lemma 3, (i), as we will outline in detail below.

   ${k_2}^{(min)}$ (54) of the hypergeometric distributions is always greater than or equal to ${k_2}^{(min)}$ (38) of the Poisson/binomial distributions, due to the additional factor $(1 - (n+2)/(N+3))$ under the square root. Therefore the statements in Lemma 3, (i) which refer to k_1 apply here as well. That is, $k_1 \geq 1$ for all $\nu > 2+\sqrt{2}$ and $k_1 \geq 0$ for all $\nu > 2-p$ in the case ${\delta_m}^{(l)} = 0$, and $k_1 \geq 0$ for all $\nu > 2+\sqrt{2}$ in the case ${\delta_m}^{(l)} = 1$. Substituting ${k_1}^{(min)}$ for k_1 in the first case (${\delta_m}^{(l)} = 0$) then concludes our argumentation.

(2) The proof of $S(i, k_2) \leq 0$ for all $1 \leq i \leq m - k_2 - 1$ and $k_2 \in [{k_2}^{(min)}, m]$ can be split up, as in the Poisson/binomial case, into the following two parts.

   (a) Verify that $R(i, k_2) = A(k_2)\, i^2 + B(k_2) \leq 0$.

   (b) Verify that $C(k_2) \geq 0$.

   Both statements together with $N(k_2) \leq 0$ establish the feasibility of $k_2 \geq {k_2}^{(min)}$, for then $S(i, k_2) = R(i, k_2)\, i^2 + N(k_2)C(k_2) \leq 0$. For the proof of the two statements take into account that $\nu - k_2 \geq 2$, due to $1 \leq i \leq m - k_2 - 1$ and $\nu \geq m$.

For convenience and also for clarity, we introduce the quantity $z = \nu - k_2 - 2$. $z \geq 0$ which is traceable to $\nu - k_2 \geq 2$, formula (53) and the identity $\nu = \nu_p p$ are the basic conditions our proof will fall back.

(a) First of all, we set

$$\mathcal{B}(k_2) \;=\; -\,\frac{6\nu(\nu_p - \nu + 1 - p)}{\mathcal{N}+1}\,k_2 \;+\; \frac{2\nu({\nu_p}^2 - \nu^2 - (1-p^2))}{\mathcal{N}+1}\,. \tag{64}$$

$\mathcal{B}(k_2)$ is the fractional part of $B(k_2)$ and satisfies (in consequence of (53)) the sequence of inequalities

$$\mathcal{B}(k_2) \;\leq\; 0 \;+\; \frac{2(1-p^2)\nu(\nu^2 - p^2)}{(\mathcal{N}+1)p^2} \;\leq\; 2(1-p^2)(\nu^2 - p^2)\,.$$

Since $A(k_2)$ attains positive as well as negative values, we now have to distinguish between two cases, depending on the sign of $A(k_2)$:

- $A(k_2) \leq 0$:

  $R(i,\, k_2)$ attains its maximum, with respect to i, at $i_{min} = 1$. Consequently $R(i,\, k_2) \leq 0$ for all $1 \leq i \leq m - k_2 - 1$ whenever $\mathcal{R}_{\_}(k_2) = R(1,\, k_2) \leq 0$. To establish this inequality, we substitute $k_2(z) = \nu - 2 - z$ for k_2 in the non–fractional part $\mathcal{R}_{\_}(k_2) - \mathcal{B}(k_2)$ and rewrite the resulting expression as a sum of terms containing different powers of z. Proceeding in this manner, we obtain the following representation for $\mathcal{R}_{\_}(k_2)$:

$$\begin{aligned}\mathcal{R}_{\_}(k_2) \;=\; & -4z^3 \;+\; 2\Big((1-2p)\mathcal{N} - 2\nu_p + 4\nu - 9\Big)z^2 \\ & -2\Big(((1-2p)(\nu-3) + p\nu)\mathcal{N} - \nu\nu_p + \nu^2 + 6\nu_p - 3(3+p)\nu + 13\Big)z \\ & -2\Big(((1-p^2)(\nu-2) + 2p(2-p))\mathcal{N} \\ & \qquad - 3(1-p)\nu(\nu-2) + 4\nu_p - (9-p^2)\nu + 6\Big) \;+\; \mathcal{B}(k_2)\,. \end{aligned} \tag{65}$$

  That $\mathcal{R}_{\_}(k_2) \leq 0$ follows directly from $\mathcal{B}(k_2) \leq 2(1-p^2)\nu^2$ and the inequalities below which are an immediate consequence of (53), the identity $\nu = \nu_p p$ and the fact that $\nu - 2 = z + k_2 \geq z + 1$ ($k_2 \geq 1$ due to $k_2 \geq (m-1)/2$ and $m - k_2 \geq 2$).

$$\begin{aligned} -4z^3 - 18z^2 - 2(\nu^2 + 13)z - 12 &\leq 0\,, \\ 2(1-2p)\mathcal{N}z^2 \;-\; 2(1-2p)\mathcal{N}(\nu-3)z &\leq 0\,, \\ -2(2\nu_p - 4\nu)z^2 \;-\; 2(6\nu_p - 3(3+p)\nu)z &\leq 0\,, \\ -2(p\nu\mathcal{N} - \nu\nu_p)z &\leq 0\,, \\ -2((1-p^2)(\nu-2)\mathcal{N} - (4+p)(1-p)\nu(\nu-2)) &\leq 0\,, \\ -2(2p(2-p)\mathcal{N} + 4\nu_p - (11-3p^2)\nu) &\leq 0\,. \end{aligned}$$

- $A(k_2) \geq 0$:

  $R(i,\, k_2)$ attains its maximum, with respect to i, at $i_{max} = m - k_2 - 1$. We will use the little bit larger value $\hat{\imath}_{max} = \nu - k_2 - 1$ instead of i_{max} to avoid the variable m in our test condition. Because of

$$R(i,\, k_2) \;\leq\; R(i_{max},\, k_2) \;\leq\; R(\hat{\imath}_{max},\, k_2) \;=\; R(\nu - k_2 - 1,\, k_2)$$

  it suffices to show that $\mathcal{R}_+(k_2) = R(\nu - k_2 - 1,\, k_2) \leq 0$. Using the definitions of z and $\mathcal{B}(k_2)$, $\mathcal{R}_+(k_2)$ may be written as follows.

$$\begin{aligned}
\mathcal{R}_+(k_2) \;=\; & -2z^3 + \Big((1-2p)\mathcal{N} - 2\nu_p + 4\nu - 11\Big) z^2 \\
& - 2\Big(((1-2p)(\nu-2) + p\nu)\mathcal{N} - \nu\nu_p + \nu^2 + 4\nu_p - (5+3p)\nu + 10\Big) z \\
& - 2\Big(((1-p^2)(\nu-2) + 2p(2-p))\mathcal{N} \\
& \qquad - 3(1-p)\nu(\nu-2) + 4\nu_p - (9-p^2)\nu + 6\Big) + \mathcal{B}(k_2)\,. \qquad (66)
\end{aligned}$$

  That $\mathcal{R}_+(k_2) \leq 0$ follows directly from $\mathcal{B}(k_2) \leq 2(1-p^2)\nu^2$ and the inequalities below which again are an immediate consequence of (53), the identity $\nu = \nu_p p$ and the fact that $\nu - 2 = z + k_2 \geq z$.

$$\begin{aligned}
-2z^3 - 11z^2 - 2(\nu^2 + 10)z - 12 &\leq 0\,, \\
(1-2p)\mathcal{N} z^2 \;-\; 2(1-2p)\mathcal{N}(\nu-2)z &\leq 0\,, \\
-(2\nu_p - 4\nu)z^2 \;-\; 2(4\nu_p - (5+3p)\nu)z &\leq 0\,, \\
-2(p\nu\mathcal{N} - \nu\nu_p)z &\leq 0\,, \\
-2((1-p^2)(\nu-2)\mathcal{N} - (4+p)(1-p)\nu(\nu-2)) &\leq 0\,, \\
-2(2p(2-p)\mathcal{N} + 4\nu_p - (11 - 3p^2)\nu) &\leq 0\,.
\end{aligned}$$

(b) Writing $C(k_2)$ as a sum of terms containing different powers of z leads to the following representation for $C(k_2)$:

$$\begin{aligned}
C(k_2) \;=\; & \;\; 2z^3 - \Big((1-2p)\mathcal{N} - 2\nu_p + 4\nu - 9\Big) z^2 \\
& + \Big((2(1-p)\nu - 3(1-2p))\mathcal{N} - 2\nu\nu_p + 2\nu^2 + 6\nu_p - 12\nu + 13\Big) z \\
& + \Big((3(1-p)\nu - 2(1-2p))\mathcal{N} - 3\nu\nu_p + 3\nu^2 + 4\nu_p - 8\nu + 6\Big). \qquad (67)
\end{aligned}$$

That $C(k_2) \geq 0$ follows directly from the inequalities below.

$$
\begin{aligned}
2\,z^3 \;+\; 9\,z^2 \;+\; (2\nu^2 + 13)z \;+\; (3\nu^2 + 6) &\geq 0\,,\\
(2\nu_p - 4\nu)z^2 \;+\; (6\nu_p - 12\nu)z \;+\; (2p\mathcal{N} + 2\nu_p - 8\nu) &\geq 0\,,\\
-\,(1-2p)\mathcal{N}z^2 \;+\; 2(1-2p)\mathcal{N}(\nu-2)z &\geq 0\,,\\
((2p\nu + (1-2p))\mathcal{N} - 2\nu\nu_p)z &\geq 0\,,\\
((1-p)(3\nu-2)\mathcal{N} - (3\nu-2)\nu_p) &\geq 0\,.
\end{aligned}
$$

This concludes the proof of $S(i,\,k_2) \leq 0$ for all $1 \leq i \leq m - k_2 - 1$ and $k_2 \in [{k_2}^{(min)},\, m]$ which establishes the feasibility of all $k_2 \in [{k_2}^{(min)},\, m - 1 + {\delta_m}^{(l)}]$, provided $\nu > 2 - p$ in the case ${\delta_m}^{(l)} = 0$ and $\nu > 2 + \sqrt{2}$ in the case ${\delta_m}^{(l)} = 1$.

Case (ii): Reflection centre k_4

A preliminary range for the reflection centres k_4 is given by the interval $[m + 1 - {\delta_m}^{(r)},\, {k_4}^{(max)}]$, due to Lemma 1. We will examine the case ${\delta_m}^{(r)} = 1$ (m is part of the transformation to the right) only, since all statements we establish here hold likewise for the case ${\delta_m}^{(r)} = 0$.

In the sequel we will show that for ${\delta_m}^{(r)} = 1$ all $k_4 \in [m,\, {k_4}^{(max)}]$ (and consequently for ${\delta_m}^{(r)} = 0$ all $k_4 \in [m+1,\, {k_4}^{(max)}]$) are feasible (meet the requirement (15)), provided $n \geq 3$. Similar to Case (i), the proof of this statement splits up into two parts.

(1) Show that ${k_5}^{(max)} \leq n$, where ${k_5}^{(max)} = 2{k_4}^{(max)} - m$.

(2) Show that $S(i,\,k_4) \geq 0$ for all $1 \leq i \leq k_4 - m - 1$ and $k_4 \in [m,\, {k_4}^{(max)}]$.

As in the binomial case, we can establish the slightly sharper restraint that ${k_5}^{(max)} \leq n-1$ for all $\nu \geq 3$ and ${k_5}^{(max)} \leq n$ for all $0 < \nu < 3$, provided $n \geq 3$.

We can see with half an eye that ${x_4}^{(max)} \geq \nu$, that is ${k_4}^{(max)} \geq m$ for all $\nu > 0$. This allows us to set $k_4 = {k_4}^{(max)}$ in our algorithm HPRS^r without the need to explicitly check that $k_4 \geq m$ is satisfied. (In the case ${\delta_m}^{(r)} = 0$, however, we are forced to guarantee $k_4 \geq m + 1$ by some extra statements.)

(1) That ${k_5}^{(max)} \leq n$ for all $\nu > 0$ and ${k_5}^{(max)} \leq n - 1$ for all $\nu \geq 3$, provided $n \geq 3$, follows directly from Lemma 3, (ii), as we will outline in detail below.

${k_4}^{(max)}$ (55) of the hypergeometric distributions is always less than or equal to ${k_4}^{(max)}$ (39) of the Poisson/binomial distributions, due to the additional factor $(1 - (n + 2)/(N+3))$ under the square root. Furthermore, the domain of $p_N(k)$ is $\{0,\,1,\,\ldots\,n\}$, due to the parameter region $1 \leq n \leq M \leq N/2$ we only need to consider here. But this domain coincides with the domain of the corresponding binomial distribution with parameter n (and parameter $p = (M+1)/(N+2)$). Therefore the statements in Lemma 3, (ii) which refer to k_5 apply to the hypergeometric distributions as well. That is, $k_5 \leq n - 1$ for all $\nu \geq 3$ and $k_5 \leq n$ for all $\nu > 0$, provided $n \geq 3$. Substituting ${k_5}^{(max)}$ for k_5 then concludes our argumentation.

(2) As in the Poisson/binomial case, the proof of $S(i,\, k_4) \geq 0$ for all $1 \leq i \leq k_4 - m - 1$ and $k_4 \in [m,\, {k_4}^{(max)}]$ can be split up into the following two parts.

(a) Verify that $R(i,\, k_4) = A(k_4)\, i^2 + B(k_4) \geq 0$.

(b) Verify that $C(k_4) \leq 0$.

Both statements together with $N(k_4) \leq 0$ establish the feasibility of $k_4 \leq {k_4}^{(max)}$, for then $S(i,\, k_4) = R(i,\, k_4)\, i^2 + N(k_4) C(k_4) \geq 0$. For the proof of the two statements take into account that $k_4 - \nu \geq 1$ which is due to $1 \leq i \leq k_4 - m - 1$ and $\nu < m+1$.

For clarification, we again introduce the quantity z. But here we set $z = k_4 - \nu - 1$. Since $k_4 - \nu \geq 1$, we once again have $z \geq 0$.

(a) Analogous to Case (i), we set

$$\mathcal{B}(k_4) \;=\; -\,\frac{6\nu(\nu_p - \nu + 1 - p)}{\mathcal{N}+1}\, k_4 \;+\; \frac{2\nu({\nu_p}^2 - \nu^2 - (1 - p^2))}{\mathcal{N}+1}\,. \tag{68}$$

$\mathcal{B}(k_4)$, the fractional part of $B(k_4)$, satisfies the sequence of inequalities

$$\mathcal{B}(k_4) \;\geq\; -\,\frac{2(1-p)\nu(\nu_p+1)(3k_4+1+p)}{\mathcal{N}+1} \;\geq\; -\,6(1-p)\nu(\nu+p)\,.$$

The first inequality can be quickly verified, using the identity $\nu = \nu_p\, p$. The second one is a direct consequence of $3k_4 + 1 + p \leq 3(k_4+1) \leq 3\nu_p$ and $\mathcal{N} + 1 \geq \nu_p/p$.

$R(i,\, k_4)$ attains its minimum, with respect to i, at $i_{max} = k_4 - m - 1$, due to $A(k_4) \leq 0$ for all $k_4 \geq m$. We will use the little bit larger value $\hat{\imath}_{max} = k_4 - \nu$ instead of i_{max} in order to avoid the variable m in our test condition. Since

$$R(i,\, k_4) \;\geq\; R(i_{max},\, k_4) \;\geq\; R(\hat{\imath}_{max},\, k_4) \;=\; R(k_4 - \nu,\, k_4)$$

it suffices to show that $\mathcal{R}(k_4) = R(k_4 - \nu,\, k_4) \geq 0$. Because of $z = k_4 - \nu - 1$ we can read k_4 as a function of z. Substituting $k_4(z) = \nu + 1 + z$ for k_4 in $\mathcal{R}(k_4) - \mathcal{B}(k_4)$ and arranging the resulting expression in terms of powers of z brings us the following representation for the test function $\mathcal{R}(k_4)$:

$$\begin{aligned}
\mathcal{R}(k_4) \;=\;\;& 2z^3 + \Big((1-2p)\mathcal{N} - 2\nu_p + 4\nu + 11\Big) z^2 \\
&+ 2\Big(((1-p)\nu + 2(1-2p))\mathcal{N} - \nu\nu_p + \nu^2 - 4\nu_p + (11-3p)\nu + 10\Big) z \\
&+ 2\Big(((1-p)(2-p)\nu + 2(1-2p))\mathcal{N} - 3\nu\nu_p \\
&\qquad + 3(2-p)\nu^2 - 4\nu_p + (12 - 3p - p^2)\nu + 6\Big) + \mathcal{B}(k_4)\,.
\end{aligned} \tag{69}$$

That $\mathcal{R}(k_4) \geq 0$ follows directly from $\mathcal{B}(k_4) \geq -6(1-p)\nu(\nu+p)$ and the inequalities below which again are an immediate consequence of (53) and the identity $\nu = \nu_p p$.

$$2z^3 + 11z^2 + 2((3(1-p)\nu + 10)z + (4 - 3p - p^2 - 3p(1-p))\nu + 6) \geq 0\,,$$
$$((1-2p)\mathcal{N} - 2\nu_p + 4\nu)(z^2 + 4z + 4) \geq 0\,,$$
$$2((1-p)\nu\mathcal{N} - \nu\nu_p + \nu^2)z \geq 0\,,$$
$$2((1-p)(2-p)\nu\mathcal{N} - 3(1-p)^2\nu\nu_p - 3(1-p)\nu^2) \geq 0\,.$$

(b) Writing $C(k_4)$ as a sum of terms containing different powers of z leads to the following representation for $C(k_4)$:

$$\begin{aligned} C(k_4) \;=\; & -2z^3 - \Big((1-2p)\mathcal{N} - 2\nu_p + 4\nu + 9\Big)z^2 \\ & - \Big((2(1-p)\nu + 3(1-2p))\mathcal{N} - 2\nu\nu_p + 2\nu^2 - 6\nu_p + 12\nu + 13\Big)z \\ & - \Big((3(1-p)\nu + 2(1-2p))\mathcal{N} - 3\nu\nu_p + 3\nu^2 - 4\nu_p + 8\nu + 6\Big). \end{aligned} \tag{70}$$

Considering the inequalities below, it becomes obvious that $C(k_4) \leq 0$.

$$-2z^3 - 9z^2 - 13z - 6 \leq 0\,,$$
$$-((1-2p)\mathcal{N} - 2\nu_p + 4\nu)(z^2 + 3z + 2) \leq 0\,,$$
$$-((1-p)\nu\mathcal{N} - \nu\nu_p + \nu^2)(2z + 3) \leq 0\,.$$

This establishes the feasibility of all $k_4 \in [m + 1 - \delta_m{}^{(r)},\ k_4{}^{(max)}]$, provided $n \geq 3$. ■

Appendix D

Characteristics of Poisson, Binomial and Hypergeometric Algorithms

Table 33: Average number α of samples X from $h(x)$ to generate one sample K from $p(k)$, Poisson algorithms.

Algorithm \ μ	3.5	5.0	10	20	50	200	1000	10000
PRUA	1.869	1.727	1.616	1.540	1.475	1.421	1.392	1.376
PRUE	1.812	1.702	1.599	1.525	1.462	1.415	1.389	1.375
PAMS	1.607	1.470	1.310	1.230	1.182	1.158	1.152	1.150
PDEX	—	1.678	1.556	1.478	1.415	1.364	1.337	1.322
PNEX	—	—	1.476	1.323	1.203	1.101	1.046	1.015
PDAC_2	—	—	1.068	1.048	1.030	1.015	1.007	1.002
P2PE	—	1.431	1.378	1.378	1.380	1.387	1.374	1.373
PTPE	—	—	1.623	1.473	1.304	1.148	1.141	1.145
PTRS	—	—	1.329	1.252	1.197	1.158	1.139	1.128
PTRD	—	—	1.329	1.252	1.197	1.158	1.139	1.128
PPRS^c	1.449	1.269	1.131	1.037	1.053	1.037	1.024	1.024
PPRS^l	1.357	1.304	1.152	1.042	1.058	1.039	1.025	1.024
PPRS^r	1.327	1.174	1.107	1.033	1.053	1.038	1.025	1.024

Table 34: Average number of uniforms N needed to generate one sample K from $p(k)$, Poisson algorithms.

Algorithm \ μ	3.5	5.0	10	20	50	200	1000	10000
PRUA	3.737	3.455	3.232	3.079	2.949	2.841	2.783	2.752
PRUE	3.624	3.404	3.197	3.050	2.923	2.830	2.779	2.751
PAMS	3.136	2.896	2.610	2.459	2.364	2.316	2.303	2.300
PDEX	—	3.357	3.113	2.957	2.829	2.727	2.673	2.644
PNEX	—	—	3.246	3.001	2.811	2.649	2.560	2.510
PDAC_2	—	—	1.883	1.908	1.934	1.959	1.974	1.982
P2PE	—	2.862	2.756	2.756	2.761	2.773	2.749	2.746
PTPE	—	—	3.247	2.946	2.609	2.297	2.282	2.289
PTRS	—	—	2.657	2.504	2.395	2.316	2.277	2.257
PTRD	—	—	2.194	1.886	1.661	1.496	1.414	1.371
PPRS^c	2.309	1.852	1.529	1.385	1.426	1.434	1.439	1.449
PPRS^l	2.050	1.972	1.625	1.446	1.465	1.453	1.447	1.451
PPRS^r	1.959	1.586	1.480	1.414	1.447	1.449	1.447	1.451

Table 35: Average number α of samples X from $h(x)$ to generate one sample K from $p_N(k)$, Algorithm HRUA (Stadlober, 1989a).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	2.112							
20	10	2.038	2.207						
40	20	1.963	2.024	1.932					
80	20	1.924	1.884						
	40	1.930	1.959	1.811	1.748				
200	50	1.903	1.827	1.705					
	100	1.911	1.926	1.762	1.657	1.596			
800	100	1.884	1.771	1.654					
	200	1.892	1.802	1.670	1.583	1.514			
	400	1.902	1.910	1.741	1.628	1.535	1.477		
4000	250	1.876	1.747	1.632					
	500	1.880	1.760	1.639	1.558	1.489			
	1000	1.890	1.796	1.662	1.572	1.496	1.437		
	2000	1.899	1.906	1.736	1.621	1.524	1.447	1.415	
40000	1250	1.872	1.735	1.622	1.544				
	2500	1.874	1.742	1.626	1.547	1.480			
	5000	1.879	1.758	1.636	1.553	1.483	1.426		
	10000	1.889	1.795	1.661	1.570	1.493	1.430	1.397	
	20000	1.899	1.905	1.735	1.619	1.522	1.443	1.402	1.383

Table 36: Average number α of samples X from $h(x)$ to generate one sample K from $p_n(k)$, Algorithms BRUA and PRUA (Stadlober, 1989a).

	$n \backslash \mu$	3.5	5.0	10	20	50	200	1000	10000
BRUA	7	1.899							
	10	1.895	1.905						
	20	1.883	1.795	1.735					
	40	1.876	1.758	1.661	1.619				
	100	1.872	1.739	1.632	1.562	1.522			
	400	1.869	1.730	1.620	1.545	1.483	1.443		
	2000	1.869	1.728	1.617	1.541	1.476	1.424	1.401	
	20000	1.869	1.727	1.616	1.540	1.475	1.421	1.392	1.379
PRUA	∞	1.869	1.727	1.616	1.540	1.475	1.421	1.392	1.376

Table 37: Average number α of samples X from $h(x)$ to generate one sample K from $p_N(k)$, Algorithm HRUE (Stadlober, 1989a).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	—							
20	10	1.745	1.719						
40	20	1.731	1.719	1.688					
80	20	1.710	1.707						
	40	1.714	1.715	1.590	1.593				
200	50	1.683	1.673	1.598					
	100	1.703	1.706	1.602	1.542	1.506			
800	100	1.750	1.661	1.583					
	200	1.688	1.653	1.569	1.527	1.470			
	400	1.697	1.700	1.600	1.531	1.470	1.437		
4000	250	1.785	1.686	1.596					
	500	1.761	1.672	1.592	1.516	1.461			
	1000	1.694	1.647	1.572	1.525	1.466	1.418		
	2000	1.695	1.699	1.598	1.525	1.468	1.420	1.400	
40000	1250	1.801	1.695	1.598	1.523				
	2500	1.790	1.689	1.597	1.521	1.464			
	5000	1.764	1.674	1.593	1.514	1.465	1.416		
	10000	1.696	1.646	1.574	1.524	1.463	1.416	1.390	
	20000	1.694	1.698	1.598	1.524	1.468	1.418	1.391	1.378

Table 38: Average number α of samples X from $h(x)$ to generate one sample K from $p_n(k)$, Algorithms BRUE and PRUE (Stadlober, 1989a).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BRUE	7	1.694							
	10	1.684	1.698						
	20	1.739	1.646	1.598					
	40	1.780	1.674	1.574	1.524				
	100	1.800	1.692	1.595	1.520	1.468			
	400	1.809	1.700	1.599	1.522	1.465	1.418		
	2000	1.812	1.702	1.599	1.525	1.462	1.415	1.390	
	20000	1.812	1.702	1.599	1.525	1.461	1.415	1.389	1.376
PRUE	∞	1.812	1.702	1.599	1.525	1.462	1.415	1.389	1.375

Table 39: Average number α of samples X from $h(x)$ to generate one sample K from $p_n(k)$, Algorithms BAMS (Thurner, 1990) and PAMS (Stadlober, 1989b).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BAMS	7	1.893							
	10	1.807	1.670						
	20	1.707	1.570	1.410					
	40	1.657	1.520	1.360	1.280				
	100	1.627	1.490	1.330	1.250	1.202			
	400	1.612	1.475	1.315	1.235	1.187	1.163		
	2000	1.608	1.471	1.311	1.231	1.183	1.159	1.153	
	20000	1.607	1.470	1.310	1.230	1.182	1.158	1.152	1.150
PAMS	∞	1.607	1.470	1.310	1.230	1.182	1.158	1.152	1.150

Table 40: Average number α of samples X from $h(x)$ to generate one sample K from $p_n(k)$, Algorithms BDEX and PDEX (Ahrens & Dieter, 1980).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BDEX	7	—							
	10	—	—						
	20	—	—	1.678					
	40	—	1.709	1.600	1.556				
	100	—	1.690	1.571	1.500	1.460			
	400	—	1.681	1.560	1.483	1.422	1.384		
	2000	—	1.679	1.557	1.479	1.416	1.366	1.346	
	20000	—	1.678	1.556	1.479	1.415	1.364	1.337	1.325
PDEX	∞	—	1.678	1.556	1.478	1.415	1.364	1.337	1.322

Table 41: Average number α of samples X from $h(x)$ to generate one sample K from $p_n(k)$, Algorithms BNEX (Devroye & Naderisamani, 1980) and PNEX (Devroye, 1986).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BNEX	7	1.000							
	10	1.000	1.000						
	20	1.000	1.000	2.196					
	40	1.000	1.000	2.270	1.805				
	100	1.000	1.000	2.471	1.928	1.491			
	400	1.000	1.000	2.483	1.965	1.593	1.243		
	2000	1.000	1.000	2.486	1.973	1.611	1.296	1.111	
	20000	1.000	1.000	2.487	1.975	1.614	1.307	1.138	1.037
PNEX	∞	—	—	1.476	1.323	1.203	1.101	1.046	1.015

Table 42: Average number α of samples X from $h(x)$ to generate one sample K from $p_N(k)$, Algorithm H2PE (Kachitvichyanukul & Schmeiser, 1985).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	—							
20	10	—	1.458						
40	20	—	1.577	1.475					
80	20	—	1.440						
	40	—	1.514	1.390	1.450				
200	50	—	1.402	1.399					
	100	—	1.486	1.483	1.449	1.427			
800	100	—	1.503	1.450					
	200	—	1.391	1.488	1.402	1.363			
	400	—	1.473	1.455	1.397	1.440	1.382		
4000	250	—	1.462	1.406					
	500	—	1.486	1.421	1.433	1.395			
	1000	—	1.389	1.472	1.379	1.363	1.384		
	2000	—	1.470	1.448	1.387	1.403	1.395	1.372	
40000	1250	—	1.444	1.388	1.391				
	2500	—	1.455	1.396	1.399	1.407			
	5000	—	1.482	1.415	1.420	1.367	1.381		
	10000	—	1.388	1.468	1.375	1.412	1.391	1.367	
	20000	—	1.469	1.447	1.385	1.396	1.366	1.372	1.376

Table 43: Average number α of samples X from $h(x)$ to generate one sample K from $p_n(k)$, Algorithms B2PE and P2PE (Kachitvichyanukul & Schmeiser, 1985).

	$n \backslash \mu$	3.5	5.0	10	20	50	200	1000	10000
B2PE	7	—							
	10	—	1.469						
	20	—	1.388	1.447					
	40	—	1.481	1.468	1.385				
	100	—	1.450	1.406	1.451	1.395			
	400	—	1.436	1.384	1.392	1.364	1.403		
	2000	—	1.432	1.379	1.380	1.388	1.391	1.375	
	20000	—	1.431	1.378	1.378	1.381	1.390	1.377	1.374
P2PE	∞	—	1.431	1.378	1.378	1.380	1.387	1.374	1.373

Table 44: Average number α of samples X from $h(x)$ to generate one sample K from $p_n(k)$, Algorithms BTPE (Kachitvichyanukul & Schmeiser, 1988) and PTPE (Schmeiser & Kachitvichyanukul, 1982).

	$n \backslash \mu$	3.5	5.0	10	20	50	200	1000	10000
BTPE	7	—							
	10	—	—						
	20	—	—	1.998					
	40	—	—	1.906	1.793				
	100	—	—	1.895	1.754	1.499			
	400	—	—	1.897	1.679	1.427	1.229		
	2000	—	—	1.899	1.663	1.376	1.202	1.145	
	20000	—	—	1.899	1.660	1.365	1.193	1.135	1.145
PTPE	∞	—	—	1.623	1.473	1.304	1.148	1.141	1.145

Table 45: Average number α of samples X from $h(x)$ to generate one sample K from $p_n(k)$, Algorithms BTRD (Hörmann, 1993) and PTRD (Hörmann 1992).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BTRD	7	—							
	10	—	—						
	20	—	—	1.410					
	40	—	—	1.368	1.343				
	100	—	—	1.350	1.304	1.273			
	400	—	—	1.343	1.291	1.241	1.205		
	2000	—	—	1.341	1.288	1.235	1.186	1.164	
	20000	—	—	1.340	1.287	1.234	1.184	1.155	1.140
PTRD	∞	—	—	1.329	1.252	1.197	1.158	1.139	1.128

Table 46: Average number α of samples X from $h(x)$ to generate one sample K from $p_N(k)$, Algorithm HPRS^c (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	*1.250*							
20	10	*1.290*	1.248						
40	20	*1.340*	1.314	1.368					
80	20	1.384	1.409						
	40	*1.367*	1.351	1.070	1.107				
200	50	1.403	1.230	1.132					
	100	*1.384*	1.373	1.096	1.172	1.090			
800	100	1.424	1.254	1.100					
	200	1.413	1.243	1.157	1.100	1.050			
	400	*1.393*	1.385	1.107	1.059	1.077	1.033		
4000	250	1.437	1.263	1.118					
	500	1.429	1.259	1.112	1.087	1.044			
	1000	1.416	1.246	1.092	1.070	1.054	1.037		
	2000	*1.395*	1.388	1.110	1.064	1.089	1.041	1.038	
40000	1250	1.443	1.267	1.126	1.102				
	2500	1.439	1.265	1.123	1.100	1.042			
	5000	1.430	1.260	1.114	1.092	1.055	1.035		
	10000	1.417	1.246	1.093	1.073	1.059	1.039	1.028	
	20000	*1.396*	1.389	1.110	1.065	1.035	1.051	1.033	1.023

Table 47: Average number α of samples X from $h(x)$ to generate one sample K from $p_n(k)$, Algorithms BPRS^c and PPRS^c (Zechner, 1993).

	$n \backslash \mu$	3.5	5.0	10	20	50	200	1000	10000
BPRS^c	7	1.107							
	10	1.146	1.389						
	20	1.424	1.247	1.110					
	40	1.435	1.260	1.093	1.065				
	100	1.443	1.266	1.118	1.081	1.035			
	400	1.448	1.268	1.128	1.102	1.036	1.027		
	2000	1.449	1.269	1.131	1.107	1.050	1.034	1.028	
	20000	1.449	1.269	1.131	1.108	1.052	1.036	1.026	1.025
PPRS^c	∞	1.449	1.269	1.131	1.037	1.053	1.037	1.024	1.024

Table 48: Average number α of samples X from $h(x)$ to generate one sample K from $p_N(k)$, Algorithm HPRS^l (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	*1.316*							
20	10	*1.295*	1.208						
40	20	*1.283*	1.260	1.293					
80	20	1.329	1.308						
	40	*1.282*	1.283	1.062	1.104				
200	50	1.348	1.239	1.133					
	100	*1.282*	1.297	1.093	1.158	1.092			
800	100	1.358	1.281	1.110					
	200	1.355	1.259	1.152	1.105	1.054			
	400	*1.283*	1.304	1.105	1.060	1.080	1.034		
4000	250	1.358	1.295	1.135					
	500	1.359	1.288	1.127	1.097	1.049			
	1000	1.357	1.264	1.097	1.075	1.057	1.039		
	2000	*1.283*	1.306	1.107	1.066	1.091	1.043	1.039	
40000	1250	1.358	1.300	1.146	1.114				
	2500	1.359	1.297	1.142	1.111	1.046			
	5000	1.359	1.289	1.130	1.102	1.061	1.037		
	10000	1.358	1.265	1.099	1.078	1.064	1.041	1.029	
	20000	*1.283*	1.306	1.108	1.067	1.036	1.054	1.034	1.023

Table 49: Average number α of samples X from $h(x)$ to generate one sample K from $p_n(k)$, Algorithms BPRS^l and PPRS^l (Zechner, 1993).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BPRS^l	7	1.134							
	10	1.089	1.306						
	20	1.359	1.265	1.108					
	40	1.359	1.289	1.099	1.067				
	100	1.358	1.299	1.135	1.089	1.037			
	400	1.358	1.303	1.148	1.114	1.039	1.028		
	2000	1.357	1.303	1.151	1.119	1.055	1.036	1.029	
	20000	1.357	1.304	1.152	1.120	1.058	1.038	1.027	1.025
PPRS^l	∞	1.357	1.304	1.152	1.042	1.058	1.039	1.025	1.024

Table 50: Average number α of samples X from $h(x)$ to generate one sample K from $p_N(k)$, Algorithm HPRS^r (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	1.413							
20	10	1.464	1.208						
40	20	1.535	1.260	1.293					
80	20	1.271	1.327						
	40	1.575	1.283	1.062	1.104				
200	50	1.278	1.137	1.122					
	100	1.600	1.297	1.093	1.158	1.092			
800	100	1.296	1.157	1.082					
	200	1.284	1.146	1.142	1.097	1.051			
	400	1.613	1.304	1.105	1.060	1.080	1.034		
4000	250	1.311	1.166	1.097					
	500	1.301	1.161	1.092	1.080	1.044			
	1000	1.285	1.148	1.076	1.066	1.055	1.039		
	2000	1.617	1.306	1.107	1.066	1.091	1.043	1.039	
40000	1250	1.319	1.171	1.103	1.093				
	2500	1.313	1.168	1.101	1.090	1.042			
	5000	1.302	1.162	1.094	1.084	1.055	1.036		
	10000	1.285	1.149	1.077	1.068	1.060	1.041	1.029	
	20000	1.617	1.306	1.108	1.067	1.036	1.054	1.034	1.023

Table 51: Average number α of samples X from $h(x)$ to generate one sample K from $p_n(k)$, Algorithms BPRS^r and PPRS^r (Zechner, 1993).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BPRS^r	7	1.041							
	10	1.155	1.306						
	20	1.294	1.149	1.108					
	40	1.309	1.162	1.077	1.067				
	100	1.319	1.169	1.097	1.075	1.037			
	400	1.325	1.173	1.105	1.092	1.036	1.028		
	2000	1.327	1.174	1.107	1.096	1.050	1.034	1.029	
	20000	1.327	1.174	1.107	1.097	1.052	1.037	1.027	1.025
PPRS^r	∞	1.327	1.174	1.107	1.033	1.053	1.038	1.025	1.024

Table 52: Average number of uniforms N needed to generate one sample K from $p_n(k)$, Algorithms BAMS (Thurner, 1990) and PAMS (Stadlober, 1989b).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BAMS	7	3.746							
	10	3.576	3.322						
	20	3.357	3.115	2.818					
	40	3.246	3.006	2.715	2.560				
	100	3.180	2.940	2.652	2.499	2.404			
	400	3.147	2.907	2.620	2.469	2.374	2.326		
	2000	3.138	2.899	2.612	2.461	2.366	2.318	2.305	
	20000	3.136	2.897	2.610	2.459	2.364	2.316	2.303	2.301
PAMS	∞	3.136	2.896	2.610	2.459	2.364	2.316	2.303	2.300

Table 53: Average number of uniforms N needed to generate one sample K from $p_n(k)$, Algorithms BNEX (Devroye & al., 1980) and PNEX (Devroye, 1986).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BNEX	7	4.500							
	10	4.500	6.000						
	20	4.500	6.000	5.325					
	40	4.500	6.000	5.470	4.407				
	100	4.500	6.000	6.026	4.705	3.655			
	400	4.500	6.000	6.044	4.779	3.910	3.073		
	2000	4.500	6.000	6.050	4.793	3.946	3.200	2.755	
	20000	4.500	6.000	6.051	4.796	3.953	3.224	2.820	2.577
PNEX	∞	—	—	3.246	3.001	2.811	2.649	2.560	2.510

Table 54: Average number of uniforms N needed to generate one sample K from $p_n(k)$, Algorithms BTRD (Hörmann, 1993) and PTRD (Hörmann, 1992).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BTRD	7	—							
	10	—	—						
	20	—	—	2.453					
	40	—	—	2.266	2.153				
	100	—	—	2.190	1.994	1.872			
	400	—	—	2.159	1.943	1.750	1.621		
	2000	—	—	2.151	1.931	1.730	1.556	1.480	
	20000	—	—	2.149	1.929	1.726	1.547	1.448	1.401
PTRD	∞	—	—	2.194	1.886	1.661	1.496	1.414	1.371

Table 55: Average number of uniforms N needed to generate one sample K from $p_N(k)$, Algorithm HPRS^c (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	*1.658*							
20	10	*1.784*	1.674						
40	20	*1.930*	1.902	2.080					
80	20	2.090	2.190						
	40	*2.004*	2.012	1.323	1.412				
200	50	2.152	1.747	1.479					
	100	*2.050*	2.078	1.380	1.603	1.429			
800	100	2.229	1.809	1.457					
	200	2.183	1.778	1.552	1.452	1.399			
	400	*2.072*	2.111	1.407	1.363	1.428	1.400		
4000	250	2.270	1.833	1.498					
	500	2.243	1.821	1.483	1.452	1.404			
	1000	2.191	1.785	1.440	1.418	1.417	1.425		
	2000	*2.079*	2.120	1.414	1.373	1.454	1.422	1.437	
40000	1250	2.291	1.844	1.517	1.485				
	2500	2.276	1.838	1.509	1.478	1.412			
	5000	2.247	1.823	1.489	1.461	1.418	1.429		
	10000	2.193	1.787	1.442	1.423	1.426	1.431	1.440	
	20000	*2.080*	2.122	1.416	1.375	1.379	1.434	1.440	1.443

Table 56: Average number of uniforms N needed to generate one sample K from $p_n(k)$, Algorithms BPRS^c and PPRS^c (Zechner, 1993).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BPRS^c	7	1.394							
	10	1.487	2.122						
	20	2.225	1.787	1.416					
	40	2.265	1.823	1.443	1.375				
	100	2.291	1.841	1.498	1.439	1.380			
	400	2.305	1.849	1.521	1.484	1.403	1.412		
	2000	2.308	1.851	1.527	1.495	1.421	1.429	1.437	
	20000	2.309	1.852	1.529	1.498	1.425	1.433	1.441	1.448
PPRS^c	∞	2.309	1.852	1.529	1.385	1.426	1.434	1.439	1.449

Table 57: Average number of uniforms N needed to generate one sample K from $p_N(k)$, Algorithm HPRS^l (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	*1.662*							
20	10	*1.647*	1.622						
40	20	*1.658*	1.751	1.857					
80	20	1.995	1.891						
	40	*1.672*	1.814	1.402	1.463				
200	50	2.031	1.874	1.515					
	100	*1.683*	1.853	1.444	1.582	1.468			
800	100	2.050	1.941	1.552					
	200	2.047	1.908	1.556	1.490	1.442			
	400	*1.689*	1.873	1.462	1.424	1.464	1.431		
4000	250	2.052	1.962	1.595					
	500	2.053	1.952	1.580	1.517	1.444			
	1000	2.051	1.916	1.530	1.483	1.459	1.447		
	2000	*1.691*	1.879	1.466	1.431	1.481	1.446	1.452	
40000	1250	2.052	1.969	1.614	1.546				
	2500	2.052	1.965	1.606	1.541	1.451			
	5000	2.054	1.954	1.586	1.526	1.456	1.449		
	10000	2.052	1.917	1.534	1.487	1.466	1.451	1.449	
	20000	*1.691*	1.880	1.468	1.433	1.422	1.456	1.451	1.448

Table 58: Average number of uniforms N needed to generate one sample K from $p_n(k)$, Algorithms BPRS^l and PPRS^l (Zechner, 1993).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BPRS^l	7	1.596							
	10	1.379	1.880						
	20	2.054	1.918	1.468					
	40	2.053	1.954	1.534	1.433				
	100	2.052	1.967	1.595	1.505	1.423			
	400	2.051	1.971	1.618	1.545	1.442	1.435		
	2000	2.050	1.972	1.624	1.554	1.461	1.448	1.447	
	20000	2.050	1.972	1.625	1.556	1.464	1.451	1.449	1.451
PPRS^l	∞	2.050	1.972	1.625	1.446	1.465	1.453	1.447	1.451

Table 59: Average number of uniforms N needed to generate one sample K from $p_N(k)$, Algorithm HPRS^r (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	2.111							
20	10	2.279	1.622						
40	20	2.478	1.751	1.857					
80	20	1.729	1.955						
	40	2.581	1.814	1.402	1.463				
200	50	1.772	1.486	1.497					
	100	2.645	1.853	1.444	1.582	1.468			
800	100	1.856	1.537	1.436					
	200	1.796	1.507	1.545	1.478	1.436			
	400	2.677	1.873	1.462	1.424	1.464	1.431		
4000	250	1.907	1.563	1.460					
	500	1.872	1.548	1.452	1.455	1.434			
	1000	1.802	1.512	1.426	1.435	1.442	1.445		
	2000	2.686	1.879	1.466	1.431	1.481	1.446	1.452	
40000	1250	1.936	1.576	1.473	1.476				
	2500	1.915	1.568	1.467	1.472	1.436			
	5000	1.876	1.551	1.455	1.461	1.446	1.446		
	10000	1.804	1.513	1.428	1.438	1.448	1.448	1.449	
	20000	2.688	1.880	1.468	1.433	1.422	1.456	1.451	1.448

Table 60: Average number of uniforms N needed to generate one sample K from $p_n(k)$, Algorithms BPRS^r and PPRS^r (Zechner, 1993).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BPRS^r	7	1.316							
	10	1.632	1.880						
	20	1.846	1.514	1.468					
	40	1.900	1.551	1.428	1.433				
	100	1.935	1.572	1.460	1.448	1.423			
	400	1.953	1.583	1.475	1.476	1.430	1.435		
	2000	1.958	1.585	1.479	1.483	1.444	1.445	1.447	
	20000	1.959	1.586	1.480	1.484	1.446	1.448	1.448	1.451
PPRS^r	∞	1.959	1.586	1.480	1.414	1.447	1.449	1.447	1.451

Table 61: Probability $P(I)$ of accepting K immediately, given that K is accepted as a sample from $p_N(k)$, Algorithm HPRS^c (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	*0.842*							
20	10	*0.796*	0.821						
40	20	*0.751*	0.727	0.657					
80	20	0.678	0.628						
	40	*0.730*	0.689	0.816	0.802				
200	50	0.655	0.713	0.784					
	100	*0.718*	0.669	0.811	0.740	0.751			
800	100	0.620	0.700	0.742					
	200	0.644	0.707	0.761	0.749	0.701			
	400	*0.713*	0.659	0.806	0.754	0.726	0.666		
4000	250	0.604	0.693	0.738					
	500	0.615	0.697	0.740	0.723	0.684			
	1000	0.641	0.706	0.743	0.722	0.690	0.649		
	2000	*0.711*	0.657	0.805	0.756	0.725	0.660	0.639	
40000	1250	0.596	0.689	0.735	0.720				
	2500	0.601	0.691	0.737	0.721	0.673			
	5000	0.614	0.696	0.739	0.722	0.692	0.641		
	10000	0.641	0.706	0.743	0.723	0.693	0.648	0.617	
	20000	*0.711*	0.656	0.805	0.756	0.691	0.669	0.626	0.602

Table 62: Probability $P(I)$ of accepting K immediately, given that K is accepted as a sample from $p_n(k)$, Algorithms BPRS^c and PPRS^c (Zechner, 1993).

	$n \backslash \mu$	3.5	5.0	10	20	50	200	1000	10000
BPRS^c	7	0.820							
	10	0.804	0.656						
	20	0.624	0.706	0.805					
	40	0.606	0.696	0.743	0.756				
	100	0.596	0.690	0.738	0.723	0.691			
	400	0.591	0.688	0.735	0.720	0.668	0.642		
	2000	0.590	0.687	0.734	0.719	0.678	0.639	0.620	
	20000	0.590	0.687	0.734	0.718	0.680	0.639	0.611	0.602
PPRS^c	∞	0.590	0.687	0.733	0.689	0.680	0.640	0.609	0.599

Table 63: Probability $P(I)$ of accepting K immediately, given that K is accepted as a sample from $p_N(k)$, Algorithm HPRS^l (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
7	14	*0.971*							
20	10	*0.943*	0.794						
40	20	*0.909*	0.769	0.729					
80	20	0.663	0.725						
	40	*0.892*	0.751	0.722	0.745				
200	50	0.664	0.604	0.752					
	100	*0.882*	0.740	0.742	0.734	0.715			
800	100	0.665	0.621	0.667					
	200	0.664	0.611	0.749	0.720	0.666			
	400	*0.877*	0.734	0.747	0.695	0.696	0.637		
4000	250	0.665	0.629	0.675					
	500	0.665	0.624	0.673	0.677	0.654			
	1000	0.664	0.612	0.664	0.668	0.656	0.630		
	2000	*0.875*	0.733	0.748	0.700	0.700	0.640	0.627	
40000	1250	0.665	0.632	0.678	0.682				
	2500	0.665	0.630	0.677	0.681	0.642			
	5000	0.665	0.625	0.674	0.678	0.665	0.625		
	10000	0.664	0.613	0.664	0.669	0.661	0.631	0.609	
	20000	*0.875*	0.732	0.749	0.701	0.651	0.651	0.617	0.598

Table 64: Probability $P(I)$ of accepting K immediately, given that K is accepted as a sample from $p_n(k)$, Algorithms BPRS^l and PPRS^l (Zechner, 1993).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BPRS^l	7	0.672							
	10	0.800	0.732						
	20	0.665	0.613	0.749					
	40	0.665	0.625	0.664	0.701				
	100	0.665	0.631	0.675	0.674	0.651			
	400	0.664	0.634	0.679	0.682	0.635	0.622		
	2000	0.664	0.635	0.679	0.683	0.649	0.624	0.610	
	20000	0.664	0.635	0.680	0.683	0.651	0.624	0.605	0.599
PPRS^l	∞	0.664	0.635	0.680	0.638	0.651	0.626	0.602	0.596

Table 65: Probability $P(I)$ of accepting K immediately, given that K is accepted as a sample from $p_N(k)$, Algorithm HPRS^r (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	0.714							
20	10	0.650	0.794						
40	20	0.593	0.769	0.729					
80	20	0.813	0.699						
	40	0.568	0.751	0.722	0.745				
200	50	0.785	0.789	0.747					
	100	0.555	0.740	0.742	0.734	0.715			
800	100	0.737	0.777	0.728					
	200	0.772	0.786	0.740	0.717	0.666			
	400	0.549	0.734	0.747	0.695	0.696	0.637		
4000	250	0.714	0.769	0.733					
	500	0.730	0.774	0.732	0.705	0.654			
	1000	0.768	0.785	0.726	0.697	0.667	0.633		
	2000	0.547	0.733	0.748	0.700	0.700	0.640	0.627	
40000	1250	0.703	0.765	0.734	0.709				
	2500	0.711	0.768	0.734	0.709	0.648			
	5000	0.728	0.773	0.733	0.707	0.665	0.626		
	10000	0.767	0.784	0.726	0.698	0.672	0.633	0.609	
	20000	0.547	0.732	0.749	0.701	0.651	0.651	0.617	0.598

Table 66: Probability $P(I)$ of accepting K immediately, given that K is accepted as a sample from $p_n(k)$, Algorithms BPRS^r and PPRS^r (Zechner, 1993).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BPRS^r	7	0.766							
	10	0.679	0.732						
	20	0.743	0.784	0.749					
	40	0.718	0.773	0.726	0.701				
	100	0.704	0.766	0.733	0.703	0.651			
	400	0.697	0.763	0.734	0.709	0.642	0.622		
	2000	0.695	0.762	0.734	0.710	0.656	0.624	0.610	
	20000	0.695	0.762	0.734	0.710	0.658	0.626	0.605	0.599
PPRS^r	∞	0.695	0.762	0.734	0.652	0.659	0.627	0.602	0.597

Table 67: Probability $P(Q) = P(I) + P(S)$ of accepting K quickly, given that K is accepted as a sample from $p_N(k)$, Algorithm HPRS^c (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	*0.985*							
20	10	*0.970*	0.977						
40	20	*0.970*	0.935	0.938					
80	20	0.960	0.948						
	40	*0.979*	0.934	0.948	0.933				
200	50	0.951	0.934	0.935					
	100	*0.983*	0.944	0.941	0.923	0.933			
800	100	0.938	0.930	0.926					
	200	0.946	0.931	0.932	0.931	0.934			
	400	*0.984*	0.948	0.936	0.940	0.934	0.928		
4000	250	0.944	0.929	0.924					
	500	0.941	0.930	0.925	0.928	0.931			
	1000	0.944	0.930	0.929	0.930	0.930	0.927		
	2000	*0.984*	0.949	0.935	0.940	0.931	0.929	0.928	
40000	1250	0.947	0.927	0.923	0.926				
	2500	0.945	0.928	0.924	0.926	0.927			
	5000	0.941	0.930	0.925	0.928	0.933	0.927		
	10000	0.944	0.930	0.929	0.930	0.930	0.928	0.925	
	20000	*0.984*	0.949	0.935	0.940	0.932	0.931	0.926	0.923

Table 68: Probability $P(Q) = P(I) + P(S)$ of accepting K quickly, given that K is accepted as a sample from $p_n(k)$, Algorithms BPRS^c and PPRS^c (Zechner, 1993).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BPRS^c	7	0.938							
	10	0.947	0.949						
	20	0.939	0.930	0.935					
	40	0.944	0.930	0.929	0.940				
	100	0.947	0.928	0.924	0.930	0.932			
	400	0.948	0.926	0.922	0.926	0.927	0.926		
	2000	0.948	0.926	0.922	0.925	0.928	0.927	0.925	
	20000	0.948	0.926	0.922	0.925	0.929	0.927	0.924	0.923
PPRS^c	∞	0.948	0.926	0.922	0.925	0.929	0.927	0.923	0.923

Table 69: Probability $P(Q) = P(I) + P(S)$ of accepting K quickly, given that K is accepted as a sample from $p_N(k)$, Algorithm HPRS^l (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	*0.999*							
20	10	*0.997*	0.988						
40	20	*0.992*	0.963	0.956					
80	20	0.971	0.953						
	40	*0.988*	0.960	0.956	0.947				
200	50	0.966	0.937	0.947					
	100	*0.986*	0.963	0.953	0.937	0.941			
800	100	0.958	0.934	0.924					
	200	0.963	0.934	0.940	0.940	0.934			
	400	*0.985*	0.964	0.950	0.942	0.938	0.928		
4000	250	0.956	0.934	0.926					
	500	0.958	0.935	0.925	0.929	0.931			
	1000	0.962	0.933	0.925	0.927	0.929	0.927		
	2000	*0.984*	0.964	0.950	0.943	0.937	0.930	0.928	
40000	1250	0.954	0.933	0.927	0.929				
	2500	0.955	0.934	0.927	0.929	0.927			
	5000	0.958	0.935	0.926	0.929	0.934	0.926		
	10000	0.962	0.933	0.925	0.927	0.930	0.928	0.925	
	20000	*0.984*	0.964	0.949	0.943	0.931	0.932	0.926	0.922

Table 70: Probability $P(Q) = P(I) + P(S)$ of accepting K quickly, given that K is accepted as a sample from $p_n(k)$, Algorithms BPRS^l and PPRS^l (Zechner, 1993).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BPRS^l	7	0.938							
	10	0.960	0.964						
	20	0.960	0.933	0.949					
	40	0.957	0.935	0.925	0.943				
	100	0.954	0.934	0.926	0.928	0.931			
	400	0.953	0.933	0.927	0.929	0.925	0.926		
	2000	0.952	0.933	0.927	0.930	0.928	0.927	0.925	
	20000	0.952	0.933	0.927	0.930	0.929	0.926	0.924	0.923
PPRS^l	∞	0.952	0.933	0.927	0.922	0.929	0.927	0.923	0.923

Table 71: Probability $P(Q) = P(I) + P(S)$ of accepting K quickly, given that K is accepted as a sample from $p_N(k)$, Algorithm HPRS^r (Zechner, 1993).

		μ							
N	M	3.5	5.0	10	20	50	200	1000	10000
14	7	0.971							
20	10	0.943	0.988						
40	20	0.948	0.963	0.956					
80	20	0.964	0.961						
	40	0.969	0.960	0.956	0.947				
200	50	0.952	0.943	0.945					
	100	0.979	0.963	0.953	0.937	0.941			
800	100	0.940	0.941	0.942					
	200	0.946	0.944	0.945	0.939	0.934			
	400	0.983	0.964	0.950	0.942	0.938	0.928		
4000	250	0.948	0.938	0.937					
	500	0.943	0.940	0.939	0.937	0.932			
	1000	0.944	0.944	0.943	0.938	0.933	0.928		
	2000	0.984	0.964	0.950	0.943	0.937	0.930	0.928	
40000	1250	0.951	0.936	0.934	0.935				
	2500	0.949	0.937	0.936	0.936	0.929			
	5000	0.944	0.940	0.939	0.937	0.934	0.927		
	10000	0.944	0.944	0.943	0.937	0.934	0.929	0.925	
	20000	0.984	0.964	0.949	0.943	0.931	0.932	0.926	0.922

Table 72: Probability $P(Q) = P(I) + P(S)$ of accepting K quickly, given that K is accepted as a sample from $p_n(k)$, Algorithms BPRS^r and PPRS^r (Zechner, 1993).

	$n \setminus \mu$	3.5	5.0	10	20	50	200	1000	10000
BPRS^r	7	0.945							
	10	0.951	0.964						
	20	0.940	0.944	0.949					
	40	0.947	0.940	0.943	0.943				
	100	0.951	0.936	0.937	0.938	0.931			
	400	0.952	0.934	0.933	0.935	0.928	0.926		
	2000	0.953	0.934	0.932	0.934	0.931	0.927	0.925	
	20000	0.953	0.934	0.932	0.934	0.931	0.927	0.924	0.923
PPRS^r	∞	0.953	0.934	0.932	0.928	0.931	0.928	0.923	0.923

References

[1] Ahrens, J. H. and Dieter, U. (1980): Sampling from binomial and Poisson distributions: a method with bounded computation times. Computing *25*, 193 – 208.

[2] Ahrens, J. H. and Dieter, U. (1982): Computer generation of Poisson deviates from modified normal distributions. ACM Trans. Math. Software *8*, 163 – 179.

[3] Ahrens, J. H. and Dieter, U. (1987, 1991): A convenient sampling method with bounded computation times for Poisson distributions. In: The First Internat. Conf. Statist. Comput., Izmir, Turkey, March 30 – April 2, 1987, pp. 4 – 17. In: The Frontiers of Statist. Comput., Simul. & Modelling (Nelson, P. R., ed.), Amer. Sci. Press, 1991, pp. 137 – 149.

[4] Devroye, L. (1986): Non–Uniform Random Variate Generation. Springer, New York.

[5] Devroye, L. and Naderisamani, A. (1980): A binomial random variate generator. Technical Report, School Computer Science, McGill University, Montreal.

[6] Fishman, G. S. (1973): Concepts and methods in discrete event digital simulation. Wiley, New York.

[7] Hörmann, W. (1992): The transformed rejection method for generating Poisson random variables. Insurance: Math. Econom. *11*, 1 – 7.

[8] Hörmann, W. (1993): The generation of binomial random variates. J. Statist. Comput. Simul. *46*, 101 – 110.

[9] Hörmann, W. and Derflinger, G. (1990): The ACR method for generating normal random variables. OR Spektrum *12*, 181 – 185.

[10] Kachitvichyanukul, V. and Schmeiser, B. W. (1985): Computer generation of hypergeometric random variates. J. Statist. Comput. Simul. *22*, 127 – 145.

[11] Kachitvichyanukul, V. and Schmeiser, B. W. (1988a): Algorithm 668 H2PEC: Sampling from the hypergeometric distribution. ACM Trans. Math. Software *14*, 397 – 398.

[12] Kachitvichyanukul, V. and Schmeiser, B. W. (1988b): Binomial random variate generation. Comm. ACM *31*, 216 – 222.

[13] Kachitvichyanukul, V. and Schmeiser, B. W. (1989): Algorithm 678 BTPEC: Sampling from the binomial distribution. ACM Trans. Math. Software *15*, 394 – 397.

[14] Kemp, A. W. (1988): Simple algorithms for the Poisson modal cumulative probability. Commun. Statist. Simul. *17*, 1495 – 1508.

[15] Kemp, A. W. and Kemp, C. D. (1990): A composition–search algorithm for low–parameter Poisson generation. J. Statist. Comput. Simul. *35*, 239 – 244.

[16] Kemp, C. D. (1982): Low–storage Poisson generators for microcomputers. In: COMPSTAT 1982, Part II (Caussinus, H., et al., eds.), Physica, Vienna, Austria, pp. 145 – 146.

[17] Kemp, C. D. (1986): A modal method for generating binomial variables. Commun. Statist. A – Theory Methods *15*, 805 – 813.

[18] Kemp, C. D. (1990): New algorithms for generating Poisson variates. J. Comput. Applied Math. *31*, 133 – 137.

[19] Kemp, C. D. and Kemp, A. W. (1991): Poisson random variate generation. Applied Statistics *40*, 143 – 158.

[20] Minh, Do Le: Generating gamma variates. ACM Trans. on Math. Software *14*, 261 – 266 (1988).

[21] von Neumann, J. (1951): Various techniques used in connection with random digits. In: The Monte Carlo Method (Householder, A. S., et al., eds.), Nat. Bur. Standards Appl. Math. Ser. *12*, pp. 36 – 38.

[22] Schmeiser, B. W. and Kachitvichyanukul, V. (1982): Poisson random variate generation. Research Memorandum *81–4*, School of Industrial Engineering, Purdue University, West Lafayette, Indiana.

[23] Stadlober, E. (1989a): Sampling from Poisson, binomial and hypergeometric distributions: ratio of uniforms as a simple and fast alternative. Math. Statist. Sektion *303*, 93 pp. Forschungsgesellschaft Joanneum, Graz, Austria.

[24] Stadlober, E. (1989b): A note on Atkinson's simple Poisson generator PA. Preprint *1229*, Techn. Hochschule Darmstadt, West Germany.

[25] Stadlober, E. (1990): The ratio of uniforms approach for generating discrete random variables. J. Comput. Applied Math. *31*, 181 – 189.

[26] Thurner, J. (1990): Generation of random variates from Poisson and binomial distributions. Diploma thesis, 86 pp. Technische Universität Graz, Austria (in German).