Stat Comput
DOI 10.1007/s11222-012-9334-8

Efficient time/space algorithm to compute rectangular probabilities of multinomial, multivariate hypergeometric and multivariate Pólya distributions

R. Lebrun

Received: 22 October 2011 / Accepted: 9 May 2012

Abstract The computation of rectangular probabilities of multivariate discrete integer distributions such as the multinomial, multivariate hypergeometric or multivariate Pólya distributions is of great interest both for statistical applications and for probabilistic modeling purpose. All these distributions are members of a broader family of multivariate discrete integer distributions for which computationaly efficient approximate methods have been proposed for the evaluation of such probabilities, but with no control over their accuracy. Recently, exact algorithms have been proposed for computing such probabilities, but they are either dedicated to a specific distribution or to very specific rectangular probabilities. We propose a new algorithm that allows to perform the computation of arbitrary rectangular probabilities in the most general case. Its accuracy matches or even outperforms the accuracy exact algorithms when the rounding errors are taken into account. In the worst case, its computational cost is the same as the most efficient exact method published so far, and is much lower in many situations of interest. It does not need any additional storage than the one for the parameters of the distribution, which allows to deal with large dimension/large counting parameter applications at no extra memory cost and with an acceptable computation time, which is a major difference with respect to the methods published so far.

Keywords Rectangular probabilities · Cumulative distribution function · Multinomial · Multivariate hypergeometric · Multivariate Pólya · Poisson summation formula

R. Lebrun (✉)
Department of Applied Mathematics and Modeling, EADS Innovation Works, 12, rue Pasteur BP76, 92152 Suresnes Cedex, France
e-mail: regis.lebrun@eads.net

Published online: 06 June 2012

1 Introduction

We are interested in the computation of rectangular probabilities for a d dimensional discrete integer-valued random vector $\mathbf{X} \sim \mathcal{D}$:

$$\forall \mathbf{a}, \mathbf{b} \in \mathbb{N}^d,$$

$$p_{\mathcal{D}}(\mathbf{a}, \mathbf{b}) = \mathbb{P}(\mathbf{a} \leq \mathbf{X} \leq \mathbf{b}) = \mathbb{P}(a_1 \leq X_1 \leq b_1, \ldots, a_d \leq X_d \leq b_d) \quad (1)$$

The computation of such quantities are of uttermost interest in many statistical applications for $\mathbf{X}$ distributed according to a multinomial, multivariate hypergeometric or multivariate Pólya distribution (see Butler and Stutton 1998; Corrado 2011; Frey 2009; Good 1957; Johnson 1960; Levin 1981, 1983, 1992), but despite the existing literature on the subject, there is no function in the standard numerical softwares such as R, SAS, Matlab, Scilab or Octave to evaluate these probabilities.

Several authors (see Butler and Stutton 1998; Childs and Balakrishnan 2000; Good 1957; Levin 1981, 1983, 1992, Mallows 1968) have described in details approximate algorithms for a long time, but these algorithms provide only a limited precision and with no control on the error, which may be inadequate for some applications. Some of these authors (see Good 1957; Levin 1981, 1983, 1992) have also noticed that it could be possible to derive an exact algorithm if one was to compute exactly a given convolution, but they gave no indication on how to do it efficiently and accurately.

It is only recently that reasonably efficient algorithms have been described (see Corrado 2011; Frey 2009), using completely different roots than the previous authors. But even with these algorithms, the only distribution for which it is possible to compute arbitrary rectangular probabilities is the multinomial one, with a polynomial space and time complexity.

We propose to change this situation by providing an algorithm which is essentially exact up to machine precision for all the multivariate discrete distributions considered in Butler and Stutton (1998) and Levin (1983), amongst which the multinomial, multivariate hypergeometric and multivariate Pólya distributions. In the multinomial case, our algorithm is more efficient than the algorithm described in Frey (2009), both with respect to space and time complexity, for an equivalent or even better accuracy when implemented in double precision.

More precisely, we are interested in d-dimensional discrete distributions $\mathcal{D}$ with a $d-1$ dimensional probability function. We suppose that there exists a random vector $\mathbf{Y}_t = (Y_{t1}, \ldots, Y_{td})$ with independent components such that $\mathbf{X} \sim \mathcal{D}$ has the same distribution as $\mathbf{Y}_t \mid \sum_{j=1}^{d} Y_{tj} = N$, where $t > 0$ is a scaling parameter for the mean of $\mathbf{Y}_t$.

With these hypotheses, the rectangular probability (1) admits the following representation by a direct application of Bayes' theorem:

$$p_{\mathcal{D}}(\mathbf{a}, \mathbf{b}) = \mathbb{P}(T_t = N) \frac{\prod_{j=1}^{d} \mathbb{P}(a_j \leq Y_{tj} \leq b_j)}{\mathbb{P}(Y_t = N)} \tag{2}$$

where

$$T_{tj} = (Y_{tj} | a_j \leq Y_{tj} \leq b_j),$$
$$T_t = \sum_{j=1}^{d} T_{tj} \quad \text{and} \quad Y_t = \sum_{j=1}^{d} Y_{tj} \tag{3}$$

We also suppose that all the variables Y_{tj} are members of a parametric family of distributions $\mathcal{L}(\theta)$ for which the distribution of Y_t is known analytically. It is the case if $\mathcal{L}(\theta)$ is closed under convolution, i.e. $\forall j \in \{1, \ldots, d\}$, $Y_{tj} \sim \mathcal{L}(\theta_j)$ and $Y_t \sim \mathcal{L}(\theta)$.

This set of hypotheses covers the multinomial, multidimensional hypergeometric and multidimensional Pólya distributions.

We recall some definitions concerning these distributions and make explicit the associated family $\mathcal{L}(\theta)$. We note $\mathcal{S}$ the set $\{\mathbf{a} \in \mathbb{N}^d \mid \sum_{j=1}^{d} a_j = N\}$, and we have:

Definition 1 The multinomial distribution $\mathcal{M}_d(N, \mathbf{p})$ is defined by:

$$\forall \mathbf{x} \in \mathbb{N}^d,$$
$$\mathbb{P}(\mathbf{X} = \mathbf{x}) = \frac{N!}{\prod_{j=1}^{d} x_j!} \left(\prod_{j=1}^{d} p_j^{x_j} \right) \mathbf{1}_{\mathcal{S}}(\mathbf{x}) \tag{4}$$

where $\forall j \in \{1, \ldots, d\}$, $p_j \geq 0$ and $\sum_{j=1}^{d} p_j = 1$.

The decomposition (2) is obtained with

$$\mathcal{L}(\theta_j) = \mathcal{P}(t p_j)$$

where $\mathcal{P}(t p_j)$ is the Poisson distribution with mean $t p_j$ for any $t > 0$, and $Y_t \sim \mathcal{P}(t)$.

Definition 2 The multivariate hypergeometric distribution $\mathcal{H}_d(N, \mathbf{h})$ is defined by:

$$\forall \mathbf{x} \in \mathbb{N}^d,$$
$$\mathbb{P}(\mathbf{X} = \mathbf{x}) = \left(\prod_{j=1}^{d} \binom{h_j}{x_j} \Big/ \binom{h}{N} \right) \mathbf{1}_{\mathcal{S}'}(\mathbf{x}) \tag{5}$$

where $\forall j \in \{1, \ldots, d\}$, $h_j \in \mathbb{N}$, $\mathcal{S}' = \mathcal{S} \cap \{0, \ldots, h_1\} \times \cdots \times \{0, \ldots, h_d\}$ and $h = \sum_{j=1}^{d} h_j$.

The decomposition (2) is obtained with

$$\mathcal{L}(\theta_j) = \mathcal{B}(h_j, t)$$

the binomial distribution with mean $t h_j$ for any $t \in (0, 1)$, and $Y_t \sim \mathcal{B}(h, t)$.

Definition 3 The multivariate Pólya distribution $\mathcal{P}_d(N, \mathbf{q})$ is defined by:

$$\forall \mathbf{x} \in \mathbb{N}^d,$$
$$\mathbb{P}(\mathbf{X} = \mathbf{x}) = \frac{\Gamma(q) N!}{\Gamma(N+q)} \left(\prod_{j=1}^{d} \frac{\Gamma(x_j + q_j)}{\Gamma(q_j) x_j!} \right) \mathbf{1}_{\mathcal{S}}(\mathbf{x}) \tag{6}$$

where $\forall j \in \{1, \ldots, d\}$, $q_j > 0$ and $q = \sum_{j=1}^{d} q_j$.

The decomposition (2) is obtained with

$$\mathcal{L}(\theta_j) = \mathcal{NB}(q_j, t)$$

the negative binomial distribution with mean $q_j(1-t)/t$ for any $t \in (0, 1)$, and $Y_t \sim \mathcal{NB}(q, t)$.

For all the distributions $\mathcal{L}(\theta)$ we are interested in, there exists efficient and accurate routines to evaluate both $\mathbb{P}(a_j \leq Y_{tj} \leq b_j)$ and $\mathbb{P}(Y_t = N)$. The only difficulty is the evaluation of $\mathbb{P}(T_t = N)$, as noticed in Good (1957), Levin (1981, 1983) or Levin (1992), but they gave no clue on how to do it both efficiently and accurately. Instead, they developed several approximations of this quantity using either Edgeworth expansions or saddle-point approximations.

We list the available algorithms, the distributions they address and the possible restrictions on $\mathbf{a}$ and $\mathbf{b}$ in the computation of (1) in Table 1 for the exact algorithms and in Table 2 for the approximate ones.

Table 1 Exact algorithms applicability

Reference	Berry and Mielke (1995)	Corrado (2011)	Frey (2009)
Distributions	$\mathcal{M}_d$	$\mathcal{M}_d, \mathcal{H}_d$	$\mathcal{M}_d$
Restrictions on **a**, **b**	$\mathbf{a} = \mathbf{0}$	$\mathbf{a} = (a, \ldots, a)$ $\mathbf{b} = (b, \ldots, b)$	*none*

Table 2 Approximate algorithms applicability

Reference	Butler and Stutton (1998), Levin (1983, 1992)	Childs and Balakrishnan (2000)	Good (1957)	Levin (1981)
Distributions	$\mathcal{M}_d, \mathcal{H}_d, \mathcal{P}_d$	$\mathcal{H}_d$	$\mathcal{M}_d$	$\mathcal{M}_d$
Restrictions on **a**, **b**	*none*	*none*	$\mathbf{a} = (a, \ldots, a)$ $\mathbf{b} = (b, \ldots, b)$	$\mathbf{a} = \mathbf{0}$

Is the accurate (or even exact) evaluation $\mathbb{P}(T_t = N)$ intractable? A straightforward approach is to compute the associated convolution by multiplying the generating functions of the Y_{tj} random variables and by extracting the coefficient of degree N. All these generating functions are polynomials of degree b_j, of which only the coefficients of order not greater than N are of interest. It leads to an $\mathcal{O}(dN^2)$ time complexity if these multiplications are done using the naive polynomial multiplication algorithm, or to an $\mathcal{O}(dN \log N)$ time complexity if these multiplications are done using an FFT based algorithm. While it is clearly a much better algorithm than the brute force enumeration method, it remains costly for large values of N and d.

Using an appropriate numerical method, it is possible to evaluate the order N coefficient efficiently in both space and time, not exactly but with a user-controlled accuracy that can be made as small as the machine precision. In some sense, the resulting algorithm is essentially exact.

To contrast the performance of our algorithm with respect to the exact ones, let us introduce the following notation:

$$\begin{aligned} \sigma_a &= \sum_{j=1}^{d} a_j, \qquad \sigma_b = \sum_{j=1}^{d} b_j, \\ \sigma_{ab} &= \sigma_b - \sigma_a, \qquad N_a = N - \sigma_a \end{aligned} \tag{7}$$

In the case of the multinomial distribution, the most efficient exact algorithm proposed so far for the evaluation of (1) is the one described in Frey (2009) and we will take it as a reference. Its space complexity is $\mathcal{O}(\sigma_{ab})$ and its time complexity is $\mathcal{O}(N_a \sigma_{ab})$. The algorithm described in Corrado (2011) has the same space and time complexity, is applicable to more distributions but covers only restricted arguments of (1).

The key results are that our algorithm has a constant (and small) $\mathcal{O}(1)$ space complexity, and has a worst case $\mathcal{O}(N_a \sigma_{ab})$ time complexity that drops to $\mathcal{O}(d\sqrt{N_a})$ for most situations, which is a tremendous improvement for large scale problems. Its accuracy is on par or even better than the accuracy of the reference algorithm.

The first section of the article presents the foundations of the algorithm and the second section details some specific results that make the algorithm efficient, with a particular emphasize on the multinomial case. The last section gives experimental evidences of both the time complexity and the accuracy of the algorithm. Several test cases gathered in the literature are also detailed.

2 Foundations of the algorithm

In this section, our key result is the representation of the rectangular probability given in Proposition 4, which is the basis of our algorithm.

Here is the key result given in (Abate and Whitt 1992, Eqs. 5.35–5.37) and allowing for a fast and accurate evaluation of convolutions for discrete univariate distributions, using Poisson's summation formula:

Theorem 1 *Let f_X be the probability function of a discrete random variable X and ϕ_X its associated probability generating function $\phi_X(z) = \sum_{k \geq 0} f_X(k) z^k$.*

Then, for any non-negative integers n, $m > n$ and real number $0 < r < 1$:

$$f_X(n) = \mathbb{P}(X = n) = \frac{1}{m r^n} \sum_{k=0}^{m-1} \xi_m^{-kn} \phi_X\left(r \xi_m^k\right) - \epsilon_{n,m,r} \tag{8}$$

where $\epsilon_{n,m,r} = \sum_{k \geq 1} f_X(n + km) r^{km} \leq \frac{r^m}{1 - r^m} \simeq r^m$ and $\xi_m = e^{\frac{2i\pi}{m}}$.

This theorem provides a numerical method to compute the probability function of a discrete distribution from its generating function: the value of $f_X(n)$ is approximated by the finite sum that appears in (8), with a positive error (i.e. $f_X(n)$ is *over-estimated*) that can be made as small as

needed by a judicious choice of r and m. We note that if X has a bounded support with upper bound M, which is the case in the application we have in mind, any choice of m such that $m > M$ leads to an *exact* algorithm as $\epsilon_{n,m,r} = 0$ for such a choice.

Using $m = 2n$ in (8) gives two advantages, namely the terms of the sum can be paired in order to add to real values so the resulting formula has no more than $n + 1$ terms, and the factor ξ_m^{-kn} reduces to $(-1)^k$. The resulting formula is given in (Abate and Whitt 1992, Eqs. 5.38–5.39), and reads:

Proposition 1 *If we take $m = 2n$ in (8), we get*:

$$f_X(n) = \frac{1}{2nr^n}\sum_{k=0}^{n-1}(-1)^k\Re\big(\phi_X\big(r\zeta_n^k\big) - \phi_X\big(r\zeta_n^{k+1}\big)\big) - \epsilon_{n,r} \tag{9}$$

where

$$\epsilon_{n,r} = \sum_{k\geq 1} f_X\big((2k+1)n\big)r^{2kn} \leq \frac{r^{2n}}{1-r^{2n}} \simeq r^{2n} \tag{10}$$

where $\zeta_n = \xi_{2n} = e^{\frac{i\pi}{n}}$.

We will apply (8) if we want an exact algorithm, or (9) if we want an approximate algorithm, to evaluate $\mathbb{P}(T_t = N)$, and plug the resulting formula into (2) in order to derive our algorithm. In the approximate case, we see that the value of the error (10) can be made smaller than a given ϵ_{max} by choosing r such that $r^{2n} \leq \epsilon_{max}$:

$$r \leq \epsilon_{max}^{\frac{1}{2n}} \tag{11}$$

It remains to express the generating probability function of T_t, which is an elementary result stated without proof:

Proposition 2 *$\forall j \in \{1,\ldots,d\}$, $\forall z \in \mathbb{C}$ with $|z| \leq 1$ we have*:

$$\phi_{T_{tj}}(z) = \frac{\pi^{(j)}_{a_jb_j}(z)}{\mathbb{P}(a_j \leq Y_{tj} \leq b_j)} \tag{12}$$

with

$$\pi^{(j)}_{a_jb_j}(z) = \sum_{k=a_j}^{b_j}\mathbb{P}(Y_{tj}=k)z^k = \pi^{(j)}_{b_j}(z) - \pi^{(j)}_{a_j-1}(z) \tag{13}$$

where

$$\pi^{(j)}_{-1}(z) \equiv 0,\ \forall n \in \mathbb{N}, \quad \pi^{(j)}_n(z) = \sum_{k=0}^{n}\mathbb{P}(Y_{tj}=k)z^k \tag{14}$$

The independence of the T_{tj} leads to:

$$\phi_{T_t}(z) = \frac{\prod_{j=1}^d \pi^{(j)}_{a_jb_j}(z)}{\prod_{j=1}^d \mathbb{P}(a_j \leq Y_{tj} \leq b_j)} \tag{15}$$

We see that a key factor in the cost of (8) is the constraint $m > n$. When we are interested in computing the value of a multivariate discrete cumulative distribution function, there is no choice but to take $n = N$ and $m > N$ in (8). But when we are interested in computing a rectangular probability, i.e. when $\mathbf{a} \neq \mathbf{0}$, we can express $\mathbb{P}(T_t = N)$ in a form that leads to a less expensive summation. The elementary properties of the characteristic functions lead to the following proposition:

Proposition 3 *If $\mathbf{a} \neq \mathbf{0}$, $\forall j \in \{1,\ldots,d\}$ we set $V_{tj} = T_{tj} - a_j$. The random variables V_{tj} are such that*:

$$\mathbb{P}(V_{tj} = k) = \mathbb{P}(T_{tj} = a_j + k) \tag{16}$$

$$\phi_{V_{tj}}(z) = z^{-a_j}\phi_{T_{tj}}(z) \tag{17}$$

and

$$\mathbb{P}(T_t = N) = \mathbb{P}(V_t = N_a) \tag{18}$$

$$\phi_{V_t}(z) = z^{-\sigma_a}\phi_{T_t}(z) \tag{19}$$

where $V_t = \sum_{j=1}^d V_{tj}$ has support $\{0,\ldots,\sigma_{ab}\}$.

Replacing the evaluation of $\mathbb{P}(T_t = N)$ by the evaluation of $\mathbb{P}(V_t = N_a)$ moves the constraint $m > N$ into $m > N_a$ with $N_a < N$. Furthermore, the algorithm is now exact as soon as $m > \sigma_{ab}$.

Considering only the approximate version of the algorithm, we get:

Proposition 4 *$\forall \mathbf{a}, \mathbf{b} \in \mathbb{N}^d$, we have*:

$$p_{\mathcal{D}}(\mathbf{a},\mathbf{b}) = \Re\Bigg\{\sum_{k=0}^{N-1}\big(-\zeta_{N_a}^{-\sigma_a}\big)^k\Bigg(\prod_{j=1}^d \pi^{(j)}_{a_jb_j}\big(r\zeta_{N_a}^k\big) - \zeta_{N_a}^{-\sigma_a}\prod_{j=1}^d \pi^{(j)}_{a_jb_j}\big(r\zeta_{N_a}^{k+1}\big)\Bigg)\Bigg\} \Big/\big(2N_a r^N \mathbb{P}(Y_t = N)\big) - \eta_{N_a,r} \tag{20}$$

where

$$K = \frac{\prod_{j=1}^d \mathbb{P}(a_j \leq Y_{tj} \leq b_j)}{\mathbb{P}(Y_t = N)}$$

and $\eta_{N_a,r} = K\epsilon_{N_a,r}$.

Except for the storage of the data **a**, **b** and **p**, which is a $\mathcal{O}(d)$, the memory complexity of this algorithm is $\mathcal{O}(1)$ as no intermediate structure is needed in the evaluation of (20). The time complexity is of order $\mathcal{O}(N_a C)$, where C is the time complexity of evaluating $\pi^{(j)}_{a_1 b_1}, \ldots, \pi^{(j)}_{a_d b_d}$ at a given point. A naive evaluation of these polynomials leads to $C \simeq \sigma_{ab}$ and a total time complexity of $\mathcal{O}(N_a \sigma_{ab})$, which is the same complexity as the algorithm proposed in Frey (2009). We also note that the factor $\prod_{j=1}^{d} \mathbb{P}(a_j \leq Y_{tj} \leq b_j)$ in (1) simplifies with the denominator of (15), reducing the overall computational cost.

One can see that the error in (20) depends on t through the numerator of K, and on r through $\epsilon_{N_a, r}$. The theoretical behavior of this error is clear: we can take r small enough to get the absolute error we want. Its numerical behavior is less clear, as the summation in (20) can be subject to cancellation. The best way to take into account these cancellations is to use the recommendations in Abate and Whitt (1992) to choose r using (11), then to choose t in order to minimize K. This point will be explored numerically in the case of the multinomial distribution.

3 Making the algorithm more efficient

In this section, our key results are the efficient evaluation of the characteristic function of T_t, as a result of Proposition 5, and the original stopping criterion given in Proposition 6. Combined, these results lead to Algorithms 1 and 2, which are our core contribution.

Two remarks can lead to a dramatic improvement of the time complexity (or complexity for short) of the proposed algorithm. The first one is that in many situations, the evaluation of $\pi^{(j)}_{a_j b_j}$ can be done with $\mathcal{O}(1)$ operations instead of $\mathcal{O}(b_j - a_j)$ within machine precision when $b_j - a_j \gg 1$, counting the evaluation of a transcendental function such as the exponential as a $\mathcal{O}(1)$ time complexity operation. In this case, $C = \mathcal{O}(d)$ instead of $C = \mathcal{O}(\sigma_{ab})$, and the total complexity drops to $\mathcal{O}(N_a d)$. The second one is that the terms involved in (20) are usually of very different magnitudes, and most of them do not contribute significantly (up to machine precision) to the final result. In most cases, only $\mathcal{O}(\sqrt{N_a})$ terms are needed. The overall complexity is thus reduced to $\mathcal{O}(d\sqrt{N_a})$.

3.1 Efficient evaluation of $\pi^{(j)}_{a_j, b_j}(z)$

If $b_j - a_j = \mathcal{O}(1)$, the evaluation of $\pi^{(j)}_{a_j b_j}$ is obviously a $\mathcal{O}(1)$, so we restrict our attention to the case $b_j - a_j \gg 1$. It covers two different sub-cases: either we have $a_j = \mathcal{O}(1)$, for example in the case where one is interested in the computation of the cumulative distribution function of the distribution, or we have $a_j, b_j \gg 1$. In the first case, the following proposition gives elements to make the evaluation of $\pi^{(j)}_{a_j b_j}$ less expensive than $\mathcal{O}(b_j - a_j)$:

Proposition 5 *Let n be a nonnegative integer and z a complex number such that $|z| \leq 1$. Let $\bar{s} = \sup\{s \geq 0, \phi_{Y_{tj}}(e^s) < +\infty\}$. If $\bar{s} > 0$, then*

$$\left|\phi_{Y_{tj}}(z) - \pi^{(j)}_n(z)\right| \leq \frac{\phi_{Y_{tj}}(e^{s^*})}{e^{(n+1)s^*}} \tag{21}$$

where $s^* = \operatorname{argmin}_{0<s<\bar{s}} \frac{\phi_{Y_{tj}}(e^s)}{e^{(n+1)s}}$.

Proof By definition of $\phi_{Y_{tj}}$ and $\pi^{(j)}_n(z)$, we have:

$$\begin{aligned}\left|\phi_{Y_{tj}}(z) - \pi^{(j)}_n(z)\right| &= \left|\sum_{k>n} \mathbb{P}(Y_{tj} = k) z^k\right| \\ &\leq \sum_{k>n} \mathbb{P}(Y_{tj} = k)|z|^k \\ &\leq F^c_{Y_{tj}}(n) \quad \text{as } |z| \leq 1\end{aligned}$$

where $F^c_{Y_{tj}}(n) = \mathbb{P}(Y_{tj} > n)$ is the complementary cumulative distribution function of Y_{tj} evaluated at n.

Then, applying Markov's inequality to $e^{sY_{tj}}$ and minimizing the bound with respect to s such that $0 < s < \bar{s}$, we get (21). □

The hypothesis made on $\phi_{Y_{tj}}$ is fulfilled for the binomial, negative binomial and Poisson distributions, for which the respective values of s^* are $\log(\frac{1-t}{t}\frac{n+1}{h_j-(n+1)})$, $\log(\frac{n+1}{(1-t)(q_j+n+1)})$ and $\log(\frac{n+1}{tp_j})$. In the general case, the convergence of $\pi^{(j)}_n(z)$ to $\phi_{Y_{tj}}(z)$ is at least exponential with n, and can be even faster (e.g. in the Poisson case). It results the following algorithm to evaluate $\pi^{(j)}_n(z)$:

Algorithm 1

Given $n \in \mathbb{N}$, $z \in \mathbb{C}$, $|z| \leq 1$, do:

1. Set $k := n + 1$
2. Set $v_k := \phi_{Y_{tj}}(z)$
3. Set $dv_k = \mathbb{P}(Y_{tj} = k) z^k$
4. While $|dv_k| > |v_k| \epsilon_{machine}$ do
   (a) $v_{k+1} := v_k - dv_k$
   (b) $dv_{k+1} := \mathbb{P}(Y_{tj} = k+1) z^{k+1} = f(dv_k, k, z)$
   (c) $k := k + 1$
5. Return v_k

This algorithm performs $\mathcal{O}(|\log \epsilon_{machine}|)$ iteration. The evaluation of $\phi_{Y_{tj}}(z)$ and $\mathbb{P}(Y_{tj} = k)$ can be done in $\mathcal{O}(1)$ time complexity for the common distributions, and the update (4) can be made for usual distributions using a simple recursion $f(dv_k, k, z)$ instead of the full evaluation of $\mathbb{P}(Y_{tj} = k+1) z^{k+1}$.

Considering the case of the multinomial distribution, i.e. $Y_{tj} \sim \mathcal{P}(tp_j)$, the situation of (5) is likely to occur when $\sigma_b = \mathcal{O}(dN)$ and $\sigma_a = \mathcal{O}(d)$, i.e. when $b_j = \mathcal{O}(N)$ and $a_j = \mathcal{O}(1)$, which corresponds to the worst complexity we get using the naive evaluation of all the $\pi^{(j)}_{a_j b_j}$. In this case, the number of iterations of (1) is less than 18 for $\epsilon_{machine} = 10^{-16}$ and $tp_j = 1$. The update is given by $f(dv_k, k, z) = dv_k \times \frac{tp_j z}{k+1}$.

When both a_j and b_j are large, the situation is more involved. A naive evaluation using $\pi^{(j)}_{a_j,b_j} = \pi^{(j)}_{b_j} - \pi^{(j)}_{a_j-1}$ can suffer from massive cancellation, providing a very inaccurate result. Nevertheless, in the multinomial case, a systematic $\mathcal{O}(1)$ time complexity can be achieved for the evaluation of $\pi^{(j)}_{a_j,b_j}(z)$, in connection with the evaluation of the regularized incomplete gamma function, see Didonato and Morris (1986), Temme (1994) and the boost library (www.boost.org) for an efficient implementation of these methods.

3.2 Fast (essentially) exact evaluation of Poisson's summation

The terms involved in (20) can have very different magnitudes. As a result, only a few of them might have a significant contribution to Poisson's summation formula, and taking advantage of it could reduce very significantly the cost in the evaluation of the sum. We illustrate it in the computation of the multinomial cumulative distribution function. In this case, $N_a = N$, $V_{tj} = T_{tj}$ and $V_t = T_t$. We restrict our analysis to the case where $\phi_{T_t} \simeq \phi_{Y_t} = e^{-t(1-z)}$:

Proposition 6 *In the case where $N \gg 1$ and $\phi_{T_t}(z) \simeq e^{-t(1-z)}$, either $t = \mathcal{O}(1)$ and no term of the sum in (20) is negligible, or $t \to +\infty$ with $t = \mathcal{O}(N)$ and only the N^* first terms of (20) have a relative contribution to $p_{\mathcal{D}}(\mathbf{a}, \mathbf{b})$ greater than $\epsilon \ll 1$, with:*

$$N^* \simeq \frac{1}{\pi}\sqrt{-\frac{2\log\epsilon}{r}\frac{N^2}{t}} \tag{22}$$

The case $t \gg N$ is not relevant, as it leads to severe cancellations in (20).

Proof In order to study the magnitude of the terms occurring in (9), we use the elementary relation:

$$\left|e^{\beta} - e^{\alpha}\right|^2 = e^{2\Re(\beta)}\rho_{\alpha,\beta} \tag{23}$$

with

$$\rho_{\alpha,\beta} = 1 + e^{2\Re(\alpha-\beta)} - 2\cos\big(\Im(\alpha-\beta)\big)e^{\Re(\alpha-\beta)} \tag{24}$$

using $\beta = -t(1 - r\zeta_N^k)$ and $\alpha = -t(1 - r\zeta_N^{k+1})$.

Let ρ_k be the value of $\rho_{\alpha,\beta}$ for this choice of α and β, and $\delta_k = \phi_{T_t}(r\zeta_N^k) - \phi_{T_t}(r\zeta_N^{k+1})$. Three cases have to be considered: $t = \mathcal{O}(1)$, $t = o(N)$ with $t \to +\infty$ and $t = \Theta(N)$.[1] For the first and second cases, using $\zeta_N - 1 = \frac{i\pi}{N} + \mathcal{O}(\frac{1}{N})$ we get:

$$\frac{|\delta_k|}{|\delta_0|} = e^{-rt(1-\cos(\theta_k))} + \mathcal{O}\left(\frac{t}{N}\right) \tag{25}$$

We have $|\delta_k|/|\delta_0| < \epsilon$ as soon as $\cos(\theta_k) \le 1 + \frac{\log\epsilon}{rt}$. For typical values of ϵ, when $t = \mathcal{O}(1)$, it is not possible to fulfill this constraint so one must compute the N terms in (9). In the second case, using the expansion $\arccos(1-x) = \sqrt{2x} + \mathcal{O}(x^{3/2})$ we get the value of N^* given in (22).

When $t = \Theta(N)$, the computation is more involved as the terms in t/N are no more negligible. We proceed in two steps: first we show that $\frac{|\delta_k|}{|\delta_0|}$ is small as soon as k is greater than a bound which is an $o(N)$, justifying that one can use series expansions with respect to $\theta_k = \frac{k\pi}{N} = o(1)$, then one gets (22) by computations similar to the previous cases. More precisely, defining $\gamma = \frac{t}{N}$ and assuming that $\cos(\gamma r\pi) < 1$, we get:

$$\begin{aligned}\frac{|\delta_k|}{|\delta_0|} &= e^{-rt(1-\cos(\theta_k))} \\ &\quad\times\sqrt{\frac{1-2\cos(\gamma r\pi\cos(\theta_k))e^{-\gamma r\pi\sin(\theta_k)}+e^{-2\gamma r\pi\sin(\theta_k)}}{2(1-\cos(\gamma r\pi))}} \\ &\quad+\mathcal{O}\left(\frac{1}{N}\right)\end{aligned} \tag{26}$$

from which we deduce that:

$$\begin{aligned}\frac{|\delta_k|}{|\delta_0|} &\le e^{-rt(1-\cos(\theta_k))}\frac{1+e^{-\gamma r\pi\sin(\theta_k)}}{\sqrt{2(1-\cos(\gamma r\pi))}} \\ &\le \frac{e^{-rt(1-\cos(\theta_k))}}{\sqrt{1-\cos(\gamma r\pi)}}\end{aligned} \tag{27}$$

as $\theta_k \in [0, \pi]$. The same computation as in the case $t = o(N)$ shows that the upper bound of (27) is smaller than ϵ as soon as $k \le \frac{1}{\pi}\sqrt{N\frac{2\log\epsilon\sqrt{1-\cos(\gamma r\pi)}}{\gamma r}} = o(N)$, i.e. $\theta_k = o(1)$. It is thus possible to expand the square-root term of (26) with respect to θ_k, and one gets (22) the same way as for the previous case. □

If we choose $t = N$ as suggested in Levin (1981), we get $N^* = \mathcal{O}(\sqrt{N})$. The resulting algorithm reads:

[1] The notation $t = \Theta(N)$ means that t is bounded above and below by a linear function of N, while $t = \mathcal{O}(N)$ means that t is only bounded above by a linear function of N. Here, it is important to make this distinction as the argument in the proof is not the same if t is a $\Theta(N)$ or a $\mathcal{O}(N)$ without being a $\Theta(N)$.

Algorithm 2

Given $N \in \mathbb{N}^*$, $\mathbf{p} \in [0,1]^d$, $\mathbf{a}, \mathbf{b} \in \mathbb{N}^d$ such that $\forall j \in \{1,\dots,d\}$, $0 \le a_j \le b_j \le N-1$, $\epsilon_{max} > 0$, do:

1. Compute $r := \epsilon_{max}^{\frac{1}{2N_a}}$
2. Compute $\delta_0 := \prod_{j=1}^d \pi^{(j)}_{a_j b_j}(r) - \zeta_{N_a}^{-\sigma_a} \prod_{j=1}^d \pi^{(j)}_{a_j b_j}(r\zeta_{N_a})$
3. Set $v := \delta_0$, $k := 1$
4. Repeat
   (a) Compute $\delta_k := (-\zeta_{N_a}^{-\sigma_a})^k (\prod_{j=1}^d \pi^{(j)}_{a_j b_j}(r\zeta_{N_a}^k) - \zeta_{N_a}^{-\sigma_a} \prod_{j=1}^d \pi^{(j)}_{a_j b_j}(r\zeta_{N_a}^{k+1}))$
   (b) Compute $v := v + \delta_k$
   (c) Set $k := k+1$
5. Until $k = N$ or $|\delta_k| < \epsilon_{machine}|\delta_0|$
6. Return $\Re(v)/(2N_a r^N \mathbb{P}(Y_t = N))$

The evaluation of $\pi^{(j)}_{b_j a_j - 1}$ is done using (1) or one of the more involved $\mathcal{O}(1)$ methods. We note that the definitions of δ_0 and δ_k are not the same as in Proposition 6, but using (15) we see that the ratio $|\delta_k|/|\delta_0|$ remains the same.

It must be emphasized that in an actual implementation of this algorithm, one should include tests to detect trivial situations for which an early exit is possible. For example, when one component of $\mathbf{x}$ is larger than N, the problem is reduced to a lower dimensional one, or if $\sigma_b < N$ or $\sigma_a > N$, the probability is zero.

This algorithm is available in the Open TURNS software (www.openturns.org), an Open Source C++ library dedicated to probabilistic modeling and uncertainty propagation.

4 Numerical experiments

The objectives of these numerical experiments are to assess the accuracy of the proposed algorithm on various examples used in the literature, and to check its time complexity. All the computations have been made using Levin's recommendation for t, namely $t = N$. There is certainly more insight to be gained in the study of the influence of t on the numerical accuracy, as suggested in Butler and Stutton (1998) or Levin (1983).

4.1 Accuracy

In this numerical experiment, we check the accuracy of the proposed algorithm on the computation of the multinomial cumulative distribution function in the following settings: $d = N$, $p_1 = \cdots = p_d = 1/d$ for $N \in \{\lfloor 2^{k/2} \rfloor \,|\, k = 2,\dots,20\}$. The cumulative distribution function is computed at the points $(x_1 = \cdots = x_d = k)$ for k such that the resulting probability value is in $[10^{-5}, 1-10^{-5}]$. For each value of N, the maximum relative error is plotted against the size N on a logarithmic scale on Fig. 1, for ϵ_{max} taken in $\{10^{-7}, 10^{-9}, 10^{-11}\}$. The points with a zero maximal error are not plotted.

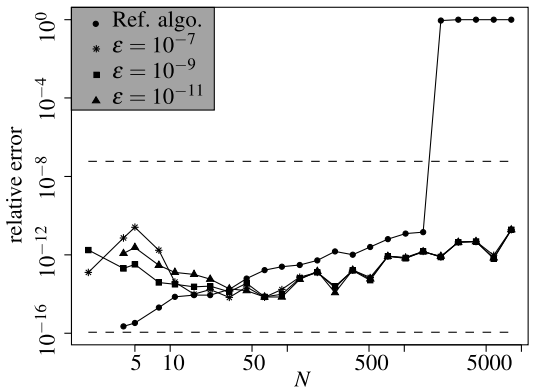

Fig. 1 Maximum relative error for various problem sizes N and various precision parameters ϵ_{max}. For problem sizes larger than 50, the proposed algorithm is consistently more accurate than the reference algorithm, and the achieved accuracy does not depend on the value of ϵ_{max}. The *two horizontal dashed lines* correspond to the single and double precision accuracies

It was not possible to explore larger values for N due to the space complexity of the reference algorithm.

We note several points from this experiment. The first one is that the accuracy of the proposed algorithm is near to the machine precision on a wide range of problem sizes, and even better than the accuracy of the *exact* algorithm as soon as the problem size is larger than a few tens. The second one is that the choice of ϵ_{max} does not seems to have a significant impact on the accuracy of the algorithm as soon as the problem size is larger than a few tens. For smaller sizes, a value of $\epsilon_{max} \simeq 10^{-9}$ seems to give the best overall precision, even if in this case the algorithm (Frey 2009) should probably be preferred. The third one is that the implementation of the algorithm (Frey 2009) as given in the reference paper seems to have numerical stability issues for problems of sizes larger than few thousands.

4.2 Some classical examples

Here are the results of our algorithm on the classical examples that can be found in Berry and Mielke (1995), Corrado (2011), Levin (1981) etc. The algorithm is implemented in Python, using double-precision for the computation. These results have been checked against both a Monte Carlo simulation with 10^9 samples, and the algorithm in Frey (2009) using the reference implementation in R provided by the author as well as a multi-precision implementation in Maple. For each example, we give the 16 digits obtained thanks to our algorithm and we underline the digits that differ from the exact result. We also give the corresponding absolute and relative error.

Example 1 This example[2] is from Berry and Mielke (1995), which consider the classification of $N = 200$ adult subjects into $d = 4$ marital status. This example leads to the following computation:

$\mathbf{X} \sim \mathcal{M}(200, [0.2, 0.35, 0.15, 0.3])$

1. $\mathbb{P}(X_1 \leq 30, X_2 \leq 80, X_3 \leq 40, X_4 \leq 50)$

Example 2 This example is exposed in both Corrado (2011) and Levin (1981), and is attributed to Mallows. It consists of the following computation:

$\mathbf{X} \sim \mathcal{M}(500, [p_1 = \cdots = p_{50} = 1/50])$

1. $\mathbb{P}(X_1 \leq 19, \ldots, X_{50} \leq 19)$

Based on the same multinomial distribution, these two other computations[3] are proposed in Corrado (2011):

2. $\mathbb{P}(4 \leq X_1, \ldots, 4 \leq X_{50})$
3. $\mathbb{P}(4 \leq X_1 \leq 19, \ldots, 4 \leq X_{50} \leq 19)$

Example 3 This example is exposed in Levin (1981), and is attributed to Barton and David. It consists of the following computation:

$\mathbf{X} \sim \mathcal{M}(12, [p_1 = \cdots = p_{12} = 1/12])$

1. $\mathbb{P}(X_1 \leq 2, \ldots, X_{12} \leq 2)$
2. $\mathbb{P}(X_1 \leq 3, \ldots, X_{12} \leq 3)$

The last computation is essentially the same as the one presented in Butler and Stutton (1998), for which the reported absolute error is of order 5×10^{-5}, which illustrates the limited precision of the best available approximate algorithm.

4.3 Time complexity assessment

The time complexity benchmark consists in the evaluation of the cumulative distribution function of the multinomial distribution in different settings for the pair (N, d). The objective is to quantify the asymptotic time complexity of the algorithm with respect to both N and d in the most demanding situations, namely the computation of $\mathbb{P}(X_1 \leq N - 1, \ldots, X_d \leq N - 1)$ for equiprobable X_i.

We will test the configurations $(N, d) \in \{\lfloor 10^{k/5} \rfloor \,|\, k = 10, \ldots, 25\} \times \{10^k \,|\, k = 2, \ldots, 5\}$ for the time complexity with respect to N, and $(N, d) \in \{10^k \,|\, k = 2, \ldots, 5\} \times \{\lfloor 10^{k/5} \rfloor \,|\, k = 10, \ldots, 25\}$ for the time complexity with respect to d.

[2]For which the wrong value of 0.030837 is reported (using a storage of 1373701 floating point numbers for the computation).

[3]For which the wrong values of resp. 0.877373 and 0.750895 are reported.

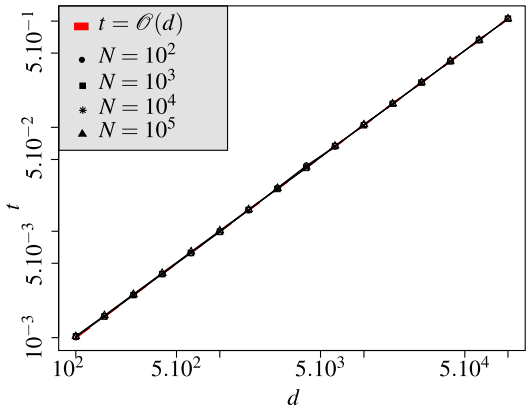

Fig. 2 Evolution of the time complexity with respect to the dimension d in logarithmic scale, for several values of N. The time is normalized such that it is equal to 1 for the largest value of d. There is a perfect agreement between the observed complexity and the theoretical $\mathcal{O}(d)$ complexity, as all the curves are superposed with the *dashed line* $t = d$

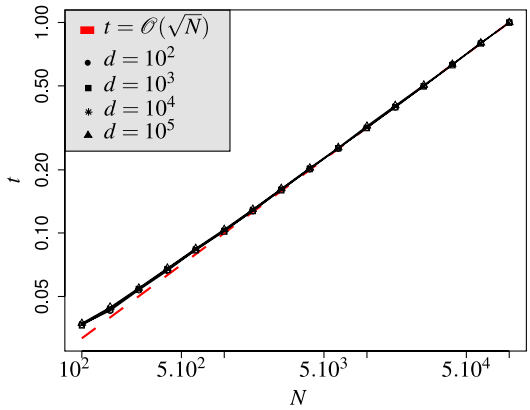

Fig. 3 Evolution of the time complexity with respect to N in logarithmic scale, for several values of the dimension d. The time is normalized such that it is equal to 1 for the largest value of N. There is an excellent agreement between the observed complexity and the theoretical $\mathcal{O}(\sqrt{N})$ complexity as all the curves are almost superposed with the *dashed curve* $t = \sqrt{N}$

The time complexity matches perfectly the theoretical bounds, as it can be seen on Figs. 2 and 3, which confirms that the algorithm is a significant improvement over the previous ones: when $d = N$, we get a time complexity of $\mathcal{O}(N^{3/2})$ instead of $\mathcal{O}(N^3)$ for the reference algorithm.

5 Conclusion

In this article, we provide an algorithm that enables to compute arbitrary rectangular probabilities with a high accuracy for a class of multidimensional discrete probability distributions that includes the multinomial, multivariate hypergeometric and multivariate Pólya distributions. This algorithm can be made exact in exact arithmetic with a constant space complexity and a polynomial time complexity that matches the best available algorithms so far.

Table 3 Probability value and precision of the examples. The exact values, rounded to the 16th significant figure, are also given in bold face

Example	Probability	Absolute error	Relative error
1-1	$0.478450946581829\underline{5} \times 10^{-5}$	1.5×10^{-17}	3.2×10^{-12}
	$\mathbf{0.4784509465802881 \times 10^{-5}}$		
2-1	$0.852726985258\underline{1543}$	1.5×10^{-14}	1.8×10^{-14}
	$\mathbf{0.8527269852581694}$		
2-2	$0.602684281137\underline{5376}$	2.3×10^{-14}	3.9×10^{-14}
	$\mathbf{0.6026842811375610}$		
2-3	$0.520266492592\underline{7378}$	2.3×10^{-14}	4.4×10^{-14}
	$\mathbf{0.5202664925927609}$		
3-1	$0.312632188766\underline{41}$	1.6×10^{-15}	4.9×10^{-15}
	$\mathbf{0.3126321887664725}$		
3-2	$0.837043537778\underline{8633}$	1.0×10^{-14}	1.2×10^{-14}
	$\mathbf{0.8370435377788733}$		

More interestingly, its approximate version allows for a significant time complexity improvement, for an actual accuracy that matches and even outperforms the accuracy of previous *exact* algorithms, which suffer from round-off errors when implemented in finite precision arithmetic.

Several numerical experiments have demonstrated the performance of this algorithm in the multinomial case, both with respect to its accuracy and its time complexity.

This algorithm allows addressing problems that were impossible to deal with using previous state-of the art algorithms, either in terms of problem size or in terms of accuracy. It has been implemented in the Open TURNS software (www.openturns.org), an Open Source software dedicated to probabilistic modeling and uncertainty propagation. It has also been implemented it as either a Maple (www.maple.soft.com) or a Python (www.python.org) script, both available upon request to the author.

Some additional work should be done, regarding the sensitivity of this algorithm to round-off error or the optimal choice for t, even if the choice $t = N$ seems to be effective in the multinomial case.

Acknowledgements I would like to thank Pr. Kenneth J. Berry and Pr. Jesse Frey, who kindly provided the source code of their algorithms and additional bibliographical material, as well as the two anonymous reviewers for their very valuable remarks and advices that improved significantly the manuscript.

References

Abate, J., Whitt, W.: The Fourier-series method for inverting transforms of probability distributions. Queueing Syst. **10**(1–2), 5–87 (1992)

Berry, K.J., Mielke, P.W. Jr.: Exact cumulative probabilities fort the multinomial distribution. Technical report 19, Colorado State University (1995)

Butler, R.W., Stutton, R.K.: Saddlepoint approximation for multivariate cumulative distribution functions and probability computations in sampling theory and outlier testing. J. Am. Stat. Assoc. **93**(442), 596–604 (1998)

Childs, A., Balakrishnan, N.: Some approximations to the multivariate hypergeometric distribution with applications to hypothesis testing. Comput. Stat. Data Anal. **35**(2), 137–154 (2000)

Corrado, C.J.: The exact distribution of the maximum, minimum and the range of multinomial/Dirichlet and multivariate hypergeometric frequencies. Stat. Comput. **21**, 349–359 (2011)

Didonato, A.R., Morris, A.H.: Computation of the incomplete gamma function ratios and their inverse. ACM Trans. Math. Softw. **12**(4), 377–393 (1986)

Frey, J.: An algorithm for computing rectangular multinomial probabilities. J. Stat. Comput. Simul. **79**(12), 1483–1489 (2009)

Good, I.J.: Saddle-point methods for the multinomial distribution. Ann. Math. Stat. **4**, 861–881 (1957)

Johnson, N.L.: An approximation to the multinomial distribution some properties and applications. Biometrika **47**(1–2), 93–102 (1960)

Levin, B.: A representation for multinomial cumulative distribution functions. Ann. Math. Stat. **9**(5), 1123–1126 (1981)

Levin, B.: On calculations involving the maximum cell frequency. Commun. Stat. **12**(11), 1299–1327 (1983)

Levin, B.: Siobhan's problem: the coupon collector revisited. Am. Stat. **46**, 76 (1992)

Mallows, C.L.: An inequality involving multinomial probabilities. Biometrika **55**, 422–424 (1968)

Maple computer algebra system, www.maplesoft.com

Open TURNS software, www.openturns.org

Python programming language, www.python.org

Temme, N.M.: A set of algorithms for the incomplete gamma functions. Probab. Eng. Inf. Sci. **8**, 291 (1994)