Statistics and Computing manuscript No.
(will be inserted by the editor)

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

Régis LEBRUN

Received: date / Accepted: date

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 family of multivariate discrete integer distributions for which the existing methods to evaluate such probabilities are either approximate, with no real control on the precision of the approximation, or exact (if the computation is made using exact arithmetic) but available only for some of these distributions or for particular rectangular probabilities.

We propose here a new approximate algorithm that allows performing these computations in the most general case for both the distribution and the rectangular region. Its accuracy matches or even outperforms the exact algorithms when the rounding errors are taken into account. In the worst case, the computational cost of our algorithm is the same as the most efficient exact method published so far, and is much lower in many situations of interest. Our algorithm does not need an intermediate storage that grows with the dimension or problem parameters, which allows dealing with large dimension/large counting parameter applications at no memory cost and 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
EADS Innovation Works
Department of applied mathematics and modeling
12, rue Pasteur BP76
92152 Suresnes Cedex, FRANCE
Tel.: +33-(0)1-46-97-35-80
Fax: +33-(0)1-46-97-35-08
E-mail: regis.lebrun@eads.net

1 Introduction

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

$$\begin{aligned}\forall \mathbf{a},\mathbf{b} \in \mathbb{N}^d,\\ p_{\mathscr{D}}(\mathbf{a},\mathbf{b}) &= \mathbb{P}(\mathbf{a} \leq \mathbf{X} \leq \mathbf{b})\\ &= \mathbb{P}(a_1 \leq X_1 \leq b_1, \ldots, a_d \leq X_d \leq b_d) \qquad (1)\end{aligned}$$

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 [3], [5], [7], [8], [9], [11], [12], [13]), but despite the existing literature on the subject, no function allows to perform this computation in the standard numerical softwares such as R, SAS, Matlab, Scilab or Octave.

Several authors (see [3], [4], [8], [11], [12], [13], [14]) have described in details approximate algorithms for a long time, but these algorithms provide only a limited precision which may be inadequate for some applications, and with no control on the error. This lack of control of the error may be the reason why these algorithms have not yet been implemented a standard numerical package. Some of these authors (see [8], [11], [12], [13]) have also indicated how to derive an exact algorithm if one was able to compute a particular convolution exactly, but with no indication on how to do it efficiently and accurately.

It is only recently that reasonably efficient algorithms for the computation of rectangular probabilities have been described (see [5], [7]), using completely different roots than the previous authors. But even with these algorithms, the only case covered with full generality (see [7]) 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 [3] and [12], amongst which the multinomial, multivariate hypergeometric and multivariate Pólya distributions. In the multinomial case, our algorithm is more efficient with respect to both space and time complexity than the algorithm described in [7], for an equivalent accuracy when implemented in double precision.

More precisely, we are interested in d-dimensional discrete distributions $\mathscr{D}$ with a $d-1$ dimensional probability function. We suppose that there exists a random vector $\mathbf{Y}_t = (Y_{t1},\dots,Y_{td})$ with independent components such that $\mathbf{X} \sim \mathscr{D}$ has the same distribution as $\mathbf{Y}_t \,|\, \sum_{j=1}^d Y_{tj} = N$, 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:

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

where

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

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

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

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

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

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

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

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

The decomposition (2) is obtained with

$$\mathscr{L}(\theta_j) = \mathscr{P}(tp_j)$$

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

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

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

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

Table 1 Exact algorithms applicability

Reference	[2]	[5]	[7]
Distributions	$\mathscr{M}_d$	$\mathscr{M}_d$, $\mathscr{H}_d$	$\mathscr{M}_d$
Restrictions on **a**, **b**	$\mathbf{a} = \mathbf{0}$	$\mathbf{a} = (a,\dots,a)$ $\mathbf{b} = (b,\dots,b)$	*none*

Table 2 Approximate algorithms applicability

Reference	[3], [12], [13]	[4]	[8]	[11]
Distributions	$\mathscr{M}_d$, $\mathscr{H}_d$, $\mathscr{P}_d$	$\mathscr{H}_d$	$\mathscr{M}_d$	$\mathscr{M}_d$
Restrictions on **a**, **b**	*none*	*none*	$\mathbf{a} = (a,\dots,a)$ $\mathbf{b} = (b,\dots,b)$	$\mathbf{a} = \mathbf{0}$

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

The decomposition (2) is obtained with

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

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

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

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

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

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

The decomposition (2) is obtained with

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

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

For all the distributions $\mathscr{L}(\theta)$ we are interested in, there exist efficient and accurate routines to evaluate both $\mathbb{P}(a_j \le Y_{tj} \le b_j)$ and $\mathbb{P}(Y_t = N)$. The only difficulty is the evaluation of $\mathbb{P}(T_t = N)$, as noticed in [8], [11], [12] or [13], 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 saddlepoint approximations.

We list the available algorithms, the distributions they adress and the possible restrictions on **a** and **b** in the computation of (1) in table 1 for the exact algorithms and in table 2 for the approximate ones.

Is the accurate (or even exact) evaluation $\mathbb{P}(T_t = N)$ intractable? A naive use of the definition of the convolution is clearly enough to address situations with small values of d,

N and $b_j - a_j$, for exemple using the following trivial representation:

$$\mathbb{P}(T_t = N) = \sum_{y_1=a_1}^{b_1} P_1 \dots \sum_{y_{d-1}=a_{d-1}}^{b_{d-1}} P_{d-1} Q_d \tag{7}$$

where $P_j = \mathbb{P}(Y_{tj} = y_j)$ and $Q_d = \mathbb{P}(Y_{td} = N - \sum_{k=1}^{d-1} y_k)$, but it is clear that even for moderate values of d and $b_j - a_j$, the cost of such an approach is prohibitive.

Using an appropriate numerical method, it is possible to evaluate such a convolution 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 performances of our algorithm with respect to the exact ones, let's introduce the following notations:

$$\sigma_a = \sum_{j=1}^{d} a_j,\ \sigma_b = \sum_{j=1}^{d} b_j,\ \sigma_{ab} = \sigma_b - \sigma_a,\ N_a = N - \sigma_a \tag{8}$$

The most efficient exact algorithm proposed so far for the evaluation of (1), in the case of the multinomial distribution, is the one described in [7]. Its space complexity is $\mathscr{O}(\sigma_{ab})$ and its time complexity is $\mathscr{O}(N_a \sigma_{ab})$. The algorithm described in [5] has the same space and time complexity, covers more distributions but for restricted arguments of (1), so we take [7] as a reference in terms of accuracy, space and time complexity . To jump directly to the conclusion, and to motivate the reader, the key results are that the proposed algorithm has a constant (and small) $\mathscr{O}(1)$ space complexity, and has a worst case $\mathscr{O}(N_a \sigma_{ab})$ time complexity that drops to $\mathscr{O}(d\sqrt{N_a})$ for most situations, for a relative precision comparable to [7], which is a tremendous improvement with respect to the best current exact algorithms.

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 new algorithm.

Here is the key result given in [1, equations 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 $0 < r < 1$:

$$f_X(n) = \mathbb{P}(X = n) = \frac{1}{m r^n} \sum_{k=0}^{m-1} \xi_m^{-kn} \phi_X\left(r \xi_m^k\right) - \varepsilon_{n,m,r} \tag{9}$$

where $\varepsilon_{n,m,r} = \sum_{k\geq 1} f_X(n+km) r^{km} \leq \frac{r^m}{1-r^m} \simeq r^m$ 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 (9), 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 $\varepsilon_{n,m,r} = 0$ for such a choice.

Using $m = 2n$ in (9) 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 [1, equations 5.38–5.39], and reads:

Proposition 1 *If we take $m = 2n$ in (9), we get:*

$$f_X(n) = \frac{1}{2nr^n} \sum_{k=0}^{n-1} (-1)^k \Re\left(\phi_X\left(r\zeta_n^k\right) - \phi_X\left(r\zeta_n^{k+1}\right)\right) - \varepsilon_{n,r} \tag{10}$$

where

$$\varepsilon_{n,r} = \sum_{k\geq 1} f_X((2k+1)n) r^{2kn} \leq \frac{r^{2n}}{1-r^{2n}} \simeq r^{2n} \tag{11}$$

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

We will apply (9) if we want an exact algorithm, or (10) 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 (11) can be made smaller than a given ε_{max} by choosing r such that $r^{2n} \leq \varepsilon_{max}$:

$$r \leq \varepsilon_{max}^{\frac{1}{2n}} \tag{12}$$

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

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

$$\phi_{T_{tj}}(z) = \frac{\pi^{(j)}_{a_j b_j}(z)}{\mathbb{P}(a_j \leq Y_{tj} \leq b_j)} \tag{13}$$

with

$$\pi^{(j)}_{a_j b_j}(z) = \sum_{k=a_j}^{b_j} \mathbb{P}(Y_{tj} = k) z^k = \pi^{(j)}_{b_j}(z) - \pi^{(j)}_{a_j - 1}(z) \tag{14}$$

where

$$\pi^{(j)}_{-1}(z) \equiv 0, \quad \forall n \in \mathbb{N}, \pi^{(j)}_{n}(z) = \sum_{k=0}^{n} \mathbb{P}(Y_{tj} = k) z^k \tag{15}$$

The independence of the T_{tj} leads to:

$$\phi_{T_t}(z) = \frac{\prod_{j=1}^{d} \pi^{(j)}_{a_j b_j}(z)}{\prod_{j=1}^{d} \mathbb{P}(a_j \leq Y_{tj} \leq b_j)} \tag{16}$$

We see that a key factor in the cost of (9) 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 (9). 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 $\mathbf{a} \neq \mathbf{0}$, $\forall j \in \{1,\dots,d\}$ we set $V_{tj} = T_{tj} - a_j$. The random variables V_{tj} are such that:*

$$\mathbb{P}(V_{tj} = k) = \mathbb{P}(T_{tj} = a_j + k) \tag{17}$$

$$\phi_{V_{tj}}(z) = z^{-a_j} \phi_{T_{tj}}(z) \tag{18}$$

and

$$\mathbb{P}(T_t = N) = \mathbb{P}(V_t = N_a) \tag{19}$$

$$\phi_{V_t}(z) = z^{-\sigma_a} \phi_{T_t}(z) \tag{20}$$

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

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:*

$$p_{\mathscr{D}}(\mathbf{a},\mathbf{b}) = \Re\left\{ \sum_{k=0}^{N-1} (-\zeta_{N_a}^{-\sigma_a})^k \left(\prod_{j=1}^{d} \pi^{(j)}_{a_j b_j}\left(r\zeta_{N_a}^k\right) - \zeta_{N_a}^{-\sigma_a} \prod_{j=1}^{d} \pi^{(j)}_{a_j b_j}\left(r\zeta_{N_a}^{k+1}\right) \right) \right\} / \left(2N_a r^N \mathbb{P}(Y_t = N)\right) - \eta_{N_a,r} \tag{21}$$

where $K = \dfrac{\prod_{j=1}^{d} \mathbb{P}(a_j \leq Y_{tj} \leq b_j)}{\mathbb{P}(Y_t = N)}$ and $\eta_{N_a,r} = K\varepsilon_{N_a,r}$.

Except for the storage of the data **a**, **b** and **p**, which is a $\mathscr{O}(d)$, the memory complexity of this algorithm is $\mathscr{O}(1)$ as no intermediate structure is needed in the evaluation of (21). The time complexity is of order $\mathscr{O}(N_a C)$, where C is the time complexity of evaluating $\pi^{(j)}_{a_1 b_1}, \dots, \pi^{(j)}_{a_d b_d}$ at a given point. A naive evaluation of these polynomials leads to $C \simeq \sigma_{ab}$ and a total time complexity of $\mathscr{O}(N_a \sigma_{ab})$, which is the same complexity as the algorithm proposed in [7]. 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 (16), reducing the overall computational cost.

One can see that the error in (21) depends on t through the numerator of K, and on r through $\varepsilon_{N_a,r}$. The theoretical behavior of this error is clear: we can take r small enough to get the absolute error we want. The numerical behavior of this error is less clear as the summation in (21) can be subject to cancellation, increasing the error. The best way to take into account these cancellations is to use the recommendations in [1] to choose r using (12), 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 $\mathscr{O}(1)$ operations instead of $\mathscr{O}(b_j - a_j)$ within machine precision when $b_j - a_j \gg 1$, counting the evaluation of a transcendental function such as exp as a $\mathscr{O}(1)$ operation. In this case, $C = \mathscr{O}(d)$ instead of $C = \mathscr{O}(\sigma_{ab})$, and the total complexity drops to $\mathscr{O}(N_a d)$. The second one is that the terms involved in (21) are usually of very different magnitude, and most of them does not contribute significantly (up to machine precision) to the final result. It is common that only $\mathscr{O}(\sqrt{N_a})$ terms are needed. The overall complexity is thus reduced to $\mathscr{O}(d\sqrt{N_a})$.

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

If $b_j - a_j = \mathscr{O}(1)$, the evaluation of $\pi^{(j)}_{a_j b_j}$ is obviously a $\mathscr{O}(1)$, so we restrict our attention to the case $b_j - a_j \gg 1$. It covers two different sub-cases: either we have $a_j = \mathscr{O}(1)$, 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 $\pi^{(j)}_{a_j b_j}$ cheaper than $\mathscr{O}(b_j - a_j)$:

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*

$$\left|\phi_{Y_{tj}}(z) - \pi^{(j)}_n(z)\right| \leq \frac{\phi_{Y_{tj}}(e^{s^*})}{e^{(n+1)s^*}} \tag{22}$$

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

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

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

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 (22). □

The hypothesis made on $\phi_{Y_{tj}}$ is fulfilled in the particular cases of the Poisson, binomial and negative binomial distributions, for which the respective values of s^* are $\log\left(\frac{n+1}{tp_j}\right)$, $\log\left(\frac{1-t}{t}\frac{n+1}{h_j-(n+1)}\right)$ and $\log\left(\frac{n+1}{(1-t)(q_j+n+1)}\right)$. 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 in specific cases (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 $|dv_k| > |v_k| \varepsilon_{machine}$ 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 $\mathscr{O}(|\log \varepsilon_{machine}|)$ iteration. The evaluation of $\phi_{Y_{tj}}(z)$ and $\mathbb{P}(Y_{tj} = k)$ can be done in $\mathscr{O}(1)$ time complexity for the common distributions, and the update (4b) 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. $Y_{tj} \sim \mathscr{P}(tp_j)$, the situation of (5) is likely to occur when $\sigma_b = \mathscr{O}(dN)$ and $\sigma_a = \mathscr{O}(d)$, i.e. when $b_j = \mathscr{O}(N)$ and $a_j = \mathscr{O}(1)$, 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 $\varepsilon_{machine} = 10^{-16}$ and $tp_j = 1$, a situation typical of interacting particle algorithms setting. The update is given by $f(dv_k, k, z) = dv_k \times \dfrac{tp_j z}{k+1}$.

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 $\mathscr{O}(1)$ 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 [6], [18] 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 (21) can have very different magnitudes. As a result, only a few of them could 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 *We consider the case where $N \gg 1$ and $\phi_{T_t}(z) \simeq e^{-t(1-z)}$. Either $t = \mathscr{O}(1)$ and no term of the sum in (21) is negligible, or $t \to +\infty$ with $t = \mathscr{O}(N)$ and only the N^* first terms of (21) have a relative contribution to $p_{\mathscr{D}}(\mathbf{a}, \mathbf{b})$ greater than $\varepsilon \ll 1$, with:*

$$N^* \simeq \frac{1}{\pi}\sqrt{-\frac{2\log\varepsilon}{r}\frac{N^2}{t}} \tag{23}$$

The case $t \gg N$ is not relevant, as it leads to severe cancellations in (21).

Proof In order to study the magnitude of the terms occurring in (10), we use the elementary relation:

$$|e^\beta - e^\alpha|^2 = e^{2\Re(\beta)} \rho_{\alpha,\beta} \tag{24}$$

with

$$\rho_{\alpha,\beta} = 1 + e^{2\Re(\alpha-\beta)} - 2\cos\left(\Im(\alpha-\beta)\right) e^{\Re(\alpha-\beta)} \tag{25}$$

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

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 $\zeta_N - 1 = \frac{i\pi}{N} + \mathcal{O}\left(\frac{1}{N}\right)$ we get:

$$\frac{|\delta_k|}{|\delta_0|} = e^{-rt(1-\cos(\theta_k))} + \mathcal{O}\left(\frac{t}{N}\right) \tag{26}$$

We have $|\delta_k|/|\delta_0| < \varepsilon$ as soon as $\cos(\theta_k) \leq 1 + \frac{\log\varepsilon}{rt}$. 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 (10). 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 (23).

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 (23) by computations similar to the previous cases. More precisely, defining $\gamma = \frac{t}{N}$ and assuming that $\cos(\gamma r\pi) < 1$, we get:

$$\begin{aligned}\frac{|\delta_k|}{|\delta_0|} &= e^{-rt(1-\cos(\theta_k))} \\ &\quad \times \sqrt{\frac{1 - 2\cos(\gamma r\pi\cos(\theta_k))e^{-\gamma r\pi\sin(\theta_k)} + e^{-2\gamma r\pi\sin(\theta_k)}}{2(1-\cos(\gamma r\pi))}} \\ &\quad + \mathcal{O}\left(\frac{1}{N}\right)\end{aligned} \tag{27}$$

from which we deduce that:

$$\begin{aligned}\frac{|\delta_k|}{|\delta_0|} &\leq e^{-rt(1-\cos(\theta_k))}\frac{1+e^{-\gamma r\pi\sin(\theta_k)}}{\sqrt{2(1-\cos(\gamma r\pi))}} \\ &\leq \frac{e^{-rt(1-\cos(\theta_k))}}{\sqrt{1-\cos(\gamma r\pi)}}\end{aligned} \tag{28}$$

as $\theta_k \in [0,\pi]$. The same computation as in the case $t = o(N)$ shows that the upper bound of (28) is smaller than ε as soon as $k \leq \frac{1}{\pi}\sqrt{N\frac{2\log\varepsilon\sqrt{1-\cos(\gamma r\pi)}}{\gamma r}} = o(N)$, i.e. $\theta_k = o(1)$. It is thus possible to expand the square-root term of (27) with respect to θ_k, and one gets (23) the same way as for the previous case. □

If we choose $t = N$ as suggested in [11], we get $N^* = \mathcal{O}(\sqrt{N})$. The resulting algorithm reads:

Algorithm 2

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

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

The evaluation of $\pi_{b_j a_j - 1}^{(j)}$ 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 is not the same as in proposition 6, but using (16) we see that the ratio $|\delta_k|/|\delta_0|$ is 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 [16], 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. An application to interacting particles algorithm will also be made through the computation of the cumulative distribution function of the most occupied site at the end of the resampling step of such algorithms. 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.

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 figure 1, for ε_{max} taken in $\{10^{-7}, 10^{-9}, 10^{-11}\}$. The points with a zero maximal error are not plotted.

[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 $\Theta(N)$.

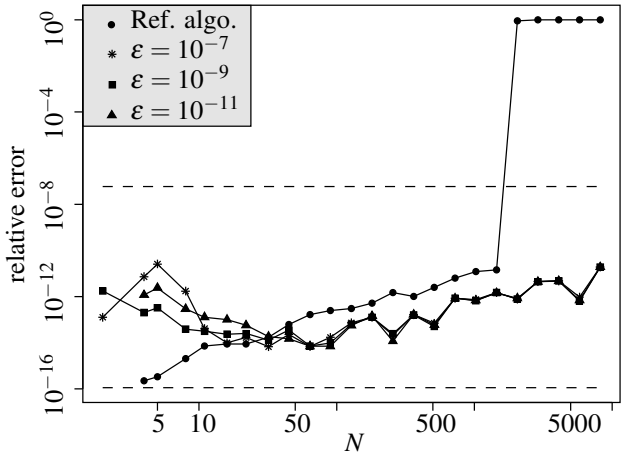

Fig. 1 Maximum relative error for various problem size N and various precision parameter ε_{max}. For problem size larger than 50, the proposed algorithm is consistently more accurate than the reference algorithm, and the achieved accuracy does not depends on the value of ε_{max}. The two horizontal 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 facts 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 size, 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 $\varepsilon_{max} \simeq 10^{-9}$ seems to give the best overall precision, even if in this case the algorithm [7] should probably be preferred. The third fact is that the implementation of the algorithm [7] as given in the reference paper seems to have an overflow for problems of size 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 [2], [5], [11] 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 [7] 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 of the computed result and underline the digits that differ from the exact result/ We also give the absolute and relative error of the computed result.

Table 3 Probability value and precision of the examples

Example	probability	absolute error	relative error
1-1	$0.4784509465\underline{81}8295\,10^{-5}$	$1.5\,10^{-17}$	$3.2\,10^{-12}$
2-1	$0.852726985258\underline{1}543$	$1.5\,10^{-14}$	$1.8\,10^{-14}$
2-2	$0.602684281137\underline{5}376$	$2.3\,10^{-14}$	$3.9\,10^{-14}$
2-3	$0.520266492592\underline{7}378$	$2.3\,10^{-14}$	$4.4\,10^{-14}$
3-1	$0.312632188766\underline{4}741$	$1.6\,10^{-15}$	$4.9\,10^{-15}$
3-2	$0.837043537778\underline{8}633$	$1.0\,10^{-14}$	$1.2\,10^{-14}$

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

$\mathbf{X} \sim \mathscr{M}(200, [0.2, 0.35, 0.15, 0.3])$
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 [5] and [11], and is attributed to Mallows. It consists in the following computation:

$\mathbf{X} \sim \mathscr{M}(500, [p_1 = \cdots = p_{50} = 1/50])$
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 [5]:

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 [11], and is attributed to Barton and David. It consists in the following computation:

$\mathbf{X} \sim \mathscr{M}(12, [p_1 = \cdots = p_{12} = 1/12])$
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 [3], for which the reported absolute error is of order $5\,10^{-5}$, which illustrate 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 verify the asymptotic time complexity of the algorithm with respect to both the N and d parameters in the most demanding situation, namely the computation of $\mathbb{P}(X_1 \leq N-1, \ldots, X_d \leq N-1)$ for equiprobable X_i.

[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.

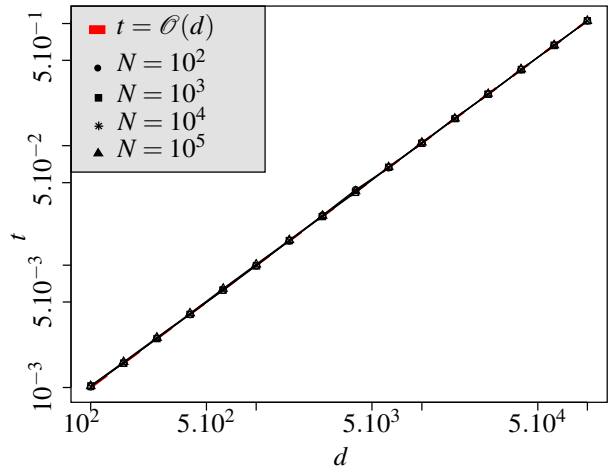

Fig. 2 Evolution of the time complexity with respect to the dimension d in logarithmic scale, for several values of N. The time is normalized such that it is equal to 1 for the largest value of d. We see the perfect matching of the $\mathcal{O}(d)$ complexity as all the curves are superposed with the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

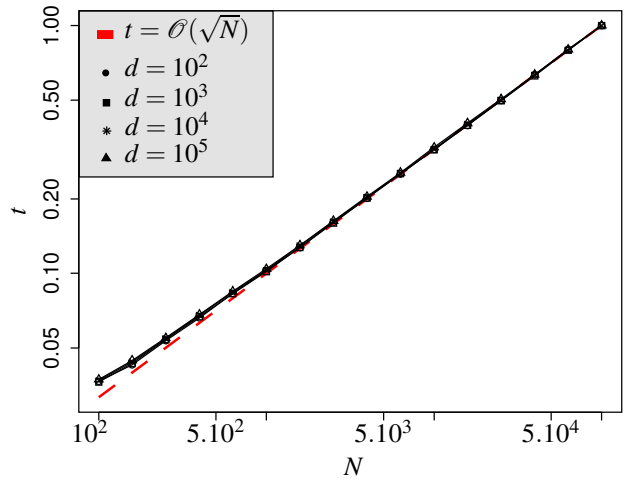

Fig. 3 Evolution of the time complexity with respect to N in logarithmic scale, for several values of the dimension d. The time is normalized such that it is equal to 1 for the largest value of N. We see the almost perfect matching of the $\mathcal{O}(\sqrt{N})$ complexity as all the curves are almost superposed with the dashed curve $t = \sqrt{N}$.

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.

The time complexity matches perfectly the theoretical bounds, 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 permits the computation of rectangular probabilities to 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.

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

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

This algorithm allows to address 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 [16], an Open Source software dedicated to probabilistic modeling and uncertainty propagation. It is also available as a Python [17] script or as a Maple [15] script upon request to the author.

Some additional work should be made regarding this algorithm, regarding its sensitivity 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 algorithm and additional bibliographical material. I would also thank the two anonymous reviewers for their very valuable remarks and advices that improved the manuscript significantly.

References

1. Joseph Abate, Ward Whitt, "The Fourier-series method for inverting transforms of probability distributions", Queueing Systems, 10:1-2, 5–87 (1992).
2. Kenneth J. Berry and Paul W. Mielke Jr., "Exact cumulative probabilities fort the multinomial distribution", Colorado State University, technical report 19 (1995).
3. Ronald W. Butler, Richard K. Stutton, "Saddlepoint Approximation for Multivariate Cumulative Distribution Functions and Probability Computations in Sampling Theory and Outlier Testing", Journal of the American Statistical Association, 93:442, 596–604 (1998).
4. Aaron Childs, N. Balakrishnan, "Some approximations to the multivariate hypergeometric distribution with applications to hypothesis testing", Computational Statistics & Data Analysis, 35:2, 137–154 (2000).
5. Charles J. Corrado, "The exact distribution of the maximum, minimum and the range of Multinomial/Dirichlet and Multivariate Hypergeometric frequencies", Statistics and Computing, 21, 349–359 (2011).

6. A. R. Didonato and A. H. Morris, "Computation of the Incomplete Gamma Function Ratios and their Inverse.", ACM TOMS, 12:4, 377–393 (1986).
7. Jesse Frey, "An algorithm for computing rectangular multinomial probabilities", Journal of Statistical Computation and Simulation, 79:12, 1483–1489 (2009).
8. I. J. Good, "Saddle-point methods for the multinomial distribution", Annals of Mathematical Statistics, 4, 861–881 (1957).
9. N. L. Johnson, "An approximation to the multinomial distribution some properties and applications", Biometrika, 47:1–2, 93–102 (1960).
10. Olav Kallenberg, "Foundations of Modern Probability second edition", 537–562. Springer, New-York (2002).
11. Bruce Levin, "A representation for multinomial cumulative distribution functions", The Annals of Statistics, 9:5, 1123–1126 (1981).
12. Bruce Levin, "On Calculations Involving the Maximum Cell Frequency", Communications in Statistics, special edition on the Analysis of Categorical Data 12(11):1299-1327 (1983).
13. Bruce Levin, In re: "Siobhan's Problem: The Coupon Collector Revisited", The American Statistician 46:76 (Letter to the Editor) (1992).
14. C. L. Mallows, "An inequality involving multinomial probabilities", Biometrika, 55, 422-424 (1968).
15. Maple computer algebra system, www.maplesoft.com.
16. Open TURNS software, www.openturns.org.
17. Python programming language, python.org.
18. N. M. Temme, "A Set of Algorithms For the Incomplete Gamma Functions.", Probability in the Engineering and Informational Sciences, 8, 291– (1994).