On Computing the Distribution Function for the Sum of Independent and Non-identical Random Indicators

Yili Hong
Department of Statistics
Virginia Tech
Blacksburg, VA 24061, USA

April 5, 2011

Abstract

The Poisson binomial distribution is the distribution of the sum of independent and non-identical random indicators. Each indicator follows a Bernoulli distribution with individual success probability. When all success probabilities are equal, the Poisson binomial distribution is a binomial distribution. The Poisson binomial distribution has many applications in different areas such as reliability, survival analysis, survey sampling, econometrics, etc. The computing of the cumulative distribution function (cdf) of the Poisson binomial distribution, however, is not straightforward. Approximation methods such as the Poisson approximation and normal approximations have been used in literature. Recursive formulae also have been used to compute the cdf in some areas. In this paper, we present a simple derivation for an exact formula with a closed-form expression for the cdf of the Poisson binomial distribution. The derivation uses the discrete Fourier transform of the characteristic function of the distribution. We develop an algorithm for efficient implementation of the exact formula. Numerical studies were conducted to study the accuracy of the developed algorithm and the accuracy of approximation methods. We also studied the computational efficiency of different methods. The paper is concluded with a discussion on the use of different methods in practice and some suggestions for practitioners.

Key Words: Characteristic function; Discrete Fourier transform; k-out-of-n system; Normal approximation; Poisson binomial distribution; Warranty returns.

1 Introduction

1.1 Motivation

The Poisson binomial distribution describes the distribution of the sum of independent and non-identical random indicators. Each indicator is a Bernoulli random variable with individual success probability. A special case of the Poisson binomial distribution is when all success probabilities are equal. In this case, the Poisson binomial distribution is a binomial distribution. The Poisson binomial distribution has many applications in different areas such as reliability, survival analysis, survey sampling, econometrics, and so on. For example, in some reliability applications, it is often of interest to predict the total number of failures for a fleet of products in the field. Hong, Meeker, and McCalley (2009) considered the prediction for the total number of field failures for a fleet of high-voltage power transformers. Due to staggered entry of units into service, individual units in the field have different failure probabilities at a specified future time. Thus the total number of field failures follows a Poisson binomial distribution. In econometrics, it is sometimes of interest to predict the number of corporation defaults (Duffie, Saita, and Wang 2007). The default probabilities differ from corporation to corporation because each corporation has its unique situation such as assets, debts and stock returns. The number of corporation defaults at a future time also follows a Poisson binomial distribution. Chen and Liu (1997) presented an example on survey sampling where each sampling unit has different probabilities to be included in the sample. Fernández and Williams (2010) provided more examples from areas such as pattern identification, multi-sensor fusion, and reliability of k-out-of-n systems.

While the Poisson binomial distribution has many applications, the computing of the cumulative distribution function (cdf) of the distribution is not straightforward. Because each indicator has different success probabilities, the naive way of computing the cdf by using enumeration is not practicable, even when the number of indicators is small (i.e., around 30). Approximation methods such as the Poisson approximation and normal approximations have been used in literature. There are situations, however, in which approximation methods do not perform well. Thus it is desirable to have a method to compute the exact values of the cdf. It is also useful to know in which situation approximation methods work well. In applications such as predictions for the number of failures and corporation defaults, the number of indicators is usually large. Thus efficiency of algorithms for computing the exact values of the cdf is also important. This motivates us to provide efficient methods to compute the exact values of the cdf of the Poisson binomial distribution.

1.2 Related Literature and This Work

The study on Poisson binomial distribution has a long history. Le Cam (1960) provided an upper bound for the error of the Poisson approximation. Normal approximations are widely used in practice. Volkova (1996) gave a normal approximation with second order correction and provided an upper bound for the error of the approximation. Hong, Meeker, and McCalley (2009) and Hong and Meeker (2010) applied the approximation in Volkova (1996) to warranty prediction applications. Recursive formulae are available in literature to compute the exact values of the cdf of the Poisson binomial distribution. For example, Barlow and Heidtmann (1984) described a recursive formula for computing the cdf. Chen, Dempster, and Liu (1994) provided another recursive formula. Details for these recursive formulae are described in Section 2.5. Fernández and Williams (2010) gave a closed-form expression for the cdf using the technique of polynomial interpolation and the discrete Fourier transform.

In this paper, we present a simple derivation for an exact formula for the cdf of the Poisson binomial distribution, which gives the same form as in Fernández and Williams (2010). We develop an algorithm that efficiently implements the exact formula. Numerical studies were conducted to compare the accuracy of the algorithm and the accuracy of approximation methods. We also compared the computational efficiency of different methods. Based on the numerical studies, we provide a discussion on the advantages and disadvantages of different methods and some guidelines for practitioners.

The statistical software R (2011) is widely used and it is free. There is no package, however, for computing the Poisson binomial distribution function. We developed an R package that efficiently implements both exact methods and approximation methods and it can be downloaded from the R website. See Section 5 for more details.

1.3 Overview

The rest of the paper is organized as follows. Section 2 describes several exact methods for computing the cdf and algorithms for their efficient implementations. Section 3 describes several approximation methods based on the Poisson and normal approximations. Section 4 conducts a comprehensive numerical studies to assess the performance of various methods in terms of accuracy and efficiency. Section 5 discusses software implementation for both the exact and approximation methods. Section 6 provides some concluding remarks and suggestions for practitioners.

2 Exact Methods

2.1 Notation

Let $I_j, j = 1, \ldots, n$ be a series of n random indicators. Each indicator is independent and follows

$$I_j \sim \text{Bernoulli}(p_j), j = 1, \ldots, n \tag{1}$$

where $p_j = \Pr(I_j = 1)$ is the success probability for indicator I_j and not all p_j's are equal. The Poisson binomial random variable N is defined as the sum of independent and non-identical random indicators. That is $N = \sum_{j=1}^{n} I_j$ and N takes value in $\{0, 1, \ldots, n\}$.

Let $\xi_k = \Pr(N = k), k = 0, 1, \ldots, n$ be the probability mass function (pmf) for the Poisson binomial random variable N. When all p_j's are identical, the distribution of N is a binomial distribution. The cdf of N, denoted by $F_N(k) = \Pr(N \leq k), k = 0, 1, \ldots, n$, gives the probability of having at least k successes out of a total of n. The cdf $F_N(k)$ can be expressed by (Wang 1993)

$$F_N(k) = \sum_{m=0}^{k} \xi_m = \sum_{m=0}^{k} \left\{ \sum_{A \in \mathcal{F}_m} \prod_{j \in A} p_j \prod_{j \in A^c} (1 - p_j) \right\} \tag{2}$$

where $\mathcal{F}_m$ is the set of all subsets of m integers that can be selected from $\{1, 2, 3, \ldots, n\}$ and A^c is the complement of set A (i.e., $A^c = \{1, 2, 3, \ldots, n\} \setminus A$). The computing of $F_N(k)$ in (2) by enumerating all elements in $\mathcal{F}_m$ is not practicable, even when n is small (e.g., $n = 30$). Thus computationally efficient methods are desirable.

2.2 Discrete Fourier Transform

In this section, we give a brief introduction to the discrete Fourier transform (DFT). For a sequence of $n+1$ complex numbers $\{y_0, y_1, \cdots, y_n\}$, the DFT transforms $\{y_0, y_1, \cdots, y_n\}$ into a sequence of $n+1$ complex numbers $\{z_0, z_1, \cdots, z_n\}$ where

$$z_k = \sum_{l=0}^{n} y_l \exp(-\boldsymbol{i}\omega kl), k = 0, 1, \ldots, n$$

and $\omega = 2\pi/(n+1)$. The inverse discrete Fourier transform (IDFT), which recovers $\{y_0, y_1, \cdots, y_n\}$ from $\{z_0, z_1, \cdots, z_n\}$, is given by

$$y_l = \frac{1}{n+1} \sum_{k=0}^{n} z_k \exp(\boldsymbol{i}\omega lk), l = 0, 1, \cdots, n. \tag{3}$$

Applying the DFT to both sides of equation (3), one can also recover $\{z_0, z_1, \cdots, z_n\}$ from $\{y_0, y_1, \cdots, y_n\}$. More details on the DFT can be found in Bracewell (2000, Chapter 11).

There are fast Fourier transform (FFT) algorithms to efficiently compute the DFT. The most commonly-used algorithm is the Cooley-Tukey algorithm (Cooley and Tukey 1965). There are also subroutines available in C or FORTRAN that implement FFT algorithms. See Bracewell (2000, Chapter 11) for details on FFT algorithms.

2.3 The DFT of the Characteristic Function

Fernández and Williams (2010) provided a closed-form expression for $F_N(k)$ which was derived by using polynomial interpolation technique and the DFT. In this paper, we present a simpler derivation, based on the characteristic function (CF) for random variables (see, for example, Athreya and Lahiri 2006, Chapter 10). The CF of $N = \sum_{j=1}^{n} I_j$ is

$$\begin{aligned}\varphi(t) = \mathbf{E}[\exp(\boldsymbol{i}tN)] &= \sum_{k=0}^{n} \xi_k \exp(\boldsymbol{i}tk) = \mathbf{E}\left[\exp\left(\boldsymbol{i}t\sum_{j=1}^{n} I_j\right)\right] \\ &= \prod_{j=1}^{n} \mathbf{E}[\exp(\boldsymbol{i}tI_j)] = \prod_{j=1}^{n}[1 - p_j + p_j \exp(\boldsymbol{i}t)],\end{aligned} \tag{4}$$

where $\boldsymbol{i} = \sqrt{-1}$. Substituting $t = \omega l, l = 0, 1, \cdots, n$ into (4) where $\omega = 2\pi/(n+1)$, one obtains

$$\frac{1}{n+1}\sum_{k=0}^{n} \xi_k \exp(\boldsymbol{i}\omega lk) = \frac{1}{n+1}\prod_{j=1}^{n}[1 - p_j + p_j \exp(\boldsymbol{i}\omega l)] = \frac{1}{n+1}x_l, \quad l = 0, 1, \cdots, n, \tag{5}$$

where $x_l = \prod_{j=1}^{n}[1 - p_j + p_j \exp(\boldsymbol{i}\omega l)]$. Note that the left hand side of equation (5) is the IDFT of the sequence $\{\xi_0, \xi_1, \cdots, \xi_n\}$. Apply the DFT to both sides of equation (5), one recovers $\{\xi_0, \xi_1, \cdots, \xi_n\}$. In particular,

$$\xi_k = \frac{1}{n+1}\sum_{l=0}^{n} \exp(-\boldsymbol{i}\omega lk)\prod_{j=1}^{n}[1 - p_j + p_j \exp(\boldsymbol{i}\omega l)] = \frac{1}{n+1}\sum_{l=0}^{n} \exp(-\boldsymbol{i}\omega lk)x_l. \tag{6}$$

The expression in equation (6) gives the same closed-form expression as in Fernández and Williams (2010). From (6), the cdf of N can be expressed as

$$F_N(k) = \sum_{m=0}^{k} \xi_m = \frac{1}{n+1}\sum_{l=0}^{n}\sum_{m=0}^{k} \exp(-\boldsymbol{i}\omega lm)x_l = \frac{1}{n+1}\sum_{l=0}^{n} \frac{\{1 - \exp[-\boldsymbol{i}\omega l(k+1)]\}x_l}{1 - \exp(-\boldsymbol{i}\omega l)}. \tag{7}$$

The last equality in (7) follows from the fact that $\exp(-\boldsymbol{i}\omega lm), m = 0, 1, \ldots, k$ is a geometric sequence. We refer to the closed-form expression in (7) for $F_N(k)$ as the DFT-CF method.

2.4 Efficient Implementation of the DFT-CF Method

In this section, we describe details on efficient implementation for computing the cdf $F_N(k)$ in (7). To compute $\xi_k, k = 0, 1, \ldots, n$, one first needs to compute x_l. Let $x_l = a_l + \boldsymbol{i}b_l, l = 0, 1, \ldots, n$, where a_l and b_l are the real and imaginary parts of x_l, respectively. From (5), $x_l = \sum_{k=0}^{n} \xi_k \exp(\boldsymbol{i}\omega lk), l = 0, 1, \cdots, n$. Note that $x_0 = \sum_{k=0}^{n} \xi_k = 1$. Because all ξ_k's are real numbers and $\exp[\boldsymbol{i}\omega(n+1)k] = 1$, the conjugate of x_l is

$$\begin{aligned}\overline{x_l} = a_l - \boldsymbol{i}b_l &= \sum_{k=0}^{n} \xi_k \exp(-\boldsymbol{i}\omega lk) = \sum_{k=0}^{n} \xi_k \exp[\boldsymbol{i}\omega(n+1-l)k] \\ &= x_{n+1-l} = a_{n+1-l} + \boldsymbol{i}b_{n+1-l}, \quad l = 1, \ldots, n.\end{aligned}$$

Thus $a_l = a_{n+1-l}$, and $b_l = -b_{n+1-l}$ for $l = 1, \ldots, n$. Let $z_j(l) = 1 - p_j + p_j \cos(\omega l) + \boldsymbol{i}p_j \sin(\omega l)$, $|z_j(l)|$ be the modulus of $z_j(l)$, and $\mathrm{Arg}[z_j(l)]$ be the principal value of the argument of $z_j(l)$. Note that

$$\begin{aligned}x_l &= \exp\left\{\sum_{j=1}^{n} \log\left[z_j(l)\right]\right\} = \exp\left\{\sum_{j=1}^{n} \log\left(|z_j(l)| \exp\{\boldsymbol{i}\mathrm{Arg}[z_j(l)]\}\right)\right\} \\ &= \exp\left\{\sum_{j=1}^{n} \log\left[\,|z_j(l)|\,\right]\right\} \exp\left(\boldsymbol{i}\sum_{j=1}^{n} \mathrm{Arg}[z_j(l)]\right) \\ &= \exp\left\{\sum_{j=1}^{n} \log\left[\,|z_j(l)|\,\right]\right\} \left(\cos\left\{\sum_{j=1}^{n} \mathrm{Arg}[z_j(l)]\right\} + \boldsymbol{i}\sin\left\{\sum_{j=1}^{n} \mathrm{Arg}[z_j(l)]\right\}\right).\end{aligned}$$

Here $|z_j(l)| = \{[1 - p_j + p_j \cos(\omega l)]^2 + [p_j \sin(\omega l)]^2\}^{1/2}$ and $\mathrm{Arg}[z_j(l)] = \mathrm{atan2}[p_j \sin(\omega l), 1 - p_j + p_j \cos(\omega l)]$. The function $\mathrm{atan2}(y, x)$ is defined as

$$\mathrm{atan2}(y, x) = \begin{cases} \arctan(\frac{y}{x}) & x > 0 \\ \pi + \arctan(\frac{y}{x}) & y \geq 0, x < 0 \\ -\pi + \arctan(\frac{y}{x}) & y < 0, x < 0 \\ \frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ 0 & y = 0, x = 0 \end{cases}.$$

Thus explicit expressions for a_l and b_l are

$$a_l = d_l \cos\left\{\sum_{j=1}^{n} \mathrm{Arg}[z_j(l)]\right\} \text{ and } b_l = d_l \sin\left\{\sum_{j=1}^{n} \mathrm{Arg}[z_j(l)]\right\} \tag{8}$$

where $d_l = \exp\left\{\sum_{j=1}^{n} \log\left[\,|z_j(l)|\,\right]\right\}, l = 1, \ldots, n$. The following algorithm is used to compute the cdf $F_N(k)$ for $k = 0, 1, \cdots, n$.

Algorithm A:

1. Let $x_0 = 1$. For $l = 1, \ldots [n/2]$, compute the real and imaginary parts of x_l by using the formulae in (8). Here $[\,\cdot\,]$ is the ceiling function.

2. For $l = [n/2]+1, \ldots, n$, compute the real and imaginary parts of x_l by using the formula $a_l = a_{n+1-l}$, and $b_l = -b_{n+1-l}$.

3. Apply the FFT algorithm to the set $\{x_0/(n+1), x_1/(n+1), \ldots, x_n/(n+1)\}$ to obtain $\{\xi_0, \xi_1, \ldots, \xi_n\}$.

4. Compute the cdf by using $F_N(k) = \sum_{m=0}^{k} \xi_m, k = 0, 1, \cdots, n$.

The above algorithm returns the entire cdf by doing the FFT once. Because there are C or FORTRAN subroutines available to do the FFT, the implementation of **Algorithm A** is not difficult. The FFT algorithm that is used for the implementation in this paper is due to Singleton (1969), which is an FFT algorithm based on the Cooley-Tukey algorithm. The original subroutine was written in FORTRAN. It was translated to C and was included in the R library.

2.5 Recursive Formulae

Recursive formulae (RF) are available in literature to compute $F_N(k)$. Barlow and Heidtmann (1984) described the following recursive formula. A better description of the algorithm is available in Kuo and Zuo (2003, Chapter 7). Let $N_j = \sum_{m=1}^{j} I_m$ and $\xi_{k,j} = \Pr(N_j = k)$ where the random indicator I_m is defined in (1). Note that $N = N_n$ and $\xi_k = \xi_{k,n}$. The recursive formula is given by

$$\xi_{k,j} = (1 - p_j)\xi_{k,j-1} + p_j \xi_{k-1,j-1}, \quad 0 \le k \le n, 0 \le j \le n. \tag{9}$$

The boundary conditions for (9) are $\xi_{-1,j} = \xi_{j+1,j} = 0, j = 0, 1, \ldots, n-1$ and $\xi_{0,0} = 1$. We refer to (9) as the RF1 method.

Chen, Dempster, and Liu (1994) introduced another recursive formula for computing ξ_k. The algorithm requires all $p_j < 1$. In particular, the formula is given by

$$\xi_0 = \prod_{j=1}^{n} (1 - p_j), \text{ and } \quad \xi_k = \frac{1}{k} \sum_{l=1}^{k} (-1)^{l-1} t_l \xi_{k-l}, \ k = 1, \ldots, n \tag{10}$$

where $t_l = \sum_{j=1}^{n} [p_j/(1-p_j)]^l$. We refer to (10) as the RF2 method. This formula is sometimes not numerically stable. This is caused by round-off error in ξ_0 and the explosion of the term $[p_j/(1-p_j)]^l$ in t_l, especially when p_j is close to 1 and n is large.

3 Approximation Methods

In this section, we describe several commonly-used approximation methods for computing the cdf $F_N(k)$. Approximation methods are still widely used because of their computational efficiency, especially when n is large and the cdf $F_N(k)$ needs to be evaluated many times. For example, in the prediction application in Hong, Meeker, and McCalley (2009), the cdf needs to be evaluated $B = 10{,}000$ times in the calibration of prediction intervals for the number of field failures. We will need moments or functions of moments of N in the description of approximation methods. The expectation, standard deviation, and skewness of the distribution of N are

$$\mu = \mathbf{E}(N) = \sum_{j=1}^{n} p_j, \quad \sigma = [\mathrm{Var}(N)]^{1/2} = \left[\sum_{j=1}^{n} p_j(1-p_j)\right]^{1/2}, \tag{11}$$
$$\gamma = [\mathrm{Var}(N)]^{-3/2}\,\mathbf{E}\,[N-\mu]^3 = \sigma^{-3}\sum_{j=1}^{n} p_j(1-p_j)(1-2p_j),$$

respectively.

3.1 Poisson Approximation

In literature, the Poisson distribution has been used to approximate the distribution of N, which is referred to as the Poisson approximation (PA) method. In particular, the pmf the Poisson binomial distribution ξ_k is approximated by

$$\xi_k \approx \frac{\mu^k \exp(-\mu)}{k!}, k = 0, 1, \cdots, n \tag{12}$$

where μ is defined in (11). By Le Cam's theorem (Le Cam 1960), the approximation error for the PA method is $\sum_{k=0}^{n} \left|\xi_k - \mu^k \exp(-\mu)/(k!)\right| < 2\sum_{j=1}^{n} p_j^2$. Thus the PA method only works well when the expected number of successes μ is small.

3.2 Normal Approximation

The normal approximation (NA) method is based on the central limit theorem. In particular, the NA method with continuous correction approximates the cdf of a Poisson binomial distribution by

$$F_N(k) \approx \Phi\left(\frac{k + 0.5 - \mu}{\sigma}\right), k = 0, 1, \cdots, n \tag{13}$$

where $\Phi(x)$ is the cdf of the standard normal distribution, and μ and σ are defined in (11).

3.3 Refined Normal Approximation

Volkova (1996) described a refined normal approximation (RNA) which makes a correction to the skewness of the distribution of N. For the RNA method, the cdf $F_N(k)$ is approximated by

$$F_N(k) \approx G\left(\frac{k+0.5-\mu}{\sigma}\right), k = 0, 1, \cdots, n \tag{14}$$

where $G(x) = \Phi(x) + \gamma(1-x^2)\phi(x)/6$, $\phi(x)$ is the pdf of the standard normal distribution, and γ is defined in (11). The values of the cdf approximated by the RNA method are not always within $[0, 1]$. Thus those values less than 0 are corrected to 0 and those values larger than 1 are corrected to 1.

4 Numerical Studies

4.1 Accuracy of Exact Methods

In this section, we use the distribution function of the binomial distribution to verify the accuracy of algorithms using the DFT-CF, RF1 and RF2 methods. The pmf of a binomial distribution can be easily computed by using existing software. Note that the distribution of the sum of three independent and non-identical binomial distributions is a special case of the Poisson binomial distribution. The pmf in this special case is

$$\xi_k = \sum_{j=0}^{k} b_{k-j,n_3}\left(\sum_{i=0}^{j} b_{i,n_1} b_{j-i,n_2}\right) = \sum_{j=0}^{k}\sum_{i=0}^{j} b_{i,n_1} b_{j-i,n_2} b_{k-j,n_3} \tag{15}$$

where $n_1+n_2+n_3 = n$, and b_{i,n_1}, b_{i,n_2} and b_{i,n_3} are the pmfs of Binomial(n_1, p_1), Binomial(n_2, p_2) and Binomial(n_3, p_3), respectively. Here p_1, p_2 and p_3 are the success probabilities for these three binomial distributions. The pmfs b_{i,n_1}, b_{i,n_2} and b_{i,n_3} can be accurately computed by using existing software. With different values of n_1, n_2, n_3 and p_1, p_2, p_3, one can obtain various Poisson binomial distributions. We use the total absolute error (TAE) between two cdfs as a metric for accuracy comparisons. The TAE is defined by

$$\text{TAE} = \sum_{k=0}^{n} |F(k) - F_{\text{bin}}(k)|$$

where $F(k)$ is a cdf computed by using one of the exact methods, and $F_{\text{bin}}(k)$ is the cdf computed by using the formula in (15). Table 1 shows the results from the accuracy study for the DFT-CF, RF1 and RF2 methods. Various values of n_1, n_2, n_3 and p_1, p_2, p_3 were chosen to generate different scenarios and the TAE for each scenario was computed. The TAEs

Table 1: Accuracy comparisons for the DFT-CF, RF1 and RF2 methods.

n	n_1	n_2	n_3	p_1	p_2	p_3	TAE		
							DFT-CF	RF1	RF2
30	10	10	10	0.500	0.500	0.500	1.6×10^{-14}	7.4×10^{-15}	7.4×10^{-15}
30	10	5	15	0.500	0.500	0.500	1.3×10^{-14}	5.2×10^{-15}	5.2×10^{-15}
30	10	5	15	0.010	0.500	0.990	1.4×10^{-14}	7.0×10^{-16}	na
300	100	50	150	0.010	0.500	0.990	1.9×10^{-12}	4.7×10^{-14}	na
3,000	1,000	500	1,500	0.010	0.500	0.990	3.6×10^{-10}	1.1×10^{-11}	na
3,000	1,000	500	1,500	0.001	0.010	0.020	3.1×10^{-11}	9.4×10^{-11}	1.6×10^{-10}
3,000	1,000	500	1,500	0.999	0.990	0.980	1.4×10^{-09}	1.1×10^{-14}	na
3,000	1,000	500	1,500	0.001	0.500	0.999	3.4×10^{-10}	7.2×10^{-12}	na
3,000	1,000	500	1,500	0.300	0.500	0.700	3.8×10^{-10}	7.7×10^{-11}	na

are generally less than 1×10^{-10} for the DFT-CF and RF1 methods. Thus the results show that the DFT-CF and RF1 methods can accurately compute the cdf for the Poisson binomial distribution. The RF2 method does not work for most cases because the algorithm is not numerically stable.

4.2 Accuracy Comparisons for Approximation Methods

Being able to compute the exact values of the cdf $F_N(k)$ allows us to study the performance of approximation methods. To see the performance of different approximation methods, we simulate success probabilities p_j's from various patterns. Figure 1 shows the six different patterns in p_j's used in this numerical study. These patterns in the p_j's are simulated from the uniform distribution, beta distribution with various values of scale and shape parameters, and mixtures of beta distributions. For each pattern, various values of n, which is the number of random indicators in N, were chosen to see the effect of n on the accuracy of approximation methods. In particular, the values of n were chosen from $n = 10, 20, 50, 100, 200, 500, 1{,}000, 2{,}000, 5{,}000, 10{,}000, 15{,}000$.

Figure 2 shows an illustration of computed cdfs from various methods for Pattern (b) in Figure 1 when $n = 200$. The x-axis is on the logarithm scale and the y-axis is on the scale of the quantile function of the standard normal distribution. The RNA method approximates the cdf well and the NA method approximates the cdf moderate well (there are departures in the upper and lower tails of the cdf). The cdf computed by the PA method deviates from the true cdf. Thus the PA method does not perform well. The RF1 method gives exactly the same values (agree to the ninth decimal places) as the DFT-CF method. The RF2 method does not work because it is not numerically stable. Thus the results for recursive formulae are

not shown in Figure 2.

Table 2 shows the accuracy comparison for different methods under various patterns in p_j's and values of n. The TAE is used as a metric for comparisons, which is computed by

$$\text{TAE} = \sum_{k=0}^{n} |F(k) - F_N(k)|$$

where $F(k)$ is a cdf computed by using one of the approximation methods, and $F_N(k)$ is the cdf computed by using the DFT-CF method. As we can see from the results in Table 2, the PA method does not perform well for most cases. The PA method only works reasonable well when μ is small, for example in Pattern (b) when $n \leq 50$.

For the normal approximation methods, the RNA method performs better than the NA method for almost all cases. For Patterns (b) and (c) where N is highly skewed, the RNA method performs much better than the NA method. When $n \geq 2000$, the TAE for the RNA method is generally less than 0.005. Thus the RNA method is recommended when an approximation method needs to be used.

For all combinations of patterns in the p_j's and values of n considered in Table 2, both the DFT-CF and RF1 methods provide results that agree to the ninth decimal places. The RF2 method, however, does not work for most cases for the same reason mentioned previously. Thus the results for the RF1 and RF2 methods are not shown in Table 2.

4.3 Efficiency Comparisons for Exact Methods

The computing time for the exact and approximation methods needs to be considered when n is large. Table 3 gives the time for computing the entire cdf using the RNA, DFT-CF and RF1 methods for all combinations of patterns in the p_j's and values of n as in Section 4.2. The unit of time is the second. The computations were done by using the 64-bit R in a workstation. The workstation has an Intel Xeon CPU (X5570, 2.93GHz) and 24G RAM installed.

The results in Table 3 show that the computing time for the RNA method is generally negligible (less than four milliseconds). The computing time for both the DFT-CF and RF1 methods are generally negligible (less than ten milliseconds) when $n \leq 500$. When $n \geq 1{,}000$, the RF1 method requires more computing time than the DFT-CF method. The RF1 method also requires more RAM. For example, when $n = 15{,}000$, approximately 4GB memory is needed for computing the entire cdf. The DFT-CF method, however, is less demanding in memory. Thus the DFT-CF method is recommended for computing the exact values for the cdf $F_N(k)$, especially when n is large.

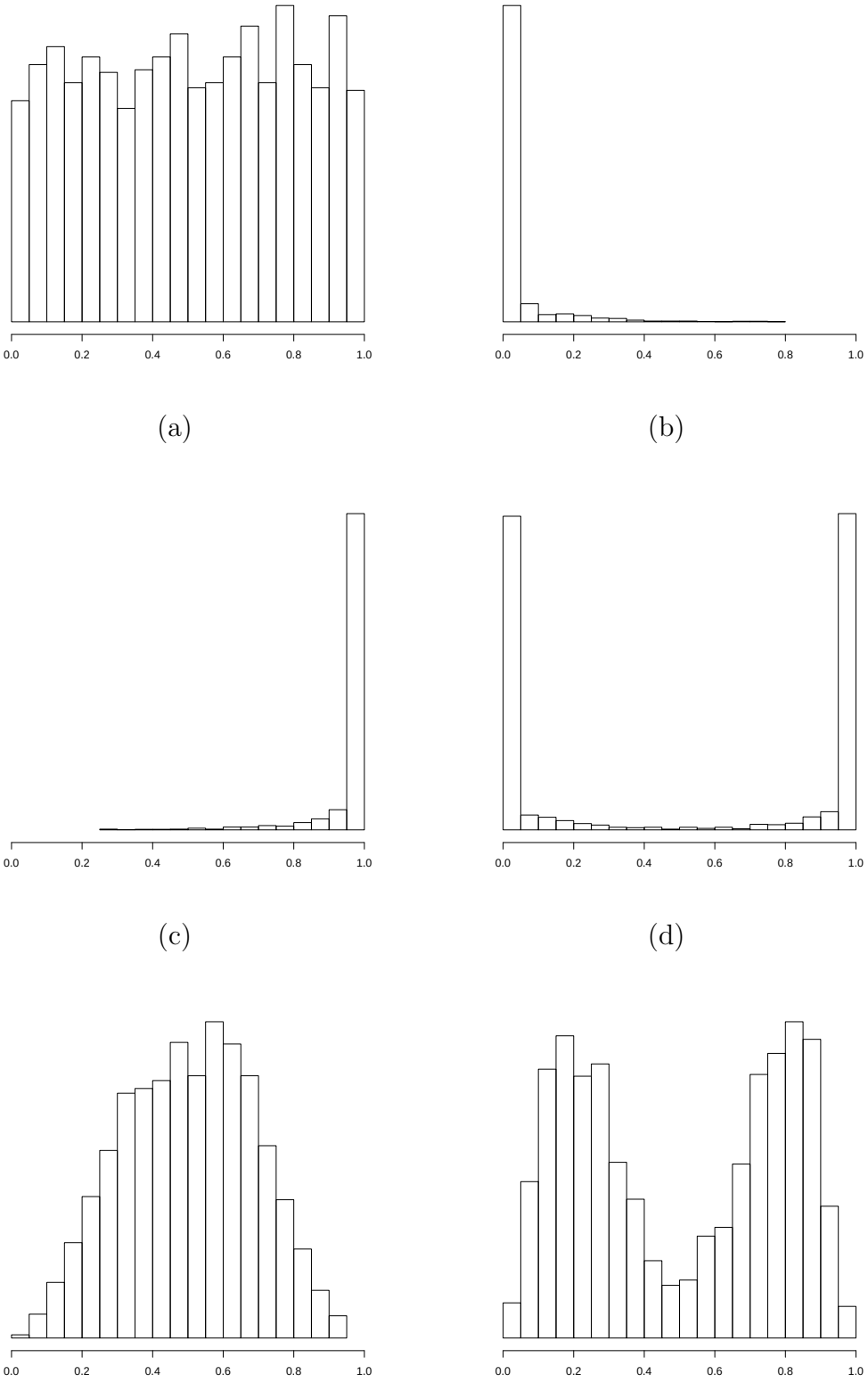

Figure 1: Six different patterns in the p_j's used in the numerical study.

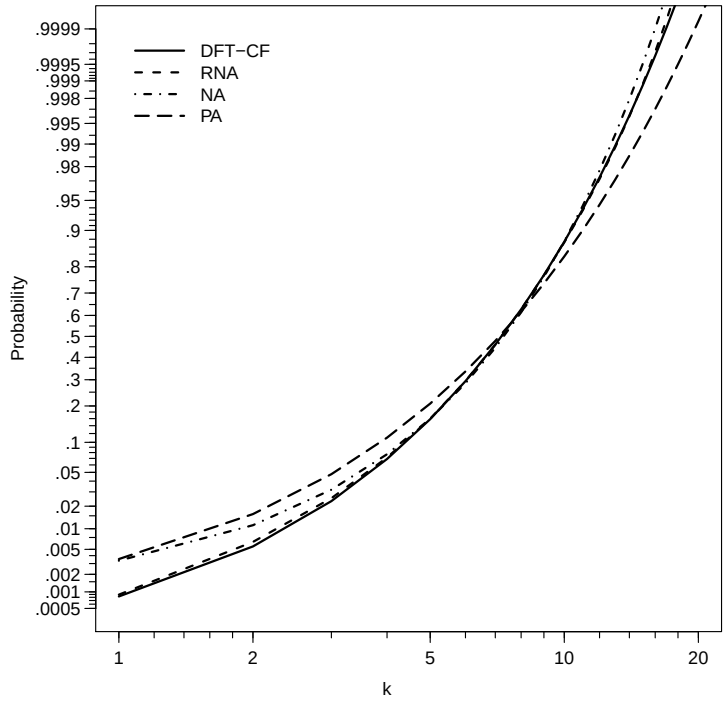

Figure 2: An illustration of computed cdfs with various methods for Pattern (b) in Figure 1 when $n = 200$. The x-axis is on the logarithm scale and the y-axis is on the scale of the quantile function of the standard normal distribution.

Table 2: Accuracy comparisons for approximation methods.

Pattern	(a)			(b)		
Method	RNA	NA	PA	RNA	NA	PA
$n = 10$	0.0286	0.0291	0.7287	0.0234	0.0600	0.1113
$n = 20$	0.0143	0.0207	0.8473	0.0085	0.0876	0.0124
$n = 50$	0.0082	0.0109	1.5700	0.0150	0.0782	0.0915
$n = 100$	0.0063	0.0105	2.6048	0.0193	0.0696	0.2345
$n = 200$	0.0046	0.0054	3.5082	0.0142	0.0801	0.3909
$n = 500$	0.0029	0.0039	5.4100	0.0100	0.0982	0.3839
$n = 1{,}000$	0.0021	0.0021	7.4680	0.0067	0.0945	0.5755
$n = 2{,}000$	0.0015	0.0024	10.819	0.0046	0.0947	0.8741
$n = 5{,}000$	0.0009	0.0012	17.081	0.0027	0.0891	1.5384
$n = 10{,}000$	0.0006	0.0011	23.628	0.0020	0.0916	2.0953
$n = 15{,}000$	0.0005	0.0006	29.029	0.0016	0.0926	2.4991
Pattern	(c)			(d)		
Method	RNA	NA	PA	RNA	NA	PA
$n = 10$	0.0597	0.1743	1.3854	0.0435	0.0604	1.6163
$n = 20$	0.0493	0.1261	1.8745	0.0352	0.0333	1.8703
$n = 50$	0.0212	0.0665	3.2784	0.0359	0.0401	3.0619
$n = 100$	0.0192	0.0900	4.4739	0.0287	0.0291	4.5203
$n = 200$	0.0116	0.0839	6.9814	0.0167	0.0167	6.1238
$n = 500$	0.0087	0.0884	12.132	0.0118	0.0123	9.9468
$n = 1{,}000$	0.0060	0.0807	18.970	0.0079	0.0085	13.922
$n = 2{,}000$	0.0044	0.0899	28.309	0.0056	0.0059	19.769
$n = 5{,}000$	0.0028	0.0906	46.528	0.0036	0.0046	31.307
$n = 10{,}000$	0.0020	0.0924	66.407	0.0025	0.0025	44.199
$n = 15{,}000$	0.0016	0.0912	80.915	0.0021	0.0021	54.089
Pattern	(e)			(f)		
Method	RNA	NA	PA	RNA	NA	PA
$n = 10$	0.0176	0.0191	0.7665	0.0261	0.0262	0.7921
$n = 20$	0.0100	0.0104	0.8061	0.0172	0.0181	1.0737
$n = 50$	0.0071	0.0095	1.3319	0.0103	0.0114	1.6842
$n = 100$	0.0048	0.0055	1.8916	0.0078	0.0079	2.3522
$n = 200$	0.0032	0.0039	2.6391	0.0053	0.0055	3.3915
$n = 500$	0.0021	0.0031	4.3815	0.0031	0.0031	5.1914
$n = 1000$	0.0016	0.0029	6.4646	0.0023	0.0023	7.5615
$n = 2{,}000$	0.0011	0.0015	8.7089	0.0016	0.0017	10.750
$n = 5{,}000$	0.0007	0.0007	13.683	0.0010	0.0011	16.876
$n = 10{,}000$	0.0005	0.0007	19.598	0.0007	0.0008	24.007
$n = 15{,}000$	0.0004	0.0008	23.686	0.0006	0.0007	29.387

Table 3: Computational efficiency comparisons for the RNA, DFT-CF and RF1 Methods. The unit of time is the second.

Pattern	(a)			(b)		
Method	RNA	DFT-CF	RF1	RNA	DFT-CF	RF1
$n = 10$	0.000	0.000	0.000	0.000	0.000	0.000
$n = 20$	0.000	0.000	0.000	0.000	0.000	0.000
$n = 50$	0.000	0.000	0.000	0.000	0.000	0.000
$n = 100$	0.000	0.000	0.000	0.000	0.000	0.000
$n = 200$	0.000	0.000	0.001	0.000	0.000	0.002
$n = 500$	0.000	0.006	0.006	0.000	0.004	0.005
$n = 1{,}000$	0.000	0.026	0.050	0.000	0.020	0.023
$n = 2{,}000$	0.000	0.106	0.134	0.000	0.081	0.131
$n = 5{,}000$	0.000	0.672	0.811	0.002	0.511	0.796
$n = 10{,}000$	0.003	2.661	3.354	0.001	2.018	3.305
$n = 15{,}000$	0.003	6.033	7.721	0.003	4.557	7.619
Pattern	(c)			(d)		
Method	RNA	DFT-CF	RF1	RNA	DFT-CF	RF1
$n = 10$	0.000	0.000	0.000	0.000	0.000	0.000
$n = 20$	0.000	0.000	0.000	0.000	0.000	0.000
$n = 50$	0.000	0.000	0.000	0.000	0.000	0.000
$n = 100$	0.000	0.000	0.000	0.000	0.000	0.000
$n = 200$	0.000	0.002	0.000	0.000	0.000	0.002
$n = 500$	0.000	0.006	0.005	0.000	0.006	0.006
$n = 1{,}000$	0.000	0.025	0.024	0.000	0.021	0.022
$n = 2{,}000$	0.000	0.097	0.131	0.000	0.087	0.132
$n = 5{,}000$	0.001	0.602	0.794	0.000	0.555	0.798
$n = 10{,}000$	0.002	2.380	3.304	0.002	2.201	3.321
$n = 15{,}000$	0.003	5.382	7.624	0.003	4.974	7.671
Pattern	(e)			(f)		
Method	RNA	DFT-CF	RF1	RNA	DFT-CF	RF1
$n = 10$	0.000	0.000	0.000	0.000	0.000	0.000
$n = 20$	0.000	0.000	0.000	0.000	0.000	0.000
$n = 50$	0.000	0.002	0.000	0.000	0.000	0.000
$n = 100$	0.000	0.000	0.000	0.000	0.001	0.000
$n = 200$	0.000	0.001	0.000	0.000	0.002	0.001
$n = 500$	0.000	0.008	0.005	0.000	0.007	0.005
$n = 1000$	0.000	0.026	0.022	0.000	0.025	0.024
$n = 2{,}000$	0.001	0.108	0.132	0.001	0.100	0.139
$n = 5{,}000$	0.002	0.672	0.811	0.002	0.637	0.863
$n = 10{,}000$	0.002	2.666	3.365	0.001	2.521	3.628
$n = 15{,}000$	0.003	6.017	7.736	0.003	5.685	8.483

5 Software Implementation

The DFT-CF, RF1, RNA and NA methods have been implemented in R. The computationally intensive components such as the FFT are implemented in C and are linked to R. The R functions have been wrapped into an R package `poibin` which can be downloaded from the Comprehensive R Archive Network (http://cran.r-project.org/). The R function in the package for computing the cdf $F_N(k)$ is `ppoibin()`, which has an option that allows users to specify the method to be used for computing.

6 Concluding Remarks

In this paper, we focus on the computing of the distribution function for the Poisson binomial distribution. We present a simple derivation for an exact formula with a closed-form expression. We develop an algorithm for efficient implementation of the exact formula and review the advantages and disadvantages of various approximation methods. Numerical studies were conducted to study the accuracy of the exact and approximation methods. The DFT-CF, RF1, RNA and NA methods have been implemented in an R package.

In practice, the DFT-CF method is generally recommended for computing. The RF1 method can also been used when $n < 1{,}000$ because there is not much difference in computing time from the DFT-CF method. When $n > 2{,}000$ and the cdf needs to be evaluated many times, the RNA method is recommended. This is because the RNA method can approximate the cdf well when n is large and is more computationally efficient, as shown in the numerical study.

Acknowledgement

The author would like to thank William Q. Meeker from Iowa State University and Qingyu Yang from Wayne State University for their valuable comments on an earlier version of the paper.

References

Athreya, K. B. and S. N. Lahiri (2006). *Measure Theory and Probability Theory*. New York: Springer.

Barlow, R. E. and K. D. Heidtmann (1984). Computing k-out-of-n system reliability. *IEEE Transactions on Reliability 33*, 322–323.

Bracewell, R. (2000). *The Fourier Transform & Its Applications* (third ed.). Singapore: McGraw-Hill, Inc.

Chen, S. X., A. P. Dempster, and J. S. Liu (1994). Weighted finite population sampling to maximize entropy. *Biometrika 81*, 457–469.

Chen, S. X. and J. S. Liu (1997). Statistical applications of the Poisson-binomial and conditional Bernoulli distributions. *Statistica Sinica 7*, 875–892.

Cooley, J. W. and J. W. Tukey (1965). An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation 19*, 297–301.

Duffie, D., L. Saita, and K. Wang (2007). Multi-period corporate default prediction with stochastic covariates. *Journal of Financial Economics 83*, 635–665.

Fernández, M. and S. Williams (2010). Closed-form expression for the Poisson-binomial probability density function. *IEEE Transactions on Aerospace Electronic Systems 46*, 803–817.

Hong, Y. and W. Q. Meeker (2010). Field-failure and warranty prediction based on auxiliary use-rate information. *Technometrics 52*, 148–159.

Hong, Y., W. Q. Meeker, and J. D. McCalley (2009). Prediction of remaining life of power transformers based on left truncated and right censored lifetime data. *The Annals of Applied Statistics 3*, 857–879.

Kuo, W. and M. Zuo (2003). *Optimal Reliability Modeling: Principles and Applications*. Hoboken, NJ: John Wiley & Sons, Inc.

Le Cam, L. (1960). An approximation theorem for the Poisson binomial distribution. *Pacific Journal of Mathematics 10*, 1181–1197.

R (2011). *The R Project for Statistical Computing*. http://www.r-project.org/.

Singleton, R. C. (1969). An algorithm for computing the mixed radix fast Fourier transform. *IEEE Transactions on Audio and Electroacoustics AU-17*, 93–103.

Volkova, A. Y. (1996). A refinement of the central limit theorem for sums of independent random indicators. *Theory of Probability and its Applications 40*, 791–794.

Wang, Y. H. (1993). On the number of successes in independent trials. *Statistica Sinica 3*, 295–312.