

# Computing the Joint Distribution of General Linear Combinations of Spacings or Exponential Variates

Fred W. Huffer  
Department of Statistics  
Florida State University  
Tallahassee, FL 32306  
huffer@stat.fsu.edu

Chien-Tai Lin  
Department of Mathematics  
Tamkang University  
Tamsui, Taiwan 251  
chien@math.tku.edu.tw

## Abstract

We present an algorithm for computing exact expressions for the distribution of the maximum or minimum of an arbitrary finite collection of linear combinations of spacings or exponential random variables with rational coefficients. These expressions can then be manipulated or evaluated using symbolic math packages such as MAPLE. As examples, we apply this algorithm to obtain the distributions of the maximum and minimum of a moving average process, and the distribution of the Kolmogorov-Smirnov statistic.

*Key words and phrases:* Kolmogorov-Smirnov statistic, moving average process, symbolic computations.

## 1 Introduction

Let  $\mathbf{S}^{(n)}$  denote the vector of spacings between  $n$  random points on the interval  $(0, 1)$ . More precisely, suppose that  $X_1, X_2, \dots, X_n$  are i.i.d. from a uniform distribution on the interval  $(0, 1)$ , and let  $X_{(1)} \leq X_{(2)} \leq \dots \leq X_{(n)}$  be the corresponding order statistics. We define the spacings  $S_1, S_2, \dots, S_{n+1}$  to be the successive differences between the order statistics  $S_i = X_{(i)} - X_{(i-1)}$ , where we take  $X_{(0)} = 0$  and  $X_{(n+1)} = 1$ . Finally, we define  $\mathbf{S}^{(n)} = (S_1, S_2, \dots, S_{n+1})'$ . This paper is concerned with the evaluation of probabilities involving linear combinations of spacings with arbitrary rational coefficients. We present an algorithm for evaluating

$$P(\mathbf{A}\mathbf{S}^{(n)} > t\mathbf{b}) \tag{1}$$

where  $\mathbf{A}$  is any matrix of rational values,  $\mathbf{b}$  is any vector of rational values, and  $t > 0$  is a real-valued scalar. (Note that, for vectors  $\mathbf{x} = (x_i)$  and  $\mathbf{y} = (y_i)$ , we define  $\mathbf{x} > \mathbf{y}$  to mean that  $x_i > y_i$  for all  $i$ .) This algorithm produces an exact expression for the probability in (1) which is a piecewise polynomial in the argument  $t$ . With this expression, we can use symbolic math packages such as MAPLE to evaluate (1) to any required degree of precision. The computer programs we have written are also convenient for evaluating quantities which can be expressed as sums of probabilities of the form (1).

Our methods and programs can also be used for computations involving linear combinations of exponential random variables. In fact, by simply re-interpreting the symbols, the expression we obtain for (1) is also valid for the corresponding problem with exponential variates obtained by replacing  $\mathbf{S}^{(n)}$  by a vector  $\mathbf{Z}$  of i.i.d. exponentials.

Our approach in this paper has much in common with our earlier work reported in Huffer and Lin (1997a, 1999a). However, in this earlier work, our methods were very limited in the matrices  $\mathbf{A}$  and vectors  $\mathbf{b}$  they could handle;  $\mathbf{A}$  had to belong to a special class of binary matrices, and  $\mathbf{b}$  had to be a vector of ones. The method in Lin (1993) can compute (1) for some, but not all, rational matrices  $\mathbf{A}$ . Unfortunately, we are unable to characterize the problems that can be solved by this method, and thus cannot tell in advance whether a given problem is solvable. With our new algorithm, we can now handle arbitrary rational  $\mathbf{A}$  and  $\mathbf{b}$ . This permits us to solve a much greater variety of problems.

The approach we use to evaluate (1) depends on the repeated, systematic use of two basic recursions given later in equations (17) and (18). Each of these recursions is used to re-express a probability like that in (1) by decomposing it into a sum of similar, but simpler components. The same recursions are then applied to each of these components and so on. The process is continued until we obtain components which are simple and easily expressed in closed form.

This paper is organized as follows. In Section 2 we present two examples to illustrate our methods. Section 3 gives some definitions and results we use in our algorithm. Section 4 contains a detailed description of the algorithm.

## 2 Examples

Before proceeding, we establish one notational convention. It is convenient to regard the probability in (1) as being defined even when the number of columns in  $\mathbf{A}$  is less than  $n + 1$ , the number of entries in  $\mathbf{S}^{(n)}$ . Let  $k$  be the number of columns in  $\mathbf{A}$ . If  $k < n + 1$ , then in computing  $\mathbf{A}\mathbf{S}^{(n)}$  we simply discard the extra entries of  $\mathbf{S}^{(n)}$ , or equivalently, we pad the matrix  $\mathbf{A}$  with extra columns of zeros and define

$$\mathbf{A}\mathbf{S}^{(n)} = (\mathbf{A} \mid \mathbf{0})\mathbf{S}^{(n)}. \quad (2)$$

Our expressions for (1) are written in terms of a function  $R(j, \lambda)$  defined for integers  $j \geq 0$  and real values  $\lambda \geq 0$  by

$$R(j, \lambda) = \begin{cases} \binom{n}{j} t^j (1 - \lambda t)^{n-j} & \text{for } \lambda t < 1, \\ 0 & \text{for } \lambda t \geq 1. \end{cases} \quad (3)$$

The dependence of  $R$  on  $n$  and  $t$  can be left implicit because these values are fixed in any given application of our methods. If we replace  $\mathbf{S}^{(n)}$  in (1) by a vector  $\mathbf{Z}$  of i.i.d. exponential random variables with mean 1, then our expressions remain valid so long as we redefine  $R$  to be

$$R(j, \lambda) = \frac{t^j}{j!} e^{-\lambda t}. \quad (4)$$

**Example 1:** For our first example, we use our methods to compute the probability of the event

$$\bigcap_{i=0}^3 \{S_{i+1} + 2S_{i+2} + 3S_{i+3} + 2S_{i+4} + S_{i+5} > t\}. \quad (5)$$

This problem has the form in (1) with  $\mathbf{A}$  and  $\mathbf{b}$  given by

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 2 & 3 & 2 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}. \quad (6)$$

The probability of (5) as a function of  $t$  gives the distribution of the minimum (call it  $L$ ) of a particular finite moving average of spacings (or exponential random variables). Moving averages of i.i.d. exponential random variables occur when considering the null distribution of spectral estimates in time series. See, for example, Theorem 6.1.1 and formula 7.6.18 in Priestley (1981).

Our algorithm leads to the following expression for the probability of (5).

$$\begin{aligned}
& -17415/64 R(0, 2/3) - 243/8 R(1, 2/3) + 27/4 R(2, 2/3) + 5120/3 R(0, 3/4) \\
& -21875/24 R(0, 4/5) - 1944/1 R(0, 5/6) + 823543/576 R(0, 6/7) - 40/3 R(0, 1/1) \\
& -17/12 R(1, 1/1) - 11/12 R(2, 1/1) + 3125/576 R(0, 6/5) + 3/64 R(0, 2/1) \quad (7)
\end{aligned}$$

This expression is easy to manipulate and evaluate using symbolic math packages such as MAPLE. For example, it is easy to study the cdf and density of  $L$  by plotting (7) and its derivative as functions of  $t$ . It is also routine to use (7) to obtain the exact moments of  $L$ .

A closely related problem is to calculate the probability of the event

$$\bigcap_{i=0}^3 \{S_{i+1} + 2S_{i+2} + 3S_{i+3} + 2S_{i+4} + S_{i+5} \leq t\}. \quad (8)$$

As a function of  $t$ , this probability gives the distribution of the maximum of the moving average process.

Since this problem involves “ $\leq$ ” instead of “ $>$ ”, it does not quite fit the form in (1). However, it is easily converted into this form in either of two ways. The first is to note that the event

$$\mathbf{A}\mathbf{S}^{(n)} \leq t\mathbf{b} \quad \text{is equivalent to} \quad (-\mathbf{A})\mathbf{S}^{(n)} \geq t(-\mathbf{b}) \quad (9)$$

which does have the required form (except for the unimportant difference between “ $>$ ” and “ $\geq$ ”). The second is to use an inclusion-exclusion argument to write

$$P(\mathbf{A}\mathbf{S}^{(n)} \leq t\mathbf{b}) = 1 + \sum_{\pi} (-1)^{\#(\pi)} P(\mathbf{A}_{[\pi]}\mathbf{S}^{(n)} > t\mathbf{b}). \quad (10)$$

Here, the sum is taken over all non-empty subsets  $\pi$  of  $\{1, \dots, r\}$  where  $r$  is the number of rows of  $\mathbf{A}$ . We define  $\#(\pi)$  to be the number of elements in  $\pi$ , and use  $\mathbf{A}_{[\pi]}$  to denote the matrix consisting of the given subset of the rows of  $\mathbf{A}$ . Now the terms on the right hand side have the required form.

Either method must lead to the same answer. The second method turns out to require less computational effort in this case. Using (10), we express the probability of the event (8) as a sum of 15 terms and evaluate each of these terms using our algorithm. This leads to the result

$$1 - 81/1 R(0, 1/3) + 9375/64 R(0, 2/5) - 2696/27 R(0, 1/2) + 4/1 R(1, 1/2)$$

$$\begin{aligned}
& +1701/20 R(0, 2/3) + 40625/72 R(0, 4/5) - 3125/72 R(1, 4/5) - 648/1 R(0, 5/6) \\
& +823543/576 R(0, 6/7) - 65536/45 R(0, 7/8) + 319/6 R(0, 1/1) + 1/3 R(1, 1/1) \\
& -3125/288 R(0, 6/5) + 512/27 R(0, 5/4) - 8/3 R(0, 3/2) + 1/16 R(0, 2/1) \quad (11)
\end{aligned}$$

**Example 2:** For our second example, we use our algorithm to compute the distribution of the Kolmogorov-Smirnov (K-S) statistic  $D_n = \sup_x |F_n(x) - F(x)|$  where  $F_n$  is the empirical cdf of  $n$  i.i.d. observations from a continuous distribution  $F$ . The distribution of  $D_n$  does not depend on  $F$ , so we can assume that  $F$  is the uniform distribution on  $(0, 1)$ .

We may express  $D_n$  in terms of the spacings as  $D_n = \max(D_n^+, D_n^-)$  where

$$\begin{aligned}
D_n^+ &= \sup_x (F_n(x) - x) = 0 \vee \max_{1 \leq k \leq n} \left( \frac{k}{n} - (S_1 + \cdots + S_k) \right) \quad \text{and} \\
D_n^- &= \sup_x (x - F_n(x)) = 0 \vee \max_{1 \leq k \leq n} \left( (S_1 + \cdots + S_k) - \frac{k-1}{n} \right). \quad (12)
\end{aligned}$$

Thus, for any integer  $i$  from 1 to  $n-1$ , we have

$$\{D_n < i/n\} = \{D_n^+ < i/n\} \cap \{D_n^- < i/n\}$$

where

$$\begin{aligned}
\{D_n^+ < i/n\} &= \bigcap_{i+1 \leq k \leq n} \{S_1 + \cdots + S_k > (k-i)/n\} \quad \text{and} \\
\{D_n^- < i/n\} &= \bigcap_{1 \leq k \leq n-i} \{S_1 + \cdots + S_k < (k+i-1)/n\}. \quad (13)
\end{aligned}$$

Now, after switching the direction of the inequalities in (13) as in equation (9), we obtain a problem in the form (1) which can be solved by our algorithm. Note that for this problem we do not need the variable  $t$  in equation (1), that is, we set  $t = 1$ .

Here are some numerical results obtained using our algorithm.

$$\begin{aligned}
P(D_{14} < 3/14) &= \frac{14831925552873}{28346956187648} \approx .52322815383388 \\
P(D_{14} < 5/14) &= \frac{2661531270146463}{2778001706389504} \approx .95807402278582 \quad (14)
\end{aligned}$$

The algorithm actually produces expressions like those in (7) and (11). But, in this example, these expressions have no useful interpretations as functions of  $t$ , so we have simply evaluated them at  $t = 1$ .

It is, of course, well known how to compute the exact distribution of  $D_n$ , at least when  $n$  is not too large. For instance, the original approach of Kolmogorov (see Birnbaum (1952)) can also be used to obtain the results in (14). Our algorithm has no advantage over Kolmogorov's approach in this particular problem. The advantage of our method is its flexibility. For example, a popular variant of the K-S statistic uses  $k/(n+1)$  in place of  $k/n$  and  $(k-1)/n$  in (12). Our method can handle this statistic by making the obvious minor changes to the setup above. As another example, if the larger values of  $X_i$  are censored, then one may wish to define a K-S type statistic based only on the lower order statistics  $X_{(1)}, \dots, X_{(m)}$  where  $m < n$ . Such statistics are also easily handled by our approach.

**Other Applications:** We briefly mention some other applications of our algorithm. First, our approach can be used to evaluate the joint distribution of statistics which can be expressed as linear combinations of spacings or exponential random variables. For example, we can obtain the joint distribution of the sample mean and sample median for random samples from the uniform distribution. Secondly, we can evaluate various probabilities and moments related to the number of clumps or gaps among randomly distributed points on an interval or circle. In particular, we can use our procedure to compute the distribution of the scan statistic on the interval or the circle. (See Glaz and Balakrishnan (1999) for a review of research on the scan statistic.) Huffer and Lin (1997b, 1999b) discuss these applications. Many of these problems can be solved using the earlier algorithm in Huffer and Lin (1997a), but not all. In particular, our earlier algorithm could not handle problems involving random points on a circle. See Example 3 of Huffer and Lin (1997a).

One potential application of our method is to the Bayesian bootstrap. Suppose  $\mathbf{A}$  is a  $(n+1) \times p$  matrix whose columns are an i.i.d. sample from an unknown  $p$ -variate distribution  $F$  with mean vector  $\mu$ . The Bayesian bootstrap distribution for  $\mu$ , which is a certain limit of posterior distributions, coincides with the distribution of  $\mathbf{A}\mathbf{S}^{(n)}$  (see Choudhuri (1998)). Similarly, the Bayesian bootstrap distribution for the lower moments of  $F$  is the same as that of  $\mathbf{A}\mathbf{S}^{(n)}$  with a different choice of the matrix  $\mathbf{A}$  (see Gasparini (1995)). Thus, our algorithm may prove to be of use in calculating characteristics of Bayesian bootstrap distributions.

### 3 Recursions and Elementary Properties

In this section we present the two recursions upon which our algorithm is based. We also list some elementary properties which are used in the course of the algorithm. The recursions (given in equations (17) and (18) below) are stated in terms of a function  $Q$  defined by

$$Q(\mathbf{A}, \mathbf{b}, \lambda, p) = p! R(p, \lambda) P\left((1 - \lambda t) \mathbf{A} \mathbf{S}^{(n-p)} > t \mathbf{b}\right). \quad (15)$$

We chose this particular definition of  $Q$  because it leads to a fairly simple form for the recursion (18) below. As in the definition of  $R$ , the dependence of  $Q$  on  $n$  and  $t$  is left implicit. The algorithm works by using the recursions to successively reduce the dimensionality of  $\mathbf{A}$  and  $\mathbf{b}$ . When the dimensions reach zero and both  $\mathbf{A}$  and  $\mathbf{b}$  are empty, we define

$$Q(\emptyset, \emptyset, \lambda, p) = p! R(p, \lambda). \quad (16)$$

We now present our first recursion. Let  $\mathbf{A}$  be an arbitrary matrix. Let  $r$  and  $q$  be the number of rows and columns of  $\mathbf{A}$ . For any  $r \times 1$  vector  $\mathbf{x}$ , we define  $A_{i,\mathbf{x}}$  to be the matrix obtained by replacing the  $i^{\text{th}}$  column of  $\mathbf{A}$  by  $\mathbf{x}$ . Let  $\mathbf{c} = (c_1, \dots, c_q)'$  be any  $q \times 1$  vector satisfying  $\sum_{i=1}^q c_i = 1$ . Define  $\boldsymbol{\xi} = \mathbf{A} \mathbf{c}$ . Then

$$Q(\mathbf{A}, \mathbf{b}, \lambda, p) = \sum_{i=1}^q c_i Q(\mathbf{A}_{i,\boldsymbol{\xi}}, \mathbf{b}, \lambda, p). \quad (17)$$

This recursion is an immediate consequence of the more general recursion given in Huffer (1988). See Huffer (1988), Lin (1993), and Huffer and Lin (1999a) for applications of this general recursion.

Our second recursion is the following. If a matrix  $\mathbf{A} = (a_{ij})$  and vector  $\mathbf{b} = (b_j)$  satisfy (for some  $k \geq 1$ ) the following three conditions:

- (R1)  $a_{1j} = 0$  for  $j > k$ ,
- (R2)  $a_{ij} = a_{i1}$  for  $j \leq k$ , (i.e., the first  $k$  columns of  $\mathbf{A}$  are identical),
- (R3)  $a_{11} > 0$  and  $b_1 > 0$ ,

then

$$Q(\mathbf{A}, \mathbf{b}, \lambda, p) = \sum_{i=0}^{k-1} \frac{\delta^i}{i!} Q(\mathbf{A}_{(-i)}^*, \mathbf{b}^* - \delta \mathbf{a}^*, \lambda + \delta, p + i), \quad (18)$$

where  $\delta = b_1/a_{11}$ ,  $\mathbf{A}^*$  is a matrix obtained by deleting the first row of  $\mathbf{A}$ ,  $\mathbf{A}_{(-i)}^*$  is a matrix obtained by deleting the first  $i$  columns of  $\mathbf{A}^*$ ,  $\mathbf{b}^*$  is a vector obtained by

deleting the first entry of  $\mathbf{b}$ , and  $\mathbf{a}^*$  is a vector obtained by taking the first column of  $\mathbf{A}$  and deleting the first entry. We give a proof of this recursion at the end of this section. Note that condition R2 is vacuous when  $k = 1$ . Under the stated conditions, recursion (18) allows us to reduce the dimension of  $\mathbf{A}$  (and  $\mathbf{b}$ ) by deleting one row.

In connection with (18), we also use the following simple recursion. If conditions R1 and R2 hold, but instead of R3 we have  $a_{11} < 0$  and  $b_1 < 0$ , then

$$Q(\mathbf{A}, \mathbf{b}, \lambda, p) = Q(\mathbf{A}^*, \mathbf{b}^*, \lambda, p) - Q(\mathbf{A}^\circ, \mathbf{b}^\circ, \lambda, p) \quad (19)$$

where  $\mathbf{A}^*$  and  $\mathbf{b}^*$  are as described above, and  $\mathbf{A}^\circ$  and  $\mathbf{b}^\circ$  are obtained by negating the first row of  $\mathbf{A}$  and the first entry of  $\mathbf{b}$  respectively. Note that  $\mathbf{A}^\circ$  and  $\mathbf{b}^\circ$  now satisfy R1–R3, so that (18) can now be applied to  $Q(\mathbf{A}^\circ, \mathbf{b}^\circ, \lambda, p)$ . The recursion (19) is merely a special case of the fact that  $P(C \cap D) = P(C) - P(C \cap D^c)$  for any events  $C$  and  $D$ .

We now list some other properties which are useful in the process of evaluating  $Q$ . These properties are straightforward and their proofs are omitted. For any permutation matrix  $\mathbf{G}$ ,

$$(E1) \quad Q(\mathbf{A}, \mathbf{b}, \lambda, p) = Q(\mathbf{A}\mathbf{G}, \mathbf{b}, \lambda, p),$$

$$(E2) \quad Q(\mathbf{A}, \mathbf{b}, \lambda, p) = Q(\mathbf{G}\mathbf{A}, \mathbf{G}\mathbf{b}, \lambda, p).$$

For any diagonal matrix  $\mathbf{D}$  with strictly positive entries on the diagonal

$$(E3) \quad Q(\mathbf{A}, \mathbf{b}, \lambda, p) = Q(\mathbf{D}\mathbf{A}, \mathbf{D}\mathbf{b}, \lambda, p).$$

Let  $(\mathbf{A}, \mathbf{b})$  denote the set of inequalities in (15). Property E1 follows from the exchangeability of the spacings. Property E2 reflects the fact that we can arbitrarily re-order the inequalities in  $(\mathbf{A}, \mathbf{b})$ . Property E3 ensures that we can always make the entries in  $\mathbf{b}$  to be  $\pm 1$  or 0. Note that property E1 allows us to delete any columns in  $\mathbf{A}$  which consist entirely of zeros; we permute the columns of  $\mathbf{A}$  so that the columns of zeros are at the end, and then delete them by using the convention in (2).

The value of  $Q$  remains the same when we delete redundant inequalities from  $(\mathbf{A}, \mathbf{b})$ . Also, if any inequalities in  $(\mathbf{A}, \mathbf{b})$  are contradictory, then  $Q(\mathbf{A}, \mathbf{b}, \lambda, p) = 0$ . In particular, we can delete the  $i^{\text{th}}$  row of  $\mathbf{A}$  and the  $i^{\text{th}}$  entry of  $\mathbf{b}$  without changing the value of  $Q$  whenever any of the following conditions are true:

$$(S1) \quad a_{ik} \geq a_{jk} \text{ for all } k \text{ and } b_i \leq b_j,$$

$$(S2) \quad a_{ik} \geq 0 \text{ for all } k \text{ (with } a_{ik} > 0 \text{ for some } k) \text{ and } b_i \leq 0,$$

$$(S3) \quad a_{ik} = 0 \text{ for all } k \text{ and } b_i < 0.$$

The value of  $Q$  is 0 whenever any of the following conditions are true:

(S4)  $a_{ik} + a_{jk} \leq 0$  for all  $k$  and  $b_i + b_j \geq 0$ ,

(S5)  $a_{ik} \leq 0$  for all  $k$  and  $b_i \geq 0$ .

Properties S3 and S5 allow us to eliminate terms  $Q$  in which  $\mathbf{A}$  has any rows consisting entirely of zeros.

The remainder of this section is devoted to a proof of the recursion in equation (18). Readers can skip this proof without loss of continuity.

The expression  $\mathbf{A}\mathbf{S}^{(n)} > t\mathbf{b}$  stands for a conjunction of inequalities. Under the assumptions R1–R3, the first of these inequalities is

$$D \equiv \{a_{11}(S_1 + \cdots + S_k) > b_1 t\} = \{X_{(k)} > \delta t\}.$$

(Recall that  $X_{(k)}$  is the  $k$ -th order statistic of our  $n$  random points.) Clearly  $D = \cup_{i=0}^{k-1} D_i$  where  $D_i$  is the event that exactly  $i$  of the random points fall in the interval  $(0, \delta t]$ , that is,  $D_i = \{X_{(i)} \leq \delta t < X_{(i+1)}\}$ . Thus

$$P(\mathbf{A}\mathbf{S}^{(n)} > t\mathbf{b}) = P(D \cap \{\mathbf{A}^*\mathbf{S}^{(n)} > t\mathbf{b}^*\}) = \sum_{i=0}^{k-1} P(D_i)P(\mathbf{A}^*\mathbf{S}^{(n)} > t\mathbf{b}^* | D_i). \quad (20)$$

Condition on the event  $D_i$  where  $i < k$ . Define the  $(n - i + 1)$ -dimensional vector  $\mathbf{T}^i$  by  $\mathbf{T}^i = (X_{(i+1)} - \delta t, S_{i+2}, \dots, S_{n+1})'$ . The vector  $\mathbf{T}^i$  gives the spacings between the  $n - i$  random points lying in the interval  $(\delta t, 1)$ . Under assumptions R1–R3, the inequalities in  $\mathbf{A}^*\mathbf{S}^{(n)} > t\mathbf{b}^*$  are

$$\begin{aligned} \left\{ \sum_{\ell=1}^{n+1} a_{j\ell} S_\ell > b_j t \right\} &= \left\{ a_{j1} X_{(i+1)} + \sum_{\ell=i+2}^{n+1} a_{j\ell} S_\ell > b_j t \right\} \\ &= \left\{ a_{j1} (X_{(i+1)} - \delta t) + \sum_{\ell=i+2}^{n+1} a_{j\ell} S_\ell > (b_j - \delta a_{j1}) t \right\} \\ &= \left\{ \sum_{\ell=i+1}^{n+1} a_{j\ell} T_{\ell-i}^i > (b_j - \delta a_{j1}) t \right\} \end{aligned}$$

for  $j \geq 2$ . Thus we have

$$P(\mathbf{A}^*\mathbf{S}^{(n)} > t\mathbf{b}^* | D_i) = P(\mathbf{A}_{(-i)}^* \mathbf{T}^i > t(\mathbf{b}^* - \delta \mathbf{a}^*) | D_i).$$

Now note that, conditional on  $D_i$ , the points in  $(\delta t, 1)$  are i.i.d. uniformly distributed on this interval. Thus

$$\mathcal{L}(\mathbf{T}^i | D_i) = \mathcal{L}((1 - \delta t)\mathbf{S}^{(n-i)})$$

which leads to

$$P(\mathbf{A}^* \mathbf{S}^{(n)} > t \mathbf{b}^* | D_i) = P\left((1 - \delta t) \mathbf{A}_{(-i)}^* \mathbf{S}^{(n-i)} > t(\mathbf{b}^* - \delta \mathbf{a}^*)\right).$$

Substituting this back in (20), we obtain

$$P(\mathbf{A} \mathbf{S}^{(n)} > t \mathbf{b}) = \sum_{i=0}^{k-1} \binom{n}{i} (\delta t)^i (1 - \delta t)_+^{n-i} P\left((1 - \delta t) \mathbf{A}_{(-i)}^* \mathbf{S}^{(n-i)} > t(\mathbf{b}^* - \delta \mathbf{a}^*)\right).$$

Here we use  $(x)_+ = \max(x, 0)$  to denote the positive part of  $x$ .

This last equation holds for all  $n$  and all matrices  $\mathbf{A}$ . If we replace  $n$  by  $n - p$  and  $\mathbf{A}$  by  $(1 - \lambda t) \mathbf{A}$ , and also multiply both sides of the equation by  $p! R(p, \lambda)$ , then we obtain (after a little algebra) the recursion (18). Note that, when replacing  $\mathbf{A}$  by  $(1 - \lambda t) \mathbf{A}$ , we must also replace the quantities  $\delta = b_1/a_{11}$  and  $\mathbf{a}^*$  (which depend on  $\mathbf{A}$ ) by  $\delta/(1 - \lambda t)$  and  $(1 - \lambda t) \mathbf{a}^*$  respectively.

When  $\mathbf{A}$  consists of a single row, then  $\mathbf{A}^*$  and  $\mathbf{b}^*$  are “empty”. The definition in (16) has been chosen so that the recursion (18) remains true even in this case.

## 4 The Algorithm

Our goal is to evaluate  $P(\mathbf{A} \mathbf{S}^{(n)} > t \mathbf{b}) = Q(\mathbf{A}, \mathbf{b}, 0, 0)$ . Our algorithm to do this consists of a series of steps. At each step one of the recursions (17) or (18) is used. In the course of the algorithm, each application of these recursions produces terms (on the right hand side of (17) or (18)) which are “simpler” than the parent term (on the left hand side). The net effect is to successively reduce the dimensionality of the  $\mathbf{A}$ -matrices in these terms until at last we arrive at terms which can be evaluated using (16) and (3).

In carrying out our algorithm, we assume that the matrix  $\mathbf{A}$  consists of blocks arranged in a lower triangular fashion, that is, the right upper region contains only zero entries. A typical example of such a matrix is given in Figure 1. In this figure, the letters  $a, b, c, \dots$ , denote distinct nonzero rational numbers. We say that such a matrix has been put in “standard form”. To be precise, a matrix  $\mathbf{A}$  is in standard form if, for some value of  $B \geq 1$ , we have the following:

- (i) The rows and columns of  $\mathbf{A}$  have each been partitioned into  $B$  groups, thus partitioning  $\mathbf{A}$  into a  $B \times B$  array of blocks. We refer to these blocks by their positions  $(\ell, m)$  in this array where  $1 \leq \ell, m \leq B$ .

$$\left( \begin{array}{cc|ccc|cc} a & a & 0 & 0 & 0 & 0 & 0 \\ b & b & c & d & d & 0 & 0 \\ 0 & 0 & e & e & f & 0 & 0 \\ \hline g & h & 0 & 0 & q & r & r \\ s & t & u & u & u & v & w \end{array} \right)$$

Figure 1: Example of a matrix in standard form.

- (ii) The blocks above the diagonal (with  $\ell < m$ ) consist entirely of zeros.
- (iii) The blocks along the diagonal (with  $\ell = m$ ) contain no zeros.

Any matrix  $\mathbf{A}$  can be arranged in standard form as follows. First, the matrix is simplified (or the corresponding term is deleted) by using properties S1–S5. In particular, any rows or columns consisting entirely of zeros are eliminated. Then the permutation properties E1 and E2 are employed to arrange the matrix into blocks satisfying (ii) and (iii). There are many ways to do this. In our algorithm, we try to make the region of zeros in the upper right as large as possible, but the success of our algorithm does not depend on our finding the best such arrangement. The approach in our current program is to use E2 to arrange the rows according to the number of nonzero entries in each row, the rows with the fewest nonzero entries being placed at the top of the matrix. Then we use E1 to move the zero entries to the rightmost columns of the matrix as much as possible.

Note that the blocks of a matrix in standard form can (and often do) consist of a single row or column, or even a single entry. Also, if a matrix contains no zero entries, then it is already in standard form (with  $B = 1$ ).

At the start of our algorithm, we always arrange the input matrix (or matrices)  $\mathbf{A}$  in standard form. As the algorithm proceeds, every time either recursion (17) or (18) is used, the  $\mathbf{A}$ -matrices of the resulting terms are put back into standard form.

To motivate the form of our algorithm, note that

$$\begin{array}{l} \text{we can always reduce the dimensionality of terms which} \\ \text{satisfy conditions R1 and R2.} \end{array} \tag{21}$$

Here are the various cases: If  $a_{11} > 0$  and  $b_1 > 0$ , then we can apply (18) to this term and thus reduce the number of rows. If  $a_{11} < 0$  and  $b_1 < 0$ , we can apply (19) followed by (18) to reduce the number of rows. If  $a_{11} > 0$  and  $b_1 \leq 0$ , then (using property S2) we can delete the first row of  $\mathbf{A}$  and the first entry of  $\mathbf{b}$  without changing the value of

the term. Finally, if  $a_{11} < 0$  and  $b_1 \geq 0$ , then (by property S5) the value of the term is zero.

The strategy we adopt in our algorithm is to use repeated applications of the recursion (17) to enlarge the region of zeros (the union of those blocks above the diagonal) and “drive” the terms closer to satisfying conditions R1 and R2. Then, when terms satisfy R1 and R2, we use recursion (18) (or as noted above) to reduce the dimensionality. Many different algorithms can be constructed using this basic strategy. We will present one such scheme below.

We first describe how the recursion (17) gets used in our algorithm. Suppose that  $\mathbf{D}$  is one of the blocks in the standard form for  $\mathbf{A}$ , and that the first row of  $\mathbf{D}$  contains distinct values  $\alpha$  and  $\beta$  in columns  $i$  and  $j$  of  $\mathbf{A}$ . To be precise, assume that  $\mathbf{D}$  is the  $(\ell, m)$  block and that it begins in row  $r$  of  $\mathbf{A}$ . Then  $\alpha = a_{ri}$  and  $\beta = a_{rj}$ . We can use (17) to simplify  $\mathbf{A}$  in either of the following two cases.

Case 1:  $\ell = m$ . Take the vector  $\mathbf{c}$  in (17) to have entries  $c_i = \beta/(\beta - \alpha)$ ,  $c_j = -\alpha/(\beta - \alpha)$ , and  $c_k = 0$  for  $k \neq i, j$ .

Case 2:  $\ell > m$  and  $a_{ki} = a_{kj}$  for  $k < r$ . Suppose the last nonzero entry in row  $r$  of  $\mathbf{A}$  occurs in column  $s$ , and the value of this last nonzero entry is  $\gamma$ . Take the vector  $\mathbf{c}$  in (17) to have entries  $c_i = \gamma/(\beta - \alpha)$ ,  $c_j = -\gamma/(\beta - \alpha)$ ,  $c_s = 1$ , and  $c_k = 0$  for  $k \neq i, j, s$ .

In either Case 1 or Case 2, the vector  $\boldsymbol{\xi} = \mathbf{A}\mathbf{c}$  will have zeros in entries 1 through  $r$ . Thus, if we use this vector in (17) and rearrange the resulting terms back into standard form, we obtain new terms in which the region of zeros is strictly larger than in the parent term.

We illustrate these cases using the example matrix in Figure 1. The  $(2, 2)$  block gives an instance of Case 1 with  $i = 3$ ,  $j = 4$ , and  $r = 2$ . The  $(3, 1)$  block gives an instance of Case 2 with  $i = 1$ ,  $j = 2$ , and  $r = 4$ .

Combining the ingredients given above, we can now give a complete description of our algorithm. To evaluate  $Q(\mathbf{A}, \mathbf{b}, \lambda, p)$ , do the following.

- (a) Search the nonzero blocks of  $\mathbf{A}$  and find the first block which contains two distinct values in the same row. If there are no such blocks, go to (b). The blocks may be searched in any order so long as a block is never searched until all the blocks lying directly above it have already been searched. (One possible search order is

1			
2	5		
3	6	8	
4	7	9	10

Figure 2: One possible ordering for searching the blocks.

that in Figure 2.) Using property E2, move the row containing the two distinct values so that it becomes the top row of the block. The resulting block must satisfy the conditions of Case 1 or Case 2. Now apply (17) as in Case 1 or Case 2.

- (b) If no blocks contain any rows with distinct values, then conditions R1 and R2 must be true. Apply the discussion following (21) to simplify this term.

Now apply this same procedure to any terms produced in (a) or (b) above. As the algorithm proceeds, terms which can be evaluated using (16) are collected together. At the termination of the algorithm, these terms make up our answer.

It is clear that every term  $Q(\mathbf{A}, \mathbf{b}, \lambda, p)$  where  $\mathbf{A}$  is in standard form can be simplified by using (a) or (b) above. Moreover, all the new terms produced by (a) or (b) involve matrices having either a larger region of zeros or a smaller number of rows than  $\mathbf{A}$ . Thus, the algorithm must terminate after finitely many steps.

When writing a computer program to implement this algorithm, there are many issues that must be dealt with. We shall briefly comment on some of these. In our current work we use a C program (available from the authors) to carry out the algorithm and construct expressions like those in (7). We then use MAPLE to manipulate and evaluate these expressions.

When our algorithm (the C program) is executed, a large number of terms are generated by the successive application of the recursions (17) and (18). At any intermediate point during the execution of the algorithm, the terms that remain to be evaluated may be thought of as a sum of the form

$$\sum_i w_i Q(\mathbf{A}_i, \mathbf{b}_i, \lambda_i, p_i) \quad (22)$$

where the values  $w_i$  are rational numbers. The computer program must decide which of the terms in (22) is the next to be simplified using the process in (a) and (b) above. Our current program orders the terms in (22) primarily according to the dimension

of the matrix  $\mathbf{A}_i$ ; the term with the largest dimension is evaluated first. The terms are ordered first according to the number of rows in  $\mathbf{A}_i$ . Terms having the same number of rows are then ordered by the number of columns. Any terms having the same matrix  $\mathbf{A}_i$  are grouped together; this improves the efficiency of the program since terms with the same  $\mathbf{A}_i$  are simplified in the same way during applications of (17). New terms are combined with old terms whenever possible. That is, whenever a new term  $w'_i Q(\mathbf{A}_i, \mathbf{b}_i, \lambda_i, p_i)$  is created which is identical to an old term  $w_i Q(\mathbf{A}_i, \mathbf{b}_i, \lambda_i, p_i)$  (except perhaps for  $w_i$ ), they are combined into a single term  $(w'_i + w_i) Q(\mathbf{A}_i, \mathbf{b}_i, \lambda_i, p_i)$ . The rational values  $w_i$  and  $\lambda_i$ , and the entries in  $\mathbf{A}_i$  and  $\mathbf{b}_i$  are all represented by pairs of integers (the numerator and denominator) and manipulated using exact rational arithmetic.

The algorithm we describe in this paper can, in principle, compute (1) exactly for any  $\mathbf{A}$  and  $\mathbf{b}$  with rational entries. However, as we increase the dimensionality of  $\mathbf{A}$ , more and more computer time and memory is required by the algorithm. This is because the total number of terms that must be evaluated (simplified) during the course of the algorithm grows rapidly with the size of the problem. These difficulties are probably unavoidable in any algorithm based primarily upon (17) and (18).

## References

- Birnbaum, Z. W. (1952). Numerical tabulation of the distribution of Kolmogorov's statistic for finite sample size. *J. Amer. Statist. Assoc.* **47**, 425–441.
- Choudhuri, N. (1998). Bayesian bootstrap credible sets for multidimensional mean functional. *Ann. Stat.* **26**, 2104–2127.
- Gasparini, M. (1995). Exact multivariate Bayesian bootstrap distributions of moments. *Ann. Stat.* **23**, 762–768.
- Glaz, J. and Balakrishnan, N. (1999). Introduction to scan statistics. *Scan Statistics and Applications*, 3–24. Birkhauser, Boston.
- Huffer, F. W. (1988). Divided differences and the joint distribution of linear combinations of spacings. *J. Appl. Prob.* **25**, 346–354.

- Huffer, F. W. and Lin, C. T. (1997a). Computing the exact distribution of the extremes of sums of consecutive spacings. *Comput. Stat. Data Analysis* **26**, 117–132.
- Huffer, F. W. and Lin, C. T. (1997b). Approximating the distribution of the scan statistic using moments of the number of clumps. *J. Amer. Stat. Assoc.* **92**, 1466–1475.
- Huffer, F. W. and Lin, C. T. (1999a). An Approach to Computations Involving Spacings With Applications to the Scan Statistic. In *Scan Statistics and Applications* (Edited by J. Glaz and N. Balakrishnan), 141–163. Birkhauser, Boston.
- Huffer, F. W. and Lin, C. T. (1999b). Using Moments to Approximate the Distribution of the Scan Statistic. In *Scan Statistics and Applications* (Edited by J. Glaz and N. Balakrishnan), 165–190. Birkhauser, Boston.
- Lin, C. T. (1993). The computation of probabilities which involve spacings, with applications to the scan statistic. Ph.D. Dissertation, Dept. of Statistics, Florida State University, Tallahassee, FL.
- Priestley, M. B. (1981). *Spectral Analysis and Time Series*. Academic Press, London.