Two binomial methods for evaluating the aggregate claims distribution in De Pril’s individual risk model

Sundt, Bjørn
Vital Forsikring ASA
P.O.Box 250, N-1326 Lysaker, Norway
Phone: (+47) 67 83 44 71
Fax: (+47) 67 83 42 60
E-mail: bjoern.sundt@vital.no

Vernic, Raluca
Department of Mathematics and Informatics
‘Ovidius’ University of Constanta
124, Bd. Mamaia, 8700 Constanta, Romania
Fax: (+40) 241 618372
E-mail: rvernic@univ-ovidius.ro

April 16, 2005

Abstract

In this paper, we deduce two new methods for evaluating the aggregate claims distribution in De Pril’s individual risk model and compare them with some other methods by counting the number of dot operations (multiplications and divisions). Finally we consider one of the new methods in the individual life model.

Keywords: De Pril’s individual risk model, aggregate claims distribution, recursions.
1 Introduction

1A. In the present paper, we shall represent probability distributions on the non-negative integers by their probability function, and we shall therefore normally mean the probability function when we refer to a distribution.

De Pril (1989) introduced a two-way individual risk model. We consider a portfolio of independent insurance policies during a specified period. It is assumed that a policy can have at most one claim during the period. The claim amounts are positive and integer-valued. In cell \((i, j)\) for \(i = 1, 2, \ldots, I\) and \(j = 1, 2, \ldots, J\), the probability that a claim occurs, is \(\pi_j\) and the claim amount distribution given that a claim has occurred, is \(h_i\). In this cell, there are \(n_{ij}\) policies. We are interested in the aggregate claims distribution \(f\) of the whole portfolio. De Pril (1989) presented two exact methods for recursive evaluation of \(f\) and discussed three approximations. Dhaene & Vandebroek (1995) presented another exact method that could be more efficient than De Pril’s exact methods.

In the present paper, we shall introduce two other exact methods and compare them with the methods of De Pril (1989) and Dhaene & Vandebroek (1995).

The first one is based on developing a recursive method for evaluating the probability function \(f_i\) of the subportfolio of the policies with probability function \(h_i\). Finally, we evaluate the convolution \(f = \ast_{i=1}^I f_i\). For the evaluation of \(f_i\), we utilise that the number of claims in cell \((i, j)\) is binomially distributed. Thus, \(f_i\) is a compound distribution where the counting distribution is a convolution of binomial distributions and the severity distribution is \(h_i\), and we apply methodology from Sundt (1992) to deduce a recursion for \(f_i\).

In the second method, we introduce some features from the first method into Dhaene-Vandebroek’s method.

Because of the relation between the new methods and the binomial distribution, we shall refer to them as binomial methods.

1B. After having introduced some notation in Section 2, we deduce the first binomial method in Section 3. In this connection, we also recapitulate some results from Sundt (1992). In Section 4, we describe some of the earlier methods for evaluation of \(f\) under the present conditions. Section 5 introduces the second binomial method. In Section 6, the methods will be compared by counting the number of dot operations (multiplication and division). Although this criterion is not perfect, it can give an idea of the efficiency of the methods. However, to not put too much into it and not drowning in details, we restrict to the case when the support of all the \(h_i\)s
is the set of positive integers, and compare the number of dot operations for the methods in an asymptotic setting. In this situation, we show that the winner is normally the first binomial method when there is a high number of non-empty cells for each $i$, and Dhaene-Vandebroek’s method when there is a low number of non-empty cells for each $i$. It is also shown that sometimes it can be efficient to combine Dhaene-Vandebroek’s method with one of the binomial methods by using the binomial method for $is$ with many non-empty cells and Dhaene-Vandebroek’s method for $is$ with few non-empty cells. Finally, in Section 7, we discuss application of the first binomial method in the individual life model where each severity distribution is concentrated in one point.

## 2 Notation

Let $\mathcal{P}_0$ denote the class of distributions on the non-negative integers with a positive probability at zero and $\mathcal{P}_+$ the class of distributions on the positive integers.

We denote by $p \vee h$ a compound distribution with counting distribution $p \in \mathcal{P}_0$ and severity distribution $h \in \mathcal{P}_+$, that is, $p \vee h = \sum_{n=0}^{\infty} p(n) h^{n\ast}$. As $h^{n\ast}(x) = 0$ when $n > x$, we have

$$(p \vee h)(x) = \sum_{n=0}^{x} p(n) h^{n\ast}(x). \quad (x = 0, 1, 2, \ldots)$$

In subsection 1A, we have already introduced some notation for our De Pril model. Let us now introduce some more. To see how the methods are affected by a finite range of the $h_i$s, we assume that $h_i$ has range $\{1, 2, \ldots, m_i\}$, where $m_i$ could either be some positive integer or infinity. We let $g_{ij}$ denote the aggregate claims distribution for a policy in cell $(i, j)$, that is, $g_{ij} = p_j \vee h_i$, where $p_j$ is the claim number distribution of the policy, that is, the Bernoulli distribution given by $p_j(1) = 1 - p_j(0) = \pi_j$. Then we have

$$g_{ij}(x) = \begin{cases} 1 - \pi_j & (x = 0) \\ \pi_j h_i(x) & (x = 1, 2, \ldots, m_i) \end{cases} \quad (2.1)$$

The aggregate claims distribution of all the policies in cell $(i, j)$ is $f_{ij} = g_{ij}^{n_{ij\ast}}$. Thus, we have $f_i = \ast_{j=1}^{i} f_{ij} \ast_{j=1}^{i} g_{ij}^{n_{ij\ast}}$. Continuing on this, we obtain

$$f = \ast_{i=1}^{I} f_i = \ast_{i=1}^{I} \ast_{j=1}^{i} f_{ij} = \ast_{i=1}^{I} \ast_{j=1}^{i} g_{ij}^{n_{ij\ast}} = \ast_{i=1}^{I} \ast_{j=1}^{i} (p_j \vee h_i)^{n_{ij\ast}} = \ast_{i=1}^{I} (\ast_{j=1}^{I} (p_j \vee h_i)^{n_{ij\ast}}) \vee h_i) = \ast_{i=1}^{I} (\ast_{j=1}^{I} q_{ij}) \vee h_i) = \ast_{i=1}^{I} (q_i \vee h_i), \quad (2.2)$$

3
where we have introduced the distribution of the number of claims in cell 
\((i, j)\), \(q_{ij} = p_j n_{ij}^*,\) and the distribution of the number of claims from the 
union of these cells for fixed \(i\), \(q_i = \sum_{j=1}^{J_i} q_{ij}.\)

For \(i = 1, 2, \ldots, I,\) we let \(J_i\) be the number of non-empty cells among the 
cells \((i, 1), (i, 2), \ldots, (i, J)\) and \(n_i = \sum_{j=1}^{J_i} n_{ij}\) the total number of policies in 
these cells, and we let \(J_* = \sum_{i=1}^I J_i\) be the total number of non-empty cells.

We let \([z]\) denote the greatest integer less than or equal to \(z\) and by \(\{z\}\) 
the smallest integer greater than or equal to \(z;\) when \(x\) is a positive integer 
amd \(m_i = \infty,\) we let \(\{x/m_i\} = 1.\)

We let \(P_b k = a = 0\) when \(b < a.\)

3 The first binomial method

3A. Sundt (1992) studied the classes \(R_k\) of distributions \(p \in \mathcal{P}_0\) that satisfy 
a recursion in the form

\[
p(n) = \sum_{u=1}^{\min(k,n)} \left( a(u) + \frac{b(u)}{n} \right) p(n-u); \quad (n = 1, 2, \ldots)
\]

we denote this distribution by \(R_k[a,b],\) and when \(k = 1,\) we drop the argument 
of \(a\) and \(b.\) In particular, he proved the following results:

**Lemma 3.1.** A distribution in \(\mathcal{P}_0\) can be represented as \(R_k[a,b]\) if and only 
if its probability generating function \(\psi\) satisfies

\[
\frac{d}{ds} \ln \psi(s) = \frac{\sum_{u=1}^{k} (ua(u) + b(u)) s^{u-1}}{1 - \sum_{u=1}^{k} a(u) s^u}.
\]

**Lemma 3.2.** The convolution between a distribution in \(\mathcal{R}_k\) and a distribu-
tion in \(\mathcal{R}_l\) is a distribution in \(\mathcal{R}_{k+l}.\)

**Lemma 3.3.** If \(p\) is \(R_k[a,b]\) and \(h \in \mathcal{P}_+,\) then \(p \vee h\) is \(R_\infty[c,d]\) with

\[
c(y) = \sum_{u=1}^{\min(k,y)} a(u) h^{u\ast}(y); \quad d(y) = y \sum_{u=1}^{\min(k,y)} \frac{b(u)}{u} h^{u\ast}(y). \quad (y = 1, 2, \ldots)
\]

Lemma 3.2 easily follows from Lemma 3.1.

Panjer (1981) studied the class \(\mathcal{R}_1.\) In particular, he proved the following 
lemma.
Lemma 3.4. The binomial distribution

\[ p(n) = \binom{m}{n} \pi^n (1 - \pi)^{m-n} \quad (n = 0, 1, 2, \ldots, m; 0 < \pi < 1; m = 1, 2, \ldots) \]

(3.3)

is \( R_1 [-\pi/(1 - \pi), (m + 1) \pi/(1 - \pi)] \).

We denote this distribution by \( \text{bin}(m, \pi) \).

We now have the main tools for evaluation of \( f_i \) as defined in subsection 1A: We express the binomial claim number distribution of the policies in cell \((i, j)\) as an \( R_1 \) distribution. Then we express the distribution of the aggregate number of claims from these cells as an \( R_k \) distribution, and finally we evaluate \( f_i \) recursively by Lemma 3.3 and (3.1).

3B. Theorem 8 in Sundt (1992) gives explicit expressions for the functions \( a \) and \( b \) when expressing the convolution of \( k \) \( R_1 \) distributions in the form \( R_k [a, b] \). In practice, it seems more convenient to do this recursively. We shall deduce such a recursion and start with the following lemma.

Lemma 3.5. The convolution of \( R_{k-1} [a, b] \) and \( R_1 [c, d] \) is \( R_k [\alpha, \beta] \) with

\[ \alpha(u) = a(u) - ca(u - 1) \]

(3.4)

\[ \beta(u) = b(u) - cb(u - 1) - da(u - 1) \].

(3.5)

for \( u = 1, 2, \ldots, k \) with \( a(0) = -1 \) and \( a(k) = b(k) = b(0) = 0 \).

Proof. Let \( \psi_{k-1}, \psi, \) and \( \psi_k \) denote the probability generating functions of \( R_{k-1} [a, b], R_1 [c, d] \), and \( R_k [\alpha, \beta] \). By using Lemma 3.1, we obtain

\[
\frac{d}{ds} \ln \psi_k(s) = \frac{d}{ds} \ln \psi_{k-1}(s) \psi(s) = \frac{d}{ds} \ln \psi_{k-1}(s) + \frac{d}{ds} \ln \psi(s) =
\]

\[
\frac{\sum_{u=1}^{k-1} (ua(u) + b(u)) s^{u-1}}{1 - \sum_{u=1}^{k-1} a(u) s^u} + \frac{c + d}{1 - cs} =
\]

\[
\frac{\sum_{u=1}^{k-1} (ua(u) + b(u)) s^{u-1}}{1 - \sum_{u=1}^{k-1} a(u) s^u} \left(1 - cs \right) + \left(c + d\right) \left(1 - \sum_{u=1}^{k-1} a(u) s^u \right) =
\]

\[
c + d + \sum_{u=1}^{k-1} ((ua(u) + b(u)) s^{u-1} - (c(ua(u) + b(u)) + (c + d) a(u)) s^u) =
\]

\[
1 - \left(cs + \sum_{u=1}^{k-1} (a(u) s^u - ca(u) s^{u+1}) \right) =
\]

\[
\sum_{u=1}^{k} (ua(u) + \beta(u)) s^{u-1} =
\]

\[
1 - \sum_{u=1}^{k} a(u) s^u
\]
with $\alpha$ given by (3.4) and

$$ua (u) + \beta (u) = ua (u) + b (u) - c ((u - 1) a (u - 1) + b (u - 1)) - (c + d) a (u - 1).$$

$$(u = 1, 2, \ldots, k)$$

Solving for $\beta (u)$ and insertion of (3.4) gives (3.5).

This completes the proof of Lemma 3.5. Q.E.D.

**Theorem 3.1.** For $k = 1, 2, \ldots$, the convolution of $\text{bin} \ (m_1, \pi_1), \text{bin} \ (m_2, \pi_2), \ldots, \text{bin} \ (m_k, \pi_k)$ is $R_k [\alpha_k, \beta_k]$ where $\alpha_k$ and $\beta_k$ can be evaluated recursively by

$$\alpha_k (u) = \alpha_{k-1} (u) + \frac{\pi_k}{1 - \pi_k} \alpha_{k-1} (u - 1)$$

$$\beta_k (u) = \beta_{k-1} (u) + \frac{\pi_k}{1 - \pi_k} (\beta_{k-1} (u - 1) - (m_k + 1) \alpha_{k-1} (u - 1))$$

for $k = 2, 3, \ldots$ and $u = 1, 2, \ldots, k$ with

$$\alpha_{k-1} (0) = -1; \quad \alpha_{k-1} (k) = \beta_{k-1} (k) = \beta_{k-1} (0) = 0$$

$$\alpha_1 (1) = -\frac{\pi_1}{1 - \pi_1}; \quad \beta_1 (1) = (m_1 + 1) \frac{\pi_1}{1 - \pi_1}.$$  (3.8)

**Proof.** Formula (3.8) follows from Lemma 3.4, and by letting

$$a = \alpha_{k-1}; \quad b = \beta_{k-1}; \quad c = \alpha_k; \quad \beta = \beta_k$$

$$c = -\frac{\pi_k}{1 - \pi_k}; \quad d = (m_k + 1) \frac{\pi_k}{1 - \pi_k}$$

in Lemma 3.5, we obtain (3.6) and (3.7).

This completes the proof of Theorem 3.1. Q.E.D.

3C. As pointed out at the end of subsection 3A, $f_i$ is a compound distribution with severity distribution $h_i$, and the counting distribution is a convolution of $J_i$ binomial distributions. Thus, the counting distribution can be expressed as $R_{J_i} [a_i, b_i]$, and the functions $a_i$ and $b_i$ can be evaluated by Theorem 3.1. From Lemma 3.3 and (3.1), we obtain that $f_i$ can be evaluated recursively by

$$f_i (x) = \sum_{y=1}^{\min(J,m_i,x)} \left( c_i (y) + \frac{d_i (y)}{x} \right) f_i (x - y) \quad (x = 1, 2, \ldots)$$  (3.9)
with
\[ c_i(y) = \sum_{u=\{y/m_i\}} a_i(u) h_i^{u*}(y), \quad d_i(y) = y \sum_{u=\{y/m_i\}} b_i(u) h_i^{u*}(y) \] (3.10)
\[(y = 1, 2, \ldots, J_i m_i)\]

\[ f_i(0) = q_i(0) = \prod_{j=1}^{J} g_{ij}(0) = \prod_{j=1}^{J} p_j(0)^{n_{ij}} = \prod_{j=1}^{J} (1 - \pi_j)^{n_{ij}}. \] (3.11)

3D. We can now summarise the algorithm for evaluation of \( f \) as follows:

1. For each non-empty cell \((i, j)\), express the claim number distribution \( q_{ij} \) as a binomial distribution.

2. For each \( i \):
   
   (a) Express the claim number distribution \( q_i \) in the form \( R_{J_i}[a_i, b_i] \) by using Theorem 3.1.
   
   (b) Express the aggregate claims distribution \( f_i \) in the form \( R_{\infty}[c_i, d_i] \) with \( c_i \) and \( d_i \) given by (3.10).
   
   (c) Evaluate \( f_i \) recursively by (3.9) and (3.11).

3. Evaluate \( f = \bigotimes_{i=1}^{I} f_i \) (3.12)
   
   by brute force convolution.

4 Earlier methods

4.1 Brute force convolution

The most obvious way to evaluate \( f \) is by brute force convolution, that is, if \( g_i \in \mathcal{P}_0 \) for \( k = 1, 2, \ldots \), then we evaluate \( \bigotimes_{i=1}^{k} g_i \) recursively by

\[ (\bigotimes_{i=1}^{k} g_i)(x) = \left( (\bigotimes_{i=1}^{k-1} g_i) \ast g_k \right)(x) = \sum_{y=0}^{x} g_k(y) \left( \bigotimes_{i=1}^{k-1} g_i \right)(x - y). \] (4.1)
\[(x = 0, 1, 2, \ldots; k = 2, 3, \ldots)\]

We apply this procedure to evaluate

\[ f = \bigotimes_{i=1}^{I} \bigotimes_{j=1}^{J} f_{ij}. \] (4.2)
In particular, this method can be applied by first evaluating \( f_i = \ast_{j=1}^j f_{ij} \) for each \( i \) and then \( f = \ast_{i=1}^f f_i \).

For each non-empty cell \((i,j)\), we evaluate \( g_{ij} \) by (2.1) and then \( f_{ij} = g_{ij}^{n_{ij}} \) by brute force convolution or De Pril’s (1985) recursion

\[
f_{ij}(x) = \frac{1}{g_{ij}(0)} \sum_{y=1}^{\min(m_i,x)} \left( (n_{ij} + 1) \frac{y}{x} - 1 \right) g_{ij}(y) f_{ij}(x-y),
\]

\[(x = 1, 2, \ldots, n_{ij}m_i)\]

which can also be expressed as a recursion for a compound binomial distribution

\[
f_{ij}(x) = \frac{\pi_j}{1 - \pi_j} \sum_{y=1}^{\min(m_i,x)} \left( (n_{ij} + 1) \frac{y}{x} - 1 \right) h_i(y) f_{ij}(x-y)
\]

\[(x = 1, 2, \ldots, n_{ij}m_i)\]

(cf. Lemma 3.4). Using the number of algebraic operations as measure of efficiency, Sundt & Dickson (2000) compare these methods and discuss the most efficient way of applying brute force convolution.

### 4.2 De Pril’s methods

Inspired by De Pril (1989), Sundt (1995) pointed out that for any \( g \in \mathcal{P}_0 \) there exists a unique function \( \varphi_g \) such that \( g \) is \( R_\infty[0, \varphi_g] \), and called \( \varphi_g \) the De Pril transform of \( g \). From (3.1), we immediately get

\[
g(x) = \frac{1}{x} \sum_{y=1}^{x} \varphi_g(y) g(x-y) \quad (x = 1, 2, \ldots)
\]

and solving for \( \varphi_g(x) \) gives

\[
\varphi_g(x) = \frac{1}{g(0)} \left( xg(x) - \sum_{y=1}^{x-1} \varphi_g(y) g(x-y) \right). \quad (x = 1, 2, \ldots)
\]

Furthermore, Lemma 3.3 gives that

\[
\varphi_{p\oplus h}(x) = x \sum_{y=1}^{x} \frac{\varphi_p(y)}{y} h^y(x). \quad (x = 1, 2, \ldots)
\]
Sundt (1995) also showed that
\[ \varphi_{*k} = \sum_{i=1}^{k} \varphi_{gi} \quad (g_1, g_2, \ldots, g_k \in \mathcal{P}_0) \quad (4.7) \]

In De Pril’s first exact method, for each non-empty cell \((i, j)\), we evaluate \(g_{ij}\) by (2.1) and its De Pril transform \(\varphi_{g_{ij}}\) by (4.5). From (4.7), we obtain
\[ \varphi_f = \sum_{i=1}^{I} \sum_{j=1}^{J} n_{ij} \varphi_{g_{ij}}, \quad (4.8) \]
from which we evaluate \(f\) by (4.4), starting with
\[ f(0) = \prod_{i=1}^{I} \prod_{j=1}^{J} (1 - \pi_j)^{n_{ij}}. \quad (4.9) \]

For De Pril’s second exact method, we utilise that
\[ \varphi_{pj} (x) = -\left( \frac{\pi_j}{\pi_j - 1} \right)^x, \quad (x = 1, 2, \ldots) \quad (4.10) \]
By insertion of (4.6) in (4.8) and some reorganisation, we obtain
\[ \varphi_f (x) = x \sum_{i=1}^{I} \sum_{y=\{x/m_i\}}^{x} \frac{h^{y^*}_i(x)}{y} \sum_{j=1}^{J} n_{ij} \varphi_{pj}(y), \quad (x = 1, 2, \ldots) \quad (4.11) \]
and insertion of (4.10) gives
\[ \varphi_f (x) = -x \sum_{i=1}^{I} \sum_{y=\{x/m_i\}}^{x} \frac{h^{y^*}_i(x)}{y} \sum_{j=1}^{J} n_{ij} \left( \frac{\pi_j}{\pi_j - 1} \right)^y, \quad (x = 1, 2, \ldots) \quad (4.12) \]
by which we evaluate \(\varphi_f\). Finally, we evaluate \(f\) recursively by (4.4), starting with (4.9).

The approximations studied by De Pril (1989) are based on approximating each \(\varphi_{pj}\) in (4.11) in De Pril’s second exact method by a function that is equal to zero for all \(y\) greater than some finite \(r\), and he gave error bounds for the approximations of \(f\). Such approximations have later been discussed by Dhaene & De Pril (1994) and Dhaene & Sundt (1998).

As an example of these approximations, we shall consider the \(r\)th order De Pril approximation \(f^{(r)}\), which is obtained by simply summing up to \(y = r\) in the second summation in (4.12), that is,
\[ \varphi_{f^{(r)}} (x) = -x \sum_{i=1}^{I} \sum_{y=\{x/m_i\}}^{x} \frac{h^{y^*}_i(x)}{y} \sum_{j=1}^{J} n_{ij} \left( \frac{\pi_j}{\pi_j - 1} \right)^y, \quad (x = 1, 2, \ldots) \]

which is equal to zero when \( x > r \max m_i \). Finally, we evaluate \( f^{(r)} \) recursively by (4.4) with \( f^{(r)}(0) = f(0) \).

### 4.3 Dhaene-Vandebroek’s method

By inserting (4.8) in (4.4), we obtain

\[
f(x) = \frac{1}{x} \sum_{i=1}^{I} \sum_{j=1}^{J} n_{ij} \psi_{ij}(x) \quad (x = 1, 2, \ldots)
\]  

with

\[
\psi_{ij}(x) = \sum_{y=1}^{x} \varphi_{g_{ij}}(y) f(x - y) . \quad (x = 1, 2, \ldots)
\]

Dhaene & Vandebroek (1995) showed that for each cell \((i, j)\) we have the recursion

\[
\psi_{ij}(x) = \frac{1}{g_{ij}(0)} \min(m_i, x) \sum_{y=1}^{x} (y f(x - y) - \psi_{ij}(x - y)) g_{ij}(y) . \quad (x = 1, 2, \ldots)
\]  

with \( \psi_{ij}(0) = 0 \).

### 5 The second binomial method

For \( i = 1, 2, \ldots, I \) and \( x = 1, 2, \ldots \), let

\[
\psi_i(x) = \sum_{y=1}^{x} \varphi_{f_i}(y) f(x - y) = \sum_{y=1}^{x} \varphi_{f_i}^{*} g_{ij}^{*}(y) f(x - y) = \sum_{y=1}^{x} n_{ij} \varphi_{g_{ij}}(y) f(x - y) = \sum_{j=1}^{J} n_{ij} \psi_{ij}(x) .
\]

Then

\[
f(x) = \frac{1}{x} \sum_{i=1}^{I} \psi_i(x) . \quad (x = 1, 2, \ldots)
\]  

From Theorem 11 in Sundt (1995), we obtain that

\[
\psi_i(x) = \sum_{y=1}^{\min(I, m_i, x)} ((yc_i(y) + d_i(y)) f(x - y) + c_i(y) \psi_i(x - y)) . \quad (x = 1, 2, \ldots; i = 1, 2, \ldots, I)
\]

Using this, we can evaluate \( f \) in the following way:
1. For each non-empty cell \((i, j)\), express the claim number distribution \(q_{ij}\) as a binomial distribution.

2. For each \(i\):

   (a) Express the claim number distribution \(q_i\) in the form \(R_J [a_i, b_i]\) by using Theorem 3.1.

   (b) Express the aggregate claims distribution \(f_i\) in the form \(R_\infty [c_i, d_i]\) with \(c_i\) and \(d_i\) given by (3.10)

3. For each \(x\):

   (a) For each \(i\), evaluate \(\psi_i(x)\) recursively by (5.2).

   (b) Evaluate \(f(x)\) by (5.1).

6 Comparison

6A. A characteristic feature of De Pril’s model is the two-way classification. We know more than just that we have \(IJ\) cells and that the aggregate claims distribution of each policy in cell \((i, j)\) is \(f_{ij}\); we know that \(f_{ij} = p_j \lor h_i\). Thus, all cells with the same \(i\) have the same severity distribution, and all cells with the same \(j\) have the same claim frequency, that is, cells with the same \(i\) or \(j\) have something in common, as opposed to just knowing that all the \(f_{ij}\)s are different. One would expect this additional information to be reflected in the method of evaluating \(f\) and simplify it. This is the case with the two binomial methods and De Pril’s second exact method (as well as the approximations based on it), but not with De Pril’s first exact method and Dhaene-Vandebroek’s method. From these considerations, one should expect that one should just concentrate on De Pril’s second exact method and the two binomial methods. Unfortunately, it is not that simple in general. In particular, the outcome would depend on whether the severity distributions have finite or infinite support.

   One way of comparing the efficiency of methods, is counting elementary algebraic operations; summation, subtraction, multiplication, and division. Such operations are sometimes classified as bar operations (summation and subtraction) and dot operations (multiplication and division) as bar operations are normally less time-consuming on computers than dot operations. Some authors (e.g. Kuon, Reich, & Reimers (1987), Sundt & Dickson (2000), and Dickson & Sundt (2001)) count bar operations and dot operations separately whereas others (e.g. Bühlmann (1984), Dhaene & Vandebroek (1995),
...and Dhaene, Ribas, & Vernic (2005)) count only dot operations. Sundt & Dickson (2000) discuss reasons for not putting too much emphasis on the counting of algebraic operations. When restricting to dot operations, we also have the additional issue that dot operations can sometimes be converted to bar operations, but that should not be overdone; for a large integer $n$, it does not seem reasonable to calculate $nx$ as $\sum_{i=1}^{n} x$, but we shall do it for $n = 2$.

It may seem that the effort of pedantically counting all the algebraic operations is not proportional to the value of this measure of efficiency. To only get an idea of the efficiency of the methods, we shall restrict to dot operations and not go too much into detail. Furthermore, we count as if the support of all the severity distributions is the set of all positive integers. This does not mean that the methods break down when a distribution has probability zero for some positive integer. However, in that case, we may count multiplications even when one of the factors is equal to zero. As optimality criterion, we use the number of dot operations needed for evaluating $f(x)$ for $x = 0, 1, 2, \ldots, s$.

A more extensive comparison between some of the earlier methods by counting dot operations is done by Dhaene, Ribas, & Vernic (2005). Dickson & Sundt (2001) compare some methods for evaluation of two compound $R_1$ distributions.

6B. For a method $A$, we denote by $\tau_A(s)$ the number of dot operations needed for evaluation of $f(x)$ for $x = 0, 1, 2, \ldots, s$. The indicator $A$ can have the following values:

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>first binomial method</td>
</tr>
<tr>
<td>B2</td>
<td>second binomial method</td>
</tr>
<tr>
<td>BFC</td>
<td>brute force convolution</td>
</tr>
<tr>
<td>DP1</td>
<td>De Pril’s first exact method</td>
</tr>
<tr>
<td>DP2</td>
<td>De Pril’s second exact method</td>
</tr>
<tr>
<td>DV</td>
<td>Dhaene-Vandebroek’s method</td>
</tr>
<tr>
<td>DP($r$)</td>
<td>$r$th order De Pril approximation</td>
</tr>
</tbody>
</table>

The last method is not fully comparable with the others as it is not an exact method, only an approximation.

As a measure of efficiency between two methods $A$ and $B$, we use $\varepsilon_{A,B} = \lim_{s \rightarrow \infty} \tau_B(s) / \tau_A(s)$.

Under our present assumptions, the dot operations can typically be classified in three groups when $s$ is sufficiently large:

1. Initial operations that we need to do only once. This is a bit imprecise; there could be recursions there too, like for the evaluation of $c_i$ and
2. Operations that we have to do for each value of $x$ and where the number of dot operations is the same for all values of $x$. Let us call them $\beta_A$.

3. Operations that we have to do for each value of $x$ and where the number of dot operations is proportional with $x$. Let the proportionality factor be $\gamma_A$, so that we need $\gamma_A x$ operations at the step for $x$.

If Method A satisfies this condition, then

$$\tau_A (s) = \alpha_A + \beta_A s + \gamma_A \sum_{x=1}^{s} x = \alpha_A + \beta_A s + \gamma_A \frac{s(s+1)}{2}.$$ 

Thus, if it is satisfied by both Methods A and B with $\gamma_A$ and $\gamma_B$ positive, then

$$\varepsilon_{AB} = \lim_{s \to \infty} \frac{\tau_B (s)}{\tau_A (s)} = \lim_{s \to \infty} \frac{\alpha_B + \beta_B s + \gamma_B \frac{s(s+1)}{2}}{\alpha_A + \beta_A s + \gamma_A \frac{s(s+1)}{2}} = \frac{\gamma_B}{\gamma_A},$$

that is, we need to consider only $\gamma_A$ and $\gamma_B$. With this procedure, we avoid counting the dot operations for initiating the methods. Hence, for methods with heavy initiation, like the binomials method, the performance would be worse for low values of $s$.

6C: We shall now count the $\gamma$ operations for each of various methods.

We start with De Pril’s first exact method. Here, the $\gamma$ operations arise for each value of $y$ in the summations in (4.4) and (4.5). At each place, we have one dot operation, and we use (4.4) on one distribution and (4.5) on $J$ distributions. Hence, $\gamma_{DP1} = J + 1$.

For Dhaene-Vandebrrok’s method, we rewrite (4.14) as

$$\psi_{ij} (x) = \frac{1}{g_{ij} (0)} \sum_{y=0}^{x-1} v_{ij} (x,y) g_{ij} (x-y) \quad (x = 1, 2, \ldots)$$

with

$$v_{ij} (x,y) = (x-y) f (y) - \psi_{ij} (y),$$

which can be evaluated recursively by

$$v_{ij} (x,y) = v_{ij} (x-1,y) + f (y) \quad (x = y + 1, y + 2, \ldots)$$

$$v_{ij} (y,y) = -\psi_{ij} (y).$$
Hence, whereas at first glance it seemed that we needed two dot operations for each \( y \) in (4.14), we have now reduced it to one. As we have to apply (4.14) on \( J_i \) distributions, we obtain \( \gamma_{DV} = J_i \), so that Dhaene-Vandebroek’s method is always better than De Pril’s first exact method when all the severity distributions have infinite support.

We now turn to the first binomial method. Here, the \( \gamma \) operations arise in (3.9), for \( h_i^u \star (x) \) for \( u = 1, 2, \ldots, J_i \) in (3.10), and when taking the convolution of the \( f_i \)s in (3.12).

Let us first consider the \( h_i^u \star \)s. The natural way to do this, is to apply (4.1) on \( h_i^u \star = h_i^{(u-1)\star} \star h_i \). However, Sundt & Dickson (2000) point out that when \( u \) is even, it is usually more efficient to use \( h_i^u \star = h_i^{2\star} \star h_i^{\frac{u}{2}\star} \). As they considered distributions in \( P_0 \) whereas we have distributions in \( P_+ \), we cannot immediately transfer the results.

For the direct method, we utilise that \( h_i^0 = 0 \) and, thus, \( h_i^u \star (y) = 0 \) when \( y < u \), so that

\[
h_i^u \star (x) = \sum_{y=u-1}^{x-1} h_i^{(u-1)\star} (y) h_i (x-y), \quad (x = u, u+1, \ldots)
\]

which needs \( x - u + 1 \) dot operations.

When \( u \) is even, for \( x = u, u+1, \ldots \), we have

\[
h_i^u \star (x) = \sum_{y=u-1}^{x-1} h_i^{\frac{u}{2}\star} (y) h_i^{\frac{u}{2}\star} (x-y) = \begin{cases} 2 \sum_{y=\frac{u}{2}}^{x-1} h_i^{\frac{u}{2}\star} (y) h_i^{\frac{u}{2}\star} (x-y) & (x \text{ odd}) \\ 2 \sum_{y=\frac{u}{2}}^{x-1} h_i^{\frac{u}{2}\star} (y) h_i^{\frac{u}{2}\star} (x-y) + h_i^{\frac{u}{2}\star} \left( \frac{x}{2} \right) h_i^{\frac{u}{2}\star} \left( \frac{x}{2} \right) & (x \text{ even}) \end{cases}
\]

which gives \( (x - u + 3) / 2 \) dot operations when \( x \) is odd and \( (x - u + 2) / 2 \) when \( x \) is even.

When counting the \( \gamma \) operations, it is the operations of order \( x \) that we have to take into account, that is, one when \( u \) is odd, and \( 1/2 \) when \( u \) is even. Counting as if \( u \) is odd half the times, we count as if on the average each of these convolutions contributes to \( \gamma_{B1} \) with \( 3/4 \). For each \( i \), we have to perform \( J_i - 1 \) such convolutions, so for all \( i \)s together, we get a contribution to \( \gamma_{B1} \) of \( \frac{3}{4} (J_i - I) \).

At first glance, it seems that in (3.9), we need two dot operations for each \( y \). However, by rewriting it as

\[
f_i (x) = \frac{1}{x} \sum_{y=1}^{x} w_i (x, y) f_i (x-y) \quad (x = 1, 2, \ldots)
\]
with 
\[ w_i(x, y) = xc_i(y) + d_i(y), \]
which can be evaluated recursively by 
\[ w_i(x, y) = w_i(x - 1, y) + c_i(y) \quad (y = 1, 2, \ldots, x - 1) \]
\[ w_i(x, x) = xc_i(x) + d_i(x), \]
we have converted one of the dot operations to bar operation, so that we have only one dot operation left. We have \( I \) recursions like (3.9), so totally they contribute with \( I \) to \( \gamma_{B1} \). Finally, each of the \( I - 1 \) convolutions of \( f_i \)'s in (3.12) contributes to \( \gamma_{B1} \) with one. Thus,

\[ \gamma_{B1} = \frac{3}{4} (J_\bullet - I) + I + I - 1 = \frac{3J_\bullet + 5I}{4} - 1. \]

Comparing with Dhaene-Vandebroek’s method, we see that \( \gamma_{B1} < \gamma_{DV} \) when \( J_\bullet > 5I - 4 \). In particular, this implies that the first binomial method is always better than Dhaene-Vandebroek’s method when there are at least five non-empty cells for each value of \( i \).

For the second binomial method, the work in the \( h_i^* \)'s is the same as for the first binomial method, that is, the contribution to \( \gamma_{B2} \) is \( \frac{3}{4} (J_\bullet - I) \). Then, for each value of \( y \) in (5.2), there are two dot operations (assuming that \( yc_i(y) + d_i(y) \) is evaluated only once for each \( (y, i) \), and there are \( I \) such recursions. Hence,

\[ \gamma_{B2} = \frac{3}{4} (J_\bullet - I) + 2I = \frac{3J_\bullet + 5I}{4}. \]

As \( \gamma_{B1} - \gamma_{B2} = 1 \) this method is slightly beaten by the first binomial method, so that we have to discard it.

For De Pril’s second exact method, we also have to evaluate \( h_i^*(x) \)'s. However, whereas for the binomial methods we needed these convolutions up to order \( J_i \) in (3.10), for De Pril’s second exact method, we need them up to order \( x \) in (4.12) so that \( \tau_{DPT2} (s) \) becomes a polynomial of third order. Hence, for large \( s \), it will perform worse than the other methods. However, in this connection it should be emphasised that the strength of De Pril’s second method is as a basis for approximations that will perform better with respect to dot operations.

On the other hand, for the \( r \)th order De Pril approximation, we need the convolutions only up to order \( r \). We have to do this for \( I \) distributions, and their total contribution to \( \gamma_{DP(r)} \) is \( \frac{3}{4} (r - 1) I \). In addition, we have to apply (4.4) once, so that

\[ \gamma_{DP(r)} = \frac{3}{4} (r - 1) I + 1. \quad (6.3) \]
With brute force convolutions, the $J - 1$ convolutions in (4.2) contribute with $J - 1$ to $\gamma_{BFC}$. In addition, we have to evaluate $f_{ij} = g^{n_{ij}}_i$ when $n_{ij} > 1$. At first glance, using (4.3) for one $f_{ij}$ seems to give a contribution of two to $\gamma_{BFC}$, but by proceeding like we did with (4.14) and (3.9) above, we can reduce it to one. However, when $n_{ij} = 2$, we can reduce it to $1/2$ by proceeding similar to (6.2). Nevertheless, when comparing with Dhaene-Vandebroek’s method and the first binomial method, we see that there cannot be more than two cells with more than one policy if brute force convolution should be the best method.

These considerations lead to the following conclusions about the exact methods under the present conditions:

1. De Pril’s exact methods and the second binomial method can be discarded.
2. Brute force convolution can be discarded unless there are only one or two cells with more than one policy.
3. The first binomial method is better than Dhaene-Vandebroek’s method when $J > 5I - 4$.

It is logical that it is when there are many non-empty cells with the same severity distribution that the binomial methods perform best, as that is when the extra information through the two-way classification is most significant.

6D. The race between the first binomial method and Dhaene-Vandebroek’s method as well as De Pril’s first method and the second binomial method is remarkably close. It is amazing that it is the application of (6.2) that rescues the first binomial method. If we had used the direct method (6.1) for all $u$, then we would have got

$$\gamma_{B1} = J - I + I - 1 = J + I - 1,$$

that is, then the first binomial method is not better than Dhaene-Vandebroek’s method for any value of $I$, and it is better than De Pril’s first exact method only when $I = 1$. Furthermore, the first binomial method is more complex to program, even when using (6.1) for all $u$.

It seems surprising if utilisation of the extra information from the two-way classification does not give a larger gain for exact methods. On the other hand, from (6.3) we see that the gain is very pronounced for De Pril approximations of low order.
6E. We have
\[
\gamma_{B1} = \sum_{i=1}^{I} \frac{3J_i + 5}{4} - 1 \quad (6.4)
\]
\[
\gamma_{DV} = \sum_{i=1}^{I} J_i
\]
and analogous if we consider only a subset of the possible \(i\)s.

We could also use a combination of these two methods where we use Dhaene-Vandebroek’s method when \(J_i < (3J_i + 5)/4\), that is, \(J_i < 5\), and the first binomial method when \(J_i \geq 5\). Then we get one extra convolution between the aggregate claims distribution of all the policies treated with the first binomial method and the aggregate claims distribution of all the other policies. Let \(\mathcal{A} = \{i : J_i \geq 5\}\). Then the \(\gamma\) value of this combined method is
\[
\gamma_{\text{comb}} = \sum_{i \in \mathcal{A}} \frac{3J_i + 5}{4} + \sum_{i \notin \mathcal{A}} J_i - \frac{\sum_{i \notin \mathcal{A}} J_i - 5}{4} = \gamma_{\text{DV}} - \left( \frac{\sum_{i \notin \mathcal{A}} 5 - J_i}{4} - 1 \right).
\]
Hence, the combined method is at least as good as Dhaene-Vandebroek’s method, and it is better than the first binomial method when \(\sum_{i \notin \mathcal{A}} (5 - J_i) > 4\).

Now let us instead replace the first binomial method with the second binomial method. Then we have to drop the subtraction of one in (6.4). However, this disadvantage is canceled by the advantage that we do not need the extra convolution that we have to do when combining Dhaene-Vandebroek’s method with the first binomial method. Hence, in combination with Dhaene-Vandebroek’s method, the two binomial methods are equally good with respect to \(\gamma\) operations. However, the second binomial method is more similar to Dhaene-Vandebroek’s methods and seems to easier to program for such combinations as we do not need to perform the brute force convolutions of the \(f_i\)s, so we recommend that method.

7 The first binomial method in the individual life model

Let us finally consider the first binomial method in the individual life model where each severity distribution is concentrated in one point, that is, for each \(i\), \(h_i\) is concentrated in one positive integer \(m_i\). We now have
\[
f_i(x) = \begin{cases} q_i(x/m_i) & (x = 0, m_i, 2m_i, \ldots, m_in_i) \\ 0 & (\text{otherwise}) \end{cases}
\]
and evaluate $q_i$ by

$$q_i(x) = \sum_{y=1}^{J_i} \left( a_i(y) + \frac{b_i(y)}{x} \right) q_i(x - y). \quad (x = 1, 2, \ldots, n_i)$$

Then

$$\left( \ast_{i=1}^{k} f_i \right) (x) = \sum_{y=\max \left( 0, \frac{x-x^{k-1}_{i=1} m_i n_i}{m_k} \right)}^{\lceil x/m_k \rceil} q_k(y) \left( \ast_{i=1}^{k-1} f_i \right) (x - m_k y), \quad (7.1)$$

and finally we get $f = \ast_{i=1}^{k} f_i$.

We would normally have $\left( \ast_{i=1}^{k} f_i \right) (x) = 0$ for many values of $x$, and, hence, we ought to try to avoid evaluating the product sum in (7.1) for such values of $x$. One way to avoid some of these values, is to choose the monetary unit so large that the $m_i$s do not have any common factor. Furthermore, even in that case, $\left( \ast_{i=1}^{k} f_i \right) (x) > 0$ if and only if $x$ can be written in the form

$$x = \sum_{i=1}^{k} v_i m_i. \quad (v_i = 0, 1, 2, \ldots, n_i; i = 0, 1, 2, \ldots, k) \quad (7.2)$$

Hence, we ought to evaluate the product sum only when this condition is satisfied. This can be tested quite simply. We know that

$$\left( \ast_{i=1}^{k} f_i \right) (0) = q_k(0) \left( \ast_{i=1}^{k-1} f_i \right) (0) > 0,$$

whereas, for $x = 1, 2, \ldots, \sum_{i=1}^{k} m_i n_i$, $\left( \ast_{i=1}^{k} f_i \right) (x) > 0$ if and only if at least one of the conditions $\left( \ast_{i=1}^{k} f_i \right) (x - m_k) > 0$ and $\left( \ast_{i=1}^{k-1} f_i \right) (x - m_k) > 0$ is satisfied.

We could also reduce the summation in (7.1) to the set

$$\{ y = 0, 1, 2, \ldots, \lceil x/m_k \rceil : x - m_k y \text{ satisfies } (7.2) \text{ with } k \text{ replaced with } k - 1 \}. $$

However, then it seems simpler to just test whether $\left( \ast_{i=1}^{k-1} f_i \right) (x - m_k y) > 0$, so this seems to be stretching too far the criterion of minimising the number of dot operations.

The multiplication in $x - m_k y$ in (7.1) could be avoided by evaluating $t_k(x, y) = x - m_k y$ recursively by

$$t_k(x, y) = t_k(x, y - 1) - m_k \quad (y = 1, 2, \ldots, \lceil x/m_k \rceil)$$

$$t_k(x, 0) = x.$$
For a discussion of application of the earlier methods in the individual life model, we refer to Dhaene, Ribas, & Vernic (2005). The procedure we have suggested for the convolution of $f_1, f_2, \ldots, f_I$, can of course also be applied in connection with the brute force convolution method of subsection 4.1.

References


