A COMPUTER ALGORITHM FOR MULTIPLE CLAIMS MODELS

by

Gottfried Berger

Cologne Life Reinsurance Company

INTRODUCTION

Some time ago, I got rather excited by a paper by Tellenbach (1977), which applies renewal theory models to actuarial problems. The input consists of assumptions as to the waiting time between consecutive claims (interclaim time). The output is a set of probabilities $p_k(t)$ for k claims in the time period (o,t).

This can be applied, for instance, to catastrophe insurance. Here, the set $p_k(t)$ measures the frequency of accidental deaths as a result of (a) claim events (accidents), and (b) fatalities per accident.

Tellenbach tested the models on data from automobile insurance. Then $p_k(t)$ is the probability of k insurance claims from the same policyholder in the accounting period (o,t). He used Monte Carlo techniques, because the formulae became, after some convolutions, much too complicated.

When I tried to repeat these tests, I noticed two things. One, Monte Carlo runs use excessive CPU time. Two, my results were different from Tellenbach's.

At this stage, I made a bold decision. I expressed the input assumptions not in analytical form, but as power expansions. The operational time t is so small that the power expansions could be truncated after a few terms. I thus replaced infinite power series by polynomials. Now the complicated process of successive convolutions becomes a simple matter of vector multiplications. The results were so encouraging that I wrote a paper on the method; this paper is reprinted in the Appendix.

In the following, we shall first describe the model, and how it translates into multiplications between Laplace-Transforms. After this preparation we shall review the computer program. Here, the Laplace-Transforms are represented by arrays, which can be easily manipulated within the computer language APL.

THE MODEL

Table 1 is a summary of the mathematical notation, and of the equations which describe the model. The first three equations:

$$\phi_0 = (1-\phi_1)/\alpha_1 s$$

$$\psi_1 = \phi_0(1-\phi_1)$$

$$\psi_k = \phi_1 \psi_{k-1} \qquad (k \geq 2)$$

describe an equiblibrium renewal process, meaning that

$$\mu(t) = \Sigma\, k p_k(t) = t/\alpha_1$$

Here, α_1 is the mean of distribution p_1 associated with $\phi_1 = \phi_1(s)$.

The waiting time for the first claim has the c.d.f. $P_o(t)$, associated with $\phi_1(s)$. The interpretation of the 3 equations is easy:

$1 - P_o(t)$ <===> ϕ_o (no first claim)

$P_1(t)$ <===> $\zeta_o(1-\phi_1)$ (first claim, but no second claim)

$P_k(t)$ <===> $\phi_1\ \psi_{k-1}$ (k-1 claims, plus one more)

So much for the general model. We now turn to the special assumption:

$P_1(t) = h\,P(t) + (1-h)u(t)$ <===> $\phi_1 = h\phi + 1-h$

With probability h ($o \leq h \leq 1$), the interclaim time has the c.d.f. P(t), for which we chose the one-sided Gauss distribution.

With probability 1-h, the interclaim time is zero, allowing for multiple claims. (u(t) is the step function which jumps at t=0 from 0 to 1.)

The model is a generalization of Poisson. The latter case results if we chose for P(t) the negative exponential distribution.

Furthermore, the model can be interpreted as a cumulative claims process: Claims events are controlled by P(t). The number of claims per claims event follows the geometrical distribution: $g_n = h\,(1-h)^{n-1}$.

After the choice of P(t), we try to match the empirical data as closely as possible. First, we fix t to match the observed mean. Second, we vary h to minimize, say, the sum of the error squares.

THE PROGRAM

The APL program, called GAUSS is displayed in Table 2.*

The essential idea is to represent power expansions by the vector of its coefficients. In program line [20] we construct the vector Y holding the coefficients y_i of

$$\phi_0(s) = y_1/s^1 + y_2/s^2 + \ldots$$

From $\phi_0(s) = \frac{1}{\alpha s} - \frac{1}{s}\,\mathrm{LTRF}\,(\alpha e^{-t^2/2})$

We obtain:

$$Y = \frac{1}{\alpha},\ 0,\ \frac{(2\cdot 1)!}{1!(-2)},\ 0,\ \frac{(2\cdot 2)!}{2!(-2)^n}$$

Program line [24] generates for $\phi_1(s)$ the associated vector $Z = z_0, z_1, \ldots$

To make use of the APL primitive function "inner product", we must upgrade Z into a matrix:

$$Z = \begin{matrix} z_0 & 0 & 0 \\ z_1 & z_0 & 0 \\ z_2 & z_1 & z_0 \end{matrix}$$

* Also reprinted is Table 2a which the author used in the presentation to explain the program steps.

Now the APL code "R← Z + .xY" yields the result vector

$$R = (z_o y_1), \quad (z_1 y_1 + z_o y_2), \ldots$$

This corresponds to the product $\phi_1(s).\phi_o(s)$.

In the course of the program, the matrix Z (corresponding with ϕ_1) is kept fixed, while the vector Y is always loaded with the new coefficients, resulting from the convolution. Of particular interest is the loop L2 in program line [33]. Here we see two inner products. The left one:

T+.xY (equivalent to +/TxY) translates the Laplace-Transform back to the original c.d.f.

So far so good. But there remains a disquieting question: How far should we extend the power expansion, i.e., what should be the length n of the vector Y we start with in line [20]?

To answer this question, the program starts with n=5, and is repeated (loop L1) with n=9, n=13, etc. until the last term of T, $t^n/n!$, becomes small enough to be ignored.

In the example displayed in Table 3, the results of the first run (with n=5) are already final. This indicates that the simple method presented in this paper may in many instances well serve to make costly Monte Carlo experiments obsolete.

REMARK ON ACCURACY

To test the accuracy of the method, we set h=1 and P(t)=Exponential. Then the computer algorithm should match the Poisson probabilities (which can be calculated directly). We found no theoretical but practical machine limitations:

t =	.1	.5	1	2	5
n =	9	13	17	25	45

Here, t is the number of expected claims, i.e., the operational time in units of the average waiting time between claims; n is the number of expansion terms needed to bring the calculation error squares total below 1E-20.

Gottfried Berger
Stamford, CT
June 26, 1981

References

Seal, H. (1969): Stochastic Theory of a Risk Business, John Wiley & Sons.
Tellenbach, U. (1977): Berechnung der Verteilung der Schadenzahlen bei bekannter Verteilung der Wartzeiten, MSV 77.1, 35-46.
Berger, G. (1978): A Computer Algorithm for the Cumul Model, MSV 78.2.

Table 1

NOTATION

c.d.f.	p.d.f.	Image	
$P(t)$	$P'(t)$	$\phi(s) = E(e^{-st})$	Inter claim time
$P_0(t)$	$P_0'(t)$	$\phi_0(s)$	• First claim
$P_1(t)$	$P_1'(t)$	$\phi_1(s)$	• Next claims
$p_K(t)$	$p_K'(t)$	$\psi_K(s)$	Number of claims
$u(t)$	$\delta(t)$	1	< Step function >
$\frac{t^n}{n!}$	$\frac{t^{n-1}}{(n-1)!}$	$\frac{1}{s^n}$	< Expansion terms >

MODEL

$$\begin{cases} \phi_0 = (1-\phi_1)/\alpha_1 s & < \alpha_1 = \text{Mean } P_1 > \\ \psi_1 = \phi_0 (1-\phi_1) & \\ \psi_K = \phi_1 \psi_{K-1} & K \geq 2 \end{cases}$$

$$P_1(t) = h\, P(t) + (1-h)\, u(t)$$

$$P(t) = \alpha \int_0^t e^{-x^2/2}\, dx \qquad \alpha = \sqrt{\frac{2}{\pi}} = \text{Mean } P$$

$$t = \alpha_1 \hat{\mu} = \alpha h \hat{\mu} \quad (= \text{operational time})$$

$\hat{\mu}$ = observed mean $\mu(t) = \sum K p_K(t)$.

Table 2

```
      ∇GAUSS[⎕]∇
    ∇ GAUSS H;A;E;I;K;N;T;Y;Z;W
[1]   '... Cumulative claims model with parameter h=';H
[2]   '... P1(t) = h×P(t) + (1-h)×u(t)   ,   P'(t)=GAUSS(t≥0)'
[3]   ⍝ ... Set N=1 ,  A=1/α ,   α = ∫tdP(t) = SQROOT 2/π
[4]    N←1
[5]    A←(○0.5)*0.5
[6]   L1:''
[7]   ⍝ ... W=Vector [ p(0,t)  p(1,t) ...] to be compared with
[8]   ⍝ ... W=Observed claims frequencies, taken from TB
[9]   ⍝ ... Z=Mean of W
[10]   Z←+/W×¯1+⍳⍴W←W÷+/W←TB
[11]  ⍝ ... Set T=t such that  Z = Σ k×p(k,t)
[12]   T←Z×H÷A
[13]  ⍝ ... T = Vector [ t ... (t*n)/n! ]  , n=3×2×N
[14]   T←(T*K)÷!K←⍳3+2×N
[15]  ⍝ ... Print last, first element of T
[16]   '⍷≈';E←¯1↑T
[17]   't=';1↑T
[18]  ⍝ ... Vector Y :   Φ0=(1-Φ)/αs = Σ Y × [ 1/s ... 1/s*n ]
[19]  ⍝ ... lefthand side from power expansion of Φ=LTRF P(t)
[20]   Y←A,((2×N+1)⍴ 1 0)\-1,(!2×⍳N)÷(!⍳N)×¯2*⍳N
[21]  ⍝ ... Build  W = p(0,t)=1-P0(t), reversing Φ0=LTRF P0(t)
[22]   W←1-+/T×Y
[23]  ⍝ ... Vector Z :   Φ1=1-αhsΦ0  = Σ Z × [ 1 ... 1/s*n-1 ]
[24]   Z←((⍴Y)↑1)-Y×H÷A
[25]  ⍝ ... Build matrix :  Z →Z[∘;1] →Z[∘+j;1+j], else zeros
[26]   Z←(0,Z)[⌊2+I∘.-I←⍳⍴Z]
[27]  ⍝ ... Perform :  Ψ1 = Φ0 - Φ1 ×  Φ0 = LTRF p(1,t)
[28]  ⍝ ...              Y  ←  Y  - Z +.× Y
[29]  ⍝ ... Append p(1,t) ... "+/T×" or "T+.×" reverses LTRF
[30]   W←W,T+.×Y←Y-Z+.×Y
[31]  ⍝ ... Perform :  Ψk =  Φ1  ×   Ψk-1 = LTRF p(k,t)
[32]  ⍝ ...              Y  ←  Z  +.×  Y
[33]  L2:W←W,T+.×Y←Z+.×Y
[34]  ⍝ ... W now holds p(0,t)...p(k,t). Go on until ⍷>p(k,t)
[35]   →(E<¯1↑W)/L2
[36]  ⍝ ... Print results
[37]   'Σ p(k,t)×k*0 1 2 = ';W+.×(0,⍳¯1+I←⍴W)∘.* 0 1 2
[38]   'Σ (p-P̲)*1 2 = ';+⌿I←((W←I↑W)-(I←I⌊⍴W)↑W)∘.*⍳2
[39]   (3 0 ,6⍴ 12 6)⍉ 10 4 ↑(¯1+⍳⍴W),W,I
[40]   '  k       p(k,t)       p-P̲          (p-P̲)*2'
[41]  ⍝ ... Conditional return to L1 with increased N
[42]   →((E>1E¯16)∧92≥N←N+2)/L1
    ∇


      ∇TB[⎕]∇
    ∇ R←TB
[1]   R← 7840 1317 239 42 14 4 4 1
    ∇
```

Table 2[a]

Laplace T+f.	APL
$\phi_0(s) \doteq \frac{y_1}{s} + \frac{y_2}{s^2}$	$y \leftarrow \{ y_1, y_2, y_3 \}$
$\psi_1(s) = \phi_0(s) - \phi_0(s) \cdot \phi_1(s)$	$y \leftarrow y - Z +.\times y$
$\psi_k(s) = \psi_{k-1}(s) \cdot \phi_1(s)$	$y \leftarrow Z +.\times y$

Initialization :

	$\phi_1(s) \doteq z_0 + \frac{z_1}{s} + \frac{z_2}{s^2}$	$Z \leftarrow \{ z_0, z_1, z_2 \}$
[20]	$\phi_0(s) = \frac{1}{a_1 \cdot s} \cdot [1 - \phi_1(s)]$	$y \leftarrow \{ y_1, y_2, y_3 \}$

[26] $Z \leftarrow \begin{Bmatrix} z_0 & 0 & 0 \\ z_1 & z_0 & 0 \\ z_2 & z_1 & z_2 \end{Bmatrix}$

Calculate ψ_1 :

[30] $y \leftarrow y - Z +.\times y$

Calculate ψ_k (Loop L2):

[33] $y \leftarrow Z +.\times y$

Transform back at each step :

[22]	$p_0(t)$:	$W \leftarrow 1 - +/T \times y$
[30]	$p_1(t)$:	$W \leftarrow W, T +.\times y$
[33]	$p_k(t)$:	$W \leftarrow W, T +.\times y$

14] $T \leftarrow \{ t, \frac{t^2}{2!}, \frac{t^3}{3!} \}$

Table 3

```
      GAUSS .86
... Cumulative claims model with parameter h=0.86
... P1(t) = h×P(t) + (1-h)×u(t)   ,   P'(t)=GAUSS(t≥0)

ε=5.7367E-7
t=0.14709
Σ p(k,t)×k*0 1 2 = 1 0.21435 0.31451
Σ (p-P̲)*1 2 = -3.2897E-8 1.0159E-5
  0      .826453      -.002212      .000005
  1      .140327       .001124      .000001
  2      .027016       .001755      .000003
  3      .005067       .000628      .000000
  4      .000931      -.000549      .000000
  5      .000168      -.000254      .000000
  6      .000030      -.000393      .000000
  7      .000005      -.000100      .000000
  8      .000001       .000001      .000000
  9      .000000       .000000      .000000
  k       p(k,t)        p-P̲          (p-P̲)*2

ε=8.879E-14
t=0.14709
Σ p(k,t)×k*0 1 2 = 1 0.21435 0.31452
Σ (p-P̲)*1 2 = -3.2964E-15 1.0158E-5
  0      .826453      -.002212      .000005
  1      .140327       .001124      .000001
  2      .027016       .001755      .000003
  3      .005067       .000628      .000000
  4      .000931      -.000549      .000000
  5      .000168      -.000254      .000000
  6      .000030      -.000393      .000000
  7      .000005      -.000100      .000000
  8      .000001       .000001      .000000
  9      .000000       .000000      .000000
  k       p(k,t)        p-P̲          (p-P̲)*2

ε=2.4217E-21
t=0.14709
Σ p(k,t)×k*0 1 2 = 1 0.21435 0.31452
Σ (p-P̲)*1 2 = 1.1547E-17 1.0158E-5
  0      .826453      -.002212      .000005
  1      .140327       .001124      .000001
  2      .027016       .001755      .000003
  3      .005067       .000628      .000000
  4      .000931      -.000549      .000000
  5      .000168      -.000254      .000000
  6      .000030      -.000393      .000000
  7      .000005      -.000100      .000000
  8      .000001       .000001      .000000
  9      .000000       .000000      .000000
  k       p(k,t)        p-P̲          (p-P̲)*2
```

Appendix

A Computer Algorithm for the Cumul Model

By Gottfried Berger

Abstract

The first two sections of this paper describe a risk theory model which was introduced recently by Tellenbach. The model involves Laplace-Stieltjes transforms which pose severe computational difficulties.
The remainder of this paper describes a simple algorithm which appears to work in the special case where the time span considered is reasonably small. The latter condition is typically met if the model refers to cumulative claims. A thorough mathematical treatment of the subject is not even attempted. Rather, the emphasis is on the computational aspect of the problem.

1. A Claims Process

Let t_j denote the interclaim time between the $j-th$ and the next following claim ($j = 0, 1, 2\ldots$). Suppose the probability distribution functions $P_j(t)$ for the stochastic variables t_j are known. We assume $t_j \geq 0$ and thus $P_j(t) = 0$ for $t < 0$. Our objective is to find the probabilities $p_k(t)$ for k claims in the time interval $(0, t)$.
Clearly, $p_0(t) = 1 - P_0(t)$. The rest is not as easy, since we have to perform convolutions. This can be done, at least in theory, by means of Laplace-Stieltjes transforms. See for instance Seal (1969), Appendix A, or Tellenbach (1977).
We thus define the following Laplace-Stieltjes transforms:

$$\Phi_j = \Phi_j(s) = L\{P_j(t)\} = \int_{0-}^{\infty} e^{-st}\, dP_j(t), (j = 0, 1, 2\ldots).$$
$$\Psi_k = \Psi_k(s) = L\{p_k(t)\} = \int_{0-}^{\infty} e^{-st}\, dp_k(t), (k = 0, 1, 2\ldots). \qquad (1)$$

Mitteilungen der Vereinigung schweiz. Versicherungsmathematiker, Heft 2, 1978

We now introduce two simplifications. First, we require that all interclaim times t_j (except possibly t_0) are independently and identically distributed, i.e., $\Phi_j = \Phi_1$ for $j \geq 1$. The theory of the renewal process then shows that:

$$\begin{aligned} L\{1-p_0(t)\} &\equiv \Phi_0 \\ L\{p_1(t)\} &\equiv \Psi_1 = \Phi_0(1-\Phi_1) \\ L\{p_k(t)\} &\equiv \Psi_k = \Psi_{k-1}\,\Phi_1 = \Psi_1\Phi_1^{k-1} \text{ for } k > 1 \end{aligned} \tag{2}$$

Second, we stipulate a stationary claims process:

$$\mu_1(t) = \sum_1^{\infty} k \cdot p_k(t) = \text{const} \cdot t = t/\alpha_1 \tag{3}$$

If (3) holds, $\mu_1(t)$, the expected claims number in $(0, t)$, is proportional to t, and inversely proportional to α_1 = average interclaim time = mean of $P_1(t)$. We obtain from (2) and (3):

$$L\{\mu_1(t)\} = \Psi_1(1+2\Phi_1+3\Phi_1^2+\ldots) = \Psi_1/(1-\Phi_1)^2 = \Phi_0/(1-\Phi_1) = 1/\alpha_1 s$$

Thus, we can rewrite equations (2) as follows:

$$\begin{aligned} L\{1-p_0(t)\} &\equiv \Phi_0 = (1-\Phi_1)/\alpha_1 s \\ L\{p_1(t)\} &\equiv \Psi_1 = \Phi_0(1-\Phi_1) \\ L\{p_k(t)\} &\equiv \Psi_k = \Psi_{k-1}\Phi_1 \text{ for } k > 1 \end{aligned} \tag{4}$$

Equations (4) confirm that the stationary claims process is completely determined by the function $\Phi_1 = \Phi_1(s)$, the Laplace-Stieltjes transform of $P_1(t)$, the d.f. of the interclaim time.

2. The Tellenbach Model

Tellenbach (1977) considered the following choice of $P_1(t)$:

$$P_1(t) = h \cdot P(t) + (1-h) \cdot u(t) \tag{5}$$

Here, $P(t)$ is an arbitrary d.f. which applies to the claims process with the probability h $(0 < h \leq 1)$. Multiple claims may occur with the probability $(1-h)$, since

$$u(t) = \begin{cases} 0 \text{ if } t < 0, \\ 1 \text{ if } t \geq 0. \end{cases}$$

Intuitively, $P(t)$ controls the number of events, while $P_1(t)$ determines the number of claims. The smaller the parameter h is chosen, the more claims one associates with each claim event.
Let $\Phi = \Phi(s)$ denote the Laplace-Stieltjes transform of $P(t)$, and let α be the mean of $P(t)$. If $\alpha < \infty$ we obtain from (5):

$$L\{P_1(t)\} = \Phi_1(s) = (1-h) + h \cdot \Phi(s). \tag{6}$$

Tellenbach applied this model to actual data on auto insurance, published by Thyrion (1961). In doing so, he determined t such that $\mu_1(t)$ matches the empirical mean $\hat{\mu}_1$. That is,

$$t = \alpha_1 \mu_1(t) = \alpha h \mu_1(t) = \alpha h \hat{\mu}_1. \tag{7}$$

This leaves the free parameter h which may be chosen so that an appropriate error measure is minimized.

Tellenbach considered for $P(t)$ the negative exponential as well as the one-sided Gauss distribution. The results were superior in the latter case which, however, involves computational difficulties. Tellenbach solved the problem by Monte-Carlo techniques.

3. An Algorithm

We shall now assume that the distribution function $P(t)$ can be expressed as a power series of t which converges reasonably fast. This assumption will hold if t is small enough. The data of Thyrion, for instance, require according to equation (7) for the one-sided Gauss distribution:

$$t = \alpha h \mu = \sqrt{2/\pi} \cdot 0.214 \cdot h = 0.171 \cdot h.$$

Hoping for convergence we thus develop:

$$P'(t) = \alpha e^{-t^2/2} = \alpha(1 - t^2/2 + t^4/8 - \ldots), \alpha = \sqrt{2/\pi}$$

Hence,

$$\Phi(s) = \alpha(1/s - 1/s^3 + 3/s^5 - \ldots),$$

and from (6):

$$\Phi_1(s) = (1-h) + h\alpha(1/s - 1/s^3 + 3/s^5 - \ldots).$$

The first equation of (4) yields:

$$\Phi_0(s) = 1/z_0 s - 1/s^2 + 1/s^3 - 3/s^6 - \ldots$$

$$1 - p_0(t) = t/z_0 - t^2/2! + t^4/4! - 3t^6/6! + \ldots$$

An algorithm to calculate the probabilities $p_k(t)$ for $k = 0, 1, 2 \ldots$ and for the special value t obtained from condition (7) is described below. Please note that the three steps refer to the three equations (4).

Step 1 – Define: $\Phi_1(s) = z_0 + z_1/s + z_2/s^2 + \ldots$

Calculate: $\Phi_0(s) = (1 - \Phi_1)/z_0 s = y_1/s + y_2/s^2 + \ldots$

Calculate: $p_0(t) = 1 - y_1 t - y_2(t^2/2!) - \ldots$

Step 2 – Calculate: $\Psi_1 = \Phi_0(1 - \Phi_1) = \big((1 - z_0)y_1\big)/s +$

$\big((1 - z_0)y_2 - z_1 y_1\big)/s^2 + \ldots$

Redefine: $\Psi_1 = y_1/s + y_2/s^2 + \ldots$

Calculate: $p_1(t) = y_1 t + y_2 \cdot t^2/2! + \ldots$

Set: $k = 1$

Step 3 – Calculate: $\Psi_{k+1} = \Psi_k \Phi_1 = z_0 y_2/s + (z_0 y_2 + z_1 y_1)/s^2 + \ldots$

Redefine: $\Psi_{k+1} = y_1/s + y_2/s^2 + \ldots$

Calculate: $p_{k+1}(t) = y_1 t + y_2 t^2/2! + \ldots$

Set: $k = k + 1$, return to Step 3.

In the computer language APL, Step 3 would read:

$$W \leftarrow W,\ T + \cdot \times Y \leftarrow Z + \cdot \times Y. \tag{8}$$

Please note the APL statements are evaluated from the right to the left. The symbols contained in (8) have the following meanings:

Y is a vector which holds the first n coefficients y_j of Ψ_k, to be replaced by the coefficients y_j of Ψ_{k+1}.

Z is a matrix built from the first n coefficients of Φ_1. For $n = 3$, Z looks like:

$$\begin{matrix} z_0 & 0 & 0 \\ z_1 & z_0 & 0 \\ z_2 & z_1 & z_0 \end{matrix}$$

T is a vector holding the values $t, t^2/2!, \ldots, t^n/n!$

W is a vector holding the values $p_k(t)$. Each time Step 3 is traversed, the value $p_{k+1}(t)$ is appended to W.

4. Numerical Results

The APL program described above runs fast even on a micro-computer. The main practical difficulty is that we do not know beforehand how far to extend the vector Z. The length of n of Z should be determined by the condition that

$$t^n/n! = (\alpha h \hat{\mu}_1)^n/n!$$

becomes insignificantly small. However, it is advisable to make additional control runs with increased values of n. Of course, the required length n of Z may exceed the computer space; then the method suggested in this paper has to be abandoned.

For the data considered by Tellenbach, final results were already achieved for $n = 5$.

Attached are copies of two sample runs; namely, for $h = 0.86$ and 0.83. The former run yields the lowest sum of error-squares; this is the measure used by Tellenbach. The latter run (with $h = 0.83$) approximates the actual second moment $\mu_2(t) = \sum k^2 p_k(t)$. The printouts apply the terminology of Tellenbach which differs from the terminology used in this paper as follows:

Tellenbach	This paper	Comments
$Q(s)$	$P_0(t)$	1 − Error Function
$P_1(s)$	$P(t)$	Gauss
p	h	

The printouts display:

$$\varepsilon = t^n/n!$$

$$P, T = h, t$$

$$\text{MOM} = \sum k^j p_k(t) \quad \text{for } j = 0, 1 \text{ and } 2$$

$$\text{ERR} = \sum (p_k(t) - \hat{p}_k)^j \quad \text{for } j = 1 \text{ and } 2$$

The columns show:

$$N = k$$
$$P[N] = p_k(t)$$
$$P - \underline{P} = p_k(t) - \hat{p}_k$$
$$(P - \underline{P})^{*}2 = (p_k(t) - \hat{p}_k)^2$$

Appendix 1 compares the actual number of claims (namely, $\hat{p}_k \cdot 9{,}461$) with the corresponding figures from Tellenbach and the runs for $h = 0.86$ and $h = 0.83$, respectively.

The run $h = 0.86$ comes reasonably close to the Monte-Carlo results of Tellenbach. This may justify the "naive" approach suggested in this paper.

The run $h = 0.83$ fits better to the tail than the run $h = 0.86$, but is less accurate for $k = 0$ through 4.

Appendix 1

Number of Claims	Number of Policies: Actual	Tellenbach (Monte-Carlo)	Computer-Algorithm h = .86	Computer-Algorithm h = .83
0	7,840	7,831	7,819.1	7,872.9
1	1,317	1,311	1,327.6	1,242.1
2	239	255	255.6	271.8
3	42	54	47.9	58.5
4	14	9	8.8	12.4
5	4	1.3	1.6	2.6
6	4	0	0.3	0.5
7	1	0	0	0.1
	9,461	9,461.3	9,460.9	9,460.9
$\hat{\mu}_1$ =	0.214	0.214	0.214	0.214
$\hat{\mu}_2$ =	0.335	0.316	0.315	0.333
$\sum(p_n - \hat{p}_n)^2$ =		566	907	8,058

Note: Computer printout figures are multiplied by 9,461. For example: 0.826453 · 9,461 = 7,819.1.

```
0.83
*** G=1-ERF *** P1=GAUSS ***

ε=4.8036E-7
P,T = 0.83 0.14195
MOM = 1 0.21435 0.33249
ERR = -2.4438E-8 9.0058E-5
  0    .832145    .003480    .000012
  1    .131286   -.007917    .000063
  2    .028732    .003471    .000012
  3    .006180    .001740    .000003
  4    .001310   -.000169    .000000
  5    .000275   -.000148    .000000
  6    .000057   -.000366    .000000
  7    .000012   -.000094    .000000
  8    .000002    .000002    .000000
  9    .000000    .000000    .000000
 10    .000000    .000000    .000000
  N     PEN3       P-P       (P-P)*2

ε=6.4503E-14
P,T = 0.83 0.14195
MOM = 1 0.21435 0.3325
ERR = -1.0693E-14 9.006E-5
  0    .832145    .003480    .000012
  1    .131286   -.007917    .000063
  2    .028732    .003471    .000012
  3    .006180    .001740    .000003
  4    .001310   -.000169    .000000
  5    .000275   -.000148    .000000
  6    .000057   -.000366    .000000
  7    .000012   -.000094    .000000
  8    .000002    .000002    .000000
  9    .000000    .000000    .000000
 10    .000000    .000000    .000000
 11    .000000    .000000    .000000
 12    .000000    .000000    .000000
 13    .000000    .000000    .000000
 14    .000000    .000000    .000000
 15    .000000    .000000    .000000
 16    .000000    .000000    .000000
 17    .000000    .000000    .000000
 18    .000000    .000000    .000000
 19    .000000    .000000    .000000
  N     PEN3       P-P       (P-P)*2
```

```
0.86
*** G=1-ERF *** P1=GAUSS ***

ε=5.7367E-7
P,T = 0.86 0.14709
MOM = 1 0.21435 0.31451
ERR = -3.2897E-8 1.0159E-5
  0    .826453   -.002212    .000005
  1    .140327    .001124    .000001
  2    .027016    .001755    .000003
  3    .005067    .000628    .000000
  4    .000931   -.000549    .000000
  5    .000168   -.000254    .000000
  6    .000030   -.000393    .000000
  7    .000005   -.000100    .000000
  8    .000001    .000001    .000000
  9    .000000    .000000    .000000
  N     PEN3       P-P       (P-P)*2

ε=8.879E-14
P,T = 0.86 0.14709
MOM = 1 0.21435 0.31452
ERR = -3.3131E-15 1.0158E-5
  0    .826453   -.002212    .000005
  1    .140327    .001124    .000001
  2    .027016    .001755    .000003
  3    .005067    .000628    .000000
  4    .000931   -.000549    .000000
  5    .000168   -.000254    .000000
  6    .000030   -.000393    .000000
  7    .000005   -.000100    .000000
  8    .000001    .000001    .000000
  9    .000000    .000000    .000000
 10    .000000    .000000    .000000
 11    .000000    .000000    .000000
 12    .000000    .000000    .000000
 13    .000000    .000000    .000000
 14    .000000    .000000    .000000
 15    .000000    .000000    .000000
 16    .000000    .000000    .000000
 17    .000000    .000000    .000000
 18    .000000    .000000    .000000
  N     PEN3       P-P       (P-P)*2
```

References

Seal, H. (1969): Stochastic Theory of a Risk Business, John Wiley & Sons

Tellenbach, U. (1977): Berechnung der Verteilung der Schadenzahlen bei bekannter Verteilung der Wartzeiten, MSV 77.1, 35–46.

Thyrion, P. (1961): Contribution à l'étude du bonus pour un sinistre en assurance automobile, ASTIN Bull. 1, 142–162.

DISCUSSION OF PRECEDING PAPER

VOICE FROM FLOOR: Could you tell us why your results differ from Tellenbach's?

GOTTFRIED BERGER: Yes. You must distinguish between the APL algorithm, which renders exact numerical results, on the one hand, and Monte-Carlo techniques on the other hand. As to the APL algorithm, I came rather close to the Monte Carlo results of Tellenbach. But when I tried Monte Carlo techniques myself, I could not match the numberical findings of Tellenbach. It may well be that, for instance, my particular APL machine has a bad random number generator. Or maybe I did not go far enough, say to 10,000 random experiments rather than 1,000. At any rate, I feel uneasy about the confidence interval of Monte Carlo experiments, and my point is, one can avoid them in many instances and use exact algorithms instead.

BOB WILLIAMS: Are you acquainted with the recent paper by Harry Panjer? On the risk model? The paper that calculates the probability exactly? Starting from the left and going to the right, one by one?

GOTTFRIED BERGER: Harry Panjer discovered a recursion formula which makes it possible to calculate claims frequencies step by step, starting at zero claims. This algorithm works within a wide family of claim number distributions, including Poisson, Binomial and Negative Binomial. This can be useful for numerical Stop Loss calculations.

The APL algorithm which I have presented, has quite different applications. The idea is to try various assumptions as to the in-

terclaim time distribution, these assumptions are the input. The output are numerical values for the claims frequency. Suppose now we have empirical data. Then we can try to find input assumptions whcih match these empirical data. The APL algorithm runs so fast that we can do many such experiments. If successful, we find a model which can be used for rate making purposes.

As an example, we considered empirical data from automobile insurance, but we can think of many other applications, particularly if we remove the restriction that interclaim times are identically distributed for all renewal claims.

Thank you.