ETH Zürich, D-MATH
HS 2022
Prof. Dr. Mario V. Wüthrich

Coordinator
Davide Apolloni

Non-Life Insurance: Mathematics and Statistics

Solution sheet 8

Solution 8.1 Panjer Algorithm

For the expected yearly claim amount π_0 we have

$$\pi_0 \;=\; \mathbb{E}[S] \;=\; \mathbb{E}[N]\,\mathbb{E}[Y_1] \;=\; 1\cdot\mathbb{E}[k+Z] \;=\; k+\mathbb{E}[Z] \;=\; k+\exp\left\{\mu+\frac{\sigma^2}{2}\right\} \;\approx\; 4'124.$$

Let Y_i^+ denote the discretized claim sizes using a span of $s=10$, where we put all the probability mass to the upper end of the intervals. Note that $k=10s$. If we write $g_l=\mathbb{P}[Y_1^+=sl]$ for all $l\in\mathbb{N}$, then we have

$$g_1 \;=\; g_2 \;=\; \cdots \;=\; g_{10} \;=\; 0,$$

since $\mathbb{P}[Y_1^+\le 10s]=\mathbb{P}[k+Z\le 10s]=\mathbb{P}[Z\le 0]=0$. For all $l\ge 11$ we get

$$\begin{aligned} g_l \;&=\; \mathbb{P}[Y_1^+=sl] \;=\; \mathbb{P}[Y_1^+=k+s(l-10)] \;=\; \mathbb{P}[k+s(l-11)<Y_1\le k+s(l-10)] \\ &=\; \mathbb{P}[Y_1\le k+s(l-10)]-\mathbb{P}[Y_1\le k+s(l-11)] \;=\; \mathbb{P}[Z\le s(l-10)]-\mathbb{P}[Z\le s(l-11)] \\ &=\; \Phi\left(\frac{\log[s(l-10)]-\mu}{\sigma}\right)-\Phi\left(\frac{\log[s(l-11)]-\mu}{\sigma}\right), \end{aligned}$$

where Φ is the distribution function of the standard Gaussian distribution and where we define $\log 0=-\infty$. From now on we replace the original claim sizes Y_i with the discretized claim sizes Y_i^+, but, by a slight abuse of notation, we still write S for the yearly claim amount.
Note that $N\sim\text{Poi}(1)$ has a Panjer distribution with parameters $a=0$ and $b=1$, see Corollary 4.8 of the lecture notes (version of February 7, 2022). Applying the Panjer algorithm given in Theorem 4.9 of the lecture notes (version of February 7, 2022), we have for $r\in\mathbb{N}_0$

$$f_r \;\overset{\text{def.}}{=}\; \mathbb{P}[S=sr] \;=\; \begin{cases} \mathbb{P}[N=0], & \text{for } r=0,\\ \sum_{l=1}^{r}\frac{l}{r}g_l f_{r-l}, & \text{for } r>0. \end{cases}$$

Since the yearly amount that the client has to pay by himself is given by

$$S_{\text{ins}} \;=\; \min\{S,d\}+\min\{\alpha\cdot(S-d)_+,M\} \;=\; \min\{S,d\}+\alpha\cdot\min\left\{(S-d)_+,\frac{M}{\alpha}\right\},$$

$M/\alpha=7'000$ and the maximal possible franchise is 2'500, we have to apply the Panjer algorithm until we reach $\mathbb{P}[S=9'500]=f_{950}$. Here we limit ourselves to determine the values of $f_0,\ldots,f_{12}$ to illustrate how the algorithm works. We have

$$f_0 \;=\; \mathbb{P}[N=0] \;=\; e^{-1} \;\approx\; 0.37$$

and

$$f_1 \;=\; f_2 \;=\; \cdots \;=\; f_{10}=0,$$

since $g_1=g_2=\cdots=g_{10}=0$. For $r=11$ and $r=12$ we get

$$f_{11} \;=\; \sum_{l=1}^{11}\frac{l}{11}g_l f_{11-l} \;=\; g_{11}f_0 \;=\; \left[\Phi\left(\frac{\log s-\mu}{\sigma}\right)-\Phi\left(\frac{\log 0-\mu}{\sigma}\right)\right]e^{-1} \;\approx\; 7.089\cdot 10^{-9}$$

and

$$f_{12} = \sum_{l=1}^{12} \frac{l}{12} g_l f_{12-l} = g_{12} f_0 = \left[\Phi\left(\frac{\log 2s - \mu}{\sigma} \right) - \Phi\left(\frac{\log s - \mu}{\sigma} \right) \right] e^{-1} \approx 2.786 \cdot 10^{-7}.$$

Using the discretized claim sizes, the yearly expected amount π_{ins} paid by the customer is given by

$$\pi_{\text{ins}} = \mathbb{E}[S_{\text{ins}}] = \mathbb{E}\left[\min\{S, d\}\right] + \alpha\, \mathbb{E}\left[\min\left\{(S-d)_+, \frac{M}{\alpha}\right\}\right],$$

where we have

$$\mathbb{E}\left[\min\{S, d\}\right] = \sum_{r=0}^{d/s} f_r sr + d\left(1 - \sum_{r=0}^{d/s} f_r\right) = d + \sum_{r=0}^{d/s} f_r (sr - d)$$

and

$$\begin{aligned}
\mathbb{E}\left[\min\left\{(S-d)_+, \frac{M}{\alpha}\right\}\right] &= \sum_{r=d/s+1}^{d/s+M/s\alpha} f_r (sr - d) + \frac{M}{\alpha}\left(1 - \sum_{r=0}^{d/s+M/s\alpha} f_r\right) \\
&= \frac{M}{\alpha} + \sum_{r=d/s+1}^{d/s+M/s\alpha} f_r \left(sr - d - \frac{M}{\alpha}\right) - \frac{M}{\alpha} \sum_{r=0}^{d/s} f_r.
\end{aligned}$$

Therefore, we get

$$\begin{aligned}
\pi_{\text{ins}} &= d + \sum_{r=0}^{d/s} f_r (sr - d) + \alpha \left[\frac{M}{\alpha} + \sum_{r=d/s+1}^{d/s+M/s\alpha} f_r \left(sr - d - \frac{M}{\alpha}\right) - \frac{M}{\alpha} \sum_{r=0}^{d/s} f_r \right] \\
&= d + M + \sum_{r=0}^{d/s} f_r (sr - d - M) + \sum_{r=d/s+1}^{d/s+M/s\alpha} \alpha f_r \left(sr - d - \frac{M}{\alpha}\right).
\end{aligned}$$

Finally, if the customer has chosen franchise d, then the monthly pure risk premium π is given by

$$\begin{aligned}
\pi &= \frac{\pi_0 - \pi_{\text{ins}}}{12} \\
&= \frac{1}{12} \left[k + \exp\left\{\mu + \frac{\sigma^2}{2}\right\} - d - M - \sum_{r=0}^{d/s} f_r (sr - d - M) - \sum_{r=d/s+1}^{d/s+M/s\alpha} \alpha f_r \left(sr - d - \frac{M}{\alpha}\right) \right].
\end{aligned}$$

In the end, we get the following monthly pure risk premiums π for the different franchises d:

franchise d	300	500	1'000	1'500	2'000	2'500
monthly pure risk premium π	307	297	274	253	233	216

Table 1: Monthly pure risk premiums π for the different franchises d.

More generally, the monthly pure risk premium π as a function of the franchise d, which is allowed to vary between 300 CHF and 2'500 CHF, is given in Figure 1. The R code used to calculate the values in Table 1 and to generate Figure 1 is given in Listings 1 and 2. Note that these monthly premiums only represent pure risk premiums. In order to get the premiums that the customer has to pay in the end, we would need to add an appropriate risk-loading, which may vary between different health insurance companies.

Listing 1: R code for Exercise 8.1 (Function to calculate risk premium).

```
KK_premium <- function(lambda, mu, sigma2, span, shift){
  M <- 9500
  m <- floor(M/span)
  k0 <- shift/span
  g_min <- array(0, dim=c(m+1,1))   ### mass put to the lower end of the interval
  for (k in (k0+1):(m+1)){
  g_min[k,1] <- pnorm(log((k-k0)*span), mean=mu, sd=sqrt(sigma2))-pnorm(log((k-k0-1)*span),
                mean=mu, sd=sqrt(sigma2))
  }
  g_max <- array(0, dim=c(m+1,1))   ### mass is put to the upper end of the interval
  g_max[2:(m+1),1] <- g_min[1:m,1]
  f1 <- matrix(0, nrow=m+1, ncol=3)     ### probability to get zero claims
  f1[1,1] <- exp(-lambda*(1-g_min[1,1]))
  f1[1,2] <- exp(-lambda*(1-g_max[1,1]))
  h1 <- matrix(0, nrow=m, ncol=3)     ### for values "l*g_{l}" of the discretized claim sizes
  for (i in 1:m){
    h1[i,1] <- g_min[i+1,1]*(i+1)
    h1[i,2] <- g_max[i+1,1]*(i+1)
  }
  for (r in 1:m){   ### Panjer algorithm (a=0 and b=lambda*v, which is just lambda here)
    f1[r+1,1] <- lambda/r*(t(f1[1:r,1])%*%h1[r:1,1])
    f1[r+1,2] <- lambda/r*(t(f1[1:r,2])%*%h1[r:1,2])
    f1[r+1,3] <- r*span
  }
  m1 <- 2500    ### maximal franchise
  m0 <- 300     ### minimal franchise
  i1 <- floor(m1/span+1)    ### number of iterations to m1
  i0 <- floor(m0/span+1)    ### number of iterations to m0
  franchise <- array(NA, c(i1,3))
  for (i in i0:i1){
    franchise[i,1] <- f1[i,3]   ### this represents the franchise
    franchise[i,2] <- sum(f1[1:i,1]*f1[1:i,3])+f1[i,3]*(1-sum(f1[1:i,1]))
    franchise[i,2] <- franchise[i,2]+sum(f1[(i+1):floor(i+7000/span),1]
                      *f1[2:floor(7000/span+1),3])*0.1
                      +700*(1-sum(f1[1:floor(i+7000/span),1]))
    franchise[i,3] <- sum(f1[1:i,2]*f1[1:i,3])+f1[i,3]*(1-sum(f1[1:i,2]))
    franchise[i,3] <- franchise[i,3]+sum(f1[(i+1):floor(i+7000/span),2]
                      *f1[2:floor(7000/span+1),3])*0.1+700*(1-sum(f1[1:floor(i+7000/span),2]))
  }
  price <- array(NA, c(i1, 3))
  price[,1] <- franchise[,1]    ### this represents the franchise
  price[,2:3] <- (lambda*(exp(mu+sigma2/2)+shift)-franchise[,2:3])/12
  price
}
```

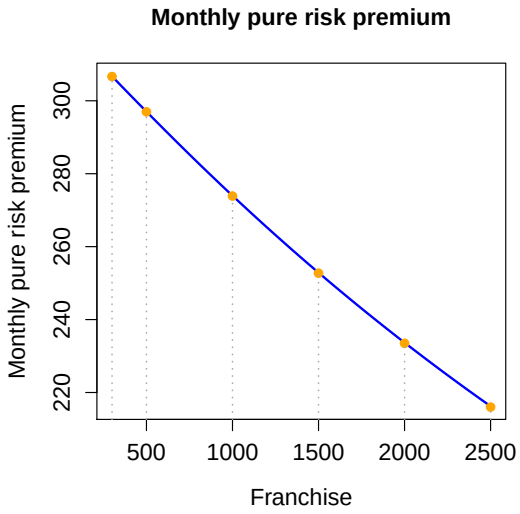

Figure 1: Plot of the monthly pure risk premium π as a function of the franchise d.

Listing 2: R code for Exercise 8.1 (Risk premium).

```
require(stats)
require(MASS)

### Run the function KK_premium
lambda <- 1
mu <- 7.8
sigma2 <- 1
span <- 10
shift <- 100
price <- KK_premium(lambda, mu, sigma2, span, shift)

### Plot the monthly pure risk premium as a function of the franchise
plot(x=price[,1], y=price[,2], lwd=2, col="blue", type='l', ylab="Monthly pure risk premium",
     xlab="Franchise", main="Monthly pure risk premium", cex.lab=1.25, cex.main=1.25,
     cex.axis=1.25)
points(x=c(300,500, 1000, 1500, 2000, 2500),
       y=price[c(300,500, 1000, 1500, 2000, 2500)/span+1,3], pch=19, col="orange")
lines(x=c(300,300), y=c(0,price[300/span+1,3]),lty=3, lwd=1.5, col="darkgray")
lines(x=c(500,500), y=c(0,price[500/span+1,3]),lty=3, lwd=1.5, col="darkgray")
lines(x=c(1000,1000), y=c(0,price[1000/span+1,3]),lty=3, lwd=1.5, col="darkgray")
lines(x=c(1500,1500), y=c(0,price[1500/span+1,3]),lty=3, lwd=1.5, col="darkgray")
lines(x=c(2000,2000), y=c(0,price[2000/span+1,3]),lty=3, lwd=1.5, col="darkgray")
lines(x=c(2500,2500), y=c(0,price[2500/span+1,3]),lty=3, lwd=1.5, col="darkgray")

### Give the monthly pure risk premiums for the six franchises listed on the exercise sheet
round(price[floor(c(300, 500, 1000, 1500, 2000, 2500)/span+1),2])
round(price[floor(c(300, 500, 1000, 1500, 2000, 2500)/span+1),3])
```

Solution 8.2 Monte Carlo Simulations

(a) We assume that for this comparably simple problem with no heavy tails 100'000 Monte Carlo simulations are enough to provide an empirical distribution function of S which is close to the true distribution function of S. The R codes used for part (a) are given in Listings 3 - 6.

Listing 3: R code for Exercise 8.2 (a) (Monte Carlo simulations).

```
compound.poisson.distribution <- Vectorize(function(n, lambdav, shape, rate){
  number.of.claims <- rpois(n=n, lambda=lambdav)
  sum(rgamma(n=number.of.claims, shape=shape, rate=rate))
},"n")
n <- 100000
lambdav <- 1000
shape <- 100
rate <- 1/10
set.seed(100)
claims <- compound.poisson.distribution(rep(1,n), lambdav, shape, rate)
```

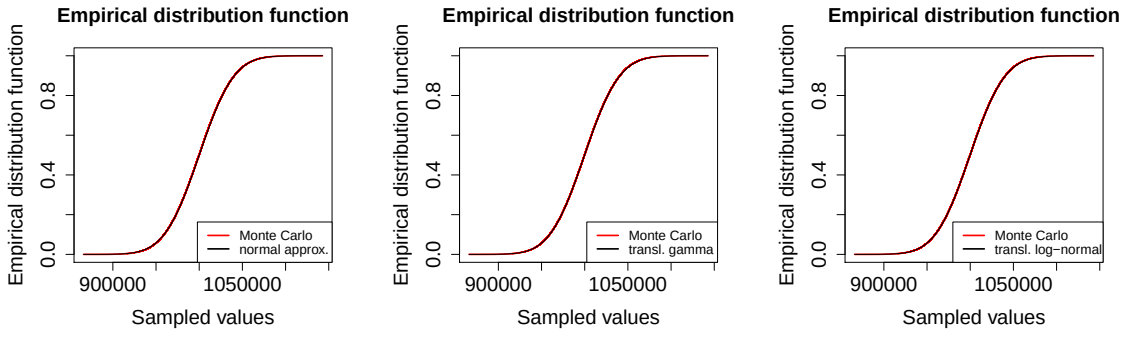

Figure 2: Comparison of the empirical distribution function of S resulting from 100'000 Monte Carlo simulations to the approximate distribution functions when using the normal (left), the translated gamma (middle) and the translated log-normal (right) approximation.

In Figure 2 we compare the empirical distribution function of S resulting from 100'000 Monte Carlo simulations to the approximate distribution functions when using the normal (left), the translated gamma (middle) and the translated log-normal (right) approximation. From these plots we cannot spot any differences between the various distribution functions. In Figure 3 we consider the log-log plot of the 100'000 Monte Carlo simulations of S and compare it to the normal (left), the translated gamma (middle) and the translated log-normal (right) approximation. We observe that all three approximations have a rather good fit to the tail of the distribution of S, but the translated gamma and the translated log-normal approximation seem slightly more accurate than the normal approximation. We conclude that in the absence of heavy tailed distributions the translated gamma and the translated log-normal approximation are very convincing in this example. Moreover, the skewness of S is small enough ($\varsigma_S \approx 0.0321$, see Exercise 7.4) and the expected number of claims large enough ($\lambda v = 1$'000, see Exercise 7.3) for the normal approximation to be a valid approximation, too.

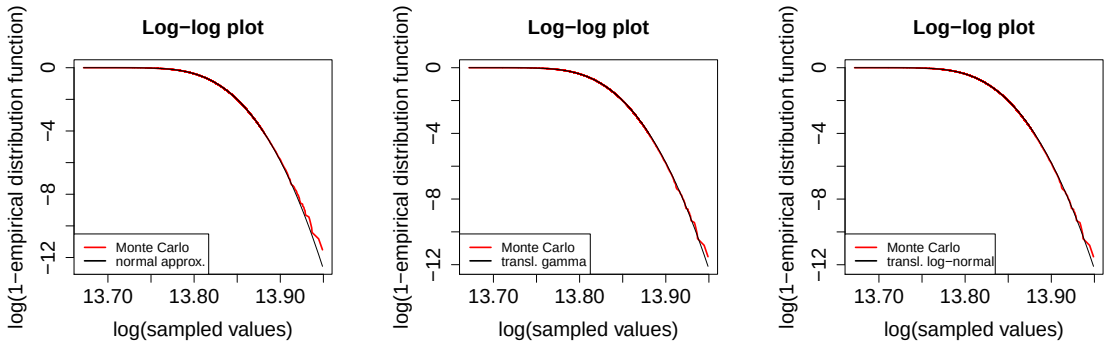

Figure 3: Log-log plot of the 100'000 Monte Carlo simulations of S compared to the normal (left), the translated gamma (middle) and the translated log-normal (right) approximation.

Listing 4: R code for Exercise 8.2 (a) (Normal approximation).

```
mu <- lambdav*shape/rate
sigma <- sqrt(lambdav*shape*(shape+1)/(rate^2))
par(mar=c(5.1, 4.4, 4.1, 2.1))
plot(claims[order(claims)], 1:n/(n+1), xlim=c(min(claims),max(claims)), type="l", col="red",
     main="Empirical distribution function", xlab="Sampled values",
     ylab="Empirical distribution function", cex.lab=1.5, cex.main=1.5, cex.axis=1.5, lwd=2)
lines(claims[order(claims)], pnorm((claims[order(claims)]),mu,sigma))
legend("bottomright", lty=1, lwd=2, col=c("red","black"),
       legend=c("Monte Carlo","normal approx.   "), cex=1)
plot(log(claims[order(claims)]), log(1-1:n/(n+1)), xlim=c(min(log(claims)),max(log(claims))),
     ylim=c(min(log(1-n/(n+1)),log(1-pnorm((claims[order(claims)]),mu,sigma))),0), type="l",
     col="red", main="Log-log plot", xlab="log(sampled values)",
     ylab="log(1-empirical distribution function)", cex.lab=1.5, cex.main=1.5, cex.axis=1.5,
     lwd=2)
lines(log(claims[order(claims)]), log(1-pnorm((claims[order(claims)]),mu,sigma)), col="black")
legend("bottomleft", lty=1, lwd=2, col=c("red","black"),
       legend=c("Monte Carlo","normal approx.   "), cex=1)
```

(b) Replicating 10'000 Monte Carlo simulations 100 times already requires some time. This is also the reason why we chose 10'000 as maximum number of simulations and not 100'000 as in part (a). Note that every single time we use Monte Carlo simulations to derive quantities like for example the quantiles $q_{0.95}$ and $q_{0.99}$, we get different results. This is something one needs to be aware of, and it is in contrast to the normal, the translated gamma and the translated log-normal approximation. In Figure 4 we show the densities of the 100 quantiles $q_{0.95}$ (left) and $q_{0.99}$ (right) resulting from Listing 7 where we replicate the $n \in \{100, 1'000, 10'000\}$ Monte Carlo simulations 100 times. We see that increasing the number of simulations n for every replication, the uncertainty regarding the quantiles $q_{0.95}$ and $q_{0.99}$ is reduced.

Listing 5: R code for Exercise 8.2 (a) (Translated gamma approximation).

```
skews <- (lambdav*shape*(shape+1)*(shape+2)/rate^3)/(lambdav*shape*(shape+1)/rate^2)^(3/2)
shape2 <- 4/skews^2
rate2 <- sqrt(shape2/(lambdav*shape*(shape+1)/rate^2))
k <- lambdav*shape/rate-shape2/rate2
plot(claims[order(claims)], 1:n/(n+1), xlim=c(min(claims),max(claims)), type="l", col="red",
     main="Empirical distribution function", xlab="Sampled values",
     ylab="Empirical distribution function", cex.lab=1.5, cex.main=1.5, cex.axis=1.5, lwd=2)
lines(claims[order(claims)], pgamma((claims[order(claims)])-k,shape=shape2,rate=rate2))
legend("bottomright", lty=1, lwd=2, col=c("red","black"),
       legend=c("Monte Carlo","transl. gamma   "), cex=1)
plot(log(claims[order(claims)]), log(1-1:n/(n+1)), xlim=c(min(log(claims)),max(log(claims))),
     ylim=c(min(log(1-n/(n+1)),log(1-pgamma((claims[order(claims)])-k,shape=shape2,
     rate=rate2))),0), type="l", col="red", main="Log-log plot", xlab="log(sampled values)",
     ylab="log(1-empirical distribution function)", cex.lab=1.5, cex.main=1.5, cex.axis=1.5,
     lwd=2)
lines(log(claims[order(claims)]), log(1-pgamma((claims[order(claims)])-k,shape=shape2,
      rate=rate2)))
legend("bottomleft", lty=1, lwd=2, col=c("red","black"),
       legend=c("Monte Carlo","transl. gamma   "), cex=1)
```

Listing 6: R code for Exercise 8.2 (a) (Translated log-normal approximation).

```
sigma.squared <- 0.00011444
mu2 <- 1/2*(log((exp(sigma.squared)-1)^(-1)*lambdav*shape*(shape+1)/rate^2)-sigma.squared)
k2 <- lambdav*shape/rate-exp(mu2+sigma.squared/2)
plot(claims[order(claims)], 1:n/(n+1), xlim=c(min(claims),max(claims)), type="l", col="red",
     main="Empirical distribution function", xlab="Sampled values",
     ylab="Empirical distribution function", cex.lab=1.5, cex.main=1.5, cex.axis=1.5, lwd=2)
lines(claims[order(claims)], pnorm(log((claims[order(claims)])-k2),mu2,sqrt(sigma.squared)))
legend("bottomright", lty=1, lwd=2, col=c("red","black"),
       legend=c("Monte Carlo","transl. log-normal    "), cex=1)
plot(log(claims[order(claims)]), log(1-1:n/(n+1)), xlim=c(min(log(claims)),max(log(claims))),
     ylim=c(min(log(1-n/(n+1)),log(1-pnorm(log((claims[order(claims)])-k2),mu2,
     sqrt(sigma.squared)))),0), type="l", col="red", main="Log-log plot",
     xlab="log(sampled values)", ylab="log(1-empirical distribution function)", cex.lab=1.5,
     cex.main=1.5, cex.axis=1.5, lwd=2)
lines(log(claims[order(claims)]), log(1-pnorm(log((claims[order(claims)])-k2),mu2,
      sqrt(sigma.squared))))
legend("bottomleft", lty=1, lwd=2, col=c("red","black"),
       legend=c("Monte Carlo","transl. log-normal    "), cex=1)
```

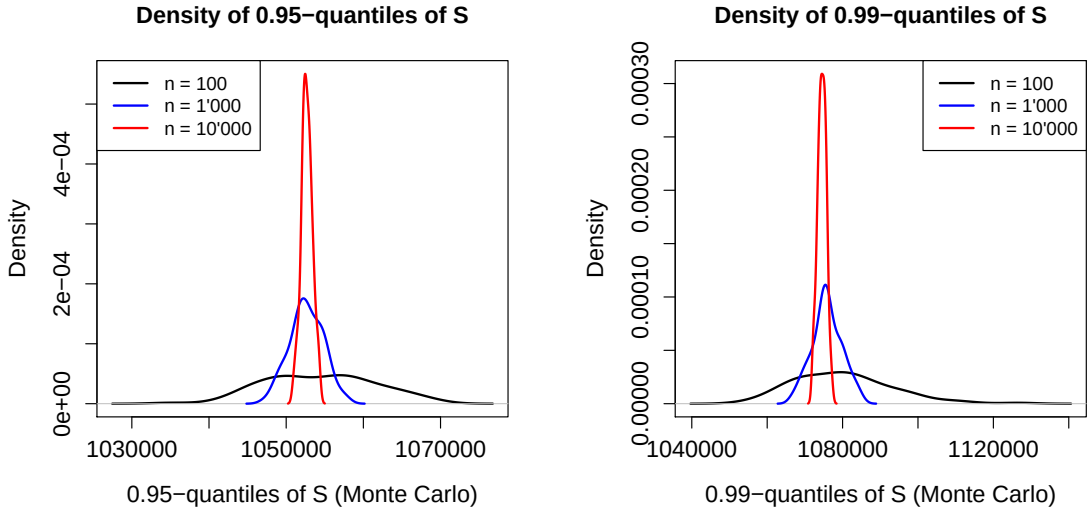

Figure 4: Densities of the 100 quantiles $q_{0.95}$ (left) and $q_{0.99}$ (right) resulting from replicating the $n \in \{100, 1'000, 10'000\}$ Monte Carlo simulations 100 times.

Listing 7: R code for Exercise 8.2 (b) (Quantiles).

```
### Monte Carlo simulations
k <- 100
n <- c(100,1000,10000)
set.seed(100)
claims.1 <- array(compound.poisson.distribution(n=rep(1,k*n[1]), lambdav=1000, shape=100,
                  rate=1/10), dim=c(n[1],k))
set.seed(200)
claims.2 <- array(compound.poisson.distribution(n=rep(1,k*n[2]), lambdav=1000, shape=100,
                  rate=1/10), dim=c(n[2],k))
set.seed(300)
claims.3 <- array(compound.poisson.distribution(n=rep(1,k*n[3]), lambdav=1000, shape=100,
                  rate=1/10), dim=c(n[3],k))

### Function calculating alpha-quantiles of S on the basis of Monte Carlo simulations of S
quantiles.monte.carlo <- function(claims, alpha){
  n <- nrow(claims)
  claims.sorted <- apply(claims, 2, sort)
  quantiles.alpha <- claims.sorted[floor(alpha*n)+1,]
}

### 0.95-quantiles
range(quantiles.1 <- quantiles.monte.carlo(claims=claims.1, alpha=0.95))
range(quantiles.2 <- quantiles.monte.carlo(claims=claims.2, alpha=0.95))
range(quantiles.3 <- quantiles.monte.carlo(claims=claims.3, alpha=0.95))

### Density
ymax <- max(density(quantiles.1)$y, density(quantiles.2)$y, density(quantiles.3)$y)
plot(density(quantiles.1), col="black", ylim=c(0,ymax), main="Density of 0.95-quantiles of S",
     xlab="0.95-quantiles of S (Monte Carlo)", cex.lab=1.25, cex.main=1.25, cex.axis=1.25,
     lwd=2)
lines(density(quantiles.2), col="blue", lwd=2)
lines(density(quantiles.3), col="red", lwd=2)
legend("topleft", col=c("black", "blue", "red"), lwd=2, lty=1,
       legend=c("n = 100","n = 1'000","n = 10'000"))

### 0.99-quantiles
range(quantiles.1 <- quantiles.monte.carlo(claims=claims.1, alpha=0.99))
range(quantiles.2 <- quantiles.monte.carlo(claims=claims.2, alpha=0.99))
range(quantiles.3 <- quantiles.monte.carlo(claims=claims.3, alpha=0.99))

### Density
ymax <- max(density(quantiles.1)$y, density(quantiles.2)$y, density(quantiles.3)$y)
plot(density(quantiles.1), col="black", ylim=c(0,ymax), main="Density of 0.99-quantiles of S",
     xlab="0.99-quantiles of S (Monte Carlo)", cex.lab=1.25, cex.main=1.25, cex.axis=1.25,
     lwd=2)
lines(density(quantiles.2), col="blue", lwd=2)
lines(density(quantiles.3), col="red", lwd=2)
legend("topright", col=c("black", "blue", "red"), lwd=2, lty=1,
       legend=c("n = 100","n = 1'000","n = 10'000"))
```

	$q_{0.95}$		$q_{0.99}$	
Monte Carlo	smallest	largest	smallest	largest
$n = 100$	1'035'018	1'069'209	1'053'719	1'126'533
$n = 1'000$	1'047'186	1'057'829	1'066'770	1'084'902
$n = 10'000$	1'050'955	1'054'282	1'072'045	1'077'195
Approximations				
normal	1'052'274		1'073'932	
translated gamma	1'052'563		1'074'682	
translated log-normal	1'052'562		1'074'684	

Table 2: Smallest and largest observed values of the quantiles $q_{0.95}$ and $q_{0.99}$ among the 100 replications of the $n \in \{100, 1'000, 10'000\}$ Monte Carlo simulations together with the values of the quantiles $q_{0.95}$ and $q_{0.99}$ resulting from the normal, the translated gamma and the translated log-normal approximation.

One can reach the same conclusions from Table 2, where we give the smallest and the largest observed values of the quantiles $q_{0.95}$ and $q_{0.99}$ among the 100 replications of the $n \in \{100, 1'000, 10'000\}$ Monte Carlo simulations. Moreover, we also give the values of the quantiles $q_{0.95}$ and $q_{0.99}$ resulting from the normal, the translated gamma and the translated log-normal approximation, see Exercises 7.3 and 7.4. We see that the quantiles resulting from the approximations are always between the smallest and the largest observed value resulting from the Monte Carlo simulations. Of course, one can argue that we could choose the number of simulations n large enough such that the results do not vary considerably anymore. However, a too high number of simulations n will lead to an excessive computation time. This is especially true if one considers heavy tailed distributions. Therefore, one is often inclined to use other algorithms for compound distributions, such as the Panjer algorithm and fast Fourier transforms.

Solution 8.3 Fast Fourier Transform

Assume that $\widetilde{Y}$ follows the claim size distribution given on the exercise sheet. Let Y denote the discretized version of $\widetilde{Y}$ that takes values in $\mathbb{N}_0$. More precisely, we shift the probability masses of $\widetilde{Y}$ to the right and define

$$\mathbb{P}[Y = 0] \,=\, 0 \qquad \text{and} \qquad \mathbb{P}[Y = l] \,=\, \mathbb{P}\left[\widetilde{Y} \le l\right] - \mathbb{P}\left[\widetilde{Y} \le l - 1\right],$$

for all $l \in \mathbb{N}$. By a slight abuse of notation, we still write S for the compound Poisson distribution with discrete claim size distribution Y. In particular, also S takes values in $\mathbb{N}_0$. We define

$$g_l \,=\, \mathbb{P}[Y = l] \qquad \text{and} \qquad f_l \,=\, \mathbb{P}[S = l],$$

for all $l \in \mathbb{N}_0$. We choose a threshold of $n = 2'000'000$, i.e. we determine the distribution function of S up to $n - 1$. Note that n is chosen sufficiently high such that we approximately have

$$P[Y > n - 1] \,\approx\, 0. \tag{1}$$

We define $\mathcal{A} = \{0, \dots, n - 1\}$ and calculate the discrete Fourier transform $(\hat{g}_z)_{z \in \mathcal{A}}$ of $(g_l)_{l \in \mathcal{A}}$ by

$$\hat{g}_z \,=\, \sum_{l=0}^{n-1} g_l \exp\left\{2\pi i \frac{zl}{n}\right\}, \tag{2}$$

for all $z \in \mathcal{A}$. Due to (1), we approximately have

$$\hat{g}_z \,\approx\, \mathbb{E}\left[\exp\left\{2\pi i \frac{zY}{n}\right\}\right] \,=\, M_Y\left(2\pi i \frac{z}{n}\right),$$

for all $z \in \mathcal{A}$, where M_Y denotes the moment generating function of Y. Note that we use an extended version of the moment generating function also allowing for complex numbers. If M_S denotes the moment generating function of S, again extended to complex numbers, then, according to Proposition 2.11 of the lecture notes (version of February 7, 2022), we have

$$M_S\left(2\pi i \frac{z}{n}\right) \,=\, \exp\left\{\lambda v \left[M_Y\left(2\pi i \frac{z}{n}\right) - 1\right]\right\} \,\approx\, \exp\left\{\lambda v \left(\hat{g}_z - 1\right)\right\}, \tag{3}$$

for all $z \in \mathcal{A}$. The left hand side of equation (3) can be written as

$$M_S\left(2\pi i \frac{z}{n}\right) \,=\, \sum_{l=0}^{\infty} f_l \exp\left\{2\pi i \frac{zl}{n}\right\} \,=\, \sum_{l=0}^{n-1} \left(f_l + \sum_{k=1}^{\infty} f_{l+kn}\right) \exp\left\{2\pi i \frac{zl}{n}\right\},$$

for all $z \in \mathcal{A}$. Using the approximation

$$f_l \;\approx\; f_l + \sum_{k=1}^{\infty} f_{l+kn}, \tag{4}$$

for all $l \in \mathcal{A}$, we compute the discrete Fourier transform $(\hat{f}_z)_{z\in\mathcal{A}}$ of $(f_l)_{l\in\mathcal{A}}$ by

$$\hat{f}_z \;=\; \sum_{l=0}^{n-1} f_l \exp\left\{2\pi i \frac{zl}{n}\right\} \;\approx\; M_S\left(2\pi i \frac{z}{n}\right) \;\approx\; \exp\left\{\lambda v\left(\hat{g}_z - 1\right)\right\},$$

for all $z \in \mathcal{A}$. Applying the inversion formula of the discrete Fourier transform, we finally calculate

$$f_l \;=\; \frac{1}{n}\sum_{z=0}^{n-1} \hat{f}_z \exp\left\{-2\pi i \frac{zl}{n}\right\}, \tag{5}$$

for all $l \in \mathcal{A}$. Note that due to the approximation in (4), instead of f_l we actually calculate

$$f_l + \sum_{k=1}^{\infty} f_{l+kn} \;>\; f_l,$$

for all $l \in \mathcal{A}$. This error is called wrap around error (or aliasing error), and n should be chosen large enough in order to keep this wrap around error small.

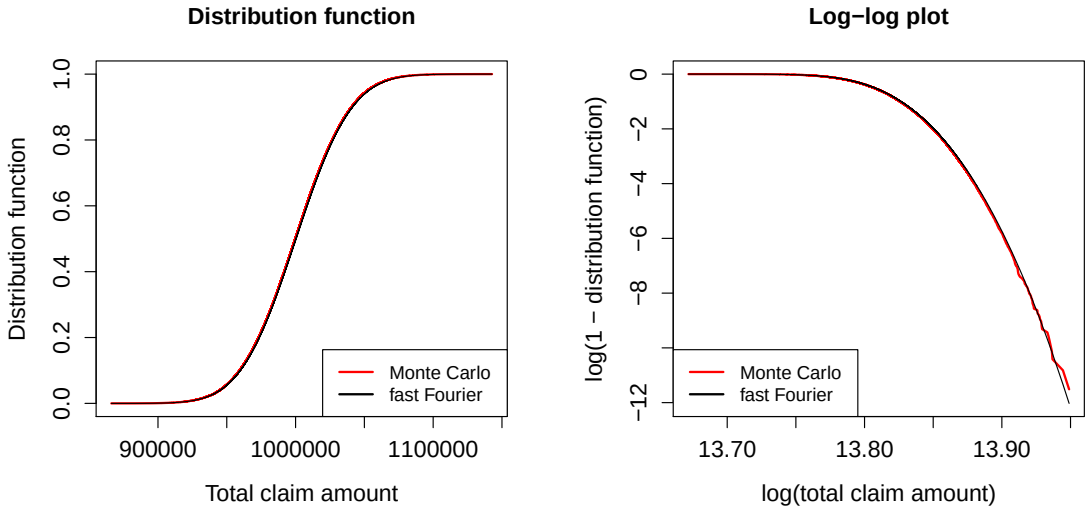

Figure 5: Comparison of the distribution function (left) and the log-log plot (right) of S resulting from the fast Fourier transform algorithm to the Monte Carlo simulations.

In R, the calculations in equations (2) and (5) can be done using the command `fft`. The corresponding R code is given in Listing 8. In Figure 5 we compare the distribution function (left) and the log-log plot (right) of S resulting from the fast Fourier transform algorithm to the Monte Carlo simulations of Exercise 8.2. We see that we get a very good fit. In particular, the threshold $n = 2'000'000$ seems to be high enough. For the 0.95-quantile $q_{0.95}$ and the 0.99-quantile $q_{0.99}$ we get

$$q_{0.95} \;=\; 1'053'089 \qquad \text{and} \qquad q_{0.99} \;=\; 1'075'215.$$

We see that we get values which are very close to the ones derived in Exercises 7.3 and 7.4, where we used the normal, the translated gamma and the translated log-normal approximation.

Listing 8: R code for Exercise 8.3.

```
### Fast Fourier transfom
n <- 2000000
lambdav <- 1000
claim.size <- c(0, pgamma(1:(n-1), shape=100, rate=1/10)-pgamma(0:(n-2), shape=100, rate=1/10))
claim.size.ft <- fft(claim.size)
total.claim.amount.ft <- exp(lambdav*(claim.size.ft-1))
total.claim.amount <- Re(fft(total.claim.amount.ft,inverse=TRUE)/length(total.claim.amount.ft))

### Monte Carlo simulations from Exercise 8.2
compound.poisson.distribution <- Vectorize(function(n, lambdav, shape, rate){
  number.of.claims <- rpois(n=n, lambda=lambdav)
  sum(rgamma(n = number.of.claims, shape=shape, rate=rate))
},"n")
m <- 100000
set.seed(100)
claim.amounts <- compound.poisson.distribution(n=rep(1,m), lambdav=1000, shape=100, rate=1/10)

### Calculate values of the distribution function of S using the fast Fourier transfrom
probabilities <- cumsum(total.claim.amount)[floor(claim.amounts[order(claim.amounts)])+1]

### Check the fast Fourier transform result
par(mar=c(5.1, 4.4, 4.1, 2.1))
plot(claim.amounts[order(claim.amounts)], 1:m/(m+1),
     xlim=c(min(claim.amounts),max(claim.amounts)), type="l", col="red",
     main="Distribution function", xlab="Total claim amount", ylab="Distribution function",
     cex.lab=1.25, cex.main=1.25, cex.axis=1.25, lwd=2)
lines(claim.amounts[order(claim.amounts)], probabilities, lwd=1)
legend("bottomright", lty=1, lwd=2, col=c("red","black"), legend=c("Monte Carlo","fast Fourier
"), cex=1)
plot(log(claim.amounts[order(claim.amounts)]), log(1-1:m/(m+1)),
     xlim=c(min(log(claim.amounts)),max(log(claim.amounts))),
     ylim=c(min(log(1-m/(m+1)),log(1-probabilities)),0), type="l", col="red",
     main="Log-log plot", xlab="log(total claim amount)", ylab="log(1 - distribution function)",
     cex.lab=1.25, cex.main=1.25, cex.axis=1.25, lwd=2)
lines(log(claim.amounts[order(claim.amounts)]), log(1-probabilities), col="black", lwd=1)
legend("bottomleft", lty=1, lwd=2, col=c("red","black"), legend=c("Monte Carlo","fast Fourier
"), cex=1)

### Determine the 0.95- and the 0.99-quantiles
which(cumsum(total.claim.amount) > 0.95)[1]-1
which(cumsum(total.claim.amount) > 0.99)[1]-1
```

Solution 8.4 Panjer Distribution

If we write $p_k = \mathbb{P}[N = k]$, for all $k \in \mathbb{N}$, then, by definition of the Panjer distribution, we have

$$p_k = p_{k-1}\left(a + \frac{b}{k}\right),$$

for all $k \in \mathbb{N}$. We can use this recursion to calculate $\mathbb{E}[N]$ and $\mathrm{Var}(N)$. Note that the range of N is $\mathbb{N}$, if $a \geq 0$, and $\{0, 1, \ldots, n\}$ for some $n \in \mathbb{N}_{\geq 1}$, if $a < 0$.
First, we consider the case where $a < 0$, i.e. where the range of N is $\{0, 1, \ldots, n\}$. According to the proof of Lemma 4.7 of the lecture notes (version of February 7, 2022), we have

$$n = -\frac{a+b}{a}. \tag{6}$$

For the expectation of N we get

$$\begin{aligned}
\mathbb{E}[N] &= \sum_{k=0}^{n} k\, p_k = \sum_{k=1}^{n} k\, p_k = \sum_{k=1}^{n} k\, p_{k-1}\left(a + \frac{b}{k}\right) = a\sum_{k=1}^{n} k\, p_{k-1} + b\sum_{k=1}^{n} p_{k-1} \\
&= a\sum_{k=0}^{n-1}(k+1)\, p_k + b\sum_{k=0}^{n-1} p_k = a\sum_{k=0}^{n-1} k\, p_k + (a+b)\sum_{k=0}^{n-1} p_k = a\,(\mathbb{E}[N] - n p_n) + (a+b)(1-p_n) \\
&= a\,\mathbb{E}[N] + a + b + p_n(-an - a - b).
\end{aligned}$$

Using (6), we get

$$-an - a - b = a\,\frac{a+b}{a} - a - b = 0. \tag{7}$$

Hence, the above expression for $\mathbb{E}[N]$ simplifies to

$$\mathbb{E}[N] = a\mathbb{E}[N] + a + b,$$

from which we can conclude that

$$\mathbb{E}[N] = \frac{a+b}{1-a}.$$

In order to get the variance of N, we first calculate the second moment of N:

$$\begin{aligned}
\mathbb{E}[N^2] &= \sum_{k=0}^{n} k^2\, p_k = \sum_{k=1}^{n} k^2\, p_k = \sum_{k=1}^{n} k^2\, p_{k-1}\left(a + \frac{b}{k}\right) = a\sum_{k=1}^{n} k^2\, p_{k-1} + b\sum_{k=1}^{n} k\, p_{k-1} \\
&= a\sum_{k=0}^{n-1}(k+1)^2\, p_k + b\sum_{k=0}^{n-1}(k+1)\, p_k = a\sum_{k=0}^{n-1} k^2\, p_k + (2a+b)\sum_{k=0}^{n-1} k\, p_k + (a+b)\sum_{k=0}^{n-1} p_k \\
&= a\,(\mathbb{E}[N^2] - n^2 p_n) + (2a+b)(\mathbb{E}[N] - np_n) + (a+b)(1-p_n) \\
&= a\,\mathbb{E}[N^2] + (2a+b)\,\mathbb{E}[N] + a + b + p_n[-an^2 - (2a+b)n - a - b].
\end{aligned}$$

Using (6), we get

$$\begin{aligned}
-an^2 - (2a+b)n - a - b &= -a\left(\frac{a+b}{a}\right)^2 + (2a+b)\frac{a+b}{a} - a - b \\
&= -\frac{a^2+2ab+b^2}{a} + \frac{2a^2+3ab+b^2}{a} - \frac{a^2+ab}{a} \\
&= 0.
\end{aligned} \tag{8}$$

Hence, the above expression for $\mathbb{E}[N^2]$ simplifies to

$$\mathbb{E}[N^2] = a\,\mathbb{E}[N^2] + (2a+b)\,\mathbb{E}[N] + a + b,$$

from which we get

$$\begin{aligned}
\mathbb{E}[N^2] &= \frac{(2a+b)\,\mathbb{E}[N] + a + b}{1-a} = \frac{(2a+b)\,(a+b) + (a+b)(1-a)}{(1-a)^2} \\
&= \frac{2a^2+3ab+b^2+a-a^2+b-ab}{(1-a)^2} = \frac{(a+b)^2+a+b}{(1-a)^2}.
\end{aligned}$$

Finally, the variance of N then is

$$\mathrm{Var}(N) = \mathbb{E}[N^2] - \mathbb{E}[N]^2 = \frac{(a+b)^2+a+b}{(1-a)^2} - \frac{(a+b)^2}{(1-a)^2} = \frac{a+b}{(1-a)^2}.$$

In the case where $a \geq 0$, i.e. where the range of N is $\mathbb{N}$, we can perform analogous calculations with the only difference that the index of summation in all the sums involved goes up to ∞ instead of stopping at n. As a consequence, the calculations in (7) and in (8) aren't necessary anymore. The formulas for $\mathbb{E}[N]$ and $\mathrm{Var}(N)$, however, remain the same.
The ratio of $\mathrm{Var}(N)$ to $\mathbb{E}[N]$ is given by

$$\frac{\mathrm{Var}(N)}{\mathbb{E}[N]} = \frac{a+b}{(1-a)^2}\,\frac{1-a}{a+b} = \frac{1}{1-a}.$$

Note that if $a < 0$, i.e. if N has a binomial distribution, we have $\mathrm{Var}(N) < \mathbb{E}[N]$. If $a = 0$, i.e. if N has a a Poisson distribution, we have $\mathrm{Var}(N) = \mathbb{E}[N]$. Finally, in the case of $a > 0$, i.e. for a negative-binomial distribution, we have $\mathrm{Var}(N) > \mathbb{E}[N]$.