UNIVERSITY OF CALGARY

On The Computation Of Ruin Probabilities

by

Lin Tan

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF MATHEMATICS AND STATISTICS

CALGARY, ALBERTA

December, 2009

UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "On The computation Of Ruin Probabilities" submitted by Lin Tan in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.

R. S. Ambagaspitiya

Supervisor
Dr. R. Ambagaspitiya
Department of Mathematics and Statistics

X. Lu

Dr. X. Lu
Department of Mathematics and Statistics

Dr. A. Fapojuwo
Department of Electrical & Computer Engineering

Dec 11 / 2009

Date

Abstract

In this thesis we study the of probability of ultimate ruin and the ruin-related quantities, including the distribution of the severity of ruin, the distribution of the surplus immediately prior to ruin, the joint distribution of the surplus immediately prior to ruin and the severity of ruin, the moments of the time to ruin and the density of the time to ruin.

We implement three methods: recursive approximation algorithms; lower and upper bounds; and simulation. The first method, based on a discrete model, uses a stable formula presented by Dickson and Waters. The second method, derived from Goovaerta and De Vylder, uses the connection between the probability of ruin and the maximal aggregate loss random variable, and that the latter has a compound geometric distribution. For the third method one observes that the probability of ruin is related to the stationary distribution of a certain associated process allowing it to be determined by simulation of the latter.

Keywords: Probability of ultimate ruin, Ruin theory, Probability of ultimate survival, Simulation, Recursive calculation, Stable algorithm.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

It is difficult to assess the risk associated with a portfolio of insurance contracts. It is nevertheless important to attempt to do so in order to ensure the viability of an insurance operation. The distribution of total claims over a fixed period of time is an obvious input parameter to such a process. The common approach to follow the fortunes of the policy or portfolio is called **ruin theory** in which the quantity of interest is the amount of surplus with ruin occurring when the surplus drops to below zero.

In order to track the variations in amount of an insurer's surplus over an extend period of time, we build a mathematical model by simplifying a real life insurance operation. This idealized model includes the initial fund, premiums collected over claims paid and claims paid as they occur.

Many ruin-related measures and quantities will be derived and calculated, including the probability of ultimate ruin, the distribution of an insurer's surplus immediately prior to ruin, the deficit at the time of ruin, the distribution of the first drop in surplus given that the drop occurs, etc.

In the next two sections we introduce the basic continuous time surplus model as well as the discrete model that approximates the basic model, including the definitions and notation. In last section, we define the ultimate ruin probability based on the continuous model and discrete model respectively.

1.1 The Classical Continuous-time Risk Model

An insurer's surplus is modeled as the result of two opposing cash flows: an incoming cash flow of premium income collected continuously at the rate of c; and an outgoing cash flow due to a sequence of insurance claims X_i that are mutually independent and identically distributed with common distribution function $P(x)$. The frequency of claims is assumed to follow a Poisson process with intensity rate λ, which means that the number of incurred claims $N(t)$ at time t is governed by a Poisson distribution with mean λt. Hence, the insurer's surplus $U(u,t)$ at any time t is given by

$$U(u,t) = u + c(t) - S(t) \qquad t \geq 0 \tag{1.1}$$

where u is the insurer's initial surplus. $c(t)$ denote premiums collected through time t will be a deterministic, not a stochastic process. $S(t) = \sum_{i=1}^{N(t)} X_i$ denote aggregate claims occurred through time t, called aggregate claims process.

To make this model simple, Dickson and Waters [8] assume

1. The distribution function of individual claim amounts $P(x) = 0$ for $x < 0$, so that all claim amounts are non-negative.

2. The mean of individual claims X_i which we denote p_1 is finite and that any other moments of X_i which we require are also finite.

3. The insurer's premium income is received continuously at positive rate c per unit time, then $c(t) = ct$.

4. The insurer's premium income exceeds the insurer's expected aggregate claim amounts, that is $c > \lambda p_1$. Relative security loading factor θ is defined to be $c = (1+\theta)\lambda p_1$

Without loss of generality, we can set both c and p_1 to be 1. We will refer to the process described above as our "basic process". It can be written as

$$U(u,t) = u + ct - \sum_{i=1}^{N(t)} X_i \qquad t \geq 0 \tag{1.2}$$

Later, we will remove the condition $c = 1$ and $p_1 = 1$ and extend it to a general process.

1.2 The Discrete-time Risk Model

As the continuous risk model tends to be difficult to analyze, we produce a discrete process to approximate our basic process. A discrete time surplus process considers the values of $U(u,t)$ at only integer values of t, denoted by $U_d(u,n)$ for $n = 0,1,2,\ldots$.

We rescale the basic process by multiplying all monetary amounts by some positive scalar β and taking a new time unit to be β^{-1} times the original time unit so that the premium income per unit time c for the rescaled process is still 1.

Let $X_{d,i}$ be a sequence of i.i.d random discrete variables whose common distribution is approximately the same as that of βX_i and which are distributed on the non-negative integers. We denote the probability function of $X_{d,i}$ by $f(k)$ so that $f_k = f(k) = \Pr(X_{d,i} = k)$, for $k = 0,1,2,\ldots$. Its common distribution function is $F_d(x) = \Pr(X_{d,i} \leq x) = \Pr(\beta X_i \leq x) = \Pr(X_i \leq x/\beta) = P(x/\beta)$.

Let $N_d(t)$ be defined to be $N(\beta^{-1}t)$ so that $\{N_d(t)\}_{t\geq 0}$ is a Poisson process with parameter $\lambda\beta^{-1}$. Now consider the discrete time surplus process $\{U_d(u,n)\}_{n=0}^{\infty}$ defined as

$$U_d(u,n) = u + n - \sum_{i=1}^{N_d(n)} X_{d,i} \tag{1.3}$$

so that the initial surplus is u and premium c is 1. The implied premium loading factor for this discrete surplus process will be denoted θ_d and is given by the formula

$$1 = (1+\theta_d)\lambda\beta^{-1}E[X_{d,i}] \tag{1.4}$$

If $E[X_{d,i}] = \beta p_1$, then $\theta_d = \theta$. We will always choose β and the distribution of $X_{d,i}$ to be such that θ_d is positive.

Let S_d denote the aggregate claims over the first time period for the discrete model. Its common distribution function and probability function are denoted by $H_d(k)$ and $h_d(k)$. So that

$$H_d(k) = \sum_{j=0}^{k} h_d(j) = \Pr(S_d \leq k) = \Pr\left(\sum_{i=1}^{N_d(1)} X_{d,i} \leq k\right) \quad \text{for } k = 0,1,2,\ldots \tag{1.5}$$

Then it is clear that for any integer n, $U_d(\beta u, \beta n)$ has approximately the same distribution as $U_d(u,n)$. It should also be clear that by increasing the value of β we ought to be able to improve this approximation.

1.3 General Process

The general process is not restricted with the assumptions $c = 1$, $p_1 = 1$. If we still want to use (1.3) to approximate (1.2), we need to amend our rescale procedure.

If $\theta_d = \theta, E(X_{d,i}) = \frac{\beta}{c} E(X_i)$, then (1.4) can be rewritten as

$$1 = (1+\theta) \cdot \lambda \beta^{-1} \cdot \frac{\beta}{c} E(X_i) \tag{1.6}$$

Move c to the left side of equation

$$c = (1+\theta) \cdot \lambda \cdot E(X_i) = (1+\theta) \cdot \lambda \cdot p_1 \tag{1.7}$$

We can recognize this is our continuous process with general c and p_1.

Our way to rescale the general process to an approximate discrete process is by multiplying all the monetary amount by β/c, and taking a new time unit to be β^{-1} times the original time unit. Then the discrete model has i.i.d claim amounts $X_{d,i}$ whose common distribution

is approximately the same as that of $\frac{\beta}{c}X_i$ and claim frequency $N_d(t)$ is Poisson process with parameter $\lambda\beta^{-1}$.

1.4 Ultimate Ruin Probability

In this section, we will use the models built in prior sections to define the ultimate ruin probability.

As the surplus process involves two opposite cash flows - the incoming cash of premium and outgoing cash of claims paid - the surplus might become negative at certain times. When this first happens we speak of ruin having occurred. The ultimate ruin probability, denoted by $\psi(u)$ is written as

$$\psi(u) = \Pr(T < \infty) \tag{1.8}$$

where T denotes the time to ruin, defined by

$$\begin{aligned} T &= \inf(t : t \geq 0 \text{ and } U(u,t) < 0) \\ &= \infty \text{ if } U(u,t) \geq 0 \text{ for all } t > 0 \end{aligned}$$

We write the survival probability $\delta(u) = 1 - \psi(u)$.

The aggregate loss process $L(t)$ is defined by $L(t) = S(t) - ct$ and L denotes the maximum of the aggregate loss process, so that $\psi(u) = \Pr(L > u)$.

We are interested in the probability of ruin $\psi_d(u)$ for the discrete process. Since we will always take the initial surplus for the discrete process to be an integer we need to define "ruin" carefully. Two definitions of ruin for the discrete process will be used, depending on whether or not a surplus of zero (other than at time zero) is regarded as ruin. Accordingly we define

$$\begin{aligned} T_d &= \min\{n : U_d(u,n) < 0 \text{ for some positive integer } n\} \\ &= \infty \text{ if } U_d(u,n) \geq 0 \text{ for all } n \end{aligned}$$

$$\begin{aligned} T_d^* &= \min\{n : U_d(u,n) \leq 0 \text{ for some positive integer } n\} \\ &= \infty \text{ if } U_d(u,n) > 0 \text{ for all } n \end{aligned}$$

Then we have $\psi_d(u) = \Pr(T_d < \infty)$ and $\psi_d^*(u) = \Pr(T_d^* < \infty)$. The corresponding probabilities of ultimate survival are $\delta_d(u) = 1 - \psi_d(u)$ and $\delta_d^*(u) = 1 - \psi_d^*(u)$.

Clearly, we have $\psi_d^*(u) = \psi_d(u-1)$ and $\delta_d^*(u) = \delta_d(u-1)$ for $u = 1,2,3,\ldots$

1.5 Thesis Overview

In our later chapters, we present a few different methods to calculate the various ruin-related quantities for the discrete surplus model, and use them to approximate the continuous surplus model corresponding quantities.

In the second chapter we calculate the basic quantity - the ultimate ruin probability by using three methods: a stable recursive algorithm; averaging the upper and lower bounds; and simulation. At the end, we present some numerical examples by these three methods and compare the results.

In the third and fourth chapters we study other quantities which include the distribution of the severity of ruin, the distribution of the surplus immediately prior to ruin, the joint distribution of the surplus immediately prior to ruin and the severity of ruin. These calculations use the output from chapter two. Some numerical examples show the results derived by different methods.

Two more quantities will be introduced in chapter five. They are the moments of the time to ruin and density of the time to ruin. Four methods will be used for the calculation.

Finally, we discuss some features of the methods used for the above chapters.

All the programs used for the series quantities calculation will be presented at the Appendix.

Chapter 2

Probability of Ultimate Ruin

Since ruin theory, known as the classical compound-Poisson risk model, was introduced in 1903 by the Swedish actuary Filip Lundberg (see Dubourdieu [11]), many studies of evaluating the probability of ruin have been explored. Seal [23] discusses numerical methods for evaluating $\psi(u)$. De Vylder [24] proposes a simple approximation for $\psi(u)$ by approximating the individual claim amount distribution with an exponential distribution.

Panjer and Willmot [22] developed the recursive methods for calculating the approximate probability of ultimate ruin.

The focus of this chapter is the calculation of the probability of ultimate ruin in continuous time for a general classical risk process. We shall present three methods: a stable recursive approximate algorithm derived by Dickson and Waters [8]; a bounds algorithm proposed by Dufresne and Gerber [13]; and simulation. The common feature of these three methods are that they can be explained in elementary terms and be implemented numerically without any difficulty.

2.1 Recursive Approximation Algorithms

Recursive algorithms for the probability of ultimate ruin have already appeared in actuarial literature. However, not all of these algorithms are numerically stable. An algorithm is numerically unstable if small errors in individual numerical operations (as a result of machine rounding for example) can combine to give uncontrollably large errors in the final results. For example, Conte and De Boor [4]. In this section, we present a stable algorithm.

In Chapter 1, we introduce how to discretize and re-scale a continuous process. By choos-

ing a distribution for $X_{d,i}$ that is, a good approximation to that of $\frac{\beta}{c}X_i$, $\delta_d(\frac{\beta}{c}u)$ is a good approximation to $\delta(u)$.

Dickson and Waters [8] presents the following formulae for the calculation of $\delta_d(u)$.

$$\delta_d(0) = \frac{\theta_d}{(1+\theta_d)h_d(0)} \tag{2.1}$$

$$\delta_d(u) = \delta_d(0) + \sum_{k=1}^{u} g_d(0,k)\delta_d(u-k) \quad \text{for } u = 1,2,3,\ldots \tag{2.2}$$

where $g_d(u,y)$ denotes the defective probability that, for given initial surplus u, ruin will occur and that the deficit at the time of ruin will be less than y, which is defined as

$$g_d(u,y) = \Pr(T_d < \infty \text{ and } U_d(u,T_d) = -y) \qquad \text{for } u = 0,1,2,\ldots \text{and } y = 1,2,3,\ldots$$

$$g_d^*(u,y) = \Pr(T_d^* < \infty \text{ and } U_d(u,T_d^*) = -y) \qquad \text{for } u = 0,1,2,\ldots \text{and } y = 0,1,2,\ldots,$$

It is clear that

$$g_d^*(u,y) = g_d(u-1,y+1) \qquad \text{for } u = 1,2,3,\ldots, \text{ and } y = 0,1,2,\ldots$$

$g_d(0,y)$ can be calculated by (Dickson and Waters [9])

$$g_d(0,y) = (1-H_d(y))/h_d(0) \text{ for } y = 1,2,3,\ldots \tag{2.3}$$

Since $H_d(y) = \sum_{k=1}^{y} h_d(k)$, it can easily be resolved by Panjer [19] recursion formulae

$$h_d(0) = e^{-\lambda\beta^{-1}} \tag{2.4}$$

$$h_d(k) = \frac{\lambda\beta^{-1}}{k}\sum_{i=1}^{k} i\cdot f_i\cdot h_d(k-i) \quad \text{for } k=1,2,3,\ldots \tag{2.5}$$

and De Vylder and Goovaerts [25]

$$f_0+f_1+\cdots f_k=\int_k^{k+1} F_d(x)\cdot dx \quad \text{for } k=0,1,2,\ldots \tag{2.6}$$

Starting from (2.1), and using (2.3), $\delta_d(u)$ can be recursively calculated by (2.2).

As the claim amounts have a continuous distribution, and according the the definition of $\delta_d(\frac{\beta}{c}u)$, survival occurs as long as the surplus stays above the value -1, but it could be zero at any time. $\delta_d^*(\frac{\beta}{c}u)$ is the survival to occur where the surplus never goes below zero. Thus $\delta_d(\frac{\beta}{c}u)$ will tend to overstate $\delta(u)$, so $\delta_d^*(\frac{\beta}{c}u)$ is usually a better approximation to $\delta(u)$ than is $\delta_d(\frac{\beta}{c}u)$. We use formulae to obtain $\delta_d(\frac{\beta}{c}u)$.

$$\delta_d^*(0)=\frac{\theta_d}{(1+\theta_d)} \tag{2.7}$$

and by the relations between them

$$\delta_d^*(u)=\delta_d(u-1) \quad \text{for } u=1,2,3\ldots \tag{2.8}$$

The important feature of formula (2.2) is that it is strongly stable which has been proved by Panjer and Wang [21].

2.2 Lower and Upper Bounds

In this section we shall present a method leads to the bounds algorithm presented by Dufresne and Gerber [13] and attributed by them to Goovaerts and Devylder [16] and Panjer [20]

It is well known that $\psi(0) = 1/(1+\theta)$ (Bowers et al [1]). We can see that $\psi(0)$ depends only upon the relative security loading θ and not on the specific form of the claim amount distribution. For convenience we denote this quantity by q.

Recall that the maximal aggregate loss $L = \max_{t\geq 0}\{S(t) - ct\}$ is the maximal excess of aggregate claims over premiums received. Since $\delta(u) = 1 - \psi(u) = \Pr(L \leq u)$, for $u \geq 0$, i.e. the probability of survival is the distribution function of L. It can be written as a random sum

$$L = L_1 + L_2 + \cdots + L_N \tag{2.9}$$

The random variables $L_1, L_2, \ldots, L_N$ and N are independent. The common distribution function of the L_i's(See Bowers et al [1]) is

$$L(y) = \frac{1}{p_1}\int_0^y [1 - P(x)] \cdot dx \tag{2.10}$$

Two new random variables that are closely related to L are L^l and L^h, they are written as

$$L^l = \lfloor L_1 \rfloor + \lfloor L_2 \rfloor + \cdots + \lfloor L_N \rfloor \tag{2.11}$$

$$L^h = \lceil L_1 \rceil + \lceil L_2 \rceil + \cdots + \lceil L_N \rceil \tag{2.12}$$

where $\lfloor L_i \rfloor$ are the largest integers less than L_i, and $\lceil L_i \rceil$ are the smallest integers larger than L_i. Clearly

$$L^l \leq L \leq L^h$$

which implies for $u > 0$

$$\psi^l(u) = \Pr(L^l > u) \leq \psi(u) = \Pr(L > u) \leq \Pr(L^h > u) = \psi^h(u), \quad \text{for } u > 0$$

We will use the average of the lower and upper bounds as an approximation value to $\psi(u)$.

Let l_k^l denote the probability that a given summand in (2.11) is equal to k, i.e.,that a given summand in (2.9) is between k and $k+1$. Thus

$$l_k^l = L(k+1)-L(k), \quad \text{for } k=0,1,2,\ldots$$

Let l_k^h denote the corresponding probability for the summands in (2.12). Thus

$$l_k^h = L(k+1)-L(k), \quad \text{for } k=0,1,2,\ldots$$

Here $L(x)$ is given by formula (2.10). We want to calculate

$$t_\tau^l = \Pr(L^l=\tau), \quad \text{for } \tau=0,1,2,\ldots$$
$$t_\tau^h = \Pr(L^h=\tau), \quad \text{for } \tau=0,1,2,\ldots$$

These can be calculated recursively by the following formulae

$$\begin{aligned} t_0^l &= \frac{1-q}{1-ql_0^l} \\ t_\tau^l &= \frac{q}{1-ql_0^l}\sum_{k=1}^{\tau} l_k^l \cdot t_{\tau-k}^l, \quad \text{for } \tau=1,2,3,\ldots \\ t_0^h &= 1-q \\ t_\tau^h &= q\sum_{k=1}^{\tau} l_k^h \cdot t_{\tau-k}^h, \quad \text{for } \tau=1,2,3,\ldots \end{aligned}$$

Then

$$\delta^l(u) = \sum_{\tau=0}^{u} t_t^h \le \delta(u) \le \sum_{\tau=0}^{u-1} t_\tau^l = \delta^h(u), \quad \text{for } u=0,1,2,\ldots$$

The approximation of $\delta(u)$ is the average of lower and upper bound

$$\delta(u) = \frac{\delta^l(u) + \delta^h(u)}{2}, \text{ for } u = 0, 1, 2, \ldots \tag{2.13}$$

2.3 Simulation

Thanks to the arrival of fast personal computers, simulation technique shows its advantage over theoretical analysis in models. For extremely complicated models this may be the only way to proceed. Just as the aggregate loss distribution can be simulated the process of surplus can also be simulated.

The procedure of simulating $\delta(u)$ is described as below:

1. Set the simulation experiment repeats to be 10000
2. For each experiment, set the total number of claims n to be 4000
3. Initialize p_1, the mean of $X_1, X_2, \ldots X_n$
4. Initialize the loading factor θ, and let $\theta = \theta_d$
5. Simulate inter-arrival time t which follows exponential distribution with mean $= 1/\lambda = 1/(c/((1+\theta) * p_1))$
6. Simulate individual claim amount X_i with mean $= p_1$, for $i = 1, 2, \ldots n$
7. Calculate accumulated claim amount $\sum_{i=1}^{m} X_i$, for $m = 1, 2, \ldots n$
8. Calculate accumulated time $\sum_{i=1}^{m} t_i$, for $m = 1, 2, \ldots n$
9. Set $L_j = \max(0, \max(\sum_{i=1}^{m} X_i - c \cdot \sum_{i=1}^{m} t_i))$, so that the maximal aggregate loss are not negative

10. Repeat 10000 times from step 2 to 9, then get $L_1, L_2, \ldots L_{10000}$

11. Calculate the empirical distribution of L_j which is the simulation result of $\delta(u)$

We use Matlab to implement the simulation experiment. The codes are given in Appendix A.

2.4 Numerical Illustrations

In this section we illustrate the approximation to $\delta(u)$. For the first two examples, we consider exponential individual claim amount distribution, but with different mean p_1, premium c and loading factor θ. The final two examples are for Pareto and Weibull individual claim amount distribution. We set the discrete scalor $\beta = 100$ for all the examples.

Example 2.1 Let the individual claim amount distribution be exponential with mean $p_1 = 1$. So that $P(x) = 1 - e^{-x}$, for $x > 0$. Let $c = 1$, $\theta = 0.1$. It is well known that the explicit solution for $\delta(u)$ is

$$\delta(u) = 1 - \frac{1}{1+\theta} \exp\left\{ -\frac{\theta u}{p_1(1+\theta)} \right\}, \text{ for } u = 0, 1, 2, \ldots \tag{2.14}$$

See Gerber [14]. Table 2.1 shows the exact value, approximate value, average value of lower and upper bounds and the simulation value of the probability of the ruin $\delta(u)$. We see from the table the approximation and average values are generally excellent. Although the simulation values are not good as the previous two, it is a very good approximation. In addition, we can make following observations:

- As the value of u increases from small to large, the approximation values are slightly less than the exact value first; then they gradually converge to the exact value.

u	Exact	App	Lower	Avg	Upper	Sim
0	0.090909	0.090909	0.090909	0.090909	0.090909	0.089730
2	0.242043	0.242041	0.241418	0.242044	0.242671	0.240559
4	0.368051	0.368049	0.367008	0.368053	0.369098	0.368185
6	0.473111	0.473108	0.471806	0.473113	0.474419	0.472933
8	0.560704	0.560702	0.559254	0.560706	0.562158	0.559600
10	0.633736	0.633733	0.632224	0.633737	0.635251	0.633474
20	0.852436	0.852434	0.851215	0.852434	0.853654	0.852401
40	0.976047	0.976047	0.975649	0.976045	0.976441	0.976019
60	0.996112	0.996112	0.996015	0.996111	0.996207	0.996145
80	0.999369	0.999369	0.999348	0.999369	0.999389	0.999428
100	0.999898	0.999898	0.999893	0.999897	0.999902	0.999932

Table 2.1: $\delta(u)$, Exponential(1) claims, $p_1 = 1$, $c = 1$, $\theta = 0.1$

- The average values show different pattern. As the values of u increased from small to large, the average values are less than the exact value; then they become larger than the exact value.

- The simulation values don't show a pattern related to the change of u. But we observe that it's very close to the exact value with errors less than 2%.

Example 2.2 Let the individual claim amount distribution be exponential with mean $p_1 = 2$. so that the distribution function is $P(x) = 1 - e^{-x/2}$, for $x > 0$. Let $c = 2, \theta = 0.25$. As in Example 2.1, Table 2.2 shows the exact, approximation, average and simulation values of $\delta(u)$. We see that while the approximation and average values are excellent, the simulation is a little bit poorer but still very good, especially for large u.

Example 2.3 We now consider the situation when the individual claim amount distribution is Pareto(4,3). So that $P(x) = 1 - \left(\frac{3}{x+3}\right)^4$ $p_1 = 1$, for $x >= 0$. Let $c = 1, \theta = 0.25$. For this distribution, an explicit solution for $\delta(u)$ does not exist. Table 2.3 shows the approximate, average and simulation values. From it, we see the similar pattern as the previous two examples.

- The approximation values is slightly less than the average values for small value of u. As the surplus increases, the approximation values become larger than average value.

u	Exact	App	Lower	Avg	Upper	Sim
0	0.200000	0.200000	0.200000	0.200000	0.200000	0.199470
2	0.345015	0.345014	0.344492	0.345016	0.345540	0.344758
4	0.463744	0.463742	0.462887	0.463745	0.464603	0.464548
6	0.560951	0.560948	0.559898	0.560952	0.562005	0.562591
8	0.640537	0.640534	0.639387	0.640537	0.641688	0.641449
10	0.705696	0.705694	0.704519	0.705696	0.706874	0.706600
20	0.891732	0.891730	0.890864	0.891730	0.892596	0.891864
40	0.985347	0.985347	0.985112	0.985346	0.985581	0.984588
60	0.998017	0.998017	0.997969	0.998017	0.998064	0.997767
80	0.999732	0.999732	0.999723	0.999732	0.999740	0.999721
100	0.999964	0.999964	0.999962	0.999964	0.999965	0.999962

Table 2.2: $\delta(u)$, Exponential(2) claims, $p_1 = 2$, $c = 2$, $\theta = 0.25$

u	App	Lower	Avg	Upper	Sim
0	0.2000000	0.2000000	0.2000000	0.2000000	0.1976200
2	0.4257595	0.4251440	0.4257614	0.4263787	0.4231872
4	0.5661899	0.5654139	0.5661915	0.5669691	0.5625668
6	0.6662730	0.6654745	0.6662741	0.6670736	0.6639189
8	0.7405717	0.7398112	0.7405724	0.7413336	0.7386563
10	0.7968496	0.7961539	0.7968499	0.7975459	0.7947224
20	0.9357567	0.9354095	0.9357563	0.9361031	0.9352955
40	0.9916746	0.9916129	0.9916744	0.9917359	0.9911388
60	0.9984501	0.9984394	0.9984500	0.9984606	0.9982739
80	0.9995639	0.9995618	0.9995639	0.9995660	0.9995483
100	0.9998249	0.9998244	0.9998249	0.9998254	0.9998303

Table 2.3: $\delta(u)$, Pareto(4,3) claims, $p_1 = 1$, $c = 1$, $\theta = 0.25$

Although they are identical for the last three large u, we expect approximation values are larger if more decimals show up.

- Simulation values do not look as good as the other two values, but considering the margin of error, we still can say it's a good method.

Example 2.4 As for our final example, we consider the individual claim amount distribution to be Weibull(2,1). So that $P(x) = (1 - e^{x/2})$, Let $c = 2, \theta = 0.1$. Table 2.4 displays the calculated approximate, average and simulation values of $\delta(0)$. We observe a similar pattern

u	App	Lower	Avg	Upper	Sim
0	0.090909	0.090909	0.090909	0.090909	0.089730
2	0.169907	0.169566	0.169909	0.170252	0.168135
4	0.242041	0.241418	0.242044	0.242671	0.240559
6	0.307907	0.307053	0.307911	0.308769	0.307474
8	0.368049	0.367008	0.368053	0.369098	0.368185
10	0.422964	0.421777	0.422969	0.424161	0.423239
20	0.633733	0.632224	0.633737	0.635251	0.633474
40	0.852434	0.851215	0.852434	0.853654	0.852401
60	0.940547	0.939808	0.940545	0.941282	0.940851
80	0.976047	0.975649	0.976045	0.976441	0.976019
100	0.990349	0.990149	0.990348	0.990548	0.990183

Table 2.4: $\delta(u)$, Weibull(2,1)) claims, $p_1 = 2$, $c = 2$, $\theta = 0.1$

to that in the previous example.

We implemented the three approximate methods for four different examples. All the results suggest that the variance of the individual claim amounts distribution, the variance of premium and the variance of loading factor have little effect on the quality of the stable recursive approximation and average of the bounds approximation. They both provide excellent approximation to $\delta(u)$. Simulation method is a very good alternative because it is very easy to implement especially for complicated model. It's clear that the accuracy of simulation result can be improved by increasing the number of repetition, with the cost of consuming much more time.

Chapter 3

Distribution of the Severity of Ruin

When ruin occurs, we also want to know how serious the situation is, i.e. the severity of ruin and its probability. Gerber et al [4] are the first ones to study the probability and severity of ruin for the classical continuous time risk model. They obtain an integral equation which is satisfied by the distribution of the severity of ruin. Since then more actuarial science researchers have started paying attention to it.

Dickson and Waters [8] derive a stable recursive algorithm and also obtain lower and upper bounds.

In this chapter, we will discuss these methods and illustrate them with some numerical examples. For both methods, some of the results from previous chapter will be used.

3.1 Recursive Approximation Algorithm

For our continuous process, we define $G(u,y)$ to be the defective probability that for given initial surplus u, ruin will occur and that the deficit at the time of ruin will be less than y. This is written as Dickson and Waters c04

$$G(u,y) = \Pr(T < \infty \text{ and } U(u,T) > -y) \quad \text{for } u \geq 0 \text{ and } y > 0$$

So that $G_d(u,y)$ is its corresponding probability for the approximate discrete process:

$$G_d(u,y) = \Pr(T_d < \infty \text{ and } U_d(u,T_d) \geq -y) \quad \text{for } u = 0,1,2,\ldots \text{ and } y = 1,2,3,\ldots$$

$$G_d^*(u,y) = \Pr(T_d^* < \infty \text{ and } U_d(u,T_d^*) > -y) \quad \text{for } u = 0,1,2,\ldots \text{ and } y = 1,2,3,\ldots$$

$G_d(u,y)$ can be calculated recursively from the following formula:

$$G_d(u,y) = G_d(0,u+y) - G_d(0,u) + \sum_{k=1}^{u} g_d(0,k)G_d(u-k,y) \tag{3.1}$$

Where the starting point $G_d(0,y)$ can be calculated from (2.3) in a recursive manner. Its stability has been proved by Panjer and Wang [21].

Once $G_d(u,y)$ is calculated, $G_d^*(u,y)$ will be obtained by

$$G_d(u,y) = \sum_{j=1}^{y} g_d(u,j) = \sum_{j=1}^{y} g_d^*(u+1,j-1) = G_d^*(u+1,y) \quad \text{for } u = 1,2,3,\ldots \text{and } y = 1,2,3,\ldots \tag{3.2}$$

As explained in Chapter 2, we will use $G_d^*(\frac{\beta}{c}u, \frac{\beta}{c}y)$ to approximate $G(u,y)$.

3.2 Lower and Upper Bounds

In this section we illustrate lower and upper bounds for $G(u,y)$ derived by Dickson and Waters [8].

Let $g(u,y)$ denote the derivative of $G(u,y)$ with respect to y. Bowers et al. [1] tells us that

$$g(0,x) = \frac{\lambda}{c}(1 - P(x)) \tag{3.3}$$

We can integrate (3.3) numerically to any degree of accuracy to compute $G(0,y)$.

$$G(0,y) = \int_0^y g(0,x)dx \tag{3.4}$$

Now let $\psi^l(u)$ and $\psi^h(u)$ denote lower and upper bounds respectively for $\psi(u)$, calculated by the lower and upper bounds method in previous chapter. $G^l(u,y)$ is a lower bound for $G(u,y)$

$$\begin{aligned} G^l(u,y) &= \frac{1}{\delta(0)}\{\sum_{r=0}^{u-1}\psi^l(u-r)[G(0,r+1)-G(0,r)]- \\ &\quad \sum_{r=0}^{u-1}\psi^l(u-r)[G(0,r+y+1)-G(0,r+y)]+ \\ &\quad G(0,u+y)-G(0,u)-\psi^h(u)G(0,y)\} \end{aligned} \tag{3.5}$$

and $G^h(u,y)$ is an upper bound for $G(u,y)$

$$\begin{aligned} G^h(u,y) &= \frac{1}{\delta(0)}\{\sum_{r=0}^{u-1}\psi^h(u-r-1)[G(0,r+1)-G(0,r)]- \\ &\quad \sum_{r=0}^{u-1}\psi^h(u-r-1)[G(0,r+y+1)-G(0,r+y)]+ \\ &\quad G(0,u+y)-G(0,u)-\psi^l(u)G(0,y)\} \end{aligned} \tag{3.6}$$

3.3 Numerical illustrations

Example 3.1 Let the individual claim amount distribution be exponential with parameter 1, so $p_1 = 1$. Let $c = 1$ and loading factor $\theta = 0.1$. Thus $G(0,y) = \frac{\lambda}{c}(1-e^{-y})$.

Table 3.1 shows exact values, bounds and approximations to $G(u,y)$. Where the exact value of $G(u,y)$ was calculated from Dickson [7]

$$G(u,y) = \frac{1}{1+\theta}\exp\left(-\frac{\theta u}{(1+\theta)p_1}\right)(1-e^{-y/p_1})$$

- The approximation and average of bounds values are close to each other.
- For the smaller values of u, the average of bounds is slightly superior, the approximation is slightly lower, but for large values of u both give values very close to the exact value.

		$y=1$	$y=3$	$y=5$
$u=20$	Exact	0.093278359	0.140217403	0.146569912
	Approx.	0.093033739	0.140119271	0.146549380
	Lower	0.077090749	0.115883949	0.121134038
	Avg.	0.093279443	0.140219032	0.146571614
	Upper	0.109468136	0.164554114	0.172009191
$u=60$	Exact	0.002457696	0.003694445	0.003861821
	Approx.	0.002451319	0.003691962	0.003861387
	Lower	0.001178609	0.001771702	0.001851968
	Avg.	0.002458286	0.003695331	0.003862747
	Upper	0.003737963	0.005618960	0.005873526
$u=100$	Exact	0.000064755	0.000097341	0.000101751
	Approx.	0.000064589	0.000097278	0.000101743
	Lower	0.000008601	0.000012929	0.000013515
	Avg.	0.000064803	0.000097413	0.000101827
	Upper	0.000121006	0.000181898	0.000190139

Table 3.1: $G(u,y)$, Exponential(1) claims, $p_1 = 1$, $c = 1$, $\theta = 0.1$

- The calculation of $G^l(u,y)$ and $G^h(u,y)$ is not recursive so that each combination of u and y needs to be calculated separately. The calculation of $G_d(u,y)$ is recursive in u, and so is more convenient if values are required for several values of u.

Example 3.2 Let the individual claim amount distribution be Weibull with parameters (1, 0.5), so $p_1 = 2$. Let $c = 2, \theta = 0.25$

Table 3.2 shows bounds and approximations to $G(u,y)$. For this case, we can't calculate the exact value. We can see that average of bound values and approximated values are very close to each other with the average value is slightly higher than approximation. Therefore we can conclude both methods provide good approximation

		$y=1$	$y=5$	$y=10$
$u=20$	Approx.	0.051532160	0.177977955	0.258712082
	Lower	0.050708986	0.175850986	0.255966073
	Avg.	0.051639040	0.178138754	0.258833073
	Upper	0.052569095	0.180426521	0.261700073
$u=100$	Approx.	0.004582348	0.016067124	0.023736517
	Lower	0.004317243	0.015392090	0.022872553
	Avg.	0.004591890	0.016081834	0.023747980
	Upper	0.004866536	0.016771577	0.024623407
$u=200$	Approx.	0.000278469	0.000978154	0.001448296
	Lower	0.000250044	0.000905966	0.001356033
	Avg.	0.000279053	0.000979062	0.001449014
	Upper	0.000308061	0.001052158	0.001541994

Table 3.2: $G(u,y)$, Weibull(1, 0.5) claims, $p_1 = 2$, $c = 2$, $\theta = 0.25$

Chapter 4

Distribution of the Surplus Immediately Prior to Ruin and Severity of Ruin

When ruin occurs, there are two related quantities of interest: distribution of the surplus immediately prior to ruin, and the joint distribution of the severity of ruin and the surplus immediately prior to ruin.

Dufresne and Gerber [12] found explicit solutions for the distribution of the surplus immediately prior to ruin in the classical compound Poisson risk model. Dickson [7] found the relationship between the distribution function of surplus at ruin, the distribution function of surplus prior to ruin, and the ruin probability.

In this chapter we study the approximate numerical calculation of these two quantities. We will employ the stable recursive algorithms and lower and upper bounds presented by Dickson and Waters [8].

4.1 Distribution of the Surplus Immediately Prior to Ruin

Given initial surplus u, let $U(u,\tilde{T})$ denote the surplus immediately prior to ruin for our basic process. We define the probability that ruin occurs and that the surplus immediately prior to ruin as less than x, as by $F(u,x)$, then

$$F(u,x) = \Pr(T < \infty \text{ and } U(u,\tilde{T}) < x \qquad \text{for } u \geq 0 \text{ and } x > 0)$$

Similarly, for the discrete process, we have

$$F_d(u,x) = \Pr(T_d < \infty \text{ and } U_d(u, T_d - 1) < x) \quad \text{for } u = 0,1,2,\ldots \text{ and } x = 1,2,3,\ldots$$

$$F_d^*(u,x) = \Pr(T_d^* < \infty \text{ and } U_d(u, T_d^* - 1) < x) \quad \text{for } u = 0,1,2,\ldots \text{ and } x = 1,2,3,\ldots$$

We know for $u = 1,2,3,\ldots$ and $x = 1,2,3,\ldots$

$$F_d^*(u,x) = F_d(u-1, x-1)$$

The approximation for $F(u,x)$ is $F_d^*(\frac{\beta}{c}u, \frac{\beta}{c}x)$; to obtain it, we need to calculate $F_d(u,x)$ by using the following formulae as suggested by Dickson and Waters [8].

$$F_d(0,x) = \frac{1}{h_d(0)} \sum_{j=1}^{x} (1 - H_d(j)) = \sum_{j=1}^{x} g_d(0,j) \tag{4.1}$$

$$F_d(u,x) = \sum_{j=1}^{u} g_d(0,j) F_d(u-j,x) \text{ for } u = x, x+1, x+2, \ldots \tag{4.2}$$

$$F_d(u,x) = \sum_{j=1}^{u} g_d(0,j) F_d(u-j,x) + \sum_{j=1}^{u} g_d(0,j) \text{ for } u = 1,2,3,\ldots \tag{4.3}$$

The lower and upper bounds for $F(u,x)$ can be calculated by

$$F^l(u,x) = \frac{1 - G(0,x)}{1 - \psi(0)} \psi^l(u) - \frac{\psi(0) - G(0,x)}{1 - \psi(0)} \text{ for } 0 \le u \le x$$

$$F^h(u,x) = \frac{1 - G(0,x)}{1 - \psi(0)} \psi^h(u) - \frac{\psi(0) - G(0,x)}{1 - \psi(0)} \text{ for } 0 \le u \le x$$

$$F^l(u,x) = G^l(u-x,x) - \frac{1 - G(0,x)}{1 - \psi(0)} \left[\psi^h(u-x) - \psi^l(u)\right] \text{ for } u \ge x$$

$$F^h(u,x) = G^h(u-x,x) - \frac{1 - G(0,x)}{1 - \psi(0)} \left[\psi^l(u-x) - \psi^h(u)\right] \text{ for } u \ge x$$

4.2 Joint Distribution of the Severity of Ruin and the Surplus Immediately Prior to Ruin

Define $F(u,x,y)$ to be the defective joint distribution of the severity (the deficit at time of ruin is less than y) of ruin and the surplus immediately prior to ruin (less than x) for our basic process.

$$F(u,x,y) = \Pr(T < \infty, U(u,T) > -y \text{ and } U(u,\tilde{T}) < x)$$

For the discrete approximation process, define

for $u = 0,1,2,\ldots$ $x = 1,2,3,\ldots$and $y = 1,2,3,\ldots$

$$F_d(u,x,y) = \Pr(T_d < \infty, U_d(u,T_d) \geq -y \text{ and } U_d(u,T_d - 1) < x)$$

for $u = 1,2,3,\ldots$ $x = 1,2,3,\ldots$and $y = 1,2,3,\ldots$

$$F_d^*(u,x,y) = \Pr(T_d^* < \infty, U_d(u,T_d^*) > -y \text{ and } U_d(u,T_d^* - 1) < x)$$

and we have for $u = 1,2,3,\ldots$ $x = 1,2,3,\ldots$and $y = 1,2,3,\ldots$

$$F_d^*(u,x,y) = F(u-1,x-1,y)$$

Using the discrete approximation to our basic process, the approximation for $F(u,x,y)$ is $F_d^*(\frac{\beta}{c}u, \frac{\beta}{c}x, \frac{\beta}{c}y)$. To obtain it, we calculate $F_d(u,x,y)$ first through (Dickson and Waters [8]).

$$F_d(0,x,y) = \frac{1}{h_d(0)} \sum_{j=1}^{x} (H_d(y+j) - H_d(j))$$

Alternatively, we could write it as

$$F_d(0,x,y) = F_d(0,x) + G_d(0,y) - G_d(0,x+y)$$

for $u = 1,2,3,\ldots, x = 1,2,3,\ldots x-1$

$$F_d(u,x,y) = \sum_{j=1}^{u} g_d(0,j)F_d(u-j,x,y) + \sum_{j=u+1}^{x} (g_d(0,j) - g_d(0,j+y))$$

for $u = x, x+1, x+2, \ldots$

$$F_d(u,x,y) = \sum_{j=1}^{u} g_d(0,j)F_d(u-j,x,y)$$

The lower and upper bounds of $F(u,x,y)$ can be calculated by (Dickson and Waters [8]).

for $0 \leq u \leq x$

$$\begin{aligned} F^l(u,x,y) &= G^l(u,y) + \frac{\delta^h(u)}{\delta(0)}(G(0,x) - G(0,x+y)) \\ F^h(u,x,y) &= G^h(u,y) + \frac{\delta^l(u)}{\delta(0)}(G(0,x) - G(0,x+y)) \end{aligned}$$

for $u \geq x$

$$\begin{aligned} F^l(u,x,y) &= G^l(u,y) - G^h(u-x,x+y) + G^l(u-x,x) + \\ &\quad G(0,x)(\psi^l(u-x) - \psi^h(u))/\delta(0) + \\ &\quad G(0,x+)(\psi^l(u) - \psi^h(u-x))/\delta(0) \end{aligned}$$

$$\begin{aligned} F^h(u,x,y) &= G^h(u,y) - G^l(u-x,x+y) + G^h(u-x,x) + \\ &\quad G(0,x)(\psi^h(u-x) - \psi^l(u))/\delta(0) + \\ &\quad G(0,x+y)(\psi^h(u) - \psi^l(u-x))/\delta(0) \end{aligned}$$

		x=5	x=10	x=15
u=10	App.	0.347764352	0.365965777	0.366264655
	Avg.	0.348173476	0.365975212	0.366260989
u=30	App.	0.056450100	0.059404607	0.059452816
	Avg.	0.056518436	0.059408041	0.059454143
u=50	App.	0.009163141	0.009642725	0.009650550
	Avg.	0.009175173	0.009644250	0.009651733

Table 4.1: $F(u,x)$, Exponential(1) claims, $p_1 = 1$, $c = 1$, $\theta = 0.1$

4.3 Numerical Illustrations

Example 4.1 Let the individual claim amount distribution be exponential with mean $p_1 = 1$ with premium $c = 1$ and loading factor $\theta = 0.1$. Table 4.1 is the approximation of $F(u,x)$. Table 4.2 is the exact and approximations of $F(u,x,y)$. Where $F(u,x,y)$ is calculated by for $0 \leq u \leq x$

$$F(u,x,y) = G(u,y) + \frac{\delta(u)}{\delta(0)}(G(0,x) - G(0,x+y))$$

for $u \geq x$

$$\begin{aligned} F(u,x,y) &= G(u,y) - G(u-x,x+y) + G(u-x,x) + \\ &\quad \frac{\psi(u-x) - \psi(u)}{\delta(0)}(G(0,x) - G(0,x+y)) \end{aligned}$$

Example 4.2 Let the individual claim amount distribution be Weibull (2,1), so $p_1 = 2$. We let $c = 2$, and $\theta = 0.25$. Table 4.3 and 4.4 give the approximation of $F(u,x)$ and $F(u,x,y)$.

In each example, the approximations are close to each other and from the first example we can see that the approximations are close to the exact values. As with the approximations to $G(u,y)$ when the individual claim amount distribution is exponential, approximations to

		x=y=1	x=y=3	x=y=5
u=20	Ext.	0.023039836	0.109159143	0.139330921
	App.	0.022661899	0.108659224	0.139146266
	Avg.	0.023041014	0.109161026	0.139332698
u=60	Ext.	0.000607053	0.002876123	0.003671088
	App.	0.000597112	0.002863030	0.003666324
	Avg.	0.000607297	0.002876883	0.003671988
u=100	Ext.	0.000015995	0.000075780	0.000096726
	App.	0.000015733	0.000075437	0.000096603
	Avg.	0.000016011	0.000075839	0.000096798

Table 4.2: $F(u,x,y)$, Exponential(1) claims, $p_1 = 1$, $c = 1$, $\theta = 0.1$

		x=5	x=10	x=15
u=10	App.	0.190763919	0.274904362	0.292697582
	Avg.	0.191787305	0.275283772	0.292742314
u=30	App.	0.025817441	0.037204775	0.039412741
	Avg.	0.025957266	0.037257512	0.039426045
u=50	App.	0.003494057	0.005035186	0.005334006
	Avg.	0.003513386	0.005042836	0.005336336

Table 4.3: $F(u,x)$, Weibull(2,1) claims, $p_1 = 2$, $c = 2$, $\theta = 0.25$

		x=y=1	x=y=3	x=y=5
u=20	App.	0.003086679	0.032148949	0.064360381
	Avg.	0.003175054	0.032514150	0.064764706
u=60	App.	0.000056536	0.000588844	0.001178833
	Avg.	0.000058183	0.000595695	0.001186512
u=100	App.	0.000001036	0.000010785	0.000021592
	Avg.	0.000001066	0.000010917	0.000021743

Table 4.4: $F(u,x,y)$, Weibull(2,1) claims, $p_1 = 2$, $c = 2$, $\theta = 0.25$

$F(u,x,y)$ based on the bounds are slightly better for small values of u. The cases use the calculation result $\psi(u)$. We know from chapter two that average of bounds for $\psi(u)$ give an excellent approximation to $\psi(u)$. When $u < x$ the average of bounds for $F(u,x)$ should be a very good approximation to $F(u,x)$ since these bounds are linear functions of the bounds on $\psi(u)$.

Chapter 5

Moments of the Time to Ruin

One particular question of interest in classical ruin theory is the moments of the time of ruin, which has been studied in the literature in recent years.

Lin and Willmot [18] extend the work of Gerber and Shiu [15] to obtain an explicit solution for the moments of the time to ruin provided that an explicit solution exists for the ultimate ruin probability. Cheng et al [3] present expressions for the moments of the time to ruin for a discrete time risk model. Egidio dos Reis [23] finds a recursion scheme to calculate the moments of the time to ruin for a discrete time risk model and uses this to approximate moments of the time to ruin in classical risk model. Cardoso and Egidio dos Reis [2] focus on the direct calculation of the distribution of time to ruin by means of Markov chain application. Drekic and Willmot [10] derive explicit results for the moments of time to ruin for exponential claims.

In this chapter, we study aspects of the time to ruin in the classical risk model. In particular, we focus on the actual distribution of the time to ruin. By calculating values of both finite and infinite time ruin probabilities, we can construct numerically the conditional distribution of the time to ruin, and use this to create density function. We also show how Lin and Willmot's [18] results can be used to calculate approximate values for moments of the time to ruin when explicit solutions for the probability of ultimate ruin does not exist. At the end, we illustrate the calculation of the moments of the time to ruin, given that ruin occurs.

In chapter 1, we define the time to ruin is denoted by T, $\psi(u)$ denotes the probability of ultimate ruin from initial surplus u, where $\psi(u) = \Pr(T < \infty)$. Now let's define the probability of ruin by time t from initial surplus u, denote it as $\psi(u,t)$, where $\psi(u,t) = \Pr(T \leq t)$. Then, we define the distribution function of the time to ruin given that ruin occurs as

$$\Pr(T_c \leq t) = \Pr(T \leq t | T < \infty) = \psi(u,t)/\psi(u)$$

where $T_c = T|T < \infty$.

Let $E(T_c^k)$ denotes the kth moment of time to ruin. Delbaen [6] proves that the kth moment of T_c exists only if the $(k+1)$th moment of the individual claim amount distribution exists. In this chapter, we will calculate the first three moments of the time to ruin, so we assume that the fourth moment of the individual claim, p_4, exists.

5.1 Formulae for Moments

Lin and Willmot [18] present a recursive scheme from which explicit solutions for the moments of T_c can be found.

$$E(T_c^k) = \psi_k(u)/\psi(u) \tag{5.1}$$

where

$$\psi_k(u) = \frac{k}{\lambda p_1 \theta}\left(\int_0^u \psi(u-x)\psi_{k-1}(x)dx + \delta(u)\int_0^\infty \psi_{k-1}(x)dx - \int_0^u \psi_{k-1}(x)dx\right) \tag{5.2}$$

The value of $\psi(u)$ can be obtained from the formula (2.2). However the formula (5.2) involves integration over an infinite range, so it cannot be used for numerical calculation. Therefore, we use the formulae provided by (Dickson and Waters [5]) to calculate $\psi_1(u), \psi_2(u), \psi_3(u)$.

$$\psi_1(u) = \frac{1}{\lambda p_1 \theta}\left(E(L)\delta(u) - \int_0^u \psi(x)\delta(u-x)dx\right) \tag{5.3}$$

$$\psi_2(u) = \frac{2}{\lambda p_1 \theta}\left(\frac{E(L^2)\delta(u)}{2\lambda p_1 \theta} - \int_0^u \psi_1(x)\delta(u-x)dx\right) \tag{5.4}$$

$$\psi_3(u) = \frac{3\delta(u)E(L)E(L^2)}{(\lambda p_1 \theta)^3} + \frac{\delta(u)E(L^3)}{(\lambda p_1 \theta)^3} - \frac{3}{\lambda p_1 \theta}\int_0^u \psi_2(x)\delta(u-x)dx \tag{5.5}$$

where

$$E[L] = \int_0^\infty \psi(x)dx = \frac{p_2}{2\theta p_1} \tag{5.6}$$

$$E[L^2] = 2\int_0^\infty x\psi(x)dx = \frac{p_3}{3\theta p_1} + \frac{1}{2}\left(\frac{p_2}{\theta p_1}\right)^2 \tag{5.7}$$

$$E[L^3] = 3\int_0^\infty x\psi(x)dx = \frac{p_4}{4\theta p_1} + \frac{3}{4}\left(\frac{p_2}{\theta p_1}\right)^3 + \frac{p_2 p_3}{(\theta p_1)^2} \tag{5.8}$$

5.2 Diffusion Approximation

The surplus process $U(t)$ can be approximated by a diffusion process. We write it $\tilde{U}(t) = u + W(t)$, where $W(t) \sim N(\theta\lambda p_1 t, \lambda p_2 t)$ for all $t > 0$. It is well known that this diffusion process $\tilde{U}(t)$ has an Inverse Gaussian [17] distribution with density

$$f(t) = \frac{u}{(2\pi\lambda p_2)^{1/2}} t^{-3/2} \exp\left\{-\frac{(u-\theta\lambda p_1)^2}{2\lambda t p_2}\right\} \tag{5.9}$$

By choosing the parameters of the diffusion process appropriately, we can consider the moments of the Inverse Gaussian distribution as approximations to the moments of T_c for $u > 0$.

$$E[T_c] \approx \frac{u}{\lambda p_1 \theta}; \quad V[T_c] \approx \frac{u p_2}{\lambda^2 p_1^3 \theta^3}; \quad Sk[T_c] \approx 3\left(\frac{p_2}{\theta p_1 u}\right)^{1/2} \tag{5.10}$$

where $Sk[T_c]$ denotes the coefficient of skewness of T_c.

We see that the approximations depend on the first two moments of the individual claim size distribution. This is because the surplus process is being approximated by a diffusion process and is matched via the first two moments. Thus calculation is simple.

It should be remembered that if p_4 does not exist then the third moment, and hence the coefficient of skewness, of T_c does not exist.

	Mean			St.Dev			Coef. Skewness		
u	Exact	App.	Diffusion	Exact	App.	Diffusion	Exact	App.	Diffusion
0	10.00	10.00	-	45.83	45.83	-	13.74	13.74	-
10	100.91	100.91	100.00	148.66	148.66	141.42	4.24	4.24	4.24
20	191.82	191.82	200.00	205.18	205.18	200.00	3.07	3.07	3.00
30	282.73	282.73	300.00	249.20	249.20	244.95	2.53	2.53	2.45
40	373.64	373.64	400.00	286.53	286.53	282.84	2.20	2.20	2.12
50	464.55	464.55	500.00	319.53	319.53	316.23	1.97	1.97	1.90

Table 5.1: Mean, Standard Deviation and Coefficient of Skewness of Tc, Exponential(1) claims

	Mean			St.Dev			Coef. Skewness		
u	Exact	App.	Diffusion	Exact	App.	Diffusion	Exact	App.	Diffusion
0	4.00	4.00	-	12.00	12.00	-	8.963	8.963	-
10	36.00	36.00	40.00	37.74	37.74	35.78	2.861	2.861	2.683
20	68.00	68.00	80.00	52.00	52.00	50.60	2.076	2.076	1.897
30	100.00	100.00	120.00	63.12	63.12	61.97	1.711	1.711	1.549
40	132.00	132.00	160.00	72.55	72.57	71.55	1.488	1.486	1.342
50	164.00	163.98	200.00	80.90	81.01	80.00	1.334	1.313	1.200

Table 5.2: Mean, Standard Deviation and Coefficient of Skewness of Tc, Exponential(1) claims

5.3 Numerical Illustrations

In the examples below, the discretizing scalar $\beta = 1000$ is used for all examples to calculate $\psi(u)$.

Example 5.1 In this example, we let the individual claim amount distribution be exponential (1). Set $c = 1$. For this case, as the exact value and the approximate value of $\psi(u)$ can be obtained from (2.14) and (2.2), respectively. Thus we can get the exact, approximate and diffusion values of $E(T_c^k)$, for $k = 1,2,3$, showing in the Tables, 5.1, 5.2 for $\theta = 0.1$ and $\theta = 0.25$ respectively. We see that the diffusion values are poorer than approximate values.

Example 5.2 Now we consider the individual claim amount distribution is Weibull (1,0.5). Let $c = 2$. For this case, the explicit solution does not exist. Tables 5.3, 5.4 show approximate values and diffusion approximate values of the first three moments of T_c for $\theta = 0.1$ and $\theta = 0.25$, respectively. We observe that the approximate and diffusion values are reasonably close

	Mean		St.Dev		Skewness	
u	App.	Diffusion	App.	Diffusion	App.	Diffusion
0	33.00	-	155.95	-	13.789	-
20	150.62	110.00	335.72	269.44	6.402	7.348
40	248.41	220.00	433.34	381.05	4.960	5.196
60	343.32	330.00	511.19	466.69	4.204	4.243
80	437.02	440.00	578.23	538.89	3.716	3.674

Table 5.3: Mean, Standard Deviation and Coefficient of Skewness of Tc, Weibull(1,0.5) claims

	Mean		St.Dev		Skewness	
u	App.	Diffusion	App.	Diffusion	App.	Diffusion
0	15.00	-	48.22	-	9.132	-
20	62.62	50.00	100.33	77.46	4.384	4.648
40	99.53	100.00	128.16	109.54	3.433	3.286
60	134.07	150.00	150.23	134.16	2.929	2.683
80	167.26	200.00	169.21	154.92	2.600	2.324

Table 5.4: Mean, Standard Deviation and Coefficient of Skewness of Tc, Weibull (1,0.5) claims

to each other.

We see that the coefficients of skewness of the above tables are positive values. This indicates that the distributions of T_c are far from normal. From formula (5.10), it indicates that

$$\lim_{u\to\infty} Sk[T_c] = 0$$

Thus for these examples the limit distribution of T_c is normal. In the next chapter, we plot the distribution of the time to ruin from which we can clearly see the skewness of the distribution of T_c.

Chapter 6

Density of the Time to Ruin

In this chapter we illustrate the shape of the density of T_c. Three different methods are used to produce graphs of density functions.

6.1 Approximation Algorithms

We know the common distribution function of time to ruin given the ruin occurs is

$$H(t) = \Pr(T_c \leq t) = \frac{\psi(u,t)}{\psi(u)}$$

values of $\psi(u)$ is calculated using the stable recursive algorithm (2.2). Values of $\psi(u,t)$ are calculated using the algorithm presented in Dickson and Waters [9] which use the discrete time probability of survival $\delta_d^*(\beta u,(1+\theta)\beta t)$ to approximate $\delta(u,t)$. $\delta_d(u,t)$ is obtained by the following formulae

$$\delta_d(0,t) = \frac{1}{h_d(0)} \sum_{j=0}^{t-1} F(j,t+1) \tag{6.1}$$

$$\delta_d(u,t) = \frac{1}{h_d(0)} \left[\delta_d(u-1,t+1) - \sum_{i=1}^{u} h_d(i)\delta_d(u-i,t) \right] \tag{6.2}$$

$$\delta_d(u-1,t+1) = \sum_{i=0}^{u} h_d(i)\delta_d(u-i,t) \tag{6.3}$$

where $h_d(0)$ and $h_d(k)$ for $k = 0,1,2,\ldots$can be calculated from (2.4), (2.5). $F(j,t)$ denotes the common distribution of the aggregate claims up to time t for $j = 0,1,2,\ldots$, which can be calculated using Panjer's [19] recursive formula.

Then we know

$$\begin{aligned}\delta_d^*(0,t) &= \frac{1}{t}\sum_{j=0}^{t-1} F(j,t)\\ \delta_d^*(u,t) &= \delta_d(u-1,t)\end{aligned}$$

We estimate the density of T_c at $t = j/[(1+\theta)\beta]$ as

$$f_{T_c}\left(\frac{j}{(1+\theta)\beta}\right) = (1+\theta)\beta\left[H\left(\frac{j}{(1+\theta)\beta}\right) - H\left(\frac{j-1}{(1+\theta)\beta}\right)\right]$$

for $j = 1,2,3,\ldots$

6.2 Diffusion Approximation

As the continuous surplus process $U(t)$ can be approximated by a diffusion process $\tilde{U}(t)$ which is the Inverse Gaussian distribution. The density of this distribution can be regarded as approximations to the density of T_c. Formula (5.9) is used to approximate it.

Based on this exact result for the diffusion surplus process, we can say the distribution of T_c can also be approximated by an Inverse Gaussian distribution with parameters determined by the first two moments. This is our third method given below.

6.3 Inverse Gaussian Approximation

Using the formulation in Klugman et al (2004), the probability density function of Inverse Gaussian distribution distribution is

$$f(x) = \left(\frac{\theta}{2\pi x^3}\right)^{1/2} \cdot \exp\left(-\frac{\theta z^2}{2x}\right), \quad \text{where } z = \frac{x-\mu}{\mu} \tag{6.4}$$

$$E(x) = \mu, \quad Var(x) = \frac{\mu^3}{\theta} \tag{6.5}$$

We calculated the first two moments of T_c, $E(T_c^1)$ and $E(T_c^2)$ using formula (5.1), then we solve for μ and θ by matching the moments in formulae (6.2). Formula (6.1) can be used to calculate the values of density directly. This is another approximation to the density of T_c.

6.4 Numerical Illustrations

Example 6.1 Let the individual claim amount distribution be exponential with $p_1 = 1$. $\theta = 0.1$, $\beta = 20$, $u = 20$, thus $\psi(20) = 0.015$. Using the exact values of the mean and standard deviation from Table 5.1, we calculate the parameters of our approximating Inverse Gaussian density as 68 and 116.28. In Figure 6.1, the densities calculated by three methods are reasonably close to each other. A clear feature of the distribution of T_c is positively skewed as indicated by the value of the coefficient of skewness in Table 5.1. The straightforward approach of Diffusion and Inverse Gaussian approximation provides much better approximations than a normal distribution does.

We use $\beta = 20$ as our scaling factor. This value is sufficient to calculate accurate approximation to both finite and infinite time ruin probabilities. The larger value of β, the better approximations are, but such extra accuracy is of limited value to us to illustrate the shape of the density T_c.

Example 6.2 Pareto(3,4),$\theta = 0.1, u = 10, \psi(10) = 0.475194$, parameters are 126.824 and 41.1322.

Because the formulae (6.1) and (6.2) are not stable (Dickson and Waters [9]), when using it to approximate $\delta(u,t)$ for the values of u greater than above 30 we experience difficulties. For example, the values of $\delta(u,t)$ are outside the range zero to one.

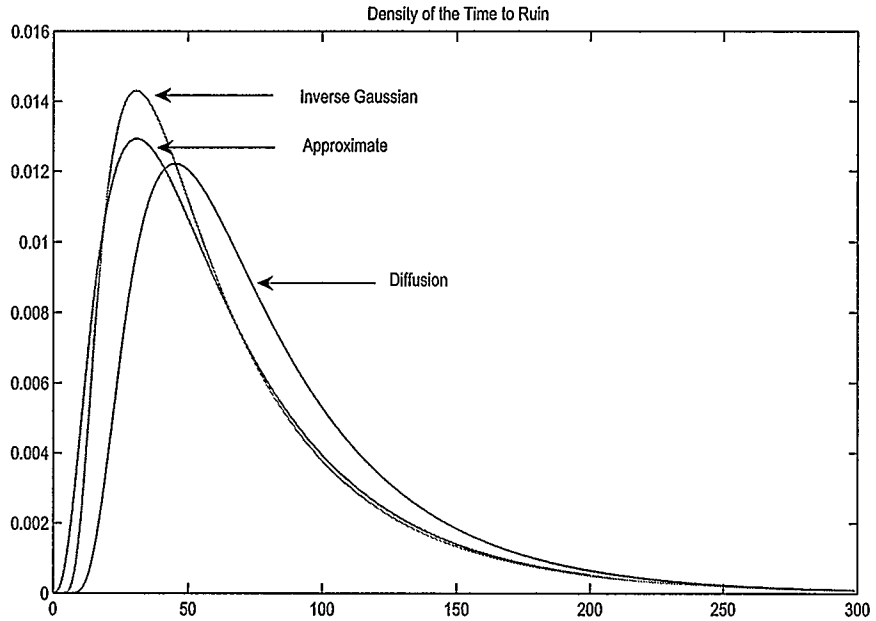

Figure 6.1: Exponential claims, $u = 20$, $\theta = 0.25, \psi(20) = 0.015$

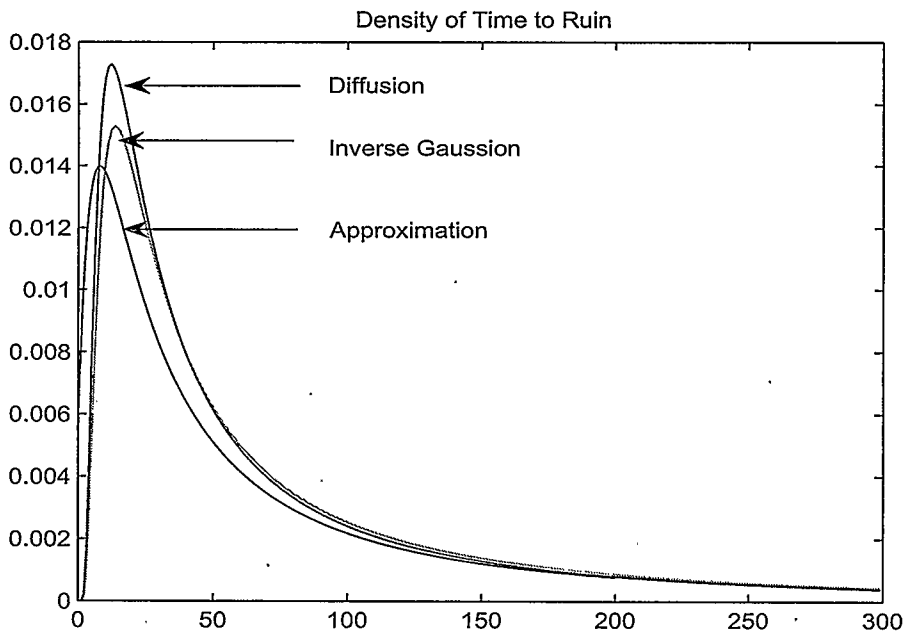

Figure 6.2: Pareto claims, $u = 10$, $\theta = 0.1, \psi(10) = 0.475194$

Chapter 7

Conclusion

In this thesis we study the classical risk process. We use different methods to compute the ruin probability for infinite time and its related quantities which include distribution of the severity of ruin, the distribution of the surplus immediately prior to ruin, the joint distribution of the surplus immediately prior to ruin and the severity of ruin.

As the continuous model is very difficult to calculate, a rescaled discrete model is built to approximate it. Based on this discrete model, Dickson and Waters [8] present a recursive algorithm to calculate the ruin probability and related quantities. By comparing its calculated results to the other two methods (average of upper and lower bounds and simulation) we find that the recursive algorithm and the bounds method both give an excellent approximation. Secondly, by comparing the computing time for each example, we see the bounds method is the fastest which only takes a few seconds; the recursive method is the second fastest which takes about 30 seconds; and the simulation method is the slowest which takes about 2 minutes. There is an important feature of the recursive method is that it is numerical stable.

When we rescale the continuous process, the choosing of scaler β determines the approximation values. The larger β the more accurate values are. However it adds computing time. In our examples, we set $\beta = 100$. From the observation this is good enough for approximation and it doesn't take extra time to complete calculation.

When calculating the ruin probability for the infinite time, a direct simulation method is presented as well. Results look poorer than other methods but it's still a good approximation. It has some advantages: it is straightforward; it is easy to implement; it may be the only way when encountering complicated problems. The quality of it can always be improved by increasing the simulation replications or using new algorithms.

From investigating the shape of density of the time to ruin we find the distribution of T_c is positively skewed. A simple approximation based on Inverse Gaussian densities can give reasonable results whereas a normal approximation would be inappropriate.

A future work we would like to do is using the translated gamma approximation method presented by Dickson and Waters [5] which is based on matching three moments. It should perform better than Inverse Gaussian method which is based on matching just two moments.

Appendix A

Matlab Codes I

1. Main function of calculation of the ruin probability and its related quantities.

```
%%% -- The following quantities are calculated: the probability of ultimate
%%%    survival Delta(u), distribution of the severity of ruin G(u,y),
%%%    distribution of the surplus immediately prior to ruin F(u,x), joint
%%%    distribution of the severity of ruin and the surplus immediately prior
%%%    to ruin F(u,x,y).

%%% -- Methods are used: Exact method (exponential distribution only), Stable
%%%    recursive alogrithm, Bounds and Simulation.

%%% -- This program can calculate the claim distribution has:
%%%    1-Expenential
%%%    2-Pareto(2,1) or Pareto(2,2)
%%%    3-Weibull(0.5,0.5) or Weibull(1,0.5)
%%%    4-Gamma(2,0.5) or Gamma(2,1)
%%%    5-Pareto(4,3)
%%%    6-Weibull(1,1) or Weibull(2,1)


function main

format long
diary result

dist=5;      %%% Chose a distribution %%%
beta=100;    %%% Set scalor %%%
theta=0.1;   %%% Set security factor %%%
c=1;         %%% Set premium collected %%%
p1=1;        %%% Set the mean of distribution %%%
beta=beta/c;


%%%----------------Rescale & Discrete the Continuous Process ----------------

umax=200;    %%% Set the maxmium initial surplus u %%%
ymax=15;     %%% Set the maxmium deficit %%%
xmax=15;     %%% Set the maxmium surplus immediately before ruin %%%


Hd=Hdy2(umax+ymax,beta,theta,p1,c,dist);
hd0=Hd(1);
gd=(1-Hd)/hd0;
gd(1)=0;
Gd0=cumsum(gd);
```

```
%%%-------------------------------------------------------------------------

%%% ------------------------Calculate Delta(u)---------------------------


%%% Example 2.1, 2.2, 2.3 & 2.4
ind=[0 2 4 6 8 10 20 40 60 80 100];
umax=200;
u=0:1:umax;

%%%%%%%%%%%%%%%%%%%%%%%%%--Recursive Algorithm--%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Approximate delta(0), delta(0.01),delta(0.02),...,delta(100),...
appdelscl=appdelta3(umax,beta,theta,gd,hd0);

%%% Approximate delta(0), delta(1),delta(2),....delta(100),...
appdel=appdelscl(u*beta+1);

%%% Display delta(u) at initial u = 0 2 4 6 8 10 20 40 60 80 100
appdel(ind+1)

%%% Plot
plot(u,appdel,'b');
hold on;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%--Lower & Upper Bounds--%%%%%%%%%%%%%%%%%%%%%%%%

%%% Bounds of delta(0),delta(0.01),delta(0.02),...,delta(100),...
bounddel=bounddelta4(umax,beta,theta,dist,p1);

deltald=bounddel(:,1); %lower bound
deltaud=bounddel(:,2); %upper bound

deltalow=deltald(u*beta+1); %lower bound at u=0,1,2,...,100
deltaup=deltaud(u*beta+1);  %upper bound at u=0,1,2,...,100
avg=(deltalow+deltaup)/2;   %avg of lower & upper bounds

%%% Display delta(u) at initial u = 0 2 4 6 8 10 20 40 60 80 100
avg(ind+1)

%%% Plot
plot(u,avg, 'g');
hold on;


%%%%%%%%%%%%%%%%%%%%%%%%--Exact Value, Exponential only--%%%%%%%%%%%%%%%%%%%%

if dist==1
    %%% Exact delta(0), delta(0.01),delta(0.02),...,delta(100)
    exactdelscl=exactdelta2(umax,beta,theta,p1);
```

```
    %%% Excat delta(0), delta(1),delta(2),....delta(100)
    exactdel=exactdelscl(u*beta+1);

    %%% Display delta(u) at initial u = 0 2 4 6 8 10 20 40 60 80 100
    exactdel(ind+1)

    %%% Plot
    plot(u,exactdel,'k');
    hold on;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%--Simulation--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

simdelta=simdelta3(c,theta,beta,p1,dist);

a=simdelta(:,1);   %%% All simulation values of initial surplus %%%
Fa=simdelta(:,2);  %%% All simulation results of Delta(u) %%%%


%%% Interplot to get delta(u) at u = 0 2 4 6 8 10 20 40 60 80 100 %%%
simdel=[];
for i=1:length(ind)
    z=min(find( a > ind(i)));
    if isempty(z)
        simdel(i)=1;
    else
        simdel(i)=(Fa(z)*(ind(i)-a(z-1))+Fa(z-1)*(a(z)-ind(i)))/(a(z)-a(z-1));
    end
end

%%% Display delta(u) at initial u = 0 2 4 6 8 10 20 40 60 80 100 %%%
[ind',simdel']

%%% Plot
plot(a,Fa,'-r');
hold off


%%%-------------------------------------------------------------------------




%%% ------------------------Calculate G(u,y)-------------------------------

%%% --Example 3.1
if dist==1
  ind=[20 60 100];
  umax=100;
```

```
  u=1:1:umax;
  y=[1 3 5];

%%% --Example 3.2
else
  ind=[20 100 200];
  umax=200;
  u=1:1:umax;
  y=[1 5 10];
end


%%%%%%%%%%%%%%%%%%%%%%--Recursive Algorithm --%%%%%%%%%%%%%%%%%%%%%%%%

%%% approximation of G(0.01,1), G(0.02,3),...,G(100,5),...
appGd=appG2(umax,y,gd,beta,p1);

%%% approximation of G(u,1), G(u,3), G(u,5), at u=1,2 3,...,100
appG=appGd(u*beta,:);

%%% Display approximation G(u,y)
appG(ind,:)


%%%%%%%%%%%%%%%%%%%%%%--Exact Value, Exponential only--%%%%%%%%%%%%%%%%%%

if dist==1
    exactGy=exactG(umax,theta,y,p1);

    %%% Display exact G(u,y)
    exactGy(ind+1,:)
end

%%%%%%%%%%%%%%%%%%%%%%%%--Lower & Upper Bounds--%%%%%%%%%%%%%%%%%%%%%%%%

boundG=boundGd3(deltald,deltaud,ind,beta,theta,y,c,p1,dist);

%%% Display
Glow=boundG(1:3,:)
Gup=boundG(4:6,:)
Gavg=boundG(7:9,:)


%%%-----------------------------------------------------------------------



%%%-------------------------Calculate F(x)-----------------------------------

%%% --Example 4.1 & 4.3
ind=[10 30 50];
x=[5 10 15];
```

```
%%%%%%%%%%%%%%%%%-- Recursive Algorithm --%%%%%%%%%%%%%%%%%%%%%%%%%%

appFxd=appFx(gd,ind,beta,theta,p1,x);

%%% Display
appFx=appFxd(ind*beta,:)


%%%%%%%%%%%%%%%%%-- Lower & Upper Bounds --%%%%%%%%%%%%%%%%%%%%%%%%%%

boundFx=boundFx4(deltald,deltaud,ind,x,beta,theta,c,p1,dist);

%%% Display
Fxlow=boundFx(1:3,:)
Fxup=boundFx(4:6,:)
Fxavg=boundFx(7:9,:)

%%%-------------------------------------------------------------------------




%%%----------------------Calculate F(x,y) ---------------------------------

%%% --Example 4.2 & 4.4
ind=[20 60 100];
umax=100;
y=[1 3 5];
x=[1 3 5];


%%%%%%%%%%%%%%%%%%%%%%%%Recursive Algorithm %%%%%%%%%%%%%%%%%%%%%%%%%

appFxyd=appFxy1(Hd,gd,beta,theta,p1,c,x,y,umax,dist);

%%% Display
appFxy=[appFxyd(ind*beta,1,1), appFxyd(ind*beta,2,2), appFxyd(ind*beta,3,3)]


%%%%%%%%%%%%%%%%%%%%%%%%Lower & Upper Bounds %%%%%%%%%%%%%%%%%%%%%%%%%

boundFxy=boundFxy2(deltald,deltaud,ind,x,y,beta,theta,c,p1,umax,dist);

%%% Display
Fxylow=boundFxy(1:3,:,:)
Fxyup=boundFxy(4:6,:,:)
Fxyavg=boundFxy(7:9,:,:)


%%%%%%%%%%%%%%%%%%-- Exact Value, Exponential only --%%%%%%%%%%%%%%%%%%
```

```
if dist==1
    %%% Exact delta(0), delta(0.01),delta(0.02),...,delta(100)
    exactdelscl=exactdelta2(umax,beta,theta,p1);

    %%% Excat delta(0), delta(1),delta(2),....delta(100)
    exactdel=exactdelscl(u*beta+1);

    exactGy=exactG(umax,theta,max(x)+max(y),p1);

    exactFxyd=exactFxy1(umax,x,y,exactGy,exactdel);

    %%% Display
    exactFxy=[exactFxyd(ind+1,1,1), exactFxyd(ind+1,2,2), exactFxyd(ind+1,3,3)]
 end

%%%-------------------------------------------------------------------------

diary off
```

2. The sub functions called by main

```
function out=Hdy2(max,beta,theta,p1,c,dist)
format long

x=1:1:max*beta;

s=length(x);


if dist==1
  ptheta=p1;   %%%----Exponential(p1)----%%%
  p1=ptheta;

  f0 =-(-exp(1/ptheta/beta)-ptheta*beta+
      ptheta*beta*exp(1/ptheta/beta))/exp(1/ptheta/beta);
  fk =ptheta*beta*exp(-(x+1)/ptheta/beta)-2*ptheta*beta*
      exp(-x/ptheta/beta)+ptheta*beta*exp(-(x-1)/ptheta/beta);

elseif dist==2
  if p1==1
    palpha=2;    ptheta=1;   %%%----Pareto(2,1)----%%%
  elseif p1==2
    palpha=2;    ptheta=2;   %%%----Pareto(2,2)----%%%
  end

  f0=1-(1/beta+ptheta)^(1-palpha)*(ptheta^palpha)*beta/(1-palpha)+
     ptheta*beta/(1-palpha);
  fk=beta.*(ptheta^palpha)/(1-palpha)*(2*(x./beta+ptheta).^(1-palpha)-
     ((x-1)./beta+ptheta).^(1-palpha)-((x+1)./beta+ptheta).^(1-palpha));

elseif dist==3
  if p1==1
    ptheta=0.5; ptau=0.5;    %%%----Weibull(0.5,0.5)----%%%
  elseif p1==2
    ptheta=1; ptau=0.5;      %%%----Weibull(1,0.5)----%%%
  end

  f0 =-(-2*ptheta*beta-exp((1/ptheta/beta)^(1/2))-2*ptheta*beta*
      (1/ptheta/beta)^(1/2)+2*ptheta*beta*exp((1/ptheta/beta)^(1/2)))/
      exp((1/ptheta/beta)^(1/2));
  fk =2*ptheta*beta*exp(-((x+1)./ptheta/beta).^(1/2)).*((x+1)./ptheta/
      beta).^(1/2)+2*ptheta*beta*exp(-((x+1)./ptheta/beta).^(1/2))-
      4*ptheta*beta*exp(-(x./ptheta/beta).^(1/2)).*(x./ptheta/beta).^(1/2)-
      4*ptheta*beta*exp(-(x./ptheta/beta).^(1/2))+2*ptheta*beta*
      exp(-((x-1)./ptheta/beta).^(1/2)).*((x-1)./ptheta/beta).^(1/2)+
      2*ptheta*beta*exp(-((x-1)./ptheta/beta).^(1/2));

elseif dist==4
  if p1==1
    palpha=2; ptheta=0.5;     %%%----Gamma(2,0.5)----%%%

    f0 =-(-1-exp(1/beta)^2-beta+beta*exp(1/beta)^2)/exp(1/beta)^2;
```

```
    fk =exp(-2*(x+1)./beta).*x+exp(-2*(x+1)./beta)+exp(-2*(x+1)./beta).*beta-
        2*x.*exp(-2*x./beta)-2*exp(-2*x./beta).*beta+exp(-2*(x-1)./beta).*x-
        exp(-2*(x-1)./beta)+exp(-2*(x-1)./beta).*beta;

  elseif p1==2
    palpha=2; ptheta=1;        %%%----Gamma(2,1)----%%%

    f0 =-(-1-exp(1/beta)-2*beta+2*beta*exp(1/beta))/exp(1/beta);
    fk =exp(-(x+1)./beta)+2*exp(-(x+1)./beta).*beta+exp(-(x+1)./beta).*x-
        4*exp(-x./beta).*beta-2*x.*exp(-x./beta)+exp(-(x-1)./beta).*x-
        exp(-(x-1)./beta)+2*exp(-(x-1)./beta).*beta;

  end


elseif dist==5 & p1==1
    palpha=4;    ptheta=3;    %%%----Pareto(4,3)-----%%%%

    p1=ptheta/(palpha-1);

    f0=1-(1/beta+ptheta)^(1-palpha)*(ptheta^palpha)*beta/(1-palpha)+
       ptheta*beta/(1-palpha);
    fk=beta.*(ptheta^palpha)/(1-palpha)*(2*(x./beta+ptheta).^(1-palpha)-
       ((x-1)./beta+ptheta).^(1-palpha)-((x+1)./beta+ptheta).^(1-palpha));

elseif dist==6
  if p1==1
      ptheta=1; ptau=1;       %%%----Weibull(1,1)----%%%
  else p1==2
      ptheta=2; ptau=1;       %%%----Weibull (2,1)----%%%
  end

  f0 = -(-exp(1/ptheta/beta)-ptheta*beta+ptheta*beta*exp(1/ptheta/beta))/
       exp(1/ptheta/beta);
  fk =ptheta*beta*exp(-(x+1)./ptheta/beta)-2*ptheta*beta*exp(-x./ptheta/beta)+
      ptheta*beta*exp(-(x-1)./ptheta/beta);

end


lambda=c/((1+theta)*p1);
lambda=lambda*(1-f0)/(beta*c);

fk=fk/(1-f0);


h=[];
h(1)=exp(-lambda);        %h(1)=h_d(0)
for i=1:1:s
    h(i+1)=(lambda/i)*(1:1:i).*fk(1:1:i)*h(i:-1:1)';
end
```

```
H=cumsum(h);

out=H;

%%%-------------------------------------------------------------------------%%%


function out=appdelta3(umax,beta,theta,g,h0)
format long


x=1:1:umax*beta;

delta=[];
delta(1)=theta/((1+theta)*h0);
del=delta;

for i=2:1:length(x)
    delta(i)=delta(1)+g(2:i)*del;
    del=[delta(i);del];
end

delta=[theta/(1+theta),delta];

out=delta';


%%%-------------------------------------------------------------------------%%%



function out=bounddelta4(umax,beta,theta,dist,p1)

u=0:1:umax*beta;
x=u/beta;
q=1/(1+theta);

size=length(x);
lambda=1;

if dist==1     %%%----Exponential ----%%%
    ptheta=p1;
    H =(-ptheta*exp(-x/ptheta)+ptheta)*lambda/ptheta;

elseif dist==2
    if p1==1
        palpha=2;   ptheta=1;  %%%----Pareto(2,1)----%%%
    elseif p1==2
        palpha=2;   ptheta=2;  %%%----Pareto(2,2)----%%%
    end
    H=1-(ptheta./(x+ptheta)).^(palpha-1);
```

```
elseif dist==3
    if p1==1
        ptheta=0.5; ptau=0.5;    %%%----Weibull(0.5,0.5)----%%%
    elseif p1==2
        ptheta=1;   ptau=0.5;    %%%----Weibull(1,0.5)----%%%
    end
    %for ptau=0.5, any ptheta
    H =1/2*(-2*ptheta*exp(-(x./ptheta).^(1/2)).*(x./ptheta).^(1/2)-
       2*ptheta*exp(-(x./ptheta).^(1/2))+2*ptheta)*lambda/ptheta;

elseif dist==4
    if p1==1    %%%----Gamma(2,0.5)----%%%
    elseif p1==2
        x=x/p1; %%%----Gamma(2,1)----%%%
    end
    H=1-(x+1).*exp(-2*x);

elseif dist==5  %%%----Pareto(4,3)----%%%
    palpha=4;
    ptheta=3;
    H=1-(ptheta./(x+ptheta)).^(palpha-1);

elseif dist==6
    if p1==1
        ptheta=1; ptau=1;  %%%----Weibull(1,1)----%%%
    elseif p1==2
        ptheta=2; ptau=1;   %%%----Weibull(2,1)----%%%
    end
    %for ptau=1, any ptheta
    H =(-ptheta*exp(-x./ptheta)+ptheta)*lambda/ptheta;

end

h=diff(H); %h1,h2,...

%%%Calculate upper bound
fu=[];
fu(1)=1-q;

for t=1:1:size-1
    k=1:t;
    fu(t+1)=q*h(k)*fu(t:-1:1)';
end

psiu=1-cumsum(fu);

hl=[];
hl0=h(1);
h(1)=[];

fl=[];
fl(1)=(1-q)/(1-q*hl0);
```

```
for t=1:1:size-2
    k=1:t;
    fl(t+1)=q/(1-q*hl0)*h(k)*fl(t:-1:1)';
end

psil=[q,1-cumsum(fl)];

deltal=1-psiu;
deltau=1-psil;

out=[deltal;deltau];

%%%-------------------------------------------------------------------------%%%


function out=exactdelta(umax,beta,theta,p1)


u=1:1:umax*beta;    %u=1,2,3...
u=u/beta;   %u=0.01,0.02,0.03....

%%% delta(0.01),delta(0.02),...,delta(100)
exact=1-1/(1+theta)*exp(-theta*u/(1+theta)/p1);

%%% delta(0);
exact0=1-1/(1+theta);

exact=[exact0,exact];
out=exact';


%%%-------------------------------------------------------------------------%%%


function out=simdelta(c,theta,beta,p1,dist)

format long

lambda=c/((1+theta)*p1);


sim=100000;
L=[];

seed = 931316785;
rand('seed',seed);

n=2000;
for s=1:sim
```

```
    inttime=exprnd(1/lambda,n,1);

    if dist==1
        indamount=exprnd(p1,n,1);     %%%----Exp(p1)----%%%

    elseif dist==2
        if p1==1
            palpha=2;     ptheta=1;    %%%----Pareto(2,1)----%%%
        elseif p1==2
            palpha=2;     ptheta=2;    %%%----Pareto(2,2)----%%%

        end
        u=rand(n,1);
        indamount=ptheta*((1-u).^(-1/palpha)-1);

    elseif dist==3
        if p1==1
            ptheta=0.5; ptau=0.5;    %%%----weibull(0.5,0.5)----%%%
        elseif p1==2
            ptheta=1; ptau=0.5;      %%%----weibull(1,0.5)----%%%
        end
        indamount=wblrnd(ptheta,ptau,n,1);

    elseif dist==4
        if p1==1
            palpha=2; ptheta=0.5;    %%%----Gamma(2,0.5)----%%%
        elseif p1==2
            palpha=2; ptheta=1;      %%%----Gamma(2,1)----%%%
        end
        indamount=gamrnd(palpha,ptheta,n,1);

    elseif dist==5
        palpha=4;     ptheta=3;    %%%----Pareto(4,3)----%%%
        u=rand(n,1);
        indamount=ptheta*((1-u).^(-1/palpha)-1);

    elseif dist==6
         if p1==1
            ptheta=1; ptau=1;       %%%----weibull(1,1)----%%%
        elseif p1==2
            ptheta=2; ptau=1;       %%%----weibull(2,1)----%%%
        end
        indamount=wblrnd(ptheta,ptau,n,1);
    end
    amount=cumsum(indamount);
    time=cumsum(inttime);

    m=amount-time*c;
    maxL=max(0,max(m));
    L=[L,maxL];

end
```

```
[Fx,x]=ecdf(L);

out=[x,Fx];


%%%-------------------------------------------------------------------------%%%


function out=appG2(umax,y,gdy,beta,p1);
format long

y=y*beta;

gdy(1)=[];   %remove gd(0,0)
G0=cumsum(gdy);

G=[];
G(1,:)=G0(y);

for u=1:1:umax*beta
    for c=1:1:length(y)
        sum=0;
        for k=1:1:u
            sum=sum+gdy(k)*G(u-k+1,c);
        end
        G(u+1,c)=G0(1,u+y(c))-G0(1,u)+sum;
    end
end

out=G;


%%%-------------------------------------------------------------------------%%%


function out=exactG(umax,theta,ymax,p1)
format long


u=0:1:umax;
y=1:1:ymax;

G=[];
for i=1:1:length(u)
    for j=1:1:length(y)
        G(i,j)=1/(1+theta).*exp(-theta*u(i)/(1+theta)/p1).*(1-exp(-y(j)/p1));
    end
end
```

```
out=G;


%%%-------------------------------------------------------------------------%%%


function out=boundGd3(deltal,deltau,ind,beta,theta,y,c,p1,dist);
format long

lambda=c/(1+theta)/p1;

psilow=[1-deltau]';
psiup=[1-deltal]';

Glow=[];
Gup=[];

for j=1:1:length(y)
  for i=1:1:length(ind)
    x=0:(1/beta):ind(i)-1/beta;
    s=ind(i)*beta+1;

    if dist==1      %%%----Exponential----%%%
      G0y=lambda/c*p1*(1-exp(-y(j)/p1));
      Gr=lambda/c*p1*exp(-x/p1)*(1-exp(-1/beta/p1));
      Gry=lambda/c*p1*exp(-(x+y(j))/p1)*(1-exp(-1/beta/p1));
      Grt=lambda/c*p1*exp(-ind(i)/p1)*(1-exp(-y(j)/p1));

    elseif dist==2
      palpha=2; ptheta=1;
      if p1==1      %%%----Pareto(2,1)----%%%
        palpha=2; ptheta=1;
        G0y=lambda/c*p1*(1-(1+y(j)).^(-palpha+1));
        Gr=lambda/c*p1*((1+x).^(-palpha+1)-(1+x+1/beta).^(-palpha+1));
        Gry=lambda/c*p1*((1+x+y(j)).^(-palpha+1)-(1+x+y(j)+1/beta).^(-palpha+1));
        Grt=lambda/c*p1*((1+ind(i))^(-palpha+1)-(1+ind(i)+y(j))^(-palpha+1));

       elseif p1==2  %%%----Pareto(2,2)----%%%
         palpha=2; ptheta=2;
         G0y = lambda/c*p1*(1-2/(2+y(j)));
         Gr = lambda/c*p1*(2./(2+x)-2./(2+x+1/beta));
         Gry = lambda/c*p1*(2./(2+x+y(j))-2./(2+x+y(j)+1/beta));
         Grt = lambda/c*p1*(2./(2+ind(i))-2./(2+ind(i)+y(j)));
       end

     elseif dist==3   %%%----Weibull(0.5,0.5), Weibull(1,0.5)----%%%
       G0y=lambda/c*p1*(1-exp(-(2*y(j)/p1)^0.5)*(1+(2*y(j)/p1)^0.5));
       Gr=lambda/c*p1*(exp(-(2*x./p1).^0.5).*(1+(2*x./p1).^0.5)-
          exp(-(2*(x+1/beta)./p1).^0.5).*(1+(2*(x+1/beta)./p1).^0.5));
       Gry=lambda/c*p1*(exp(-(2*(x+y(j))./p1).^0.5).*(1+(2*(x+y(j))./p1).^0.5)-
```

```
            exp(-(2*(x+y(j)+1/beta)./p1).^0.5).*(1+(2*(x+y(j)+1/beta)./p1).^0.5));
        Grt=lambda/c*p1*(exp(-(2*ind(i)/p1).^0.5).*(1+(2*ind(i)/p1).^0.5)-
            exp(-(2*(ind(i)+y(j))/p1).^0.5).*(1+(2*(ind(i)+y(j))/p1).^0.5));


    elseif dist==4
      if p1==1       %%%----Gamma(2,0.5)----%%%
        G0y=lambda/c*(1-(1+y(j))*exp(-2*y(j)));
        Gr=lambda/c*((1+x).*exp(-2*x)-(1+x+1/beta).*exp(-2*(x+1/beta)));
        Gry=lambda/c*((1+x+y(j)).*exp(-2*(x+y(j)))-(1+x+y(j)+1/beta).*
            exp(-2*(x+y(j)+1/beta)));
        Grt=lambda/c*((1+ind(i)).*exp(-2*ind(i))-(1+ind(i)+y(j)).*
            exp(-2*(ind(i)+y(j))));

      elseif p1==2     %%%----Gamma(2,1)----%%%
        G0y=lambda/c*(2-2*exp(-y(j))-y(j)*exp(-y(j)));
        Gr=lambda/c*((2+x).*exp(-x)-(2+x+1/beta).*exp(-(x+1/beta)));
        Gry=lambda/c*((2+x+y(j)).*exp(-(x+y(j)))-(2+x+y(j)+1/beta).*
            exp(-(x+y(j)+1/beta)));
        Grt=lambda/c*((2+ind(i)).*exp(-ind(i))-(2+ind(i)+y(j)).*
            exp(-(ind(i)+y(j))));

      end

    elseif dist==5     %%%----Pareto(4,3)----%%%
        G0y=lambda/c*(1-(3/(y(j)+3))^3);
        Gr=lambda/c*((3./(x+3)).^3-(3./(x+1/beta+3)).^3);
        Gry=lambda/c*((3./(x+y(j)+3)).^3-(3./(x+y(j)+1/beta+3)).^3);
        Grt=lambda/c*((3./(ind(i)+3)).^3-(3./(ind(i)+y(j)+1/beta+3)).^3);

    elseif dist==6    %%%----Weibull(1,1) Weibull(2,1)----%
         G0y=lambda/c*p1*(1-exp(-y(j)./p1));
         Gr=lambda/c*p1*(exp(-x./p1)-exp(-(x+1/beta)./p1));
         Gry=lambda/c*p1*(exp(-(x+y(j))./p1)-exp(-(x+y(j)+1/beta)./p1));
         Grt=lambda/c*p1*(exp(-ind(i)/p1)-exp(-(ind(i)+y(j))/p1));

    end

     Grl=psilow(s:-1:2)*(Gr-Gry)';
     Gru=psiup(s-1:-1:1)*(Gr-Gry)';
     Glow(i,j)=(Grl+Grt-psiup(s)*G0y)/(1-psilow(1)); %%%Lower bound for G(u,y)
     Gup(i,j)=(Gru+Grt-psilow(s)*G0y)/(1-psiup(1));  %%%Upper bound for G(u,y)

  end

end

out=[Glow;Gup;(Glow+Gup)/2];



%%%-------------------------------------------------------------------------%%%
```

```
function out=appFx(gdy,u,beta,theta,p1,x)

s1=length(u);
s2=length(x);
umax=u(s1);

u=u*beta-1;
x=x*beta-1;

F0=cumsum(gdy);

F=[];
F(1,:)=F0(x);

for r=1:1:u(s1)
    for c=1:1:s2
        sum=0;
        if r<x(c)
            for j=1:1:r
                sum=sum+gdy(j)*F(r-j+1,c);
            end
            sum2=0;
            for j=r+1:1:x(c)
             sum2=sum2+gdy(j);
            end
            F(r+1,c)=sum+sum2;
        else
            for j=1:1:r
                sum=sum+gdy(j)*F(r-j+1,c);
            end
            F(r+1,c)=sum;
        end

    end
end

out=F;



%%%-------------------------------------------------------------------------%%%


function out=boundFx4(deltalow,deltaup,u,x,beta,theta,c,p1,dist)
format long

umax=max(u);
xmax=max(x);
lambda=c/(1+theta)/p1;

deltal=deltalow((0:1:umax)*beta+1);%delta u=0,1,2,..
deltau=deltaup((0:1:umax)*beta+1);
```

```
posail=1-deltau;
posaiu=1-deltal;

delta0=theta/(1+theta);
posai0=1-delta0;

if dist==1          %%%----Exponential----%%%
  G0x=lambda/c*p1*(1-exp(-x/p1));
elseif dist==2
  if p1==1        %%%----Pareto(2,1)----%%%
    G0x=lambda/c*(x./(x+1));
  elseif p1==2    %%%----Pareto (2,2)----%%%
    G0x=lambda/c*p1*(x./(x+2));
  end
elseif dist==3    %%%----Weibull(0.5,0.5), Weibull(1,0.5)----%%%
  G0x=lambda/c*p1*(1-exp(-(2*x/p1).^0.5).*(1+(2*x/p1).^0.5));
elseif dist==4    %%%----Gamma----%%%
  G0x=lambda/c*p1*(1-exp(-2*x./p1)-x./p1.*exp(-2*x./p1));
elseif dist==5    %%%----Pareto(4,3)----%%%
  G0x=lambda/c*(1-(3./(x+3)).^3);
elseif dist==6    %%%----Weibull(1,1) Weibull(2,1)----%%%
  G0x=lambda/c*p1*(1-exp(-x./p1));
end

Flow=[];
Fup=[];

for r=1:1:length(u)
  for c=1:1:length(x)
    if u(r)<=x(c)
      Flow(r,c)=(1-G0x(c))/delta0*posail(u(r)+1)-(posai0-G0x(c))/delta0;
      Fup(r,c)=(1-G0x(c))/delta0*posaiu(u(r)+1)-(posai0-G0x(c))/delta0;
    else
      temp=u(r)-x(c);
      out=boundGd3(deltalow,deltaup,temp,beta,theta,x(c),c,p1,dist);
      Gl=out(1);
      Gh=out(2);

      Flow(r,c)=Gl-(1-G0x(c))/(1-posai0)*(posaiu(temp+1)-posail(u(r)+1));
      Fup(r,c)=Gh-(1-G0x(c))/(1-posai0)*(posail(temp+1)-posaiu(u(r)+1));
    end
  end
end

out=[Flow;Fup;(Flow+Fup)/2];



%%%-------------------------------------------------------------------------%%%
```

```
function out=boundFxy2(deltald,deltaud,ind,x,y,beta,theta,c,p1,umax,dist);

lambda=c/(1+theta)/p1;

s=length(ind);
u=0:1:umax;

deltal=deltald(u*beta+1);%delta u=0,1,2,..
deltau=deltaud(u*beta+1);
psil=1-deltau;
psiu=1-deltal;

delta0=theta/(1+theta);
psi0=1-delta0;

boundGd=boundGd3(deltald,deltaud,ind,beta,theta,y,c,p1,dist);
Glow=boundGd(1:s,:);
Gup=boundGd(s+1:2*s,:);

Flow=[];
Fup=[];
for r=1:1:s
  for j=1:1:length(x)
    for k=1:1:length(y)
      if dist==1          %%%----Exponential----%%%
        G0x = lambda/c*p1*(1-exp(-x(j)/p1));
        G0xy= lambda/c*p1*(1-exp(-(x(j)+y(k))/p1));
      elseif dist==2      %%%----Pareto(2,1), Pareto(2,2)----%%%
        G0x = lambda/c*p1*(x(j)./(x(j)+p1));
        G0xy= lambda/c*p1*((x(j)+y(k))./(x(j)+y(k)+p1));
      elseif dist==3       %%%---- Weibull(0.5,0.5) Weibull(1,0.5)----%%%
        G0x = lambda/c*p1*(1-exp(-(2*x(j)./p1).^0.5).*(1+(2*x(j)/p1).^0.5));
        G0xy= lambda/c*p1*(1-exp(-(2*(x(j)+y(k))./p1).^0.5).
              *(1+(2*(x(j)+y(k))./p1).^0.5));
      elseif dist==4      %%%---- Gamma(2,0.5) Gamma(2,1)----%%%
        G0x = lambda/c*p1*(1-exp(-2*x(j)./p1)-x(j)./p1.*exp(-2*x(j)./p1));
        G0xy= lambda/c*p1*(1-exp(-2*(x(j)+y(k))./p1)-(x(j)+y(k))./p1.
              *exp(-2*(x(j)+y(k))./p1));
      elseif dist==5      %%%---- Pareto(4,3)----%%%
        G0x = lambda/c*(1-(3/(x(j)+3)).^3);
        G0xy= lambda/c*(1-(3/(x(j)+y(k)+3)).^3);
      elseif dist==6      %%%----Weibull(1,1) Weibull(2,1)----%%%
        G0x = lambda/c*p1*(1-exp(-x(j)./p1));
        G0xy= lambda/c*p1*(1-exp(-(x(j)+y(j))./p1));
      end

      if ind(r)<=x(j)
        Flow(r,j,k)= Glow(r,k)+deltau(ind(r)+1)/delta0*(G0x-G0xy);
        Fup(r,j,k) = Gup(r,k)+deltal(ind(r)+1)/delta0*(G0x-G0xy);

      else
        temp= ind(r)-x(j);
        out = boundGd3(deltald,deltaud,temp,beta,theta, [x(j)+y(k),x(j)],c,
```

```
                    p1,dist);
        Gl = out(1,:);
        Gh  =out(2,:);

        Flow(r,j,k)= Glow(r,k)-Gh(1,1)+Gl(1,2)+G0x*(psil(temp+1)-
                     psiu(ind(r)+1))/delta0+G0xy*(psil(ind(r)+1)-
                     psiu(temp+1))/delta0;
        Fup(r,j,k) = Gup(r,k)-Gl(1,1)+Gh(1,2)+G0x*(psiu(temp+1)-
                     psil(ind(r)+1))/delta0+G0xy*(psiu(ind(r)+1)-
                     psil(temp+1))/delta0;
      end
    end
  end
end

[Flow;Fup];
Fxylow = [Flow(:,1,1),Flow(:,2,2),Flow(:,3,3)];
Fxyup  = [Fup(:,1,1),Fup(:,2,2),Fup(:,3,3)];
Fxyavg = (Fxylow+Fxyup)/2;

out=[Fxylow; Fxyup; Fxyavg];


%%%-------------------------------------------------------------------------%%%


function out=exactFxy(umax,x,y,G,del)

Fxy=[];
for i=0:1:umax
  for j=1:1:length(x)
    for k=1:1:length(y)
      if i<=x(j)
        Fxy(i+1,j,k)=G(i+1,y(k))+del(i+1)/del(1)*(G(1,x(j))-G(1,x(j)+y(k)));

      else
        Fxy(i+1,j,k)=G(i+1,y(k))-G(i-x(j)+1,x(j)+y(k))+G(i-x(j)+1,x(j))+
                     (del(i+1)-del(i-x(j)+1))/del(1)*(G(1,x(j))-G(1,x(j)+
                      y(k)));
      end
    end
  end
end

out=Fxy;


%%%-------------------------------------------------------------------------%%%
```

Appendix B

Matlab Codes II

1. Code of calculating the moments of time to ruin.

```
function moments
format long

scalor=100;
theta=0.1;
p1=1;
lambda=1;


c=(1+theta)*p1*lambda;

dist=5;

if dist==1          %%%---- Exponentail----%%%
    ptheta=1;
    p1=1;
    p2=2;
    p3=6;
    p4=24;
    umax=50;
    u=0:1:umax;
    ind=[0 10 20 30 40 50];
elseif dist==3
  if p1==1           %%%---- Weibull(0.5,0.5) ----%%%
    ptheta=0.5; ptau=0.5;
    p1=1;
    p2=6;
    p3=90;
    p4=2520;
    umax=80;
    u=0:1:umax;
    ind=[0 20 40 60 80];
  elseif p1==2      %%%---- Weibull(1,0.5) ----%%%
    ptheta=1; ptau=0.5;
    p1=2;
    p2=24;
    p3=720;
    p4=40320;
    umax=80;
    u=0:1:umax;
    ind=[0 20 40 60 80];
  end
elseif dist==5      %%%---- Pareto(4,3) ----%%%
```

```
    palpha=4;ptheta=3;
    p1=1;
    p2=3;
    p3=27;
    %p4 doesn't exist
    umax=80;
    u=0:1:umax;
    ind=[0 2 4 6 8 10 20 40 60 80];
end



if dist==1   %%%----Exponential(1)----%%%
    %%% exact delta(0), delta(0.01),delta(0.02),...,delta(100)
    exactdelscl=exactdelta2(umax,scalor,theta,p1);
    %%% excat delta(0), delta(1),delta(2),....delta(100)
    exactdel=exactdelscl(u*scalor+1);

    exactpsi=psi(umax,scalor,exactdelscl,theta,lambda,dist);
    exactpsi1scl=exactpsi(1,:);
    exactpsi2scl=exactpsi(2,:);
    exactpsi3scl=exactpsi(3,:);

    exactpsi1=exactpsi1scl(u*scalor+1); %psi1(u), u=0,1,2,...
    exactpsi2=exactpsi2scl(u*scalor+1); %psi2(u), u=0,1,2,...
    exactpsi3=exactpsi3scl(u*scalor+1); %psi3(u), u=0,1,2,...

    exactmean=exactpsi1'./(1-exactdel);
    exactE2=exactpsi2'./(1-exactdel);
    exactstd=(exactE2-exactmean.^2).^0.5;
    exactE3=exactpsi3'./(1-exactdel);
    exactsk=(exactE3-3*exactmean.*exactE2+2*exactmean.^3)./exactstd.^3;

    plot(u',exactsk,'r:');
    hold on;
end

Hd=Hdy2(umax,scalor,theta,p1,c,dist);
hd0=Hd(1);
gd=(1-Hd)/hd0;
gd(1)=0;

%%% approximate delta(0.00), delta(0.01),delta(0.02),...,delta(umax)
appdelscl=appdelta3(umax,scalor,theta,gd,hd0);

%%% approximate delta(0), delta(1),delta(2),....delta(umax)
appdel=appdelscl(u*scalor+1);

apppsiscl=psi(umax,scalor,appdelscl,theta,lambda,dist);

apppsi1scl=apppsiscl(1,:); %psi1(u),u=0,0.01,0.02,...
apppsi2scl=apppsiscl(2,:); %psi2(u),u=0,0.01,0.02,...
```

```
apppsi1=apppsi1scl(u*scalor+1); %psi1(u), u=0,1,2,...
apppsi2=apppsi2scl(u*scalor+1); %psi2(u), u=0,1,2,...

appmean=apppsi1'./(1-appdel);
appE2=apppsi2'./(1-appdel);
appstd=(appE2-appmean.^2).^0.5;


difmean=difmean1(umax,theta,p1,lambda);
difstd=difstddev(umax,theta,lambda,dist);


if dist==1
    apppsi3scl=apppsiscl(3,:); %psi3(u),u=0,0.01,0.02,...
    apppsi3=apppsi3scl(u*scalor+1); %psi3(u), u=0,1,2,...
    appE3=apppsi3'./(1-appdel);
    appsk=(appE3-3*appmean.*appE2+2*appmean.^3)./appstd.^3;

    difsk=difskewness(umax,theta,lambda,dist);

    plot(u',appsk,'b:');
    hold on;
    plot(u',difsk,'g:');
    hold off;
end
if dist==3 | dist==4
    apppsi3scl=apppsiscl(3,:); %psi3(u),u=0,0.01,0.02,...
    apppsi3=apppsi3scl(u*scalor+1); %psi3(u), u=0,1,2,...
    appE3=apppsi3'./(1-appdel);
    appsk=(appE3-3*appmean.*appE2+2*appmean.^3)./appstd.^3;

    difsk=difskewness(umax,theta,lambda,dist);

    plot(u',appsk,'b:');
    hold on;
    plot(u',difsk,'g:');
    hold off;
end

if dist==1
    mean=[exactmean(ind+1),appmean(ind+1),difmean(ind+1)]
    std=[exactstd(ind+1),appstd(ind+1),difstd(ind+1)]
    skeness=[exactsk(ind+1),appsk(ind+1),difsk(ind+1)]

elseif dist==3
    mean=[appmean(ind+1),difmean(ind+1)]
    std=[appstd(ind+1),difstd(ind+1)]
    skeness=[appsk(ind+1),difsk(ind+1)]
elseif dist==5
    mean=[appmean(ind+1),difmean(ind+1)]
    std=[appstd(ind+1),difstd(ind+1)]
end
```

%%%----------%%%

2. Code called by moments function.

```
function out=psi(umax,scalor,delta,theta,lambda,dist,p1)

u=0:1:umax*scalor;
u=u/scalor;           %u=0, 0.01,0.02,...50
s=length(u);

if dist==1
    p1=1;
    p2=2;
    p3=6;
    p4=24;
elseif dist==3
   if p1==1           %%%---- Weibull(0.5,0.5) ----%%%
    ptheta=0.5; ptau=0.5;
    p1=1;
    p2=6;
    p3=90;
    p4=2520;
   elseif p1==2       %%%---- Weibull(1,0.5) ----%%%
    ptheta=1; ptau=0.5;
    p1=2;
    p2=24;
    p3=720;
    p4=40320;
   end
elseif dist==5
    p1=1;
    p2=3;
    p3=27;
end

L1=p2/(2*theta*p1);
L2=p3/(3*theta*p1)+(p2/theta/p1)^2/2;

delta=delta';
psi1=[];
psi=(1-delta);
for i=1:s
  psi1(i)=L1*delta(i)-trapz(psi(1:1:i).*delta(i:-1:1))/scalor;
end
psi1=psi1/(lambda*p1*theta);

psi2=[];
for i=1:s
  psi2(i)=L2*delta(i)/(2*lambda*p1*theta)-trapz(psi1(1:1:i).*
           delta(i:-1:1))/scalor;
end
psi2=2*psi2/(lambda*p1*theta);
out=[psi1;psi2];

if dist==1 | dist==3 | dist==5
```

```
  L3=p4/(4*theta*p1)+(p2/theta/p1)^3*3/4+p2*p3/(theta*p1)^2;
  psi3=[];
  for i=1:s
    psi3(i)=3*delta(i)*L1*L2/(lambda*p1*theta)^3+delta(i)*L3/
          (lambda*p1*theta)^3-3/(lambda*p1*theta)*
           trapz(psi2(1:1:i).*delta(i:-1:1))/scalor;
  end
  out=[out;psi3];
end


%%%-------------------------------------------------------------------------%%%


function out=difmean1(umax,theta,p1,lambda);

u=0:1:umax;
mean=u/(lambda*theta*p1);
out=mean';


%%%-------------------------------------------------------------------------%%%


function d=difstddev(umax,theta,lambda,dist,p1)

u=0:1:umax;
if dist==1        %%%---- Exponential ----%%%
    p1=1;
    p2=2;
    p3=6;
    p4=24;
elseif dist==3
   if p1==1        %%%---- Weibull(0.5,0.5) ----%%%
    ptheta=0.5; ptau=0.5;
    p1=1;  %pk= ptheta^k*gamma(1+k/ptau)
    p2=6;
    p3=90;
    p4=2520;
   elseif p1==2    %%%---- Weibull(1,0.5) ----%%%
    ptheta=1; ptau=0.5;
    p1=2;  %pk= ptheta^k*gamma(1+k/ptau)
    p2=24;
    p3=720;
    p4=40320;
   end
elseif dist==5     %%%---- Pareto(4,3) ----%%%
    p1=1;
    p2=3;
    p3=27;
end
```

```
var=u*p2/(lambda^2*theta^3*p1^3);
d=var'.^0.5;

%%%-------------------------------------------------------------------------%%%


function d=difskewness(umax,theta,lambda,dist,p1)

u=1:1:umax;
if dist==1        %%%---- Exponential ----%%%
    p1=1;
    p2=2;
    p3=6;
    p4=24;
elseif dist==3
   if p1==1        %%%---- Weibull(0.5,0.5) ----%%%
    ptheta=0.5; ptau=0.5;
    p1=1;
    p2=6;
    p3=90;
    p4=2520;
  elseif p1==2     %%%---- Weibull(1,0.5) ----%%%
    ptheta=1; ptau=0.5;
    p1=2;
    p2=24;
    p3=720;
    p4=40320;
   end
elseif dist==5      %%%---- Pareto(4,3) ----%%%
    p1=1;
    p2=3;
    p3=27;
end

skw=3*(p2./(theta*p1*u)).^0.5;
skw=[0,skw];
d=skw';

%%%-------------------------------------------------------------------------%%%
```

Appendix C

Matlab Codes III

Code of calculating and plotting the density of the time to ruin.

```
%%% -- This program calculates and plots the distribution of the time to ruin.
%%% -- Three methods are used: Approximation, Invers Gaussion and Diffusion.


function density
format long

dist=5;
beta=20;
theta=0.1;
c=1;
intu=10;    %initial surplu u


t=[10 20 200]*(1+theta)*beta;  %[220,440,880]
ind=[5 10 20]*beta; %u=100 200 800

x=1:1:max(t)+max(ind); %x=1,2,...220,....1680
s=length(x);


if dist==1          %%%----Exponential(1)----%%%
  p1=1;
  p2=2;
  pmu=68; ptheta=116.284; % for initial u=20
  f0=1-beta*(1-exp(-1/beta));
  fk=beta*exp(-(x+1)/beta)*(exp(1/beta)-1)^2/(1-f0);

elseif dist==5      %%%----Pareto(4,3)----%%%
  p1=1;
  p2=3;
  pmu=126.80414; ptheta=41.1322;  % for initial u=10
  f0=1+beta*((3/(3+1/beta))^3-1);
  fk=3^3*beta.*((3+(x+1)./beta).^(-3)-2.*(3+x./beta).^(-3)+
     (3+(x-1)./beta).^(-3))./(1-f0);
end

lambda1=c/((1+theta)*p1);

%%%lambda per unit time
lambda=lambda1*(1-f0)/beta;

g=[];
```

```
g(1)=exp(-lambda);

for i=1:1:s-1
    g(i+1)=(lambda/i)*(1:1:i).*fk(1:1:i)*g(i:-1:1)';
end


tlambda=x*lambda; %lambda for time 0 to time t=1680
H=[];
h=[];
h(1,:)=exp(-tlambda); %h(1)=h_d(0) t=1,2,...1680
for i=2:1:s
  for j=1:1:i-1
    h(j+1,i)=(tlambda(i)/j)*(1:1:j).*fk(1:1:j)*h(j:-1:1,i);
  end
  H(i)=sum(cumsum(h(:,i)));
end
del0star=H./x;

%calculate delta(u), u*beta=1,2,...
del0t=H./x/g(1); %t=0,1,2,...,1680-1
del0t(1)=[];   %t=1,2,...1680-1

delt=[];
delt(1,:)=del0t;
for u=1:1:max(ind)-1
  for i=1:1:s-u-1
    delt(u+1,i)=(delt(u+1-1,i+1)-g(2:1:u+1)*delt(u:-1:1,i))/g(1);
  end
end

deltintu=delt(intu*beta);

if dist==1                  %%%---- Exponential(1) ----%%%
  apppsi=0.98534707373236; % for initial surplus u=20, delt(20)
  exctpsi=0.98534748888901;
elseif dist==5              %%%---- Pareto(4,3) ----%%%
  apppsi=0.475194;          %for initial surplus u=10, delt(10)
end

%%% Approximation method
appf = (1+theta)*beta*(deltintu(1:1:max(t)-1)-deltintu(2:1:max(t)))/(1-apppsi);
plot((1:1:max(t)-1)/(1+theta)/beta,appf,'k');
hold on;

%%% Inverse Gaussion method
x = 0.1:1:300;
invf = (ptheta./(2*pi*x.^3)).^0.5.*exp(-ptheta*((x-pmu)/pmu).^2./(2*x));
plot(x,invf,'g');
hold on;

%%% Diffusion method
f = intu/(2*pi*lambda1*p2).^0.5*x.^(-3/2).*exp(-(intu-theta*lambda1*x*p1).^2./
```

```
    (2*lambda1*x*p2));
plot(x,f,'b');
hold off;

%%%----------------------------------------------------------------------%%%
```

Bibliography

[1] N. L. Bowers, H. U. Geruer, J.C. Hickman, D.A. Jones, and C.J. Nesbitt. Actuarial mathematics. *Society of Actuaries*, 2000.

[2] Rui M.R. Cardoso and Alfredo D. Egidio dos Reis. Recursive calculation of time to ruin distributions. *Insurance: Mathematics and Economics*, 30, 2002.

[3] S. Cheng, H.U. Gerber, and E.S.W. Shiu. Discounted probabilities and ruin theory in the compound binomial model. *Insurance: Mathematics and Economics*, 26, 2000.

[4] S.D. Conte and C. De Boor. Elementary numerical analysis. 1980.

[5] Dickson David C.M. and Waters Howard R. The distribution of the time to ruin in the classical risk model. *Astin Bulletin*, 32(2):99–312, 2002.

[6] F. Delbaen. A remark on the moments of ruin time in classic risk theory. *Insurance: Mathematics and Economics*, 9, 1988.

[7] David C.M. Dickson. On the distribution of the surplus prior to ruin. *Insurance: Mathematics and Economics*, 11, 1992.

[8] David C.M. Dickson, Alfredo D. Egodio dos Reis, and Howard R. Waters. Some stable algorithms in ruin theory and their applications. *Astin Bulletin*, 25(2):153–175, 1995.

[9] David C.M. Dickson and Howard R. Waters. Recursive calculation of survival probabilities. *Astin Bulletin*, 21(2):199–221, 1991.

[10] S. Drekic and G.E. Willmot. On the density and moments of the time to ruin with exponential claims. *Astin Bulletin*, 3, 2003.

[11] J. Dubourdieu. Thkorte mathomattque du risque dans les assurances de roparttton, gauth cr-viilars, paris. 1952.

[12] F. Dufresne and H.U. Gerber. The surpluses immediately before and at ruin, and the amount of the claim causing ruin. *Insurance: Mathematics and Economics*, 7, 1988.

[13] F. Dufresne and H.U. Gerber. Three methods to calculate the probability of ruin. *Astin Bulletin*, 19(1):71–90, 1989.

[14] H.U. Gerber. An introduction to mathematical risk theory. *S.S. Huebner Foundation*, 1979.

[15] H.U. Gerber and E.S.W. Shiu. On the time value of ruin. *North American Actuarial Journal*, 2, 1998.

[16] M. Goovaerts and F. De Vylder. A stable recursive algorithm for evaluation of ultimate ruin probabilities. *Astin Bulletin*, 14(1):53–59, 1984.

[17] Stuart A. Klugman. Loss models. *Wiley Interscience*, 2004.

[18] X.S. Lin and G.E. Willmot. The moments of the time of ruin, the surplus before ruin, and the deficit at ruin. *S.S. Huebner Foundation Insurance: Mathematics & Economics*, 27:19–44, 2000.

[19] H.H. Panjer. Recusive evaluation of a family of compound distributions. *Astin Bulletin*, 12:22–26, 1981.

[20] H.H. Panjer. Direct calculation of ruin probabilities. *Astin Bulletin*, 7(1):1–7, 1986.

[21] H.H. Panjer and S. Wang. On the stability of recursive formulas. *Astin Bulletin*, 23, 1993.

[22] H.H. Panjer and G.E. Willmot. Insurance risk models. *Society of Actuaries, Schaumburg*, 1992.

[23] H.L. Seal. Survival probabilities. *John Wiley and Sons, New York*, 1978.

[24] F. De Vylder. A practical solution to the problem of ultimate ruin probability. *Scandinavian Actuarial Journal*, 1978.

[25] F. De Vylder and M.J. Goovaerts. Recursive calculation of finite-time ruin probabilities. *Insurance: Mathematics and Economics*, 7:1–7, 1988.