DIPLOMARBEIT

Simulation von Ruinwahrscheinlichkeiten im "heavy tailed case"

ausgeführt am

Institut für
Finanz- und Versicherungsmathematik der
Technischen Universität Wien

unter der Anleitung von

Ao.Univ. Prof. Dipl.-Ing. Dr. techn. Peter Grandits

durch

Mohamed Mehdi Garouachi, BSc.
Matrikelnummer: 11847609

Wien, am 13.05.2025

Kurzfassung

Die vorliegende Arbeit befasst sich mit dem klassischen Risikomodell unter der Annahme subexponentieller Schadensverteilungen. Es werden drei verschiedene Simulationsmethoden zur Schätzung der Ruinwahrscheinlichkeit vorgestellt und hinsichtlich ihrer asymptotischen Effizienz analysiert. Eine dieser Methoden basiert auf einem bedingten Monte-Carlo-Ansatz unter Verwendung von Ordnungsstatistiken und erweist sich unter bestimmten Voraussetzungen als asymptotisch effizient.

Abstract

This thesis addresses the classical risk model under the assumption of subexponential claim size distributions. Three different simulation methods for estimating the probability of ruin are presented and analyzed with regard to their asymptotic efficiency. One of these methods is based on a conditional Monte Carlo approach using order statistics and proves to be asymptotically efficient under certain conditions.

Danksagung

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ
الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ
الرَّحْمَٰنِ الرَّحِيمِ
مَالِكِ يَوْمِ الدِّينِ
إِيَّاكَ نَعْبُدُ وَإِيَّاكَ نَسْتَعِينُ
اهْدِنَا الصِّرَاطَ الْمُسْتَقِيمَ
صِرَاطَ الَّذِينَ أَنْعَمْتَ عَلَيْهِمْ غَيْرِ الْمَغْضُوبِ عَلَيْهِمْ وَلَا الضَّالِّينَ

Dieses Studium und die vorliegende Arbeit wären ohne die Unterstützung zahlreicher Menschen nicht möglich gewesen. Daher möchte ich an dieser Stelle all jenen meinen Dank aussprechen, die mich auf meinem Weg begleitet haben, auch wenn sie hier nicht namentlich erwähnt werden.
Mein besonderer Dank gilt meinem Betreuer, Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Peter Grandits, für die herausragende und engagierte Betreuung während meiner gesamten Studienzeit. Seine fachliche Expertise und seine stetige Unterstützung waren für mich von unschätzbarem Wert.
Ein herzliches Dankeschön richte ich auch an meine Lernpartnerin und langjährige Wegbegleiterin Lisa Retzer, die mich während des Studiums stets unterstützt hat. Ebenso danke ich meinem besten Freund Malek Elisa, der meine Masterarbeit sorgfältig Korrektur gelesen und mir bei zahlreichen Formulierungen geholfen hat.
Mein tiefster Dank gilt meinen Eltern, die mich nicht nur in den vergangenen Jahren, sondern bereits seit meiner Schulzeit auf meinem Weg begleitet und unterstützt haben. Ohne ihre beständige Rückendeckung wäre dieser akademische Werdegang nicht möglich gewesen.
Abschließend möchte ich meinem Bruder Hedi Garouachi danken, der mich in besonderer Weise geprägt und inspiriert hat.

Mohamed Mehdi Garouachi

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In dieser Arbeit wird die Simulation der Ruin Wahrscheinlichkeiten eines Risiko Prozesses betrachtet. Betrachtet wird die Ruin Wahrscheinlichkeit $\psi(u)$ eines klassischen Risiko Prozesses $U(t)$ mit Compound Poisson Verteilung. Der Startwert des Risikoprozesses $U(0)$ ist u und wird als Reserve bezeichnet, die Größe des Claims hat Verteilung B und ist heavy tailed. Die Claim Größen sind nicht negative Zufallsvariablen $\xi_i (i \in \mathbb{N})$ mit identsicher Verteilung, kumulativer Verteilungsfunktion $B(x)$ und endlichem Erwartungswert μ_B. Der Claim Ankunfts Prozess, ist ein homogener Poisson Prozess mit Rate $\lambda > 0$ und wird mit $N_t,\ t \geq 0$ bezeichnet, zum Zeitpunkt $t = 0$ beträgt der Prozess $N(0) = 0$. Die Prämie wird über eine konstante Rate c und über einen Zeithorizont bezahlt. Die Rate beträgt:

$$c = (1 + \theta)\lambda\mu_B$$

wobei $\theta > 0$ einen Aufschlag auf die Prämie bezeichnet, die ein Versicherungsnehmer zahlt, über den erwarteten Verlust hinaus. Dieser Aufschlag dient dazu, die Kosten für den Betrieb des Versicherungsunternehmens zu decken und einen Gewinn zu erzielen. Der Versicherungsüberschuss zum Zeitpunkt t wird mit $U(t)$ bezeichnet. Der totale Claim Prozess wird mit $R(t) = \sum_{i=1}^{N(t)} \xi_i$ bezeichnet und ist ein Compound Poisson Prozess. Für den Versicherungsüberschuss folgt

$$U(t) = u + ct - R(t).$$

Die Wahrscheinlichkeit des Ruins ist somit definiert als:

$$\psi(u) = P(\inf_{t \geq 0} U(t) < 0).$$

Ziel ist es diese Wahrscheinlichkeit für subexponentielle Schadensverteilungen durch drei verschiedene Methoden zu simulieren und zu untersuchen. Doch davor gibt es im anschlie-

ßenden Kapitel eine Einführung in die Risiko- und Ruintheorie, um etwaige Verwirrungen bezüglich Notation und Begrifflichkeiten zu vermeiden oder um eine Möglichkeit zu haben, das Wissen aufzufrischen.

2 Einführung in die stochastische Risiko- und Ruintheorie

2.1 Wichtige Prozesse

In diesem Abschnitt werden wichtige Prozesse betrachtet, mit denen in Folge häufig gearbeitet wird. Zuerst wird ein einfacher Poisson-Prozess betrachetet:

Definition 2.1. 1 (Poisson-Prozess)**.** Ein stochastischer Prozess N_t mit $t \geq 0$ über einem Wahrscheinlichkeitsraum (Ω, A, P) wird als Poisson-Prozess bezeichnet, wenn er rechtsseitig stetig ist mit Grenzwerten von links (càdlàg) und folgende Bedingungen erfüllt:

1. $N(0) = 0$
2. Der stochastische Prozess N_t hat unabhängige Inkremente für alle t_i mit $i \in 1, \ldots, n$ und $n \in \mathbb{N}$ so gilt, dass die Zuwächse $N(t_{i+1}) - N(t_i)$ unabhängig sind.
3. Füt die Inkremente gilt: $N(t_{i+1}) - N(t_i) \sim \mathcal{P}_{\lambda(t_{i+1}-t_i)}$, wobei mit$\mathcal{P}_{\lambda(t_{i+1}-t_i)}$ die Poisson-Verteilung mit Parameter $\lambda(t_{i+1} - t_i)$ bezeichnet wird.

Der Poisson-Prozess wird verwendet, um die Anzahl von Ereignissen in einem gewissen Intervall $[0, t]$ zu zählen. Der Parameter λ wird auch als die Intensität des homogenen Poisson-Prozesses bezeichnet und gibt an wie viele Sprünge in einem Zeitintervall erwartet werden. Somit ist der Erwartungswert des Prozesses N(t) gegeben als $\mathbb{E}[N(t)] = \lambda t$. Nun wird dieser Prozess genutzt, um den Zusammengesetzten Poisson-Prozess zu definieren.

Definition 2.1. 2 (Zusammengesetzter Poisson-Prozess)**.** Sei N(t) ein Poisson-Prozess mit Intensität λt und seien $\xi_1, \xi_2, \ldots$ unabhängige und identisch verteilte Zufallsvariablen die zusätzlich von N(t) unabhängig sind. Dann wird der stochastische Prozess

$$Y_t = \sum_{i=1}^{N(t)} \xi_i \tag{2.1}$$

ein zusammengesetzter Poission-Prozess oder Compound-Prozess genannt. Dieser ist ebenfalls wieder ein Sprung Prozess.

2.2 Individuelles und Kollektives Modell

Um den Gesamtschaden eines Versicherungsportfolios für ein gegebenes Intervall zu modellieren, wird in der Risikotheorie zwischen zwei Modellen unterschieden: dem individuellen Risikomodell und dem kollektiven Risikomodell.

- Individuelles Modell: Im individuellen Modell wird ein Gesamtschaden eines Portfolios als Summe von unabhängigen Zufallsvariablen betrachtet. Betrachtet wird ein Portfolio mit $n \geq 1$ Polizzen in einem festen Zeitraum und unabhängigen Einzelschäden $\xi_1, \ldots, \xi_n$ mit Verteilungsfunktionen $\mathbf{F_1}, \ldots, \mathbf{F_n}$. In diesem Modell muss nicht zwingend vorausgesetzt werden, dass $\xi_1, \ldots, \xi_n$ die gleichen Verteilungen haben. Sollten jedoch $\xi_1, \ldots, \xi_n$ unabhängig sein, spricht man von einem inhomogenen Modell. Sind die Verteilungen iid. und verschieden, spricht man von einem homogenen Portfolio. Der Gesamtschaden ist dann gegeben als:

$$S = \sum_{k=1}^{n} \xi_k. \tag{2.1}$$

  Der Erwartungswert $\mathbb{E}$ und die Varianz Var für das individuelle Modell sind:

$$\mathbb{E}[S] = \sum_{k=1}^{n} \mathbb{E}[\xi_k] \qquad \mathrm{Var}[S] = \sum_{k=1}^{n} \mathrm{Var}[\xi_k]$$

- **Kollektives Modell:** Im Vergleich zum bereits vorgestellten individuellen Modell, wird beim kollektiven Modell eine Zufallssumme betrachtet. Die Schadenhöhe $S = \sum_{k=1}^{N} \xi_k$ ist eine Zufallsvariable. N selbst ist eine Zufallsvariable. Durch $N(t)$ ist die Anzahl der Schäden bis zum Zeitpunkt t gegeben. Das kollektive Modell beschreibt den Gesamtschaden einer Risikogruppe mit Hilfe einer Zufallsvariable für die Anzahl der Schäden und einer Zufallsvariable für die Höhe des Schadens pro Schaden, ohne auf die Ebene der Einzelrisiken zurückzugreifen. Mit diesem Ansatz wird die Verteilung des Gesamtschadens ersichtlicher dargestellt. Die Zufallssumme des Erwarungswertes und Varianz sind:

$$\mathbb{E}[S] = \mathbb{E}[\xi]\mathbb{E}[N] \qquad \mathrm{Var}[S] = \mathrm{Var}[\xi]\mathbb{E}[N] + \mathbb{E}[\xi]^2\mathrm{Var}[N]$$

[Hubalek(2022)]

2.3 Das klassische Risikomodell

Die oben genannten Prozesse werden verwendet, um Risiken zu modellieren die in einem Versicherungsportfolio auftreten können. Dabei muss jedoch berücksichtigt werden, dass sowohl Ungewissheit bezüglich Zeitpunkt des Eintritts der Schäden, als auch deren Schadenhöhe herrscht. Durch mathematische Modellbildung wird das Risiko besonders eingiebig betrachtet und besser kontrollierbar dargestellt. Aus diesem Grund ist folgendes Modell von existenzieller Relevanz für Versicherungsunternehmen und deren Schutz. Also gibt es triftige Gründe Risikoprozesse zu studieren, die benötigt werden um Modelle zu bilden die Risiken beschreiben und auch Fragen über die etwaige Größe des Sicherheitszuschlages beantworten. Benötigt werden diese auch um Prämien zu kalkulieren oder um Reserven und Höhe des Selbstbehaltes zu bestimmen. Der erste Mathematiker, der ein Modell wie folgt betrachtet hat, war Filip Lundberg in seiner berühmten Dissertation [Lundberg(1903)]. Seine Arbeit wurde in späteren Jahren von Harald Cramér verallgemeinert [Cramér(1930)] und [Cramér(1955)]. Aus diesem Grund wird das Modell als Cramér-Lundberg-Modell oder klassisches Risikomodell bezeichnet.

Definition 2.1. 3 (Das Cramér-Lundberg Modell)**.** Im Cramér-Lundberg Modell beschreibt der Risikoprozess (genannt auch Surplus oder freie Reserve) das Vermögen zum Zeitpunkt t und wird mit $U(t)$ bezeichnet und hat die Form:

$$U(t) = u + ct - R(t)\,,$$

wobei u die Höhe des Vermögens zum Zeitpunkt $t = 0$ ist und c die konstante Prämienrate. Im Imtervall [0,t] wird eine Gesamtprämie von ct eingenommen. $R(t) = \sum_{i=1}^{N(t)} \xi_i$ ist eine Zufallssumme und bezeichnet die Gesamtschadenhöhe.

[Hubalek(2022)]
Die Zufälligkeit der Anzahl der Schäden wird durch den Poisson-Prozess N(t) mit Parameter λ gewährleistet. Die Einzelschäden ξ_i sind eine Folge von unabhängig und identisch verteilter Zufallsvariablen, die zusätzlich unabhängig von N(t) sind. Aus Gründen der Sinnhaftigkeit wird zusätzlich verlangt, dass die Einzelschäden positiv sind. Der Gesamtschaden R(t) ist ein Zusammengesetzter Poisson-Prozess.

Der klassische Risikoprozess ist eine dynamische Betrachtung, also eine Modellierung in stetiger Zeit, wie die Entwicklung des Gesamtschadens bzw. des Risikoprozesses über die Zeit läuft. Der Prozess lässt sich durch folgende drei Punkte präzisieren:

- Die Höhe das Startvermögens
- Die Höhe der Prämieneinnahmen bis zum Ende des Betrachtungsintervalls
- Die Höhe des Gesamtschadens bis zum Ende des Betrachtungsintervalls

In der Abbildung 2.1 sieht man eine bildliche Veranschaulichung dieses klassischen Modells:

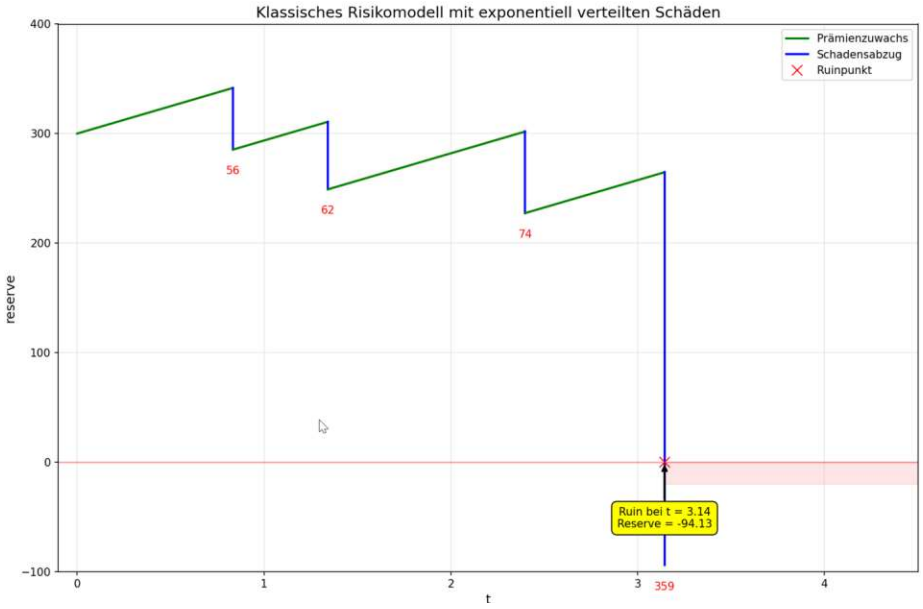

Abbildung 2.1: Klassisches Risikomodell

Der Prozess startet bei einem Wert (Startvermögen) und nimmt anfangs durch die erste Prämienzahlung zu. Es folgt ein Schadenfall und die freie Reserve sinkt. Im

Falle, dass kein Schadenfall eintritt, würde die freie Reserve anstatt zu fallen weiter steigen. Zum letzten Zeitpunkt $t = 6$ erleidet der Versicherer einen total Verlust und die freie Reserve fällt und wird negativ. Dieser Speziallfall führt zum nächsten wichtigen Begriff der Ruintheorie, nämlich der Ruinwahrscheinlichkeit. Zuvor wird jedoch ein kurzer Einblick in den Code der Abbildung gegeben:

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import random

# Zufallssamen für Reproduzierbarkeit festlegen
random.seed(44)
np.random.seed(44)

# Parameters für das Cramér-Lundberg Modell
u = 300  # Reserve zum Zeitpunkt t=0
c = 50   # Premium Rate (konstant)
t_max = 4.5  # Zeit Maximum

# Generieren nicht-äquidistante claim times
base_times = np.array([0.7, 1.5, 2.3, 3.2, 4.0])
jitter = np.random.uniform(-0.2, 0.2, size=len(base_times))
claim_times = base_times + jitter
claim_times = sorted(claim_times)
print("Schadenzeitpunkte:", [round(t, 2) for t in claim_times])

# Definieren claim Höhe - Ein großer claim um Ruin zu garantieren
claim_amounts = []
for i in range(len(claim_times)):
    if i == 3:  # Ein großer claim um Ruin zu garantieren
        claim_amounts.append(random.uniform(350, 400))
    else:
        claim_amounts.append(random.uniform(40, 80))
print("Schadenhöhen:", [round(a, 2) for a in claim_amounts])

# Berechne stetigen Prämienzuwachs
def premium_growth(t, start_time, start_value):
```

```
33      return start_value + c * (t - start_time)
34
35  # Punkte im Plot für Prämie und claims
36  times = []
37  values = []
38  current_time = 0
39  current_value = u
40
41  # Initial Punkt
42  times.append(current_time)
43  values.append(current_value)
44
45  # Ereignisse in chronologischer Reihenfolge verarbeiten
46  events = []
47  for time, amount in zip(claim_times, claim_amounts):
48      events.append((time, 'claim', amount))
49
50  # Sortiere nach Zeit
51  events.sort(key=lambda x: x[0])
52
53  # Alle Events verarbeiten
54  ruin_point = None
55  segments = []  # Store segments for coloring
56  ruin_occurred = False
57
58  for time, event_type, amount in events:
59      # Überspringe Event nach Ruin
60      if ruin_occurred:
61          continue
62
63      # Punkt vor Ruin hinzufügen
64      if time > current_time:  # Skip if events happen at the same
            time
65          new_value = premium_growth(time, current_time, current_
                value)
66
67          # Das Prämienwachstumssegment (grün) hinzufügen
68          segments.append({
69              'start': (current_time, current_value),
```

```
70              'end': (time, new_value),
71              'type': 'premium'
72          })
73
74          times.append(time)
75          values.append(new_value)
76
77
78          current_value = new_value
79
80      # Event verarbeiten
81      if event_type == 'claim':
82          # If it's a claim, reduce the value
83          prev_value = current_value
84          current_value -= amount
85
86          # Claim Fall
87          segments.append({
88              'start': (time, prev_value),
89              'end': (time, current_value),
90              'type': 'claim'
91          })
92
93          times.append(time)
94          values.append(current_value)
95
96          # Überprüfen ob Ruin stattfindet
97          if current_value < 0 and ruin_point is None:
98              ruin_point = (time, current_value)
99              ruin_occurred = True
100
101     current_time = time
102
103 # Das endgültige Segment hinzufügen, wenn Ruin nicht stattgefunden
      hat
104 if current_time < t_max and not ruin_occurred:
105     final_value = premium_growth(t_max, current_time, current_value
          )
106
```

```
107     # Das endgültige Prämienwachstumssegment hinzufügen
108     segments.append({
109         'start': (current_time, current_value),
110         'end': (t_max, final_value),
111         'type': 'premium'
112     })
113 
114     times.append(t_max)
115     values.append(final_value)
116 
117 # Plot erstellen
118 plt.figure(figsize=(12, 8))
119 
120 # Segmente mit passenden Farben zeichnen
121 for segment in segments:
122     if segment['type'] == 'premium':
123         plt.plot([segment['start'][0], segment['end'][0]],
124                  [segment['start'][1], segment['end'][1]],
125                  'g-', linewidth=2)
126     else:  # claim
127         plt.plot([segment['start'][0], segment['end'][0]],
128                  [segment['start'][1], segment['end'][1]],
129                  'b-', linewidth=2)
130 
131 # Schadensbetrag Anmerkung
132 for time, event_type, amount in events:
133     if event_type == 'claim':
134 
135         for seg in segments:
136             if seg['type'] == 'claim' and seg['start'][0] == time:
137                 claim_value = seg['end'][1]
138 
139                 plt.annotate(f"{amount:.0f}", (time, claim_value),
140                              xytext=(0, -15), textcoords='offset
                                 points',
141                              ha='center', va='top', color='red',
                                 fontsize=10)
142                 break
143 
```

```
# Exakter Ruin Zeitpunkt und ploten
exact_ruin_time = None
if ruin_point:
    for i in range(1, len(times)):
        if values[i-1] >= 0 and values[i] < 0:
            # Linear interpolation to find exact ruin time
            t_prev, t_curr = times[i-1], times[i]
            v_prev, v_curr = values[i-1], values[i]

            # Calculate exact ruin time
            exact_ruin_time = t_prev + (0 - v_prev) * (t_curr - t_prev) / (v_curr - v_prev)
            ruin_value = ruin_point[1]

            # Plot the ruin point at reserve = 0 (only this marker remains)
            plt.plot(exact_ruin_time, 0, 'rx', markersize=10)

            # Add annotation box for ruin point
            plt.annotate(f"Ruin bei t = {exact_ruin_time:.2f}\nReserve = {ruin_value:.2f}",
                         xy=(exact_ruin_time, 0), xytext=(0, -60),
                         textcoords='offset points', ha='center',
                         bbox=dict(boxstyle='round,pad=0.5', facecolor='yellow', alpha=1),
                         arrowprops=dict(arrowstyle='->', lw=2))


            plt.gca().add_patch(Rectangle((exact_ruin_time, -20), t_max-exact_ruin_time, 20, color='red', alpha=0.1))
            break

# Horizontale Linie bei y=0
plt.axhline(y=0, color='r', linestyle='-', alpha=0.3)

# Label und Titel
plt.xlabel('t', fontsize=12)
plt.ylabel('reserve', fontsize=12)
plt.title('Klassisches Risikomodell mit exponentiell verteilten
```

```
    Schäden', fontsize=14)
plt.grid(True, alpha=0.3)

# Legende erstellen
premium_line = plt.Line2D([], [], color='green', marker=None,
    linestyle='-', linewidth=2, label='Prämienzuwachs')
damage_line = plt.Line2D([], [], color='blue', marker=None,
    linestyle='-', linewidth=2, label='Schadensabzug')
ruin_marker = plt.Line2D([], [], color='red', marker='x', linestyle
    ='None', markersize=10, label='Ruinpunkt')

plt.legend(handles=[premium_line, damage_line, ruin_marker], loc='
    upper right')

# Limit für Achsen
plt.ylim(-100, 400)
plt.xlim(-0.1, t_max)

plt.tight_layout()
plt.savefig('risk_model_plot_without_markers.png', dpi=300)
plt.show()

# Print Ruin Information
if exact_ruin_time:
    print(f"\nRuin tritt ein bei t = {exact_ruin_time:.2f}")
    print(f"Reserve zum Ruinzeitpunkt = {ruin_point[1]:.2f}")
else:
    print("\nKein Ruin innerhalb des Simulationszeitraums.")
```

Listing 2.1: Python-Code klassisches Risikomodell

2.4 Ruinwahrscheinlichkeit

Der Spezialfall, dass die freie Reserve negativ wird, wird technischer Ruin genannt. Dieser kann folgendermaßen definiert werden:

Definition 2.1. 4 (Ruin)**.**

$$\text{Ruin} = \{\exists\, t \geq 0 : U(t) < 0\}$$

Der Ruin kann auch als überabzählbare Vereinigung geschrieben werden, nämlich als

$$\text{Ruin} = \bigcup_{t\geq 0} \{U(t) < 0\} \tag{2.2}$$

oder als überabzählbares Infimum

$$\text{Ruin} = \{\inf_{t\geq 0} U(t) < 0\} \tag{2.3}$$

Die Überabzählbarkeit stellt aus wahrscheinlichkeitstheoretischer Sicht kein Problem dar, da bei der überabzählbaren Vereinigung erneut eine messbare Menge entsteht. Das Infimum ergibt ebenfalls eine messbare Funktion. Damit lässt sich beantworten, dass der Ruinprozess bis zu den Ankunftszeiten T_n verläuft und genau an diesen Zeitpunkten der Prozess abstürzt. Das bedeutet, dass der Ruin ausschließlich zu den Ankunftszeiten eintritt. Dies kann wie folgt beschrieben werden:

$$\text{Ruin} = \bigcup_{n>1} \{U(T_n) < 0\}$$

Also wird aus Überabzählbaren eine abzählbare Vereinigung. Analog wird das auch für das Infimum geschrieben.

$$\inf_{t\geq 0} U(t) = \inf_{n\geq 0} U(T_n).$$

Das Event das praktischen Ruins tritt in der Praxis sehr selten auf. Sollte nämlich ein Versicherer merken dass seine freie Reserve stark abnimmt, wird das Unternehmen unverzüglich die Prämien erhöhen. Ebenfalls ist anzumerken, dass in der Praxis der Versicherer verschiedene Versicherungs-Portfolios besitzt, das heißt ein Ruinfall in einem Portfolio bedeutet nicht, dass das Unternehmen unverzüglich bankrott ist und die Verpflichtungen an die Versicherungsnehmer nicht mehr gedeckt sind. Nachdem der Ruin definiert wurde kann die Ruinzeit definiert werden:

Definition 2.1. 5 (Ruinzeit)**.** Sei $t \in (0, \infty]$. Die Ruinzeit T ist wie folgt definiert werden:

$$T = \inf\{t \mid U(t) < 0\}, \quad 0 < T \leq \infty.$$

Falls $T = \infty$ tritt kein Ruin auf.

Alternativ wird die Ruinzeit auch als Menge und folgendermaßen definiert:

Definition 2.1. 6 (Ruinzeit 2)**.**

$$T = \inf\{t \geq 0 : U(t) < 0\}$$

und der Konvention $\inf \emptyset = \infty$ gilt

$$\text{Ruin} = \{T < \infty\}$$

Nun wird ein Kernelement der Ruintheorie definiert, nämlich die Ruinwahrscheinlichkeit

Definition 2.1. 7 (Ruinwahrscheinlichkeit)**.**

$$\psi(u) = P(Ruin \mid U(0) = u) = P[T < \infty], \quad u > 0.$$

Der Zeitpunkt T beschreibt den ersten Moment, zu dem die Surplus-Reserve negativ wird. Aufgrund der Konstruktion des Risikoprozesses $U(t)$ kann ein Ruin nur zu den Zeitpunkten $t = T_n, n \geq 1$, auftreten. Dies bedeutet, dass ein Ruin nur dann möglich ist, wenn ein Schaden eintritt. Zwischen den Zeitpunkten T_n und T_{n+1} nimmt $U(t)$ linear zu.
Die Folge $(U(T_n))_{n\geq 1}$ wird als skeleton process des Risikoprozesses $U(t)$ bezeichnet.

Es ist wichtig anzumerken, dass der Begriff „Ruin“ hier nur als technischer Begriff verstanden werden sollte. Ein Versicherungsunternehmen geht nicht sofort bankrott, solange $T < \infty$. Das Anfangskapital sollte immer als Risikokapital interpretiert werden, das der Versicherer bereit ist, für das Portfolio zu riskieren.
Die Gegenwahrscheinlichkeit, also die Überlebenswahrscheinlichkeit, lautet dann:

Definition 2.1. 8 (Überlebenswahrscheinlichkeit)**.**

$$\phi(u) = \mathbb{P}[T = \infty]$$

Dabei ist es aus theoretischer Sicht günstig die Ruinwahrscheinlichkeit und Überlebenswahrscheinlichkeit als Funktion vom Anfangskapital zu schreiben. Natürlich hängt die Ruinzeit T vom Anfangskapital u ab. Bei Großem u findet der Ruin zu einem späteren Zeitpunkt statt. In der Definition der Ruinzeit wurde bewusst die Abhängigkeit der Ruinzeit vom Anfangskapital unterdrückt. Alternativ gibt es die Notation $\psi(u)$ $= \mathbb{P}[\mathbb{T} < \infty \mid \mathbb{U}(0) = \mathrm{u}]$. Allerdings ist hier nicht offensichtlich, dass es sich um eine bedingte Wahrscheinlichkeit handelt und nicht nur um reine Notation.
[Hubalek(2022)]

2.5 Verschiedene Funktionsgleichungen für die Ruinwahrscheinlichkeit

Um die Fuktionsgleichungen herleiten zu können, werden für einen Moment wichtige Annahmen der freien Reserve $U(t) = u + ct - R(t)$ betrachtet. Hierfür wird der totale Claim Prozess $R(t) = \sum_{i=1}^{N(t)} \xi_i$ angeschaut. Es wird gefordert, dass die Verteilungsfunktion von ξ_1 F_ξ bei $F_\xi(0) = 0$ ist. Für alle i soll auch der Schaden $\xi_i > 0$ sein. Drittens wird gefordert dass ξ stetig verteilt ist und die Dichte f_ξ besitzt. f_k bezeichnet das k-te Moment von ξ_i mit $f_1 = \mathbb{E}[\xi_1] < \infty$. Die momentenerzeugende Funktion von ξ_1 wird mit M_ξ bezeichnet. Existieren diese für ein $w > 0$, dann gibt es ein α, mit $0 < \alpha \leq \infty$, sodass für alle $w < \alpha$ und $\lim_{w \to \alpha^-} M_\xi(w) = \infty$ und $M_\xi(w)$ gilt.
Für die Ruinwahrscheinlichkeit $\psi(u)$ werden nun verschiedene Gleichungen hergeleitet. Jedoch wird verlangt, dass F_ξ eine exponential oder zumindest eine Mischung der

exponential Verteilung besitzt.
Sei $f = \mathbb{E}[\xi]$ wie oben erwähnt und $c > \lambda f$, betrachtet man Menge der x des ersten Moments und die Zeitkomponente t, so wird die Wahrscheinlichkeit durch folgende Gleichung dargestellt:

$$\psi(u) = \int_0^\infty \lambda e^{-\lambda t} \int_0^\infty \psi(u + ct - x) dF_\xi(x) dt,$$

mit der Annahme, dass:

$$\psi(u) = 1, \text{ für } u < 0 \quad \text{und} \quad \lim_{u \to \infty} \psi(u) = 0.$$

Durch die Substitution $s = u + ct$, $ds = cdt$, erhält man folgende Gleichung für $\psi(u)$:

$$\psi(u) = \frac{\lambda}{c} \int_u^\infty e^{-(\lambda/c)(s-u)} \int_0^\infty \psi(s - x) dF_\xi(x) ds.$$

Durch Ableiten der oberen Gleichung folgt:

$$\begin{aligned} \psi'(u) &= \frac{\lambda}{c}\psi(u) - \frac{\lambda}{c} \int_0^\infty \psi(u - x)\, dF_\xi(x) \\ &= \frac{\lambda}{c}\psi(u) - \frac{\lambda}{c} \int_0^u \psi(u - x)\, dF_\xi(x) - \frac{\lambda}{c}[1 - F_\xi(u)] \quad (*) \end{aligned}$$

Für $u > 0$. Der Fall, dass $u = 0$ ist, besitzt die ψ eine Unstetigkeit. Durch Integration über $u \in [0, t]$ folgt eine neue Darstellung. Durch den Satz von Fubini kann die Reihenfolge die Integration ändern. Durch die Integration jedes Teiles folgt diese Form:

$$\int_0^t \int_0^u \psi(u - x) dF_\xi(x) du = \int_0^t \int_x^t \psi(u - x) du dF_\xi(x) =$$

$$= \psi(0) \int_0^t F_\xi(x) dx + \int_0^t \int_x^t \psi'(u - x) du dF_\xi(x) dx =$$

$$= \int_0^t \psi(t - x) F_\xi(x) dx \tag{2.4}$$

Zusätzlich gilt auch:

$$\psi(0) = \frac{\lambda f_1}{c} = \frac{\lambda}{c} \int_0^\infty [1 - F_\xi(u)]\, du \quad (***).$$

Nun werden die erhaltenen Gleichungen zusammengefasst. Zuerst wird die Integration von $(*)$ in Grenzen von 0 bis t betrachtet. Durch die Benutzung von (2.4)und $(***)$, erhält man folgendes Resultat:

$$\psi(t) = \frac{\lambda}{c} \int_0^t \psi(t-x)[1 - F_\xi(x)]\, dx + \frac{\lambda}{c} \int_0^\infty [1 - F_\xi(u)]\, du, \tag{2.5}$$

die für $t \geq 0$ gilt. Diese Gleichung nennt man die defekte Erneuerungs Gleichung. Diese wird verwendet um in der Ruintheorie verschiedene Aspekte des Ruinrisikos zu modellieren. Vor allem bei der Berechnung von Wahrscheinlichkeiten und anderen funktionalen Größen, die mit dem Zeitpunkt des Ruins, der freien Reserve und dem Gesamtdefizit zum Ruinzeitpunkt verbunden sind.
[Hubalek(2022)]

3 Ruinwahrscheinlichkeit im Small-Claim-Case

Dieser Abschnitt behandelt die Ruinwahrscheinlichkeit für Schadenmodelle mit „small claims“, also Verteilungen, bei denen große Schäden nur sehr selten auftreten. In diesem Fall fällt der Tail exponentiell ab und die Ruinwahrscheinlichkeit lässt sich durch Exponentialfunktionen beschreiben. Anschließend wird das Setting des klassischen Risikomodells herangezogen:

$$U(t) = u + ct - S(t), \quad S(t) = \sum_{k=1}^{N(t)} X_k, \quad t \geq 0, \tag{3.1}$$

wobei:

- u der Anfangsüberschuss ist,
- c die konstante Prämienrate ist,
- $S(t)$ der aggregierte Schadensprozess ist,
- $N(t)$ die Schadenzahl bis zum Zeitpunkt t (Poisson-Prozess) ist,
- X_k die Höhe des k-ten Schadens ist.

Der Gesamtschadenprozess $(S(t), t \geq 0)$ ist ein zusammengesetzter Poissonprozess mit Poisson-Parameter λ und Schadenverteilung F:

$$N(t) \sim \text{Poi}(\lambda t), \quad S(t) \sim \text{ZPoi}(\lambda t, F), \quad t > 0.$$

Weiterhin gilt die Nettoprofitbedingung:

3.1 Die Nettoprofitbedingung

Für die Nettoprofitbedingung wählen wir das folgende Vorgehen: Wir betrachten die Prämie im Intervall $[0, t]$ mit $t > 0$. Sie soll nach dem Erwartungswertprinzip berechnet werden: Sei ein relativer Sicherheitszuschlag gegeben $\theta > 0$. Mit θ möchten wir die Prämie zum Zeitpunkt t ermitteln. Durch einfache Überlegung erhalten wir:

$$ct = (1 + \theta)\mathbb{E}[S(t)].$$

Liegt c vor, soll im nächsten Schritt θ bestimmt werden. Mit $\mu = \mathbb{E}[X]$ und $N(t) \sim \text{Poi}(\lambda t)$, folgt:

$$ct = (1 + \theta)\mathbb{E}[S(t)] = (1 + \theta)\mathbb{E}[N(t)]\mathbb{E}[X] = (1 + \theta)\lambda t\mu.$$

Es ergibt sich weiter:

$$c = (1 + \theta)\lambda\mu$$

bzw.

$$\theta = \frac{c}{\lambda\mu} - 1.$$

Die Nettoprofitbedingung besagt nun

$$c > \lambda\mu.$$

Das heißt die Prämie pro Zeiteinheit soll größer sein, als der zu erwartende Schaden (Mittlere Schaden). Äquivalent dazu soll der ausgerechnete Sicherheitszuschlag

$$\theta > 0$$

nicht negativ sein. Diese Annahme wird festgehalten, denn sollte die Nettoprofitbe-

dingung nicht erfüllt sein, beziehungsweise ist der Sicherheitszuschlag θ nicht strikt positiv, lässt sich zeigen, dass die Ruinwahrscheinlichkeit 1, unabhängig von der Höhe des Startkapitals ist. [Hubalek(2022)]

3.2 Das Exponentialprinzip und der Lundberg-Koeffizient

Das Exponentialprinzip hat eine wichtige Bedeutung in der Ruintheorie. Die Vorgehensweise ist ähnlich wie zuvor und führt zum Lundberg-Koeffizienten.
Betrachtet wird eine Prämie in einem Intervall $[0, t]$ mit $t > 0$. Anstelle des Erwartungswertprinzips wird nun das Exponentialprinzip betrachtet.
Gegeben sei ein Risikoaversio-Parameter $r > 0$. Nach dem Exponentialprinzip lässt sich die Prämie ct wie folgt ausrechnen:

$$ct = \frac{1}{r} \log M_{S(t)}(r).$$

Mit $M_{S(t)}(r)$ bezeichnen wir die Momenterzeugende Funktion des Gesamtschadens an der Stelle r. Betrachtet wird nun die Umkehraufgabe, gegeben sei c; zu ermitteln ist r. Hier ist folgendes für die Momenterzeugende Funktion des Gesamtschadens anzumerken:

$$M_{S(t)}(r) = M_{N(t)}(\log M_X(r)) = \exp\big(\lambda t(M_X(r) - 1)\big).$$

Durch Auflösen nach der Prämie ergibt sich nun folgender Ausdruck:

$$c = \frac{1}{r}\lambda(M_X(r) - 1)$$

Durch Umformen folgt die nicht lineare-Gleichung (Lundberg-Gleichung):

$$M_X(r) - 1 - \frac{c}{\lambda}r = 0. \tag{3.2}$$

Die Auflösung nach r ist nur in wenigen Spezialfällen explizit möglich. Sollte diese Gleichung aber eine Lösung für $r > 0$ besitzen und ist dieser eindeutig, so wird er

Lundberg-Koeffizient oder Anpassungskoeffizient genannt. Praktisch gesehen wird die Funktion:

$$f(r) = M_X(r) - 1 - \frac{c}{\lambda}r$$

aufgestellt und die Nullstellen gesucht. Es ist einfach zu erkennen, dass f(0) = 0 eine Lösung ist (triviale Lösung), da die Momenterzeugende Funktion an der Stelle 0 immer 1 ist. Diese ist aber uninteressant für uns. $f(r)$ kann zweimal differenziert werden. Die zweite Ableitung einer Momenterzeugenden Funktion, einer nicht-negativen Zufallsvariable ist positiv und daher ist die Funktion konvex. Drittens gilt auch: $f'(0) = \mu - \frac{c}{\lambda}$. Ist weiteres die Nettoprofitbedingung erfüllt folgt $f'(0) < 0$. Aus diesem Grund gibt es höchstens eine nichttriviale Lösung $r > 0$. Wichtig ist, dass die Momenterzeugende Funktion existieren muss, in einigen Fällen ist es möglich sie explizit auszurechnen. Die Formel die man dabei erhält funktioniert auch in einem Bereich, in dem die Momenterzeugende Funktion eigentlich nicht existiert, hier sind die berechneten Nullstellen keine Lösungen für den Anpassungskoeffizienten.
[Hubalek(2022)]

3.3 Cramer Lundberg Ungleichung und Approximation

Im folgenden, wird ständig das Setting des Cramér-Lundberg Modell (siehe hierfür Definition 2.1.3) vorausgesetzt. In diesem Abschnitt wird gezeigt, dass die Ruinwahrscheinlichkeit eine Form von asymptotischen Verhalten an den Tag legt beziehungsweise eine Abschätzung nach oben gefunden werden kann. Wie bereits in der Einleitung von Kapitel 3 erwähnt, wird der Fall mit Nettoprofitbedingung behandelt. Dies garantiert, dass Schäden mit heavy tailed Verteilungen (zum Beispiel die Log-Normalverteilung oder die Paretoverteilung) ausgeschlossen werden können [Asmussen and Albrecher(2010)]. Der Fall der Ruinwahrscheinlichkeit im heavy tailed case wird gesondert im nächsten Kapitel herangezogen. Die Herleitung der Cramér-Lundberg-Ungleichung beginnt mit einem Theorem aus [Rolski et al.(2009)Rolski, Schmidli, Schmidt, and Teugels]:

Theorem 3.3.1 (Cramér-Lundberg Schranken)**.** *Sei $\psi(u)$ die Ruinwahrscheinlichkeit im Cramér-Lundberg Modell. Unter der Annahme, dass die Anpassungsgleichung*

$$M_X(r) = 1 + \frac{c}{\lambda} r$$

eine positive Lösung $r = \gamma$ hat, gilt:

$$C_- e^{-\gamma u} \leq \psi(u) \leq C_+ e^{-\gamma u},$$

wobei $C_+, C_- > 0$ Konstanten sind und $\gamma > 0$ der Anpassungskoeffizient (Lundberg-Koeffizient).

Dieses Theorem impliziert in Folge:

$$-\gamma u + \log C_- \leq \log \psi(u) \leq -\gamma u + \log C_+.$$

und weiter

$$\gamma = -\lim_{u \to \infty} \frac{\log \psi(u)}{u}. \tag{3.3}$$

Im folgenden Abschnitt wird gezeigt, dass ein wesentlich stärkeres Ergebnis vorliegt, nämlich:

$$\lim_{u \to \infty} \psi(u) e^{\gamma u} = a \tag{3.4}$$

für ein $a > 0$. Es ist auch leicht einzusehen, dass aus $\lim_{u\to\infty} \psi(u)e^{\gamma u} = a \Rightarrow \gamma = -\lim_{u\to\infty} \frac{\log \psi(u)}{u}$ folgt. Eine Äquivalenz muss jedoch nicht immer zwingend gelten.
[Rolski et al.(2009)Rolski, Schmidli, Schmidt, and Teugels].

Das Ziel besteht darin, eine Approximation der Ruinwahrscheinlichkeit zu finden. Hierfür wird mit folgender zuvor hergeleiteter Integralgleichung (2.5) begonnen:
Aus Gründen der Notation wird es bezeichnet mit: $\overline{F}_U(x) := [1 - F_U(x)]$. Also die komplementäre Verteilungsfunktion oder auch Überlebensfunktion genannt. Zusätzlich werden die Grenzen der Integrale etwas äquivalent umgeändert, um einige Re-

chenschritte zu sparen.

$$\psi(u) = \frac{\lambda}{c}\int_u^\infty \overline{F}_U(x)dx + \int_0^u \psi(u-x)\overline{F}_U(x)dx. \tag{3.5}$$

Diese Gleichung wird auch als unvollständige Erneuerungsgleichung oder defective renewal equation bezeichnet. Da die Nettoprofitbedingung vorausgesetzt wird, gilt weiter:

$$\frac{\lambda}{c}\int_0^\infty \overline{F}_U(x)dx = \frac{\lambda\mu}{c} < 1.$$

Durch Multiplizieren von (3.5) mit $e^{\gamma u}$ folgt:

$$e^{\gamma u}\psi(u) = e^{\gamma u}\frac{\lambda}{c}\int_u^\infty \overline{F}_U(x)dx + \frac{\lambda}{c}\int_0^u \psi(u-x)e^{\gamma(u-x)}e^{\gamma x}\overline{F}_U(x)dx. \tag{3.6}$$

Setze nun:

$$z(u) := e^{\gamma u}\psi(u),$$

$$a(x) := \frac{\lambda}{c}e^{\gamma x}\overline{F}_U(x),$$

$$s(u) := \frac{\lambda}{c}e^{\gamma u}\int_u^\infty \overline{F}_U(x)dx,$$

Für (3.6) folgt mit den neu definierten Funktionen:

$$z(u) = s(u) + \int_0^u z(u-x)a(x)dx. \tag{3.7}$$

Wird ein Anpassungskoeffizient $\gamma > 0$ gefunden, sodass

$$\int_0^\infty a(x)dx = 1,$$

gilt, es folgt, dass (3.7) eine Erneurungsgleichung ist. Folgender Zusammenhang gilt:

$$\int_0^\infty a(x)dx = \int_0^\infty \frac{\lambda}{c}e^{\gamma x}\overline{F}_U(x)dx = \int_0^\infty \int_x^\infty \frac{\lambda}{c}dF_U(y)e^{\gamma x}dx$$

Mit Fubini folgt:

$$
\begin{aligned}
&= \int_0^\infty \int_0^y \frac{\lambda}{c} e^{\gamma x} dx \, dF_U(y) = \int_{y=0}^\infty \frac{\lambda}{c} \frac{e^{\gamma x}}{\gamma} \Big|_{x=0}^{x=y} dF_U(y) \\
&= \int_0^\infty \frac{\lambda}{c} \cdot \frac{e^{\gamma y} - 1}{\gamma} dF_U(y) \\
&= \frac{\lambda}{c\gamma} \left[\int_0^\infty e^{\gamma y} dF_U(y) - \int_0^\infty dF_U(y) \right] \\
&= \frac{\lambda}{c\gamma} \left[\hat{m}_U(\gamma) - 1 \right]. \qquad (3.8)
\end{aligned}
$$

Mit $\hat{m}_U(\gamma)$ wird die Momentenerzeugende Funktion von der Zufallsvariable U beschrieben:

$$\hat{m}_U(\gamma) := \int_0^\infty e^{\gamma y} dF_U(y).$$

Aufgrund der Tatsache, dass es sich bei a um eine Dichte handeln soll, kann eine der beiden Bedingungen gefordert werden.:

$$\frac{\lambda}{c\gamma} \left[\hat{m}_U(\gamma) - 1 \right] = 1, \qquad (3.9)$$

beziehungsweise:

$$\hat{m}_U(\gamma) \stackrel{!}{=} 1 + \frac{c\gamma}{\lambda}, \qquad (3.10)$$

Mit 3.10 folgt die Lundberg Gleichung, die bereits weiter oben eingeführt wurde, um den Anpassungskoeffizienten zu bestimmen.
Durch Ableiten von $\int_{x=0}^\infty \frac{\lambda}{c} e^{\gamma x} \overline{F}_U(x) dx$ nach γ, folgt:

$$\frac{\partial}{\partial\gamma}\int_{x=0}^{\infty}\frac{\lambda}{c}e^{\gamma x}\overline{F}_U(x)dx = \int_{x=0}^{\infty}\frac{\lambda}{c}\overline{F}_U(x)\left(\frac{\partial e^{\gamma x}}{\partial\gamma}\right)dx$$
$$= \int_{x=0}^{\infty}\frac{\lambda}{c}\overline{F}_U(x)xe^{\gamma x}dx$$

Weiter folgt:

$$= \frac{\partial}{\partial\gamma}\left[\frac{\lambda}{c\gamma}(\hat{m}_U(\gamma)-1)\right]$$
$$= -\frac{\lambda}{c\gamma^2}(\hat{m}_U(\gamma)-1)+\frac{\lambda}{c\gamma}\hat{m}'_U(\gamma)$$
$$= -\frac{1}{\gamma}+\frac{\lambda}{c\gamma}\hat{m}'_U(\gamma).$$

Geht man die Gleichungskette zurück, ist zu erkennen, dass:

$$\int_{x=0}^{\infty}\frac{\lambda}{c}\overline{F}_U(x)xe^{\gamma x}dx = -\frac{1}{\gamma}+\frac{\lambda}{c\gamma}\hat{m}'_U(\gamma).$$

Da $\frac{\lambda}{c}\overline{F}_U(x)e^{\gamma x}$ als $a(x)$ definiert wurde gilt nun:

$$-\frac{1}{\gamma}+\frac{\lambda}{c\gamma}\hat{m}'_U(\gamma) = \int_{x=0}^{\infty}a(x)xdx =: \mu_F$$

Es wird folgender Satz aus [Rolski et al.(2009)Rolski, Schmidli, Schmidt, and Teugels] herangezogen:

Satz 3.3.1 (Schlüssel-Erneuerungssatz)**.** *Sei $m(t)$ die Erneuerungsfunktion eines Prozesses mit mittlerer Erneuerungszeit μ_F. Sind für eine Funktion $g(t)$, die folgenden Eigenschaften erfüllt:*
- $g(t)$ ist monoton und nicht zunehmend,
- $\int_0^\infty g(t)dt < \infty$,
gilt:

$$\lim_{t\to\infty} \int_0^t g(t-x)m'(x)dx = \frac{1}{\mu_F} \int_0^\infty g(x)dx.$$

Konkret bedeutet dies, dass 3.7 eine eindeutige Lösung besitzt und

$$\lim_{u\to\infty} z(u) = \frac{1}{\mu_F} \int_0^\infty s(u)\, du., \tag{3.11}$$

erfüllt.
Die Herleitung von 3.11 ergibt sich aus der Tatsache, dass 3.7 geschrieben werden kann als $z = s + z * a$, wobei $*$ die Faltungsoperation bezeichnet. Durch wiederholtes Einsetzen und unter geeigneten Regularitätsbedingungen (insbesondere dass die Verteilung F nicht-gitterförmig ist) erhaltet man: $z = s + s * m$, m bezeichnet die Erneuerungsfunktion. Mit $u \to \infty$ gilt:

$$\lim_{u\to\infty} z(u) = \lim_{u\to\infty} s(u) + \lim_{u\to\infty} \int_0^u s(u-x)\, dm(x)$$

Durch Anwendung des Schlüssel-Erneuerungssatz auf $\lim_{u\to\infty} \int_0^u s(u-x)\, dm(x)$ und der Tatsache, dass der erste Term $\lim_{u\to\infty} z(u)$ üblicherweise unter vernünftigen Annahmen über z verschwindet folgt daher:

$$\lim_{u\to\infty} z(u) = \frac{1}{\mu_F} \int_0^\infty s(u)\, du., \tag{3.11}$$

Betrachtet wird der rechte Teil von 3.11:

$$
\begin{aligned}
&= \int_0^\infty s(u)\,du = \int_0^\infty \frac{\lambda}{c} e^{\gamma u} \int_u^\infty \overline{F}_U(x)\,dx\,du \\
&= \int_0^\infty \int_0^x \frac{\lambda}{c} e^{\gamma u} \overline{F}_U(x)\,du\,dx \\
&= \int_0^\infty \frac{\lambda}{c} \overline{F}_U(x) e^{\gamma u} \Big|_{u=0}^{u=x} \frac{1}{\gamma}\,dx \\
&= \int_0^\infty \frac{\lambda}{c} \overline{F}_U(x) \left(e^{\gamma x} - 1\right) \frac{1}{\gamma}\,dx \\
&= \frac{1}{\gamma} \left[\underbrace{\int_{x=0}^\infty \frac{\lambda}{c} \frac{e^{\gamma x}}{\gamma} \overline{F}_U(x) dx}_{=1} - \underbrace{\frac{\lambda}{c} \int_0^\infty \overline{F}_U(x) dx}_{=\mu} \right] \\
&= \frac{1}{\gamma} \left(1 - \frac{\lambda}{c}\mu\right) = \frac{c - \lambda\mu}{c\gamma}
\end{aligned}
$$

Somit gilt für die Ruinwahrscheinlichkeit:

$$\lim_{u\to\infty} e^{\gamma u}\psi(u) = \frac{\frac{c-\lambda\mu}{c\gamma}}{\frac{\lambda}{c\gamma}\hat{m}'_U(\gamma) - \frac{1}{\gamma}} = \frac{c - \lambda\mu}{\lambda\hat{m}'_U(\gamma) - c}.$$

Weiter bringt dies eine Approximation für $\psi(u)$:

$$\psi(u) \approx \frac{c - \lambda\mu}{\lambda\hat{m}'_U(\gamma) - c} e^{-\gamma u}., \tag{3.12}$$

Folgende zentrale Aussage kann getroffen werden:

Satz 3.3.2 (Cramér-Lundberg Ungleichung)**.** *Existiert der Anpassungskoeffizient $R > 0$, so gilt:*

$$\psi(u) < e^{-Ru} \tag{3.13}$$

für $u \geq 0$.

Diese Ungleichug gibt einen Einblick in die zentrale Rolle des Anpassungskoeffizienten. Sie zeigt den Zusammenhang zwischen der Ruinwahrscheinlichkeit und dem Koeffizienten. Zusätzlich wird die Abhängigkeit von der Gesamtschadenhöhe und der

Höhe der Prämienrate unterstrichen.
[Rolski et al.(2009)Rolski, Schmidli, Schmidt, and Teugels]

4 Ruinwahrscheinlichkeit im Heavy-Tailed-Case

In diesem Abschnitt werden Heavy-tailed Verteilungen betrachtet – eine wichtige Gruppe mit zahlreichen nützlichen Eigenschaften. Zusätzlich wird das Verhalten der Ruinwahrscheinlichkeit dieser Verteilungen näher untersucht. Um jegliche Verwirrung zu vermeiden, wird der Begriff des Tails in der folgenden Definition präzisiert:

Definition 2.1. 9 (Tail-Funktion)**.** *Für die Verteilungsfunktion F, ist ihr Tail $\overline{F}$ gegeben als:*

$$\overline{F}(x) = \mathbb{P}(X > x) = 1 - F(x).$$

Bei Heavy-tailed Verteilungen wird zwischen drei Klassen unterschieden: der klassischen Heavy-tailed Verteilung, der Long-tailed Verteilung und der subexponentiellen Verteilung. Diese Klassen von Verteilungen sind eng miteinander verbunden und es gilt folgender Zusammenhang:

$$\text{Subexponential Verteilung} \Longrightarrow \text{Long-tailed Verteilung} \Longrightarrow \text{Heavy-tailed Verteilung}$$

Das Konzept der Heavy-tailed Verteilung ist ein relatives: Eine Verteilung kann schwerere Tails aufweisen als eine andere, ohne selbst eine echte Heavy-tailed Verteilung zu sein. Um eine klare Unterscheidung zu treffen, wird eine Heavy-tailed Verteilung häufig im Vergleich zur Exponentialverteilung definiert, die als Referenzpunkt zwischen den Klassen der Heavy-tailed- und Light-tailed Verteilungen dient. Eine Heavy-tailed Verteilung wird als eine Verteilung beschrieben, deren Tail schwerer ist als der einer Exponentialverteilung. In den folgenden Abschnitten liegt der Fokus auf Verteilungen

mit heavy-tailed Eigenschaften – insbesondere auf subexponentiellen Schadenverteilungen (Subexponential Claims).
[Embrechts et al.(2013)Embrechts, Klüppelberg, and Mikosch]

4.1 Heavy-Tailed Verteilungen

Die Heavy-tailed Verteilungen bilden eine große Klasse von Verteilungen und umfassen verschiedene Typen von Heavy-tailed Verteilungen. Es gibt unterschiedliche Möglichkeiten, eine Heavy-tailed Verteilung zu definieren, was zu möglichen Verwirrungen führen kann. Die folgende Definition einer Heavy-tailed Verteilung fokussiert sich auf den rechten Tail der Verteilung.

Definition 2.1. 10 (Heavy-tailed Verteilung)**.** *Eine Verteilung F ist genau dann heavy-tailed, wenn:*

$$\limsup_{x\to\infty} \frac{\overline{F}(x)}{e^{-tx}} = \infty$$

für alle $t > 0$.

Eine Verteilung wird als light-tailed bezeichnet, wenn sie nicht heavy-tailed ist. Darüber hinaus gilt: Eine Zufallsvariable X ist heavy-tailed, wenn ihre Verteilungsfunktion F heavy-tailed ist. Die Definition verdeutlicht außerdem, dass zur Charakterisierung einer Heavy-tailed Verteilung lediglich der Tail betrachtet werden muss.
Das folgende Lemma zeigt, dass es tatsächlich viele Möglichkeiten gibt, eine Heavy-tailed Verteilung zu definieren. Eine typische Definition basiert auf der Momentenerzeugenden Funktion. Zudem beleuchtet das Lemma die Unterschiede zwischen Heavy-tailed und Light-tailed Verteilungen. Es erfordert jedoch die Einführung einer geeigneten Risikofunktion, die wie folgt definiert ist:“

Definition 2.1. 11 (Risikofunktion)**.** *Die Risikofunktion $R(x)$ für eine Verteilungsfunktion F ist definiert als:*

$$R(x) = -\ln \overline{F}(x).$$

Lemma 4.1.1. *Die folgenden Aussagen sind äquivalent für jede Zufallsvariable X mit Verteilungsfunktion F:*

(i) X *ist Heavy-tailed.*

(ii) $M_X(t) = \mathbb{E}[e^{tX}] = \infty \quad$ *für alle* $t > 0$.

(iii) Es gilt: $\liminf_{x\to\infty} \frac{R(x)}{x} = 0$.

Beweis: (i) $\implies$ (ii): Angenommen, X ist eine Heavy-tailed Zufallsvariable mit Verteilungsfunktion F. Basierend auf der Definition der Tail-Funktion impliziert dies, dass es eine streng monoton wachsende Folge $(x_k)_{k\geq 1}$ gibt, sodass:

$$\lim_{k\to\infty} x_k = \infty$$

und

$$\lim_{k\to\infty} e^{tx_k}\overline{F}(x_k) = \infty,$$

wenn $t > 0$. Mit der Laplace-Stieltjes-Transformation kann gezeigt werden, dass für $M_X(t)$ folgendes gilt:

$$\mathbb{E}[e^{tX}] = \int_0^\infty e^{tx} dF(x) \geq \int_{x_k}^\infty e^{tx} dF(x) \geq e^{tx_k}\overline{F}(x_k),$$

für alle k. Daher folgt:

$$M_X(t) = \mathbb{E}[e^{tX}] = \infty.$$

(ii) $\implies$ (iii): Angenommen, Bedingung (ii) gilt und Bedingung (iii) gilt nicht für X. Dies bedeutet:

$$\liminf_{x\to\infty} \frac{R(x)}{x} > 0.$$

Dies würde weiter implizieren, dass es ein $\mu > 0$ und ein $x_0 > 0$ gibt, sodass:

$$\frac{R(x)}{x} = \frac{-\ln \overline{F}(x)}{x} \geq \mu \iff \overline{F}(x) \leq e^{-\mu x}$$

für ein $x \geq x_0$. Sei t, sodass $0 < t < \mu$ gilt, es folgt:

$$\mathbb{E}[e^{tX}] = \int_0^\infty \mathbb{P}(e^{tX} > x)dx = \int_0^{e^{tx_0}} \mathbb{P}(e^{tX} > x)dx + \int_{e^{tx_0}}^\infty \mathbb{P}\left(X > -\frac{\ln x}{t}\right)dx.$$

Weiterhin, wenn beachtet wird, dass $x \geq e^{tx_0} \iff \frac{\ln x}{t} \geq x_0$, folgt:

$$\mathbb{E}[e^{tX}] \leq e^{tx_0} + \int_{e^{tx_0}}^\infty e^{-\mu\frac{\ln x}{t}}dx = e^{tx_0} + \int_{e^{tx_0}}^\infty x^{\frac{-\mu}{t}}dx.$$

Sei t so, dass $0 < t < \mu$, was bedeutet, dass $\frac{\mu}{t} > 1$. Dann gilt:

$$\int_{e^{tx_0}}^\infty x^{\frac{-\mu}{t}}dx < \infty.$$

Dies impliziert weiter, dass $M_X(t) = \mathbb{E}[e^{tX}] < \infty$, was im Widerspruch zur Annahme steht, dass (ii) für alle $t > 0$ gilt. Daher folgt aus Bedingung (ii), dass Bedingung (iii) erfüllt ist.

$(iii) \implies (i)$: Um den letzten Teil zu beweisen, nehmen wir an, dass Bedingung (iii) für die Zufallsvariable X gilt. Dann existiert eine streng monoton wachsende Folge $(x_k)_{k\geq 1}$, sodass:

$$\lim_{k\to\infty} x_k = \infty \quad \text{und} \quad \liminf_{k\to\infty} \frac{R(x_k)}{x_k} = 0.$$

Für $t > 0$ impliziert dies, dass es ein $t_0 > 0$ gibt, sodass:

$$\frac{R(x_k)}{x_k} < e^{-t/2} \iff F(x_k) > e^{-tx_k/2} \iff \frac{\overline{F}(x_k)}{e^{tx_k}} > e^{tx_k/2}.$$

Für alle $t > t_0$ folgt daraus:

$$\lim_{k\to\infty} \frac{\overline{F}(x_k)}{e^{tx_k}} = \infty,$$

und weiterhin:

$$\limsup_{x\to\infty} \frac{\overline{F}(x)}{e^{tx}} = \infty.$$

Basierend auf der Definition der Tail-Funktion folgt dass Bedingung (iii) Bedingung (i) impliziert. □

[Foss et al.(2011)Foss, Korshunov, and Zachary]

Das folgende Lemma zeigt eine interessante Eigenschaft stetiger, nicht-negativer Zufallsvariablen mit Heavy-tailed Verteilungsfunktionen: Eine Heavy-tailed Verteilungsfunktion impliziert eine Heavy-tailed Dichtefunktion. Die Umkehrung gilt jedoch nicht, da eine Zufallsvariable mit Heavy-tailed Dichtefunktion dennoch eine Light-tailed Verteilungsfunktion besitzen kann. Eine Heavy-tailed Funktion ist eine Funktion, die nicht durch eine exponentiell abfallende Funktion nach oben beschränkt ist und analog zur Heavy-tailed Verteilung definiert wird – wie in der Definition der Tail-Funktion dargestellt. Für die genaue Definition einer Heavy-tailed Funktion siehe [Foss et al.(2011)Foss, Korshunov, and Zachary].

Lemma 4.1.1. *Angenommen, eine absolut stetige Verteilung F hat eine Dichtefunktion f und F ist heavy-tailed. Dann ist auch die Dichtefunktion $f(x)$ heavy-tailed.*

Beweis: Der Beweis wird in Foss et al. [Foss et al.(2011)Foss, Korshunov, and Zachary] präsentiert und würde den Rahmen dieser Arbeit sprengen.

4.2 Long-tailed Verteilungen

Long-tailed Verteilungen sind ebenfalls Heavy-tailed Verteilungen, zeichnen sich jedoch durch eine spezifische Glattheit ihrer Tail-Funktion aus, die sie von anderen Heavy-tailed Verteilungen unterscheidet. Wie oben bereits angeführt, umfasst diese Klasse von Verteilungen auch die Subexponentiellen. Die Long-tailed Verteilungen sind wie folgt definiert:

Definition 2.1. 12 (Long-tailed Verteilung)**.** *Sei F die Verteilungsfunktion einer positiven Zufallsvariablen X. Dann ist F Long-tailed, wenn $F > 0$ gilt und für jedes*

feste positive y gilt:

$$\lim_{x\to\infty} \frac{\overline{F}(x+y)}{\overline{F}(x)} = 1$$

für alle $x \in \mathbb{R}$.

Das Phänomen der Long-tailed Verteilungen lässt sich durch das Konzept der Restlebensdauer erklären. Im Kontext von Wartezeiten beschreiben Long-tailed Verteilungen Situationen, in denen nach einer bereits sehr langen Wartezeit die erwartete verbleibende Wartezeit nicht notwendigerweise sinkt, sondern möglicherweise sogar ansteigt. Die Intuition dahinter wird am folgenden Beispiel klar: Wartet man auf einen Zug, der nach einer ungewöhnlich langen Wartezeit immer noch nicht eingetroffen ist, deutet dies wahrscheinlich auf ein größeres Problem hin (z. B. einen technischen Defekt). In einem solchen Fall sinkt die Erwartung, dass der Zug bald kommt; man stellt sich eher auf eine noch längere Verzögerung ein. Solche Szenarien lassen sich gut durch Long-tailed Verteilungen modellieren.
[Nair et al.(2022)Nair, Wierman, and Zwart]

4.3 Subexponetielle Verteilungen

Die folgende Verteilung spielt nicht nur im Rahmen dieser Arbeit eine zentrale Rolle, sondern ist auch eine der wichtigsten unter den Heavy-tailed Verteilungen. Dabei handelt es sich um die Subexponentielle Verteilung. Sie wurde das erste Mal in den 1960er Jahren eingeführt und erwies sich anfangs im Finanzkontext als sehr nützlich. Die Subexponentielle Verteilung wird häufig verwendet, um vor allem Random Walks zu untersuchen. Sie spielt aber auch in der Versicherungsmathematik eine zentrale Rolle, da es möglich ist, mit dieser Verteilung große Werte zu modellieren. Dies ist vor allem im Bereich der Claims essentiell.
Betrachten wir die formale Definition einer Subexponentiellen Verteilung. Sei $\overline{F}$ die Tail Funktion, wie in Definition der Tail-Funktion angegeben. Angenommen, $X_1, X_2, \ldots, X_n$ sind unabhängige und identisch verteilte Zufallsvariablen für alle n. Bezeichne den Tail der n-fachen Faltung von F mit:

$$\overline{F^{*n}}(x) = 1 - F^{*n}(x) = P(X_1 + \cdots + X_n > x).$$

Dann erhalten wir die folgende Definition für eine Subexponentielle Verteilung:

Definition 2.1. 13 (Subexponentielle Verteilung)**.** *Eine Verteilung F ist subexponentiell, wenn für alle $n \geq 2$ unabhängigen Zufallsvariablen $X_1, \ldots, X_n$ mit Verteilung F gilt:*

$$P(X_1 + \cdots + X_n > x) \sim nP(X_1 > x), \quad \text{wenn } x \to \infty,$$

was auch dargestellt werden kann als:

$$\overline{F^{*n}}(x) \sim n\overline{F}(x).$$

Die Definition der Subexponentiellen Verteilung besagt, dass sie für alle $n \geq 2$ gelten muss, was in der Praxis keine sehr praktische Definition ist, da man alle möglichen Werte von n überprüfen müsste, um Subexponentialität zu zeigen. Die Definition kann in einer einfacheren Form dargestellt werden: Es kann auch bewiesen werden, dass der Grenzwert aus der Subexponentiellen Verteilung für alle $n \geq 2$ gilt, genau dann wenn er für $n = 2$ gilt. Dann ist dieser Grenzwert gleich 2, wie das folgende Lemma zeigt:

Lemma 4.3.1. *Eine Verteilung F auf $[0, \infty)$ ist subexponentiell, wenn*

$$\limsup_{x \to \infty} \frac{\overline{F^{*2}}(x)}{\overline{F}(x)} = 2.$$

Beweis: Der Beweis lässt sich in [Chistyakov(1964)] nachlesen und würde den Rahmen dieser Arbeit sprengen.
Das folgende Lemma stellt formell die hinreichende Bedingung für die Subexponentialität dar.

Lemma 4.3.2 (Hinreichende Bedingung für Subexponentialität)**.** *Für jede Verteilung*

F auf $(0,\infty)$ gilt: F ist subexponentiell, wenn

$$\limsup_{x\to\infty} \frac{\overline{F^{*2}}(x)}{\overline{F}(x)} \leq 2. \tag{4.1}$$

Beweis: Sei X_1 und X_2 unabhängige nicht-negative Zufallsvariablen mit Verteilungsfunktion F. Für den Zähler gilt:

$$\overline{F^{*2}}(x) = \mathbb{P}(X_1 + X_2 > x) \geq \mathbb{P}(\max(X_1, X_2) > x),$$

für alle $x \geq 0$. Dies kann geschrieben werden als:

$$\mathbb{P}(\max(X_1, X_2) > x) = \mathbb{P}(X_1 > x) + \mathbb{P}(X_2 > x) - \mathbb{P}(X_1 > x, X_2 > x),$$

dies ergibt:

$$= 2\overline{F}(x) - (\overline{F}(x))^2.$$

Daraus folgt:

$$\liminf_{x\to\infty} \frac{\overline{F^{*2}}(x)}{\overline{F}(x)} \geq \liminf_{x\to\infty} \frac{2\overline{F}(x) - (\overline{F}(x))^2}{\overline{F}(x)} = 2.$$

Aus (4.1) schließen wir, dass Lemma 4.3.1 gilt. □

Ein wichtiges Phänomen im Zusammenhang mit subexponentiellen Verteilungen ist das sogenannte Prinzip des großen Sprungs (principle of a single big jump), das das probabilistische Verhalten von Summen unabhängiger subexponentieller Zufallsvariablen beschreibt. Das Prinzip charakterisiert die Natur von Heavy Tails, indem es das typische Verhalten von Heavy-tailed Zufallsvariablen beschreibt. Intuitiv besagt das Prinzip des großen Sprungs, dass die Summe unabhängiger und identisch verteilter subexponentieller Zufallsvariablen genau dann groß ist, wenn der maximale Wert dieser Zufallsvariablen groß ist. In der Praxis kann ein einzelner großer Wert einen erheblichen Einfluss auf das Ergebnis der Summe haben. Dies könnte beispielsweise der Effekt eines einzelnen großen Schadens auf die Gesamtschadenssumme im Kontext der Versicherung sein. Dieses Phänomen erklärt die typische Anwesenheit sehr großer Werte in Beobachtungen subexponentiell verteilter Zufallsvariablen.

Lemma 4.3.3 (Prinzip eines großen Sprungs)**.** *Seien $X_1, \ldots, X_n$ unabhängige Zu-*

fallsvariablen, die alle eine gemeinsame Subexponentielle Verteilung F besitzen. Dann gilt:

$$\lim_{x\to\infty} \frac{\mathbb{P}(X_1 + \cdots + X_n > x)}{\mathbb{P}(\max(X_1, \ldots, X_n) > x)} = 1 \quad \textit{für alle } n \geq 2.$$

Beweis: Das Lemma kann unter Verwendung der Definition der Subexponentiellen Verteilung bewiesen werden. Seien $X_1, \ldots, X_n$ nicht-negative, unabhängige und subexponentielle Zufallsvariablen mit Verteilungsfunktion F. Basierend auf der Definition gilt, für alle $n \geq 2$:

$$\mathbb{P}(X_1 + \cdots + X_n > x) \sim n\overline{F}(x), \quad \text{wenn } x \to \infty.$$

Es genügt zu zeigen, dass dasselbe für den Nenner gilt, d.h. es muss gelten:

$$\mathbb{P}(\max(X_1, \ldots, X_n) > x) \sim n\overline{F}(x),$$

für alle $n \geq 2$, für $x \to \infty$. Für den Nenner gilt:

$$\mathbb{P}(\max(X_1, \ldots, X_n) > x) = 1 - F^n(x) = \overline{F}(x) \sum_{k=0}^{n-1} F^k(x).$$

Da $F \in [0, 1]$ für alle $x \in \mathbb{R}$, und die Summe für ein festes n endlich ist, gilt:

$$\lim_{x\to\infty} \sum_{k=0}^{n-1} F^k(x) = \sum_{k=0}^{n-1} \lim_{x\to\infty} F^k(x) = \sum_{k=0}^{n-1} 1 = n.$$

Da $F^k(x)$ gegen 1 konvergiert für alle $k > 0$, folgt:

$$\lim_{x\to\infty} F^k(x) = 1.$$

Somit ergibt sich:

$$\lim_{x\to\infty}\sum_{k=0}^{n-1}F^k(x)=n.$$

Wenn also $x\to\infty$, folgt daraus:

$$\overline{F}(x)\sum_{k=0}^{n-1}F^k(x)=n\overline{F}(x).$$

Damit ist folgendes gezeigt:

$$\mathbb{P}(\max(X_1,\ldots,X_n)>x)=n\overline{F}(x),$$

□

Das Prinzip des großen Sprungs charakterisiert die Natur Heavy-tailed Verteilungen und beschreibt das typische Verhalten von Summen subexponentieller Zufallsvariablen.
[Asmussen and Albrecher(2010)] und [Embrechts et al.(2013)Embrechts, Klüppelberg, and Mikosch]

4.4 Ruinwahrscheinlichkeit Subexpontielle Verteilungen

Im folgenden Abschnitt wird die Ruinwahrscheinlichkeit für eine Subexpontielle Verteilung betrachtet. Als Modell wird das Zusammengesetzte Poisson-Modell herangezogen mit einer Ankunftsrate β und einer Schadensverteilung B. Sei $S_t=\sum_{i=1}^{N_t}U_i-t$ der Überschuss zum Zeitpunkt t und $M=\sup_{t\geq 0}S_t$, $\tau(u)=\inf\{t>0:S_t>u\}$. Es sei $\rho=\beta\mu_B<1$. Mit B_0 wird die stationäre Exzessverteilung oder Überschreitungsverteilung bezeichnet: $B_0(x)=\int_0^x\overline{B}(y)\,dy/\mu_B$. Ziel ist es eine Approximation für die Ruinwahrscheinlichkeit $\psi(u)=\mathbb{P}(M>u)=\mathbb{P}(\tau(u)<\infty)$ zu finden.

Lemma 4.4.1. *Wenn B_0 eine Subexponentielle Verteilung ist, dann gilt:*

$$\psi(u)\sim\frac{\rho}{1-\rho}\overline{B}_0(u),$$

wobei $\rho = \beta\mu_B$.

Der Beweis basiert auf dem folgenden Lemma:

Lemma 4.4.2. *Seien* $Y_1, Y_2, \ldots,$ *unabhängig und identisch verteilte Zufallsvariablen mit einer gemeinsamen Subexponentiellen Verteilung* G. *Sei* K *eine unabhängige ganzzahlige Zufallsvariable mit* $E[z^K] < \infty$ *für ein* $z > 1$. *Dann gilt:*

$$P(Y_1 + \cdots + Y_K > u) \sim \mathbb{E}[K]\,\overline{G}(u).$$

für $u \to \infty$.

Beweis: Aus Definition der Subexponentiellen Verteilung gilt, dass

$$\overline{G^{*n}}(u) \sim n\overline{G}(u),$$

wenn $u \to \infty$. Es wird auch angenommen dass es für jedes$z > 1$ ein $D < \infty$ gibt, so dass

$$\overline{G^{*n}}(u) \leq \overline{G}(u)Dz^n \quad \text{für alle } uist.$$

Dies kann in [Asmussen and Albrecher(2010)] nachgelesen werden. Es folgt:

$$\frac{\mathbb{P}(Y_1 + \cdots + Y_K > u)}{\overline{G}(u)} = \sum_{n=0}^{\infty} \mathbb{P}(K = n)\frac{\overline{G^{*n}}(u)}{\overline{G}(u)} \longrightarrow \sum_{n=0}^{\infty} \mathbb{P}(K = n) \cdot n = \mathbb{E}K.$$

Unter Verwendung der dominierten Konvergenz mit $\sum \mathbb{P}(K = n)Dz^n$ als Majorante, ist das Lemma bewiesen. □

Beweis von Lemma 4.4.1: Die Pollaczeck-Khinchine-Formel besagt, dass (im Rahmen von Lemma 4.4.2) M mit der Wahrscheinlichkeitsverteilung der Summe $Y_1 + \cdots + Y_K$, wobei die Y_i die Verteilung B_0 haben und K geometrisch verteilt ist mit Parameter ρ, $\mathbb{P}(K = k) = (1-\rho)\rho^k$. Da $\mathbb{E}K = \rho/(1-\rho)$ und $\mathbb{E}z^K < \infty$, wenn $\rho z < 1$, folgt die Aussage unmittelbar aus Lemma 4.4.2. □

Das Lemma 4.4.1 stellt eine wichtige Approximation für die Ruinwahrscheinlichkeit dar und zeigt das asymptotische Verhalten von Schadenshöhenverteilungen im Rahmen des zusammengesetzten Poisson-Modells.

5 Simulation von Ruin Wahrscheinlichkeiten für Subexponebtielle Claims

Dieses Kapitel befasst sich mit der Simulation der Ruinwahrscheinlichkeit $\psi(u)$ in einem klassischen zusammengesetzten Poisson-Risikoprozess $U(t)$ mit anfänglicher Reserve $u = U(0)$, wobei die Schadenshöhenverteilung B heavy-tailed ist. Das Hauptziel besteht darin Methoden, zur Verbesserung der einfachen Monte-Carlo-Simulation zu untersuchen. Die im folgenden untersuchten Simulationsmethoden untersuchen basieren auf der Darstellung der Ruinwahrscheinlichkeit als $\psi(u) = z = \mathbb{E}Z$ für eine Zufallsvariable Z, die durch Simulation erzeugt werden kann. Z wird durch unabhängige identisch verteilte Replikate $Z_1, \ldots, Z_n$ erzeugt. Die Ruinwahrscheinlichkeit $\psi(u)$ wird durch $\hat{z} = (Z_1 + \ldots + Z_n)/n$ geschätzt, zusätzlich wird die empirische Varianz der Z_i verwendet, um Konfidenzintervalle zu erstellen. Die Leistungskennzahl einer bestimmten Simulation ist der relative Fehler $\sigma_Z/\psi(u)$, wobei $\sigma_Z^2 = \mathrm{Var}(Z)$ ist. Diese Kennzahl ist aber nur sinnvoll, wenn für die verglichenen Simulationsmethoden basierend auf $Z(1)$, $Z(2)$, die benötigten Rechenzeiten zur Erzeugung von $Z(1)$, $Z(2)$ ungefähr gleich sind, aus einfachheitsgründen wird dies angenommen. Dennoch ergäben sich zwei Schwierigkeiten:

1. Das Ruinproblem hat einen unendlichen Zeithorizont, sodass es nicht einfach ist, die gewünschte Darstellung $\psi(u) = z = \mathbb{E}[Z]$ für eine simulierbare Zufallsvariable Z zu finden. Im Folgenden wird dies als Problem 1bezeichnet.
2. Da u groß ist, ist die Ruinwahrscheinlichkeit $\psi(u)$ klein, und wir befinden uns daher im Bereich der Simulation seltener Ereignisse. Wenn man Problem 1 für einen Moment vernachlässigt, kann angenommen werden, dass $Z = I(\tau(u) < \infty)$ generiert werden kann, wobei $I(\cdot)$ die Indikatorfunktion ist und $\tau(u)$ die Zeit des

Ruins mit Anfangskapital u bezeichnet. Dieses Verfahren ist in der Literatur als die sogenannte „Crude Monte Carlo"-Methode bekannt und führt zu einem relativen Fehler.

$$\frac{\sigma_Z}{\psi(u)} = \frac{\sqrt{\psi(u)(1-\psi(u))}}{\psi(u)} \approx \frac{1}{\sqrt{\psi(u)}} \to \infty, \quad u \to \infty. \tag{5.1}$$

[Heidelberger(1995)] und [Asmussen and Rubinstein(1995)]

5.1 B ist Light-tailed

Aus Vollständigkeitsgründen wird im folgenden Abschnitt der Fall herangezogen, dass die Schadenverteilung B Light-tailed ist. In diesem Fall wurde eine Lösung für beide Probleme von [Siegmund(1976)] & [Asmussen(1985)] präsentiert. Dabei führt man einen Maß-Wechsel durch, indem man das gegebene Wahrscheinlichkeitsmaß P durch ein anderes $\tilde{P}$ ersetzt, das $\tilde{P}(\tau(u) < \infty) = 1$ erfüllt, und setzt $Z = \frac{dP}{d\tilde{P}}$, wobei die Radon-Nikodym-Dichte für $\mathcal{F}_{\tau(u)}$ berechnet wird. Genauer gesagt entspricht $\tilde{P}$ einer exponentiellen Maßänderung, die den Lundberg-Exponenten (Anpassungskoeffizienten) R verwendet, sodass die Poisson-Intensität und die Verteilung der Schadenshöhen auf eine bestimmte durch R gegebene Weise verändert werden. Dass Problem 1 gelöst ist, folgt aus $\tilde{P}(\tau(u) < \infty) = 1$. Empirische Evidenz legt zudem nahe, dass auch Problem 2 gelöst ist. Hierfür betrachten wir ein Standard Kriterium:

$$\liminf_{u\to\infty} \frac{\log \sigma_Z}{\log \psi(u)} \geq 1. \tag{5.2}$$

Insbesondere genügt es, dass

$$\sigma_Z^2 \leq \psi(u)^2 \, p(|\log \psi(u)|) \tag{5.3}$$

für ein Polynom p gilt. Dies ist für ein konstantes p erfüllt und kann in [Siegmund(1976)] und [Asmussen(1985)] nachgelesen werden. Zu beachten ist, dass die Crude Monte Carlo-Methode gemäß (5.2) niemals effizient sein kann, da sie stets ge-

gen den Grenzwert 1/2 anstelle von 1 konvergiert.
[Asmussen and Binswanger(1997)]

5.2 Methoden zur Simulation bei subexponentiellen Schadensverteilungen

Der folgende Abschnitt beschäftigt sich mit der Simulation von $\psi(u)$ in dem Fall, dass B keine exponentiellen Momente besitzt, sodass R nicht existiert und die Methode von [Siegmund(1976)] und [Asmussen(1985)] nicht anwendbar ist. Unter solchen Verteilungen wird der Fokus auf die Klasse der subexponentiellen Verteilungen S liegen. Aus Vereinfachungsgründen wird eine Alternative Definition für die Klasse der subexponentiellen Verteilungen genommen. Diese wurde im Kapitel 4 als Lemma 4.3.3 präsentiert:

Definition 2.1. 14 (Alternative Definition subexponentielle Verteilung)**.** *Eine nichtnegative Zufallsvariable X mit Verteilungsfunktion F heißt subexponentiell ($F \in S$), wenn für alle $n \geq 2$ gilt:*

$$\lim_{x\to\infty} \frac{P(X_1 + \ldots + X_n > x)}{P(\max(X_1, \ldots, X_n) > x)} = 1, \tag{5.4}$$

wobei $X_1, \ldots, X_n$ unabhängige und identisch verteilte Zufallsvariablen mit gleicher Verteilung wie X sind.

Für das beschriebene Problem 1 am Anfang des Kapitels wird folgender Zugang für die Ruinwahrscheinlichkeit gewählt:

$$\psi(u) = 1 - (1-\rho)\sum_{n=0}^{\infty} \rho^n B_0^{*n}(u), \quad u > 0, \tag{5.5}$$

wobei $\rho = \frac{1}{1+\theta}$, $B_0(u) = \int_0^u b_0(s)ds$ und $b_0(s) = \frac{1}{\mu_B}\bar{B}(s)$ mit $\bar{B}(s) = 1 - B(s)$; B_0^{*n} bezeichnet die n-te Faltung von B_0 mit sich selbst. Beachte, dass (5.5) bedeutet, dass $1 - \psi(u)$ eine zusammengesetzte geometrische Verteilungsfunktion ist,

$$\psi(u) = P(S_K > u), \tag{5.6}$$

wobei $S_K = X_1 + ... + ...X_K$. K ist geometrisch mit dem Parameter ρ, unabhängig von den X_i, und die $X_1, X_2, ...$ sind nicht-negative unabhängige identisch verteilte Zufallsvariablen mit gemeinsamer Dichte b_0. Dies bedeutet, dass die einfache Monte-Carlo-Methode(Crude Monte Carlo simulation kurz: CMC) anwendbar ist: $\psi(u) = z = E[Z]$ wobei $Z = I(S_K > u)$. Der Algorithmus lautet wie folgt:

Algorithmus 1:

1. Generiere $K_i \sim$ geometrisch(ρ), d.h. $P(K_i = k) = (1-\rho)\rho^k$ $(k = 0, 1, 2, ...)$.
2. Generiere $X_1^i, ..., X_{K_i}^i$ aus der Dichte b_0 und setze $S_{K_i} = X_1^i + ... + X_{K_i}^i$.
3. Falls $S_{K_i} > u$, dann $Z_i = 1$, sonst $Z_i = 0$.
4. Wiederhole Schritte 1 bis 3 n-mal.
5. Schätze $E[Z]$ durch $\hat{z} = \frac{1}{n}\sum_{i=1}^{n} Z_i$.

Als CMC-Algorithmus kann dieses Verfahren nicht effizient sein. Um Problem 2 zu behandeln, werden zwei bedingte Monte-Carlo-Schätzer eingeführt. Die Idee besteht darin, den CMC-Schätzer Z durch $E(Z \mid \mathcal{G})$ für eine geeignete σ-Algebra $\mathcal{G}$ zu ersetzen, was immer zu einer Varianzreduktion führt, vgl. [Rubinstein(1981)]. Es wird gezeigt, dass einer der Schätzer im Sinne von (5.1) effizient ist, und zwar speziell in dem Fall, dass das Ende von B regulär variiert.

Neben der Simulationsmethodik werden auch die analytische Approximationen, von denen die bekanntesten die Panjer-Rekursion und

$$\psi(u) \sim \frac{1}{\theta} B_0(u), \quad u \to \infty \tag{5.7}$$

diskutiert.Im folgenden wird (5.7) auch mit $\psi_{EV}(u)$ bezeichnet.

[Asmussen and Binswanger(1997)]

Die Genauigkeit von (5.7) wird in der Arbeit von [Abate et al.(1994)Abate, Choudhury, and Whitt] diskutiert. Dort werden exakte Werte mittels Transformationsinversion berechnet (eine Zusammenfassung der Inversionsmethoden und der Anwendbarkeit dieses Ansatzes findet sich in [Abate and Whitt(1992)] sowie den dort zitierten Referenzen). In letztgenannter Arbeit wurde die Klasse PME (Pareto-Mischungen von Exponentialverteilungen, siehe auch Abschnitt Numerische Ergebnisse) mit expliziten Laplace-Transformationen konstruiert. Numerische Vergleiche zwischen exakten Werten und den durch Gleichung (5.7) gelieferten Näherungen führten zu eher negati-

ven Ergebnissen hinsichtlich der Genauigkeit dieser Formel. Im Abschnitt Numerische Ergebnisse werden zudem weitere Resultate präsentiert, wobei die exakten Werte — auch für allgemeinere Schadenshöhenverteilungen als jene in der Klasse PME — durch Simulation ermittelt wurden.

5.2.1 Bedingter-Monte-Carlo Algorithmus

In diesem Abschnitt werden Zufallsvariablen überwiegend durch Großbuchstaben bezeichnet (z. B. $Z, K, S_K, X_1, X_2, \ldots$), während die Realisationen der i-ten Simulation ($i = 1, \ldots, n$) durch indizierte Großbuchstaben dargestellt werden (z. B. $Z_i, K_i, X_1^i, X_2^i, \ldots$). Zur Erinnerung: Die CMC-Methode wird im Folgenden als Algorithmus 1 bezeichnet. Ein bedingter Monte-Carlo-Schätzer führt stets zu einer Varianzreduktion. Die asymptotischen 95%-Konfidenzintervalle ergeben sich zu:

$$\hat{\psi}(u) \pm 1{,}96 \frac{\hat{\sigma}}{\sqrt{n}}, \tag{5.8}$$

wobei $\hat{\psi}(u)$ für die geschätzte Ruinwahrscheinlichkeit steht und

$$\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (Z_i - \bar{Z})^2 \tag{5.9}$$

die empirische Varianz der Schätzwerte bezeichnet.
Sei

$$\begin{aligned} \psi(u) &= P(X_1 + \ldots + X_K > u) \\ &= E\left[P(X_1 + \ldots + X_K > u \mid X_1, \ldots, X_{K-1})\right] \\ &= E\left[\overline{B}_0\left(u - X_1 - \ldots - X_{K-1}\right)\right], \end{aligned}$$

wobei $\overline{B}_0(y)$ die Wahrscheinlichkeit bezeichnet, dass der nächste Schaden Ruin verursacht. Daher wird nur $X_1, \ldots, X_{K-1}$ generiert, $Y = u - X_1 - \ldots - X_{K-1}$ werden berechnet, zusätzlich wird für $Z = \overline{B}_0(Y)$ gesetzt, also die Wahrscheinlichkeit, dass der nächste Schaden Ruin verursacht. Genauer:

Algorithmus 2:

1. Ziehe $K_i \sim$ geometrisch(ρ), d. h. $P(K_i = k) = (1-\rho)\rho^k$ für $k = 0, 1, 2, \ldots$
2. Ziehe $X_1^i, \ldots, X_{K_i-1}^i$ aus der Dichte b_0 und setze $Y_i = u - X_1^i - \ldots - X_{K_i-1}^i$
3. Setze $Z_i = \overline{B}_0(Y_i)$ (bzw. $Z_i = 1$, falls $Y_i < 0$)
4. Wiederhole die Schritte 1 bis 3 n-mal
5. Schätze $E[Z]$ durch $\hat{z} = \frac{1}{n}\sum_{i=1}^n Z_i$

Auch $\hat{z}$ ist ein erwartungstreuer Schätzer für $\psi(u)$. Auch wenn die Varianz kleiner sein muss als bei Algorithmus I, ist die Leistung im Sinne von (5.2) asymptotisch nicht besser:

Lemma 5.2.1. *Sei B eine Subexponentielle Verteilung. Dann gilt für Algorithmus II*

$$\lim_{u\to\infty} \frac{\log \sigma_Z}{\log \psi(u)} = \frac{1}{2} \tag{5.10}$$

Beweis: Es gilt:

$$\begin{aligned}
E[Z^2] &= E\left[\overline{B}_0^2(u - X_1 - \ldots - X_{K-1})\right] \\
&\geq E\left[\overline{B}_0^2(u); K \leq 1\right] + E\left[\overline{B}_0^2(u - X_1); K \geq 2\right] \\
&= (1-\rho^2)\overline{B}_0^2(u) + E\left[\overline{B}_0^2(u - X_1); K \geq 2\right] \\
&\geq (1-\rho^2)\overline{B}_0^2(u) + E\left[\overline{B}_0^2(u - X_1); X_1 > u, K \geq 2\right] \\
&= (1-\rho^2)\overline{B}_0^2(u) + \rho^2\overline{B}_0(u).
\end{aligned}$$

Die letzte Gleichheit folgt aus der Tatsache, dass das Ereignis $(X_1 > u)$ mit der Wahrscheinlichkeit $\overline{B}_0(u)$ eintritt und dann $\overline{B}_0^2(u - X_1) = 1$ gilt. Da

$$E[Z]^2 = \psi(u)^2 \sim \frac{1}{\theta^2}\overline{B}_0^2(u), \tag{5.11}$$

folgt daraus, dass σ_Z von der Größenordnung mindestens $\overline{B}_0^{1/2} \sim (\theta\psi(u))^{1/2}$ ist. Daher kann $\log \sigma_Z$ nicht schneller gegen $-\infty$ gehen als $\log \psi(u)/2$, sodass $1/2$ eine obere Schranke für lim inf in (2) ist. Dass $1/2$ auch eine untere Schranke für lim sup ist, folgt daraus, dass der Algorithmus, der auf bedingter Monte-Carlo-Simulation basiert, eine

Verbesserung des CMC-Algorithmus darstellt. □

Der dritte Algorithmus ist etwas komplexer. Die Hauptidee dieses Algorithmus basiert auf der Tatsache, dass bei subexponentiellen Verteilungen nicht die Summe aller Schäden, sondern lediglich der größte Schaden den Ruin verursacht, wie in Alternative Definition subexponentielle Verteilung erläutert. Die folgenden zwei Lemmata werden diese Idee genauer ausführen.

Lemma 5.2.1. *Seien $X_1, X_2, \ldots, X_n \sim B_0$ nicht-negative, unabhängig und identisch verteilte Zufallsvariablen und bezeichne mit $X_{(1)} < X_{(2)} < \ldots < X_{(n)}$ die zugehörige Ordnungsstatistik. Weiterhin sei $\mathcal{F}_{(n-1)} = \sigma(X_{(1)}, \ldots, X_{(n-1)})$.*
Dann gilt

$$P(X_{(n)} > x|\mathcal{F}_{(n-1)}) = \frac{\overline{B}_0(X_{(n-1)} \vee x)}{\overline{B}_0(X_{(n-1)})}$$

wobei $a \vee b = \max(a, b)$.

Beweis: Angenommen, $X_1, \ldots, X_n$ sind u.i.v. und die X_i sind absolut stetig, dann bilden die Ordnungsstatistiken eine Markov-Kette.

$$P(X_{(n)} > x|\mathcal{F}_{(n-1)}) = P(X_{(n)} > x|X_{(n-1)})$$

und

$$P(X_{(n)} > x|X_{(n-1)} = y) = \begin{cases} 1, & x < y, \\ \int_x^\infty f_{X_{(n)}|X_{(n-1)}}(u|y)du, & x \geq y, \end{cases}$$

wobei

$$\int_x^\infty f_{X_{(n)}|X_{(n-1)}}(u|y)du = \frac{\overline{B}_0(x)}{\overline{B}_0(y)}$$

(siehe hierzu beispielsweise Arnold, Balakrishnan und Nagaraja [3], S. 23). Somit folgt

$$P(X_{(n)} > x|\mathcal{F}_{(n-1)}) = \frac{\overline{B}_0(X_{(n-1)} \vee x)}{\overline{B}_0(X_{(n-1)})}$$

□

Falls die X_i nicht absolut stetig sind, kann ein anderer Beweis mittels kombinatorischer Argumente gegeben werden.
[Asmussen and Binswanger(1997)]

Lemma 5.2.2 (Darstellung der Ruinwahrscheinlichkeit)**.** *Sei* $S_n = X_1 + ... + X_n$ *und* $S_{(k)} = X_{(1)} + ... + X_{(k)}$ *für* $1 \leq k \leq n$*. Dann gilt*

$$P(S_n > u) = E\left[\frac{\overline{B}_0((u - S_{(n-1)}) \vee X_{(n-1)})}{\overline{B}_0(X_{(n-1)})}\right] \tag{5.12}$$

Beweis: Durch Konditionierung folgt;

$$P(S_n > u) = E[P(S_n > u|\mathcal{F}_{(n-1)})] \tag{5.13}$$
$$= E[P(X_{(n)} + S_{(n-1)} > u|\mathcal{F}_{(n-1)})] \tag{5.14}$$
$$= E[P(X_{(n)} > u - S_{(n-1)}|\mathcal{F}_{(n-1)})] \tag{5.15}$$

Die Anwendung von Lemma 5.2.1 vervollständigt den Beweis. □

Algorithmus 3:

Basierend auf den vorangegangenen Lemmata kann Algorithmus III wie folgt formuliert werden:

1. Generiere K_i gemäß einer geometrischen Verteilung mit Parameter ρ, d. h. $P[K_i = k] = (1-p)p^k$ für $k = 0, 1, 2, \ldots$
2. Generiere $X^i_1, \ldots, X^i_{K_i}$ aus der Dichte b_0 und setze $Y^i = u - X^i_{(1)} - \ldots - X^i_{(K_i-1)}$ sowie $m_i = X^i_{(K_i-1)}$
3. Berechne $Z_i = \frac{\overline{B}_0(Y_i \vee m_i)}{\overline{B}_0(m_i)}$
4. Wiederhole die Schritte 1 bis 3 insgesamt n-mal
5. Schätze $E[Z]$ durch $\hat{z} = \frac{1}{n}\sum_{i=1}^n Z_i$

5.2.2 Hauptresultat

Das Hauptergebnis dieser Arbeit ist das folgende Theorem:

Theorem 5.2.1 (Effizienz des Algorithmus III)**.** *Angenommen, es gilt* $\overline{B}_0(x) = L(x)/x^\alpha$ *mit* $\alpha > 1$ *und* L *ist eine langsam variierende Funktion (d. h.* $\lim_{x\to\infty} \frac{L(\lambda x)}{L(x)} = 1$ *für alle* $\lambda > 0$*). Dann erfüllt Algorithmus III die Bedingung*

$$\liminf_{u\to\infty} \frac{\log \sigma_Z}{\log \psi(u)} \geq 1.$$

Zum Beweis von Theorem 5.2.1 werden zunächst drei Hilfslemmata benötigt.

Lemma 5.2.3 (Varianzabschätzung)**.** *Für Algorithmus III gilt die folgende Ungleichung:*

$$\sigma_Z^2 \leq E\left[K^2\left(\frac{1}{2}\overline{B}_0^2\left(\frac{u}{2}\right) + \overline{B}_0^2\left(\frac{u}{K}\right)\left|\log \overline{B}_0\left(\frac{u}{2}\right)\right|\right)\right] \tag{5.16}$$

Beweis: Zunächst wird die bedingte Dichte $f_{X_{(K-1)}}(x)$ der Zufallsvariablen $X_{(K-1)}$ gegeben K hergeleitet:

$$\begin{aligned} P(X_{K-1} \leq x) &= P(X_{(1)} \leq x, \ldots, X_{(K-1)} \leq x) \\ &= \binom{K}{1} P(X_1 \leq x, \ldots, X_{K-1} \leq x, X_K > x) \\ &= +P(X_1 \leq x, \ldots, X_{K-1} \leq x, X_K \leq x) \\ &= K B_0^{K-1}(x)\overline{B}_0(x) + B_0^K(x). \end{aligned}$$

Somit ist die Dichte:

$$f_{X_{(K-1)}}(x) = K(K-1)B_0^{K-2}(x)\overline{B}_0(x)b_0(x). \tag{5.17}$$

Als Nächstes wird folgendes berechnet:

$$E[Z^2|K] = E\left[\left(\frac{\overline{B}_0\left((u - S_{(K-1)}) \vee X_{(K-1)}\right)}{\overline{B}_0\left(X_{(K-1)}\right)}\right)^2 |K\right] \tag{5.18}$$

$$= E\left[\left(\frac{\overline{B}_0\left(u - S_{(K-1)}\right)}{\overline{B}_0\left(X_{(K-1)}\right)}\right)^2 ; X_{(K-1)} \leq \frac{u}{K}|K\right] \tag{5.19}$$

$$+ E\left[\left(\frac{\overline{B}_0\left((u - S_{(K-1)}) \vee X_{(K-1)}\right)}{\overline{B}_0\left(X_{(K-1)}\right)}\right)^2 ; \frac{u}{K} < X_{(K-1)} \leq \frac{u}{2}|K\right] \tag{5.20}$$

$$+ E\left[1; X_{(K-1)} > \frac{u}{2}|K\right] \tag{5.21}$$

Der erste Summand (5.20) kann wie folgt beschränkt werden. Falls $X_{(K-1)} \leq \frac{u}{K}$ ist

dann gilt:

$$\begin{aligned}&\overline{B}_0\left(u - S_{(K-1)}\right)\\ &\leq \overline{B}_0\left(\frac{u}{K}\right),\end{aligned}$$

sodass

$$\begin{aligned}&E\left[\left(\frac{\overline{B}_0\left(u - S_{(K-1)}\right)}{\overline{B}_0\left(X_{(K-1)}\right)}\right)^2 ; X_{(K-1)} \leq \frac{u}{K} | K\right]\\ &\leq \overline{B}_0^2\left(\frac{u}{K}\right)\int_0^{u/K}\frac{f_{X_{(K-1)}}(x)}{\overline{B}_0^2(x)}dx\\ &\leq K(K-1)\overline{B}_0^2\left(\frac{u}{K}\right)\int_0^{u/K}\frac{b_0(x)}{\overline{B}_0(x)}dx\\ &= -K(K-1)\overline{B}_0^2\left(\frac{u}{K}\right)\log\left(\overline{B}_0\left(\frac{u}{K}\right)\right).\end{aligned}$$

Der zweite Summand (5.21) kann in gleicher Weise beschränkt werden. Für $\frac{u}{K} < X_{(K-1)} \leq \frac{u}{2}$ gilt $\overline{B}_0\left((u - S_{(K-1)}) \vee X_{(K-1)}\right) \leq \overline{B}_0\left(\frac{u}{K}\right)$, womit sich folgendes ergibt:

$$E\left[\left(\frac{\overline{B}_0\left((u - S_{(K-1)}) \vee X_{(K-1)}\right)}{\overline{B}_0\left(X_{(K-1)}\right)}\right)^2 ; \frac{u}{K} < X_{(K-1)} \leq \frac{u}{2} | K\right]$$

$$\leq \overline{B}_0^2\left(\frac{u}{K}\right)\int_{u/K}^{u/2}\frac{f_{X_{(K-1)}}(x)}{\overline{B}_0^2(x)}dx$$

$$\leq K(K-1)\overline{B}_0^2\left(\frac{u}{K}\right)\int_{u/K}^{u/2}\frac{b_0(x)}{\overline{B}_0(x)}dx$$

$$= -K(K-1)\overline{B}_0^2\left(\frac{u}{K}\right)\left(\log\left(\overline{B}_0\left(\frac{u}{2}\right)\right) - \log\left(\overline{B}_0\left(\frac{u}{K}\right)\right)\right)$$

Für eine obere Schranke für (5.22) gilt:

$$\begin{aligned} E\left[1; X_{(K-1)} > \frac{u}{2}|K\right] &= \int_{u/2}^{\infty} f_{X_{(K-1)}}(x)dx \\ &= K(K-1)\int_{u/2}^{\infty} B_0^{K-2}(x)\overline{B}_0(x)b_0(x)dx \\ &\leq K(K-1)\int_{u/2}^{\infty} \overline{B}_0(x)b_0(x)dx \\ &= K(K-1)\frac{1}{2}\overline{B}_0^2\left(\frac{u}{2}\right). \end{aligned}$$

Nach Addition der obigen Ungleichungen erhalten wir

$$\begin{aligned} E[Z^2|K] &\leq K(K-1)\left(\frac{1}{2}\overline{B}_0^2\left(\frac{u}{2}\right) - \overline{B}_0^2\left(\frac{u}{K}\right)\log\left(\overline{B}_0\left(\frac{u}{2}\right)\right)\right) \\ &\leq K^2\left(\frac{1}{2}\overline{B}_0^2\left(\frac{u}{2}\right) + \overline{B}_0^2\left(\frac{u}{K}\right)\left|\log\overline{B}_0\left(\frac{u}{2}\right)\right|\right) \end{aligned}$$

Somit folgt für die Varianz:

$$\begin{aligned} \sigma_Z^2 &= E\left[E[Z^2|K]\right] - E[Z]^2 \\ &\leq E\left[K^2\left(\frac{1}{2}\overline{B}_0^2\left(\frac{u}{2}\right) + \overline{B}_0^2\left(\frac{u}{K}\right)\left|\log\overline{B}_0\left(\frac{u}{2}\right)\right|\right)\right]. \end{aligned}$$

□

Lemma 5.2.4. *Sei $\overline{B}_0(x) = \frac{L(x)}{x^\alpha}$, wobei L langsam variierend ist. Dann existieren für jedes $\varepsilon > 0$ Konstanten $C_-(\varepsilon)$ und $C_+(\varepsilon)$, sodass*

$$C_-(\varepsilon)d^{\alpha-\varepsilon}x^{-\alpha-\varepsilon} \leq \overline{B}_0\left(\frac{x}{d}\right) \leq C_+(\varepsilon)d^{\alpha-\varepsilon}x^{-\alpha+\varepsilon},\ \forall x > 0\ \forall d > 0.$$

Beweis: Aus $\overline{B}_0(x)x^{\alpha-\varepsilon} = x^{-\varepsilon}L(x)$ folgt, dass $\lim_{x\to 0} x^{-\varepsilon}L(x) = 0$ und dass L eine stetige Funktion ist. Da L langsam variierend ist, gilt auch $\lim_{x\to\infty} x^{\varepsilon}L(x) = 0$. Daher existiert eine Konstante $C_+(\varepsilon)$, sodass $L(x) \leq C_+(\varepsilon)x^{\varepsilon}$ für alle x gilt, und somit

$$\overline{B}_0\left(\frac{x}{d}\right) = \frac{L(x/d)}{(x/d)^{\alpha}} \leq \frac{C_+(\varepsilon)(x/d)^{\varepsilon}}{(x/d)^{\alpha}} = \frac{C_+(\varepsilon)d^{\alpha-\varepsilon}}{x^{\alpha-\varepsilon}}$$

Für die untere Schranke ist der Beweis ähnlich. Es sei lediglich angemerkt, dass wenn L langsam variierend ist, dann ist auch $1/L$ langsam variierend. □

Lemma 5.2.5. *Sei $\overline{B}_0(x) = \frac{L(x)}{x^{\alpha}}$, wobei L langsam variierend ist. Dann existieren für jedes $\varepsilon > 0$ Konstanten $D_1(\varepsilon)$ und $D_2(\varepsilon)$, sodass*

$$E[Z^2] \leq (D_1(\varepsilon) + D_2(\varepsilon)|\log u|)u^{2\varepsilon-2\alpha}.$$

Beweis: Aus Lemma 5.2.3 folgt:

$$E[Z^2] \leq E\left[K^2\left(\frac{1}{2}\overline{B}_0^2\left(\frac{u}{2}\right) + \overline{B}_0^2\left(\frac{u}{K}\right)|\log\overline{B}_0\left(\frac{u}{2}\right)|\right)\right]$$

Lemma 5.2.4 liefert

$$\begin{aligned}
&\leq E[K^2]\frac{1}{2}C_+^2(\varepsilon)2^{2\alpha-2\varepsilon}u^{-2\alpha+2\varepsilon} \\
&+ E[C_+^2(\varepsilon)K^{2\alpha-2\varepsilon+2}u^{-2\alpha+2\varepsilon}|\log(C_-(\varepsilon)2^{\alpha-\varepsilon}u^{-\alpha+\varepsilon})|] \\
&\leq (D_1(\varepsilon) + D_2(\varepsilon)|\log u|)u^{2\varepsilon-2\alpha}
\end{aligned}$$

wobei $D_1(\varepsilon) = E[K^2]\frac{1}{2}C_+^2(\varepsilon)2^{2\alpha-2\varepsilon} + E[K^{2\alpha-2\varepsilon+2}]C_+^2(\varepsilon)|\log(C_-(\varepsilon)2^{\alpha-\varepsilon})|$ und $D_2(\varepsilon) = E[K^{2\alpha-2\varepsilon+2}]C_+^2(\varepsilon)(\alpha+\varepsilon)$ ist. □

Nun ist es möglich mit diesen Hilfsmittel, Theorem 5.2.1 zu beweisen:

Beweis von Theorem 5.2.1: Aus Lemma 5.2.5 folgt:

$$\begin{aligned}\log \sigma_Z &\leq \log \sqrt{(D_1(\varepsilon) + D_2(\varepsilon)|\log u|)u^{2\varepsilon-2\alpha}} \\ &= \frac{1}{2}\log(D_1(\varepsilon) + D_2(\varepsilon)|\log u|) + (\varepsilon - \alpha)\log u\end{aligned}$$

und daher

$$\lim_{u\to\infty} \frac{\log \sigma_Z}{\log \psi(u)} \geq \lim_{u\to\infty} \frac{\frac{1}{2}\log(D_1(\varepsilon) + D_2(\varepsilon)|\log u|) + (\varepsilon - \alpha)\log u}{\log \psi(u)}$$

unter Verwendung von (5.7) ergibt sich

$$= \lim_{u\to\infty} \frac{\frac{1}{2}\log(D_1(\varepsilon) + D_2(\varepsilon)|\log u|) + (\varepsilon - \alpha)\log u}{\log(\overline{B}_0(u)/\theta)}$$

$$= \lim_{u\to\infty} \frac{\frac{1}{2}\log(D_1(\varepsilon) + D_2(\varepsilon)|\log u|) + (\varepsilon - \alpha)\log u}{-\log\theta + \log \mathrm{L}(u) - \alpha \log u}$$

$$= \frac{\varepsilon - \alpha}{-\alpha} = 1 - \frac{\varepsilon}{\alpha}$$

Mit $\varepsilon \to 0$ ist das Theorem bewiesen. □

[Asmussen and Binswanger(1997)]

5.3 Numerische Ergebnisse

Im folgenden Abschnitt, werden einige Beispiele von Subexponentiellen Verteilungen für die Claim Verteilung herangezogen. In der Programmiersprache R werden die Ruin Wahrscheinlichkeiten simuliert und ihre Performance wird nach (5.2) gemessen. Zusätzlich wurde für den Vergleich eine theoretischer Wert ψ_P ausgerechnet, dieser beruht auf die Panjer Rekursion:

Definition 2.1. 15 (Panjer-Rekursion). *Die **Panjer-Rekursion** ist ein rekursives*

Verfahren zur Berechnung der Ruinwahrscheinlichkeit $\psi(u)$ in diskreten Schritten. Sie basiert auf einer Diskretisierung der Dichte b_0 und liefert eine Näherung für die Überlebenswahrscheinlichkeit $\phi(u) = 1 - \psi(u)$. Die Rekursion lautet:

$$\phi^*(u) = \frac{1}{1+\theta-g(0)} \sum_{y=1}^{u} g(y)\,\phi^*(u-y), \quad u = 1, 2, \ldots$$
$$\phi^*(0) = \frac{\theta}{1+\theta-g(0)}$$

wobei g eine diskretisierte Version der Dichte b_0 ist.

Die Ruinwahrscheinlichkeit ergibt sich dann zu

$$\psi(u) \approx 1 - \sum_{y=0}^{u} \phi^*(y), \quad u = 0, 1, 2, \ldots$$

Ein großer Vorteil dieser Methode ist, dass sie auch zur Bestimmung von Ober- und Untergrenzen für $\psi(u)$ verwendet werden kann, indem g so gewählt wird, dass $g_l(x) \leq b_0(x)$ für die Untergrenze und $g_u(x) \geq b_0(x)$ für die Obergrenze gilt. Da b_0 monoton fallend ist, kann man z. B. $g_l(x) = B_0(\lfloor x \rfloor + 1) - B_0(\lfloor x \rfloor)$ und $g_u(x) = B_0(\lfloor x \rfloor) - B_0(\lfloor x \rfloor - 1)$ wählen, wobei $\lfloor x \rfloor$ die Ganzzahlfunktion bezeichnet.

Für die Approximation von $\psi(u)$, bezeichnet als $\psi_p(u)$, wählt man $g_a(x) = B_0(\lfloor x \rfloor + 1/2) - B_0(\lfloor x \rfloor - 1/2)$.

Die Panjer-Rekursion ist ein Standardwerkzeug der Versicherungsmathematik zur numerischen Berechnung der Ruinwahrscheinlichkeit. Bevor ein Vergleich der verschiedenen Simulationsmethoden stattfindet wird im folgenden, der Code von jedem Algorithmus präsentiert:

```
# Algorithm 1 Implementation for Pareto Distribution
# Based on the paper's CMC method for ruin probability estimation




# Required libraries
library(ggplot2)
library(tidyr)
```

```r
library(dplyr)

# Pareto distribution helper functions
a <- 1  # Pareto parameter
b <- 2  # Pareto parameter

B0_pareto <- function(x){
  if(x < a){
    y <- (b-1)/(a*b)*x
  } else if (x >= a){
    y <- (1 - 1/b*(a/x)^(b-1))
  }
  return(y)
}

# Inverse of the cumulative distribution function for Pareto
B0_inv_pareto <- function(x){
  y <- ifelse(x < (b-1)/b,  a*b/(b-1)*x, a/(b*(1-x))^(1/(b-1)))
  return(y)
}

#### Algorithm 1 Implementation ----

# Parameters
theta <- 0.1  # loading parameter
rho <- 1/(1 + theta)  # geometric parameter
n_sim <- 1000  # number of simulations
u_values <- c(10, 50, 100, 500, 1000)  # initial capitals to test

# Function to generate one Z value
generate_Z <- function(u, rho, a=1, b=2) {
  # 1. Generate K from geometric distribution
  K <- rgeom(1, 1-rho)   # adding 1 since R's geometric starts from
      0
  #K <- qgeom(runif(1),rho) + 1

  # 2. Generate X values from Pareto distribution using inverse
     function
  X <- B0_inv_pareto(runif(K)) #replicate(K, B0_inv_pareto(runif(1)
```

```
    ))

  # 3. Compute SK and compare with u
  SK <- sum(X)

  # Return indicator
  return(as.numeric(SK > u))
}

# Function to estimate psi(u) for a given u
estimate_psi <- function(u, n_sim, rho) {
  # Generate n Z values
  Z <- replicate(n_sim, generate_Z(u, rho))

  # Compute estimate and standard error
  psi_hat <- mean(Z)
  se <- sd(Z) #sqrt(var(Z)/n_sim)

  return(c(psi_hat, se))
}


# panjer rekursion für psi_P
psi_panjer <- function(n){

  # Gewichtungsvektor g
  g <- function(x){
    y <- as.integer(x)
    res <- B0_pareto(y+1) - B0_pareto(y)
    return(res)
  }

  g0 <- g(0)

  # Initialisierung von phi mit Startwert
  phi <- c()
  phi[1] <- theta/(1 + theta - g0)

  # Rekursive Berechnung mit Schleife
```

```
  for (x in 1:n) {
    sum <- 0
    for (k in 1:x) {
      sum <- sum + g(k) * phi[x - k + 1]
    }
    phi[x + 1] <- sum * 1/(1 + theta - g0)
  }

  # Berechnung von psi(u)
  psi_u <- 1 - sum(phi)
  psi_u

}


# Run simulations for each u value
results <- data.frame(u = u_values)
results$estimates <- NA
results$se <- NA
results$efficiency <- NA  # log^()/log^()


for(i in 1:length(u_values)) {
  res <- estimate_psi(u_values[i], n_sim, rho)
  results$estimates[i] <- res[1]
  results$se[i] <- res[2]
  # Calculate log^()/log^() as shown in Table 1
  results$efficiency[i] <- abs(log(res[2])/log(res[1]))  # abs to
      handle negative logs
}

# Print results in similar format to Table 1
print("Simulation Results:")
cat("\ n(u) ± 1.96/√n and log^()/log^():\n")
for(i in 1:nrow(results)) {
  cat(sprintf("u = %d: (%.1f ± %.1f) · 10^-%d    log^()/log^() = %.2
      f     _P = %.3f\n ",
              results$u[i],
```

```
122                 results$estimates[i] * 10^(ceiling(-log10(results$
                        estimates[i]))),
123                 results$se[i]*1.96/sqrt(n_sim) * 10^(ceiling(-log10(
                        results$estimates[i]))),
124                 ceiling(-log10(results$estimates[i])),
125                 results$efficiency[i],
126                 psi_panjer(results$u[i])))
127 }
128
129 # Optional: Create plots of the results
130 # Plot 1: Ruin probability estimates
131 plot1 <- ggplot(results, aes(x = u, y = estimates)) +
132   geom_point() +
133   geom_errorbar(aes(ymin = estimates - se, ymax = estimates + se),
          width = 20) +
134   scale_y_log10() +
135   labs(title = "Ruin Probability Estimates",
136        x = "Initial Capital (u)",
137        y = "Estimated  (u)") +
138   theme_minimal()
139
140 # Plot 2: Efficiency measure
141 plot2 <- ggplot(results, aes(x = u, y = efficiency)) +
142   geom_point() +
143   geom_line() +
144   labs(title = "Efficiency Measure",
145        x = "Initial Capital (u)",
146        y = "log^()/log^()") +
147   theme_minimal()
148
149 # Arrange plots side by side
150 library(gridExtra)
151 grid.arrange(plot1, plot2, ncol=2)
152
153
154
155
156
157 # psi_panjer <- function(n){
```

```
158 #
159 #
160 #
161 # # Gewichtungsvektor g
162 # g <- function(x){
163 #   y <- as.integer(x)
164 #   #res <- B0_pareto(y + 1/2) - B0_pareto(y - 1/2)
165 #   res <- B0_pareto(y+1) - B0_pareto(y)
166 #   return(res)
167 # }
168 #
169 # g0 <- g(0)
170 #
171 # # Anzahl Elemente die berechnet werden sollen
172 # #n <- 1000
173 #
174 # # Initialisierung von phi mit Startwert
175 # phi <- c() #numeric(n + 1)
176 # phi[1] <- theta/(1 + theta - g0)  # phi(0) = 1 (Index 1 in R
      entspricht 0 in Mathematik)
177 #
178 # # Rekursive Berechnung mit Schleife
179 # for (x in 1:n) {
180 #   sum <- 0
181 #   #for (k in 1:min(x, length(g) - 1)) {
182 #   for (k in 1:x) {
183 #     sum <- sum + g(k) * phi[x - k + 1]
184 #   }
185 #   phi[x + 1] <- sum * 1/(1 + theta - g0)   # 1 - theta works for
      u = 50
186 # }
187 #
188 # # Ausgabe: phi[1] = phi(0), phi[2] = phi(1), usw.
189 # #print(phi)
190 #
191 # #phi[1]
192 #
193 # # Berechnung von psi(u)
194 # psi_u <- 1 - sum(phi)
```

```
# psi_u
#
# }
#
# psi_panjer(10)
# psi_panjer(50)
# psi_panjer(100)
# psi_panjer(500)
# psi_panjer(1000)

# # b0
# a <- 1
# b0 <- function(x){
#   if(x<a){
#     y <- (b-1)/(a*b)
#   } else if(x >= a){
#     y <- (a/x)^b
#   }
# }
#
#
# g_u  <- function(x){
#   y <- as.integer(x)
#   #res <- B0_pareto(y + 1/2) - B0_pareto(y - 1/2)
#   res <- B0_pareto(y+1) - B0_pareto(y)
#   return(res)
# }
#
#
# x <- seq(0,10,0.1)
#
# plot(x,sapply(x,function(x)b0(x)), type = "l", col = "red")
# lines(x,sapply(x,function(x)g_u(x)), type = "l", col = "blue")
```

Listing 5.1: Simulation der Ruin Wahrscheinlichkeiten für eine Pareto Verteilung in R (Algorithmus 1)

```
# Algorithm II Implementation for Pareto Distribution
# Based on the conditional Monte Carlo method described in the
  paper
```

```r

# Required libraries
library(ggplot2)
library(tidyr)
library(dplyr)

# Parameters
theta <- 0.1  # loading parameter
rho <- 1/(1 + theta)  # geometric parameter
n_sim <- 1000  # number of simulations
u_values <- c(10, 50, 100, 500, 1000)  # initial capitals to test
a <- 1  # Pareto parameter
b <- 2  # Pareto parameter

# Import Pareto functions from sample_from_distributions (1).R
B0_inv_pareto <- function(x){
  y <- ifelse(x < (b-1)/b,  a*b/(b-1)*x, a/(b*(1-x))^(1/(b-1)))
  return(y)
}

# Function to calculate B0_bar(y) = P(X > y)
B0_bar <- function(y) {
  if(y <= 0) return(1)  # As specified in the algorithm
  # For Pareto(1,2), P(X > y) = (1/y)^2 for y  1
  return(min(1, (a/y)^b))
}

# Function to generate one Z value using Algorithm II
generate_Z_algo2 <- function(u, rho) {
  # 1. Generate K from geometric distribution
  K <- rgeom(1, 1-rho) + 1  # adding 1 since R's geometric starts
      from 0

  # 2. Generate K-1 claims and compute Y
  if(K == 1) {
    Y <- u
  } else {
    # Use B0_inv_pareto for generating Pareto random variables
    X <- replicate(K-1, B0_inv_pareto(runif(1)))
```

```
    Y <- u - sum(X)
  }

  # 3. Calculate Z = B0_bar(Y)
  Z <- B0_bar(Y)

  return(Z)
}

# Function to estimate psi(u) with confidence intervals
estimate_psi_algo2 <- function(u, n_sim, rho) {
  # Generate n Z values
  Z <- replicate(n_sim, generate_Z_algo2(u, rho))

  # Calculate estimate and standard error
  psi_hat <- mean(Z)
  sigma_hat <- sd(Z)
  se <- sigma_hat/sqrt(n_sim)

  # Calculate 95% confidence interval
  ci_lower <- psi_hat - 1.96 * se
  ci_upper <- psi_hat + 1.96 * se

  # Calculate efficiency measure log()/log()
  efficiency <- abs(log(sigma_hat)/log(psi_hat))

  return(c(psi_hat, se, ci_lower, ci_upper, efficiency))
}


# panjer rekursion für psi_P
psi_panjer <- function(n){

  # Gewichtungsvektor g
  g <- function(x){
    y <- as.integer(x)
    res <- B0_pareto(y+1) - B0_pareto(y)
    return(res)
  }
```

```
80
81    g0 <- 0.5 #g(0)
82
83    # Initialisierung von phi mit Startwert
84    phi <- c()
85    phi[1] <- theta/(1 + theta - g0)
86
87    # Rekursive Berechnung mit Schleife
88    for (x in 1:n) {
89      sum <- 0
90      for (k in 1:x) {
91        sum <- sum + g(k) * phi[x - k + 1]
92      }
93      phi[x + 1] <- sum * 1/(1 + theta - g0)
94    }
95
96    # Berechnung von psi(u)
97    psi_u <- 1 - sum(phi)
98    psi_u
99
100 }
101
102 # Run simulations for each u value
103 results <- data.frame(
104   u = u_values,
105   psi_hat = NA,
106   se = NA,
107   ci_lower = NA,
108   ci_upper = NA,
109   efficiency = NA
110 )
111
112 for(i in 1:nrow(results)) {
113   res <- estimate_psi_algo2(results$u[i], n_sim, rho)
114   results$psi_hat[i] <- res[1]
115   results$se[i] <- res[2]
116   results$ci_lower[i] <- res[3]
117   results$ci_upper[i] <- res[4]
118   results$efficiency[i] <- res[5]
```

```
}

# Print results in format exactly matching Table 1
print("Algorithm II Results with 95% Confidence Intervals:")
cat("\n u(u) ± ^1.96/√n:\n")
for(i in 1:nrow(results)) {
  # Get the order of magnitude
  order <- ceiling(-log10(results$psi_hat[i]))
  # Format the numbers
  value <- results$psi_hat[i] * 10^order
  margin <- 1.96 * results$se[i] * 10^order  # ^1.96/√n

  cat(sprintf("u = %d: (%.1f ± %.1f) · 10^-%d     log^()/log^() =
      %.2f        _P = %.3f\n",
              results$u[i],
              value,
              margin,
              order,
              results$efficiency[i],
              psi_panjer(results$u[i])))
}

# Create a table with all the statistics
results_table <- data.frame(
  u = results$u,
  estimate = sprintf("%.1f · 10^-%d",
                     results$psi_hat * 10^ceiling(-log10(results$psi
                         _hat)),
                     ceiling(-log10(results$psi_hat))),
  conf_interval = sprintf("± %.1f · 10^-%d",
                          1.96 * results$se * 10^ceiling(-log10(
                              results$psi_hat)),
                          ceiling(-log10(results$psi_hat))),
  efficiency = sprintf("%.2f", results$efficiency)
)

print("\nComparison with paper values (Algorithm II):")
print(results_table)
```

```r
# Paper values from Table 1 (Algorithm II)
paper_estimates <- c(6.0e-1, 2.0e-1, 9.0e-2, 0.9e-2, 9.5e-3)
paper_conf <- c(0.3e-1, 0.2e-1, 1.7e-2, 0.5e-2, 5.9e-3)
paper_efficiency <- c(1.36, 0.60, 0.54, 0.51, 0.51)

# Calculate absolute differences
diff_table <- data.frame(
  u = u_values,
  estimate_diff = abs(results$psi_hat - paper_estimates),
  conf_interval_diff = abs(1.96 * results$se - paper_conf),
  efficiency_diff = abs(results$efficiency - paper_efficiency)
)

# Format differences in scientific notation
diff_table$estimate_diff_formatted <- sprintf("%.1e", diff_table$
    estimate_diff)
diff_table$conf_interval_diff_formatted <- sprintf("%.1e", diff_
    table$conf_interval_diff)
diff_table$efficiency_diff_formatted <- sprintf("%.2f", diff_table$
    efficiency_diff)

print("\nAbsolute Differences between Simulation and Paper Values:"
    )
print("(Positive values indicate simulation > paper)")
cat("\nEstimate Differences (|sim - paper|):\n")
for(i in 1:nrow(diff_table)) {
  cat(sprintf("u = %d: %s\n",
              diff_table$u[i],
              diff_table$estimate_diff_formatted[i]))
}

cat("\nConfidence Interval Width Differences (|sim - paper|):\n")
for(i in 1:nrow(diff_table)) {
  cat(sprintf("u = %d: %s\n",
              diff_table$u[i],
              diff_table$conf_interval_diff_formatted[i]))
}

cat("\nEfficiency Measure Differences (|sim - paper|):\n")
```

```r
for(i in 1:nrow(diff_table)) {
  cat(sprintf("u = %d: %s\n",
              diff_table$u[i],
              diff_table$efficiency_diff_formatted[i]))
}

# Paper values table for reference
paper_table <- data.frame(
  u = c(10, 50, 100, 500, 1000),
  paper_estimate = c("6.0 · 10^-1", "2.0 · 10^-1", "9.0 · 10^-2", "
      0.9 · 10^-2", "9.5 · 10^-3"),
  paper_conf = c("± 0.3 · 10^-1", "± 0.2 · 10^-1", "± 1.7 · 10^-2",
       "± 0.5 · 10^-2", "± 5.9 · 10^-3"),
  paper_efficiency = c("1.36", "0.60", "0.54", "0.51", "0.51")
)

print("\nPaper values (Algorithm II) for reference:")
print(paper_table)

# Create visualization
p1 <- ggplot(results, aes(x = u)) +
  geom_point(aes(y = psi_hat)) +
  geom_errorbar(aes(ymin = ci_lower, ymax = ci_upper), width = 20)
      +
  scale_y_log10() +
  labs(title = "Algorithm II: Ruin Probability Estimates",
       x = "Initial Capital (u)",
       y = " (u)") +
  theme_minimal()

p2 <- ggplot(results, aes(x = u, y = efficiency)) +
  geom_point() +
  geom_line() +
  labs(title = "Efficiency Measure",
       x = "Initial Capital (u)",
       y = "log ()/log ()") +
  theme_minimal()

# Arrange plots side by side
```

```
226 library(gridExtra)
227 grid.arrange(p1, p2, ncol=2)
```

Listing 5.2: Simulation der Ruin Wahrscheinlichkeiten für eine Pareto Verteilung in R (Algorithmus 2)

```
1  # Algorithm III Implementation for Pareto Distribution
2  # Based on the conditional Monte Carlo method described in the
       paper
3
4  # Required libraries
5  library(ggplot2)
6  library(tidyr)
7  library(dplyr)
8
9  # Parameters
10 theta <- 0.1  # loading parameter
11 rho <- 1/(1 + theta)  # geometric parameter
12 n_sim <- 1000  # number of simulations
13 u_values <- c(10, 50, 100, 500, 1000)  # initial capitals to test
14 a <- 1  # Pareto parameter
15 b <- 2  # Pareto parameter
16
17 # Import Pareto functions from sample_from_distributions (1).R
18 B0_inv_pareto <- function(x){
19   y <- ifelse(x < (b-1)/b, a*b/(b-1)*x, a/(b*(1-x))^(1/(b-1)))
20   return(y)
21 }
22
23 # Function to calculate B0_bar(y) = P(X > y)
24 B0_bar <- function(y) {
25   if(y <= 0) return(1)  # As specified in the algorithm
26   # For Pareto(1,2), P(X > y) = (1/y)^2 for y   1
27   return(min(1, (a/y)^b))
28 }
29
30 # Function to generate one Z value using Algorithm III
31 generate_Z_algo3 <- function(u, rho) {
32   # 1. Generate K from geometric distribution
33   K <- rgeom(1, 1-rho) + 1  # adding 1 since R's geometric starts
```

```
        from 0
34
35    # 2. Generate K claims and sort them
36    if(K == 1) {
37      Y <- u
38      m <- 0  # No previous maximum when K=1
39    } else {
40      # Generate and sort X values
41      X <- replicate(K, B0_inv_pareto(runif(1)))
42      X_sorted <- sort(X)
43
44      # Calculate Y using all but the largest value
45      Y <- u - sum(X_sorted[1:(K-1)])
46      m <- X_sorted[K-1]  # Second largest value (X_(K-1))
47    }
48
49    # 3. Calculate Z = B0_bar(max(Y,m))/B0_bar(m)
50    Z <- if(K == 1) B0_bar(Y) else B0_bar(max(Y, m))/B0_bar(m)
51
52    return(Z)
53  }
54
55  # Function to estimate psi(u) with confidence intervals
56  estimate_psi_algo3 <- function(u, n_sim, rho) {
57    # Generate n Z values
58    Z <- replicate(n_sim, generate_Z_algo3(u, rho))
59
60    # Calculate estimate and standard error
61    psi_hat <- mean(Z)
62    sigma_hat <- sd(Z)
63    se <- sigma_hat/sqrt(n_sim)
64
65    # Calculate 95% confidence interval
66    ci_lower <- psi_hat - 1.96 * se
67    ci_upper <- psi_hat + 1.96 * se
68
69    # Calculate efficiency measure log ()/log ()
70    efficiency <- abs(log(sigma_hat)/log(psi_hat))
71
```

```
  return(c(psi_hat, se, ci_lower, ci_upper, efficiency))
}


# panjer rekursion für psi_P
psi_panjer <- function(n){

  # Gewichtungsvektor g
  g <- function(x){
    y <- as.integer(x)
    res <- B0_pareto(y+1) - B0_pareto(y)
    return(res)
  }

  g0 <- g(0)

  # Initialisierung von phi mit Startwert
  phi <- c()
  phi[1] <- theta/(1 + theta - g0)

  # Rekursive Berechnung mit Schleife
  for (x in 1:n) {
    sum <- 0
    for (k in 1:x) {
      sum <- sum + g(k) * phi[x - k + 1]
    }
    phi[x + 1] <- sum * 1/(1 + theta - g0)
  }

  # Berechnung von psi(u)
  psi_u <- 1 - sum(phi)
  psi_u

}


# Run simulations for each u value
results <- data.frame(
  u = u_values,
```

```
    psi_hat = NA,
    se = NA,
    ci_lower = NA,
    ci_upper = NA,
    efficiency = NA
)

for(i in 1:nrow(results)) {
    res <- estimate_psi_algo3(results$u[i], n_sim, rho)
    results$psi_hat[i] <- res[1]
    results$se[i] <- res[2]
    results$ci_lower[i] <- res[3]
    results$ci_upper[i] <- res[4]
    results$efficiency[i] <- res[5]
}

# Print results in format matching Table 1
print("Algorithm III Results:")
cat("\n u  ± ^1.96/√n:\n")
for(i in 1:nrow(results)) {
    # Get the order of magnitude
    order <- ceiling(-log10(results$psi_hat[i]))
    # Format the numbers
    value <- results$psi_hat[i] * 10^order
    margin <- 1.96 * results$se[i] * 10^order

    cat(sprintf("u = %d: (%.1f ± %.1f) · 10^-%d     log^()/log^() = %.2
        f         _P = %.3f\n",
                results$u[i],
                value,
                margin,
                order,
                results$efficiency[i],
                psi_panjer(results$u[i])))
}

# Create visualization
p1 <- ggplot(results, aes(x = u)) +
    geom_point(aes(y = psi_hat)) +
```

```
149   geom_errorbar(aes(ymin = ci_lower, ymax = ci_upper), width = 20)
        +
150   scale_y_log10() +
151   labs(title = "Algorithm III: Ruin Probability Estimates",
152        x = "Initial Capital (u)",
153        y = " (u)") +
154   theme_minimal()
155
156 p2 <- ggplot(results, aes(x = u, y = efficiency)) +
157   geom_point() +
158   geom_line() +
159   labs(title = "Efficiency Measure",
160        x = "Initial Capital (u)",
161        y = "log ()/log ()") +
162   theme_minimal()
163
164 # Arrange plots side by side
165 library(gridExtra)
166 grid.arrange(p1, p2, ncol=2)
167
168 # Compare with values from Table 1 (Algorithm III column)
169 paper_values <- data.frame(
170   u = u_values,
171   estimate = c("5.5 · 10^-1", "1.9 · 10^-1", "8.6 · 10^-2", "1.0 ·
        10^-2", "5.3 · 10^-3"),
172   conf_interval = c("± 0.3 · 10^-1", "± 0.2 · 10^-1", "± 1.2 ·
        10^-2", "± 0.2 · 10^-2", "± 0.6 · 10^-3"),
173   efficiency = c(1.38, 0.72, 0.69, 0.77, 0.88)
174 )
175
176 print("\nPaper values (Algorithm III):")
177 print(paper_values)
178
179 # Calculate absolute differences
180 paper_estimates <- c(5.5e-1, 1.9e-1, 8.6e-2, 1.0e-2, 5.3e-3)
181 paper_conf <- c(0.3e-1, 0.2e-1, 1.2e-2, 0.2e-2, 0.6e-3)
182 paper_efficiency <- c(1.38, 0.72, 0.69, 0.77, 0.88)
183
184 diff_table <- data.frame(
```

```
  u = u_values,
  estimate_diff = abs(results$psi_hat - paper_estimates),
  conf_interval_diff = abs(1.96 * results$se - paper_conf),
  efficiency_diff = abs(results$efficiency - paper_efficiency)
)

print("\nAbsolute Differences between Simulation and Paper Values:")
print(diff_table)
```

Listing 5.3: Simulation der Ruin Wahrscheinlichkeiten für eine Pareto Verteilung in R (Algorithmus 3)

```
Simulierte Ruin Wahrscheinlichkeiten und ihre Präzision gemessen nach (5.2) für Pareto-Verteilte Claims

Pareto(1,2), θ=0.1, n=1000
Algorithmus I
ψ(u) ± 1.96σ/√n und log(σ)/log(ψ):
u = 10:   (5.4 ± 0.3) · 10^-1    log(σ)/log(ψ) = 1.18    ψ_P = 0.561
u = 50:   (2.0 ± 0.2) · 10^-1    log(σ)/log(ψ) = 0.57    ψ_P = 0.181
u = 100:  (8.9 ± 1.8) · 10^-2    log(σ)/log(ψ) = 0.52    ψ_P = 0.075
u = 500:  (1.2 ± 0.7) · 10^-2    log(σ)/log(ψ) = 0.50    ψ_P = 0.011
u = 1000: (7.0 ± 5.2) · 10^-3    log(σ)/log(ψ) = 0.50    ψ_P = 0.005

Pareto(1,2), θ=0.1, n=1000
Algorithmus II
ψ(u) ± 1.96σ̂/√n:
u = 10:   (5.9 ± 0.3) · 10^-1    log(σ)/log(ψ) = 1.38    ψ_P = 0.561
u = 50:   (1.8 ± 0.2) · 10^-1    log(σ)/log(ψ) = 0.65    ψ_P = 0.181
u = 100:  (9.1 ± 1.7) · 10^-2    log(σ)/log(ψ) = 0.54    ψ_P = 0.075
u = 500:  (0.8 ± 0.4) · 10^-2    log(σ)/log(ψ) = 0.51    ψ_P = 0.011
u = 1000: (7.0 ± 5.2) · 10^-3    log(σ)/log(ψ) = 0.50    ψ_P = 0.005

Pareto(1,2), θ=0.1, n=1000
Algorithmus III:
ψ(u) ± 1.96σ̂/√n:
u = 10:   (5.6 ± 0.3) · 10^-1    log(σ)/log(ψ) = 1.25    ψ_P = 0.561
u = 50:   (1.9 ± 0.2) · 10^-1    log(σ)/log(ψ) = 0.71    ψ_P = 0.181
u = 100:  (7.6 ± 1.1) · 10^-2    log(σ)/log(ψ) = 0.68    ψ_P = 0.075
u = 500:  (1.3 ± 0.2) · 10^-2    log(σ)/log(ψ) = 0.79    ψ_P = 0.011
u = 1000: (5.2 ± 0.7) · 10^-3    log(σ)/log(ψ) = 0.86    ψ_P = 0.005
```

Abbildung 5.1: Simulation für eine Pareto Verteilung

5.3.1 Numerische Interpretation und Analyse

Im Folgenden werden die drei Monte-Carlo-Algorithmen zur Schätzung der Ruinwahrscheinlichkeiten analysiert und interpretiert:
Alle Algorithmen liefern Schätzungen nahe den Panjer-Approximationswerten, wobei bei Algorithmus III durchgehend präzisere Ergebnisse mit kleineren Konfidenzintervallen erzielt werden, besonders bei größeren u-Werten. Zwar ist bei Algorithmus III eine höhere Rechenintensität durch Sortieroperationen und ähnliches zu erwarten, rechtfertigt die signifikant verbesserte Präzision bei seltenen Ereignissen den Einsatz von Algorithmus III. Auf die tatsächlich notwendige zusätzlich Rechenleistung wird, aber an dieser Stelle nicht weiter eingegangen, da dies den Rahmen der Arbeit sprengen würde.

5.3.2 Weitere Simulationen

Im folgenden Abschnitt werden weitere Simulationen gezeigt:

```
Simulierte Ruin Wahrscheinlichkeiten und ihre Präzision gemessen nach (5.2) für PME-Verteilte Claims

PME(3), θ=0.25, n=1000
Algorithmus I
Simulation Results:
ψ(u) ± 1.96σ/√n and log(σ)/log(ψ):
u = 50:  (5.6 ± 4.8) · 10^-3    log(σ)/log(ψ) = 0.50    ψ_P = 0.0032
u = 60:  (3.1 ± 3.2) · 10^-3    log(σ)/log(ψ) = 0.50    ψ_P = 0.0016
u = 70:  (2.0 ± 2.8) · 10^-3    log(σ)/log(ψ) = 0.50    ψ_P = 0.0011
u = 80:       -                 log(σ)/log(ψ) =  -      ψ_P = 0.0008
u = 90:       -                 log(σ)/log(ψ) =  -      ψ_P = 0.0006
u = 100: (1.0 ± 2.0) · 10^-3    log(σ)/log(ψ) = 0.50    ψ_P = 0.0004


PME(3), θ=0.25, n=1000
Algorithmus II
ψ(u) ± 1.96σ7√n:
u = 50:  (2.1 ± 1.8) · 10^-3    log(σ)/log(ψ) = 0.57    ψ_P = 0.0032
u = 60:  (4.9 ± 4.1) · 10^-3    log(σ)/log(ψ) = 0.51    ψ_P = 0.0016
u = 70:  (1.6 ± 0.2) · 10^-3    log(σ)/log(ψ) = 0.90    ψ_P = 0.0011
u = 80:  (1.2 ± 0.1) · 10^-4    log(σ)/log(ψ) = 0.97    ψ_P = 0.0008
u = 90:  (1.1 ± 0.1) · 10^-4    log(σ)/log(ψ) = 0.96    ψ_P = 0.0006
u = 100: (8.2 ± 0.3) · 10^-5    log(σ)/log(ψ) = 1.06    ψ_P = 0.0004


PME(3), θ=0.25, n=1000
Algorithmus III
ψ(u) ± 1.96σ7√n:
u = 50:  (3.6 ± 1.1) · 10^-3    log(σ)/log(ψ) = 0.72    ψ_P = 0.0032
u = 60:  (2.3 ± 2.1) · 10^-3    log(σ)/log(ψ) = 0.56    ψ_P = 0.0016
u = 70:  (1.3 ± 0.4) · 10^-3    log(σ)/log(ψ) = 0.76    ψ_P = 0.0011
u = 80:  (9.2 ± 1.9) · 10^-4    log(σ)/log(ψ) = 0.82    ψ_P = 0.0008
u = 90:  (6.4 ± 2.2) · 10^-4    log(σ)/log(ψ) = 0.77    ψ_P = 0.0006
u = 100: (4.3 ± 0.9) · 10^-4    log(σ)/log(ψ) = 0.84    ψ_P = 0.0004
```

Abbildung 5.2: Simulation für eine PME Verteilung

```
Simulierte Ruin Wahrscheinlichkeiten und ihre Präzision gemessen nach (5.2) für Lognormal-Verteilte Claims

Lognormal(-1.62, 1.8), θ=0.1, n=1000
Algorithmus I
ψ(u) ± 1.96σ/√n and log(σ)/log(ψ):
u = 0:    (9.0 ± 0.3)  · 10^-1    log(σ)/log(ψ) = 6.88    ψ_P = 0.883
u = 100:  (3.5 ± 0.3)  · 10^-1    log(σ)/log(ψ) = 0.71    ψ_P = 0.331
u = 1000: (1.1 ± 0.5)  · 10^-2    log(σ)/log(ψ) = 0.50    ψ_P = 0.011

Lognormal(-1.62, 1.8), θ=0.1, n=1000
Algorithmus II
ψ(u) ± 1.96σ/√n and log(σ)/log(ψ):
u = 0:    (8.5 ± 0.1)  · 10^-1    log(σ)/log(ψ) = 11.2    ψ_P = 0.883
u = 100:  (3.9 ± 0.3)  · 10^-1    log(σ)/log(ψ) = 0.78    ψ_P = 0.331
u = 1000: (7.0 ± 5.6)  · 10^-3    log(σ)/log(ψ) = 0.51    ψ_P = 0.011

Lognormal(-1.62, 1.8), θ=0.1, n=1000
Algorithmus III
ψ(u) ± 1.96σ/√n and log(σ)/log(ψ):
u = 0:    (9.1 ± 0.2) · 10^-1    log(σ)/log(ψ) = 11.9    ψ_P = 0.883
u = 100:  (3.6 ± 0.2) · 10^-1    log(σ)/log(ψ) = 1.10    ψ_P = 0.331
u = 1000: (8.8 ± 1.9) · 10^-3    log(σ)/log(ψ) = 0.73    ψ_P = 0.011
```

Abbildung 5.3: Simulation für eine Lognormal Verteilung

Die Simulationen zeigen, dass Algorithmus III in der Regel deutlich bessere Effizienzwerte liefert. Ein Fall, in dem dies nicht der Regel entspricht, wurde zwar nicht simuliert, aber wird nun herangezogen. Es handelt sich dabei um die Weibull-Verteilung:

Lemma 5.3.1. *Sei B_0 eine Weibull-Verteilung mit Dichte $b_0(x) = rx^{r-1}e^{-x^r}$ und Überlebensfunktion $\overline{B}_0(x) = e^{-x^r}$. Dann ist Algorithmus III im Sinne von (5.2) nicht effizient.*

Beweis: Für den Fall $K = 2$ erhalten wir folgende untere Schranke:

$$
\begin{aligned}
E[Z^2|K=2] &\geq \int_0^{u/2} \frac{\overline{B}_0^2(u-y)}{\overline{B}_0^2(y)} f_{X_{(1)}}(y)dy \\
&= 2\int_0^{u/2} \overline{B}_0^2(u-y) y y^{r-1} dy \\
&\geq 2r(u/2)^{r-1} \int_0^{u/2} \overline{B}_0^2(u-y)dy \\
&= 2r(u/2)^{r-1} \int_{u/2}^{u} \overline{B}_0^2(y)dy \\
&\geq \int_{u/2}^{u} 2ry^{r-1}\overline{B}_0^2(y)dy \\
&= e^{-2(u/2)^r} - e^{-2u^r} = e^{-2^{1-r}u^r}(1+o(1))
\end{aligned}
$$

Somit gilt für das Effizienzmaß:

$$
\begin{aligned}
\lim_{u\to\infty} \frac{\log \sigma_Z}{\log \psi(u)} &\leq \lim_{u\to\infty} \frac{\frac{1}{2}\log\left(E[Z^2|K=2]P(K=2) - E[Z^2]\right)}{\log\left(\overline{B}_0(u)/\theta\right)} \\
&\leq \lim_{u\to\infty} \frac{2^{1-r}u^r/2}{u^r} = \frac{1}{2^r} < 1
\end{aligned}
$$

Da $\frac{1}{2^r} < 1$ für alle $r > 0$, ist Algorithmus III für Weibull-verteilte Schäden nicht asymptotisch effizient.

6 Conclusio

Das Ziel dieser Arbeit war es, Heavy-Tailed Verteilungen zu präsentieren und vor allem die Klasse der subexponentiellen Verteilungen tiefgründiger zu studieren und zu untersuchen. Genauso war es ein Ziel, verschiedene Ansätze zur Schätzung von $\psi(u)$ mittels Monte-Carlo-Simulation zu untersuchen und zu vergleichen. Jede Methode weist spezifische Vor- und Nachteile hinsichtlich Präzision und Rechenaufwand auf:

1. **Algorithmus I (Standard Monte Carlo):** Simuliert die vollständige Summe der Schäden und prüft, ob diese den Schwellenwert u überschreitet. Der Ansatz ist konzeptionell einfach, verliert jedoch bei hohen Schwellenwerten an Effizienz.
2. **Algorithmus II (Erster bedingter Monte Carlo):** Nutzt die integrierte Randverteilung, um die Varianz zu reduzieren. Dieser Ansatz führt zu stabileren Schätzungen, zeigt jedoch ähnliche asymptotische Eigenschaften wie Algorithmus I.
3. **Algorithmus III (Verbesserter bedingter Monte Carlo):** Basiert auf dem Prinzip, dass bei subexponentiellen Verteilungen der Ruin typischerweise durch einen einzelnen großen Schaden verursacht wird. Diese Eigenschaft wird genutzt, um eine deutlich verbesserte Varianzreduktion zu erzielen.

Literaturverzeichnis

[Abate et al.(1994)Abate, Choudhury, and Whitt] J. Abate, G. L. Choudhury, and W. Whitt. Waiting-time tail probabilities in queues with long-tailed service-time distributions. *Queueing Systems*, 16:311–338, 1994.

[Abate and Whitt(1992)] Joseph Abate and Ward Whitt. The fourier-series method for inverting transforms of probability distributions. *Queueing Systems*, 10:5–88, 1992.

[Asmussen(1985)] S. Asmussen. Conjugate processes and the simulation of ruin problems. *Stochastic Processes and their Applications*, 20:213–229, 1985.

[Asmussen and Binswanger(1997)] S. Asmussen and K. Binswanger. Simulation of ruin probabilities for subexponential claims. *ASTIN Bulletin: The Journal of the IAA*, 27 (2):297–318, 1997. doi: 10.2143/AST.27.2.542054.

[Asmussen and Rubinstein(1995)] Søren Asmussen and Reuven Rubinstein. Steady-state rare events simulation in queueing models and its complexity properties. In J. Dshalalow, editor, *Advances in Queueing*, pages 79–102. CRC Press, Boca Raton, Florida, 1995.

[Asmussen and Albrecher(2010)] Søren Asmussen and Hansjörg Albrecher. *Ruin Probabilities*, volume 14 of *Advanced Series on Statistical Science & Applied Probability*. World Scientific, Singapore, Hackensack, NJ, 2nd edition edition, 2010.

[Chistyakov(1964)] V. Chistyakov. A theorem on sums of independent positive random variables and its applications to branching random processes. *Theory of Probability & Its Applications*, 9(4):640–648, 1964.

[Cramér(1930)] Harald Cramér. On the mathematical theory of risk. In *Skandia Jubilee Volume*. Skandia, Stockholm, 1930.

[Cramér(1955)] Harald Cramér. Collective risk theory. In *Skandia Jubilee Volume*. Skandia, Stockholm, 1955.

[Embrechts et al.(2013)Embrechts, Klüppelberg, and Mikosch] Paul Embrechts, Claudia Klüppelberg, and Thomas Mikosch. *Modelling extremal events: for insurance and finance*, volume 33. Springer Science & Business Media, 2013.

[Foss et al.(2011)Foss, Korshunov, and Zachary] Sergey Foss, Dmitry Korshunov, and Stan Zachary. *An Introduction to Heavy-Tailed and Subexponential Distributions*. Springer, New York, 2011.

[Heidelberger(1995)] P. Heidelberger. Fast simulation of rare events in queueing and reliability models. *ACM Transactions on Modeling and Simulation*, 5:43–85, 1995.

[Hubalek(2022)] Friedrich Hubalek. Risiko- und ruintheorie vorlesungsfolien. 2022.

[Lundberg(1903)] Filip Lundberg. *1. Approximerad framställning af sannolikhetsfunktionen: 2. Återförsäkring af kollektivrisker. Akademisk afhandling.* Almqvist &Wiksells, 1903.

[Nair et al.(2022)Nair, Wierman, and Zwart] J. Nair, A. Wierman, and B. Zwart. *The Fundamentals of Heavy Tails: Properties, Emergence, and Estimation.* Cambridge University Press, Cambridge, 2022.

[Rolski et al.(2009)Rolski, Schmidli, Schmidt, and Teugels] Tomasz Rolski, Hanspeter Schmidli, Volker Schmidt, and Jozef Teugels. *Stochastic Processes for Insurance and Finance*, volume 505 of *Wiley Series in Probability and Statistics.* John Wiley & Sons, 2009. (siehe S. 5).

[Rubinstein(1981)] Reuven Y. Rubinstein. *Simulation and the Monte Carlo Method.* Wiley, New York, 1981.

[Siegmund(1976)] D. Siegmund. Importance sampling in the monte carlo study of sequential tests. *The Annals of Statistics*, 4(4):673–684, 1976. doi: 10.1214/aos/1176343541.