Higher-Order Approximations for Compound Sum Distributions

Evert Van Hecke
HEC Lausanne
MSc. in Actuarial Science
Email: evert.vanhecke@unil.ch

***Abstract*—Over the last decades, natural disasters and financial crises have shown us the need for precise risk assessment and management. Compound sums are essential in quantifying risks across insurance, environmental sciences, finance, and health sectors. This project introduces a Python-based implementation for computing higher-order approximations for the tail distribution of such compound sums. Specifically, we focus on the tail distribution $\bar{G}(s)$ of $S_N = \sum_{i=1}^{N} X_i$, where N represents claim frequency and X_i the claim severity. They are both modelled as independent and identically distributed random variables. Our implementation supports several heavy-tailed and/or sub-exponential distributions for claim severities, namely the Lognormal, Weibull, Fréchet, and Pareto distributions. The Python code approximates the tail probabilities with higher-order accuracy, by doing asymptotic expansions for small to large values of s. This work also includes a user-friendly GUI for seamless exploration and analysis. We address the effectiveness of these higher-order approximations in estimating tail probabilities and their implications for practical risk management in financial and actuarial contexts.**

I. Introduction

Compound sums are an essential component of the collective risk models, and are widely used in various fields to manage and quantify risks by aggregating multiple factors. The following are some applications of compound sums in different sectors:

- **Insurance Industry**: Insurers can estimate the total claims from policies like life, health, property,... insurance. By modeling the frequency and severity of claims, insurers can set premiums and manage financial risk effectively based on the estimates from the compound sums.
- **Environmental Sciences**: Compound sim models can be used for assessing risks related to natural hazards and environmental impacts. For example, they are used to estimate potential flood or wildfire damage. A good estimation of the compound sums can help the aiding in disaster response planning and resource allocation.
- **Financial Risk Management**: In finance, compound sums are utilized to simulate the overall risk of investment portfolios. This aids in risk management and investment strategy optimization.
- **Health Risk Assessment**: To help with intervention planning and resource allocation, public health experts can use compound sums to simulate the overall impact of environmental contaminants or disease outbreaks. In turn, this can be of help in assisting in planning interventions and/or allocating healthcare resources.

In essence, compound sums help with decision-making and resource management in a variety of industries by offering insightful information about the cumulative consequences of random events.

In this project, we present a Python-based implementation designed to compute higher-order approximations for the tail distribution $\bar{G}(s)$ of a compound sum $S_N = \sum_{i=1}^{N} X_i$, where N represents the claim frequency and X_i denotes the claim severity. Both N and X_i are modelled as independent and identically distributed (i.i.d.) random variables. The tail distribution $\bar{G}(s) = P(S_N > s)$ represents the probability that the sum exceeds a given value or threshold s. My code aims to approximate this tail distribution for different degrees of higher-order accuracy.

The implemented code supports several common heavy-tailed and/or sub-exponential distributions for the claim severities, including Lognormal, Weibull (with shape parameter $<$ 1), Fréchet, and Pareto distributions. Each distribution presents unique challenges in the conditions for the different higher-order approximation formulas for accurately approximating the tail of the sum. Given the fat tails of these distributions, combined with the computational complexity of doing convolutions, calculating $\bar{G}(s)$ directly can be computationally intensive.

To address this, we use several asymptotic expansions for small to large values of s to approximate tail probabilities $\bar{G}(s)$. While first-order asymptotic formulas are well-established in the literature and easy to compute (e.g., Greenwood (1973) [2]; Teugels & Veraverbeke (1973) [3]; von Bahr (1975) [4]; Embrechts & Veraverbeke (1982) [5]), higher-order expansions offer more refined estimates, but also bringing forth more computational complexity. For this project we include second-order expansions (e.g., Grübel (1987) [6]; Omey & Willekens (1987) [7]) and further advances in higher-order approximations (e.g., Geluk et al. (2000) [11]; Borovkov & Borovkov (2002) [12]; Barbe & McCormick (2009) [13]; Barbe et al. (2007) [14]). The higher-order approximations implemented in our Python code are based on the work by Albercher et al. [1]. We restrict ourselves to standard higher-order approximations from the 3rd to the 6th degree.

The Python implementation efficiently calculates these higher-order terms, using, among other things, numerical integration and Monte Carlo simulation. This enables accurate and computationally feasible estimation of tail probabilities, crucial for various risk assessment applications. However, some of the algorithms used still rely on computationally slower symbolic evaluations.

In addition to the core Python implementation, we have developed a user-friendly graphical interface which facilitates the analysis of these approximations. This interface integrates seamlessly with the Python code, providing an intuitive environment for users to select distributions, and input parameters, perform computations, and ultimately visualize results.

II. Research Question and Literature Review

A. Research Question

As we discussed in the introduction, compound sums are used in different fields to model and quantify risks associated with different types of events. It is quite difficult to estimate the odds of extreme events or large claims, especially when heavy-tailed severity distributions are involved. As complex as it is to predict accurately, doing so is important, even essential, for risk management and decision-making in the insurance, environmental science, finance, and health risk assessment domains.

To address this complexity, I have created a Python-based implementation that computes higher-order approximations for the tail distribution $\bar{G}(s)$ of a compound sum $S_N = \sum_{i=1}^{N} X_i$. The implementation offers useful insights into the possibility of uncommon but significant events (heavy-tailed severity distributions), assisting practitioners from these different fields in rapidly and effectively estimating the odds of extreme events.

The central research question addressed by this project is thus:

How can higher-order asymptotic approximations be effectively computed and applied to estimate the tail distribution $\bar{G}(s)$ of a compound sum $S_N = \sum_{i=1}^{N} X_i$, and what are the implications of these approximations for real-world risk assessment in financial, environmental, health and actuarial contexts?

This question aims to explore the effectiveness of these approximations, with an increasing degree of sophistication, in estimating tail probabilities and to evaluate their practical impact on risk assessment and decision-making processes.

B. Literature Review

Reviewing previous studies on tail distribution estimates (and their practical applications) is the starting point in order to answer the research question:

- **Foundational Asymptotic Approximations**: Greenwood (1973) [2], Teugels & Veraverbeke (1973) [3], and von Bahr (1975) [4] laid the groundwork for first-order asymptotic formulas. These contributions established methods to approximate tail probabilities of compound distributions under the assumption of heavy tails.

- **Second-Order Expansions**: Subsequent developments in second-order asymptotic approximations were made by Grübel (1987) [6] and Omey & Willekens (1987) [7]. These works refined the first-order estimates and provided more accurate tail probability estimates.

- **Advanced Higher-Order Approximations**: Later research, including Geluk et al. (2000) [11], Borovkov & Borovkov (2002) [12], Barbe & McCormick (2009) [13], and Barbe et al. (2007) [14], introduced methods for higher-order asymptotic expansions, enhancing the precision of tail probability approximations.

Albechet et al. (2010) [1] provides an overview of these approximations alongside some of their proofs. The methodology and implementation used in this project is build upon the presentation provided in [1]. Overall, the literature offers a thorough framework for comprehending the development and use of asymptotic approximations, which influences the computational techniques and real-world applications examined in this study.

III. Methodology

To address the research question effectively and explore the computational complexity of the given problem, we present in this section the methodology used, which incorporates first to sixth-order approximations. These approximations and their conditions require the evaluation of various symbolic and numerical methods. The methodology used in this project works as follows:

A. Initialization of Distributions

The first step is to initialize the claim frequency and severity distributions used for the compound sum distribution by defining their respective parameters. The following is a short overview of t

a) Claim Frequency Distribution: The claims amount or frequency distribution take on one of 3 distributions: Poisson, Binomial, or Negative Binomial. These 3 distributions are part of the $(a, b, 0) - class$. The corresponding parameters (a, b) are essential for the Panjer recursion ([15]) and will be further addressed in the next section. The claim amount distributions are defined as follows:

- **Poisson**: Parameter λ (rate) is used to define the Poisson distribution.
- **Binomial**: Parameters n (#trails) and p (probability of success) are used.
- **Negative Binomial**: Parameters r (#successes) and p (probability of success) are used.

b) Claim Severity Distribution: The claim severity distribution can be one of the 4 following heavy-tailed and/or sub-exponential distributions: Lognormal, Weibull, Frechet, or Pareto. They require the following parameters:

- **Lognormal**: Parameters μ (log-scale) and σ (shape) are used.
- **Weibull**: Parameters λ (scale) and k (shape) are used.
- **Frechet**: Parameters α (shape), β (scale), and x_{min} (minimum value) are used.

- **Pareto**: Parameters α (shape) and x_{min} (minimum value/ scale) are used.

B. Approximation Techniques

Now that we've initialized the claim frequency and severity distributions, we can define the first- to sixth-order approximations which we will work with:

a) First-Order Approximation: The classical first-order asymptotic approximation for the tail probability $P(X_1 + \cdots + X_N > s) = \bar{G}(s) = a_1(s) + o\left(\bar{F}(s)\right)$ is given by:

$$a_1(s) := E[N]\bar{F}(s),$$

where $\bar{F}(s)$ is the tail of the cumulative distribution function (CDF) of the severity distribution, and $E[N]$ is the expected number of claims. This approximation assumes that the tail behaviour of $G(s)$ can be well captured by this simple first-order term. This is quite restrictive and we can therefore easily refine this approximation by adding more terms. This is exactly what is done in the approximations that follow.

b) Second-Order Approximation: The second-order asymptotic approximation refines the first-order estimate: $\bar{G}(s) = a_2(s) + o\left(f(s)\right)$ The second-order term, under the assumption that $E[X] < \infty$ and F has a continuous density f that is long-tailed, dominantly varying and with an upper Matuszewska index $\alpha(f) < -1$, is given by:

$$a_2(s) := a_1(s) + 2E[\binom{N}{2}]E[X]f(s),$$

where $a_1(s)$ is the first-order approximation and $f(s)$ is the density function of $F(s)$. If one of the previously stated conditions is not fulfilled, but $\bar{F}(s)$ is regularly varying with index $-\alpha$ and $f(s)$ is also a regularly varying density function, a second-order approximation can be expressed as:

$$\mathrm{a}_2(s) := \begin{cases} a_1(s) - \frac{(2-\alpha)\Gamma(2-\alpha)}{(\alpha-1)\Gamma(3-2\alpha)} E[\binom{N}{2}] f(s) \int_0^s \bar{F}(y)\,dy, & 0 < \alpha < 1 \\ a_1(s) + 2E[\binom{N}{2}] f(s) \int_0^s \bar{F}(y)\,dy, & \alpha = 1 \end{cases}$$

c) Higher-Order Approximations: Further refinement of the estimates can be done by the higher-order approximations: For k-th order asymptotic approximations, where $k \geq 3$, the following general form is used: $\bar{G}(s) = a_k(s) + o\left(F^{(k-1)}(s)\right)$, where

$$\mathrm{a}_k(s) := a_2(s) + \sum_{j=1}^{k-2} \frac{(-1)^j E\left[N(X_1+\cdots+X_{N-1})^{j+1}\right]}{(j+1)!} f^{(j)}(s).$$

These higher-order approximations offer increasingly more refined estimates of tail probabilities of the compound sum distribution, accommodating more complex behaviours and interactions in the severity and claims distributions. Nevertheless, more refined approximations also bring forth increasingly more complexity in the Python code. Furthermore, the complexity surrounding the implementation of the previous approximation functions in Python code lies in the fact that they rely on both symbolic and numerical methods. How I decided to handle this complexity will be addressed in the next section.

C. Results and Output

The results from each approximation are computed and compared. The computed results are formatted for user-friendly visualization in the GUI. Warnings or issues encountered during computations will be reported accordingly to the user.

IV. Implementation

In this section, we delve into the implementation details of our project. We focus on the choices made to enhance performance and efficiency and tackle complexity issues raised in the previous section. Several key packages were used in our implementation, namely: `numpy`, `scipy`, `sympy`, and `numba`.

A. Distribution Initialization

The `initialize_distributions` function initializes the claim amount and severity distributions based on specified names and parameters, all of which are chosen by the user. The distributions are loaded in using `scipy.stats`, which facilitates the generation of random variables and the computation of statistical properties for their distributions. The `scipy` functions can than also be used to evaluate the expectation operator in the approximation functions.

B. Custom Probability Density Functions (PDF) and Cumulative Distribution Functions (CDF)

Although the distributions are loaded in using `scipy.stats`, we decided to implement our own PDF and CDF functions, to improve performance of our algorithms:

- **Custom Implementations**: By building custom PDF and CDF functions, we can leverage `numba`'s Just-In-Time (JIT) compilation to accelerate these computations. The JIT compiler converts Python functions into optimized machine code. This gave us significant speedups for numerical calculations of the approximation functions.
- **Performance Benefits**: Using `numba` with our own implementations avoids the limitations of `scipy.stats` functions, which do not support JIT compilation. As these numerical calculations form the core of our approximation methods, for a performance and computational efficiency standpoint, it was deemed best to use these costum implementations wherever possible.
- **Custom Numerical Derivatives**: Because we rely on the custom PDF and CDF functions using numerical methods, we needed to consequently also create our own custom derivatives. This involves finite difference techniques with a small enough value for ϵ to estimate the required derivatives.

C. Approximation Methods

Optimizing the approximations methods with regards to computational efficiency was then done as follows:

- **First and Second Order Approximations**: These methods use straightforward formulas and are implemented

using vectorized operations in numpy. By using vectorization, we can reduce the overhead of iterative computations, resulting in faster execution, rather than using standard (for-)loops.
- **Higher-Order Approximations**: For methods such as third_Order_Approximation through sixth_Order_Approximation, we unfortunately still have to rely on the, slower, symbolic evaluations from the sympy package. This is due to the difficult expectation operator ($E\left[N(X_1 + \cdots + X_{N-1})^{j+1}\right]$) in the approximation equations $a_k(s)$. To not completely fall prey to the slower calculations of the sympy.stats package, we use Monte Carlo simulations to first simulate N and subsequently use the expectation operator from the sympy.stats package to evaluate the sum in the expectation operator. Random number generation and sampling are performed using numpy to ensure both accuracy and efficiency. Although this approach results in slower computations than completely numerical methods, I didn't find any better way to handle complex expectation operators that are difficult to evaluate numerically.

D. Integration of Functionalities

The doCalculations function serves as the central component that integrates all functionalities of the framework and coordinates their execution:

- **Integration of Components**: The function combines the various modules, including the custom PDF and CDF calculations, numerical differentiation, and approximation methods.
- **Interface Communication**: First it loads the chosen distributions and parameters from the user interface and in the end it passes the computed approximations and relevant data back to the user interface. This integration allows for coherent presentation and utilization of the calculated results in subsequent analyses. We elaborate on the implementation of the GUI later on.

E. Computational Complexity, Performance and Efficiency

To sum up how we managed the computational complexity of the algorithms used, and to balance performance and efficiency:

- **Custom Implementations**: Building our own PDF and CDF functions and using JIT compilation with numba provides significant performance improvements over using scipy.stats functions.
- **Efficient Libraries**: We use numpy for (vectorized) numerical calculations and the @jit decorator from numba is used for performance-critical sections whenever possible. Slower symbolic computations with sympy are reserved for cases where numerical methods are impractical, though sped up by using Monte Carle Simulations.
- **Threshold Handling**: Thresholds are applied to prevent overflow and ensure numerical stability.
- **Manual Condition Checking**: Conditions for selecting the appropriate approximation formula are handled by the author by hand. Given that only four severity distributions are used, this approach is feasible and avoids the performance overhead associated with symbolic evaluations, which can be time-consuming.

F. Interface Overview

We have also created a graphical interface that is very user-friendly and intuitive to explore and analyze the higher-order approximations of the tail distribution $\bar{G}(s)$, in addition to the basic Python method for doing so. By integrating this interface with the Python code, users can choose and configure different distributions, carry out calculations, receive specific warnings or error messages and ultimately visually inspect the approximation results in an easy-to-use and interactive environment. I used 2 classes named DistributionCalculator and PlottingWindow. The former holds all the functionalities of initializing the user-defined distributions and communicating with the approximations.py file which holds all the implementation functions that are used for the approximation calculations. The latter is used to visualize the approximations results to the user. Overall, the Interface serves as a bridge between complex mathematical computations and practical usability, offering several key features:

- **User-Friendly Interaction**:
  - *Distribution Selection*: The interface allows users to select from a range of claim frequency and severity distributions, as shown in Figure 1. Each choice is accompanied by dynamic input fields tailored to the parameters required by the selected distribution. (see Figure 2)

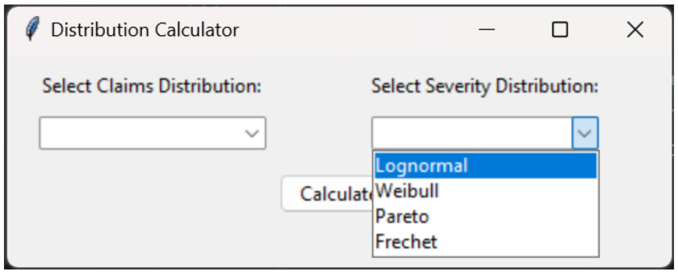
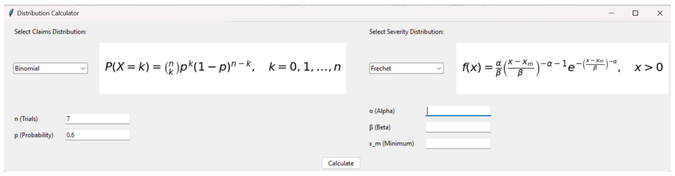


Fig. 1. Distribution Selection Interface

  - *LaTeX Formula Display*: For users who require a deeper understanding of the underlying mathematics, the interface renders LaTeX formulas corresponding to each selected distribution (see Figure 3). This feature ensures that users can clearly see the mathematical basis for the computed approximations.
  - *Parameter Input*: Users can easily input the necessary parameters for both claim frequency and severity distributions. The interface includes real-time validation to ensure that all inputs meet the necessary criteria, thus preventing common input errors. When the parameter inputs don't meet those


Distribution Calculator

Select Claims Distribution:

Binomial

$P(X=k)=\binom{n}{k}p^k(1-p)^{n-k}, \quad k=0,1,\ldots,n$

n (Trials) 7

p (Probability) 0.6

Select Severity Distribution:

Frechet

$f(x)=\frac{\alpha}{\beta}\left(\frac{x-x_m}{\beta}\right)^{-\alpha-1}e^{-\left(\frac{x-x_m}{\beta}\right)^{-\alpha}}, \quad x>0$

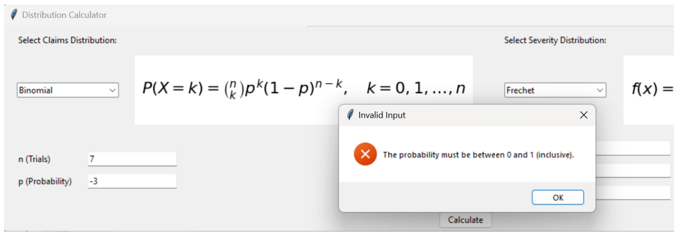
α (Alpha)

β (Beta)

x_m (Minimum)

Fig. 2. Parameter Input and LaTeX Formula's

necessary criteria, error messages will appear to the user, communicating exactly which criteria are not met. For an example, see Figure 3.


Distribution Calculator

Select Claims Distribution:

Binomial

$P(X=k)=\binom{n}{k}p^k(1-p)^{n-k}, \quad k=0,1,\ldots,n$

n (Trials) 7

p (Probability) -3

Select Severity Distribution:

Frechet

$f(x)=$

Invalid Input

The probability must be between 0 and 1 (inclusive).

Fig. 3. Parameter Validation Error Message

- **Visualization and Analysis**:
  - *Interactive Plotting*: The interface generates visualizations of the tail distribution approximations. Users can view both aggregate and detailed plots in the dual-pane plotting window. Smaller plots are displayed on the left, while larger, more detailed plots are shown on the right. The user can hand-select previously generated plots from the left to display them in more detail on the right, allowing users to interactively explore and analyze the results. (Figure 4)

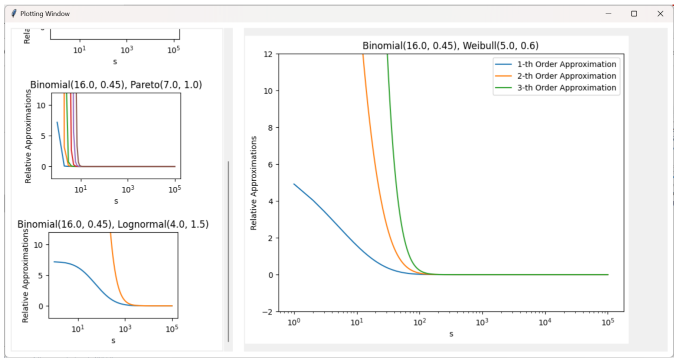
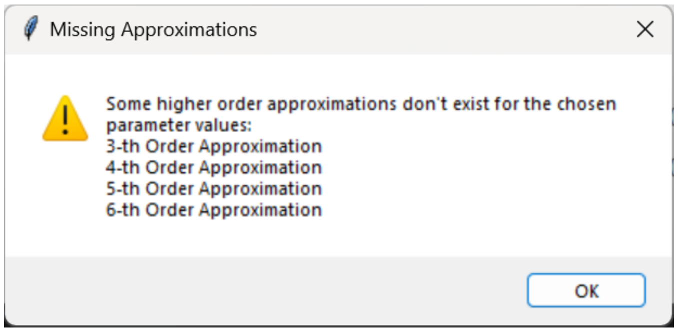


Fig. 4. Interactive Approximation Plotting

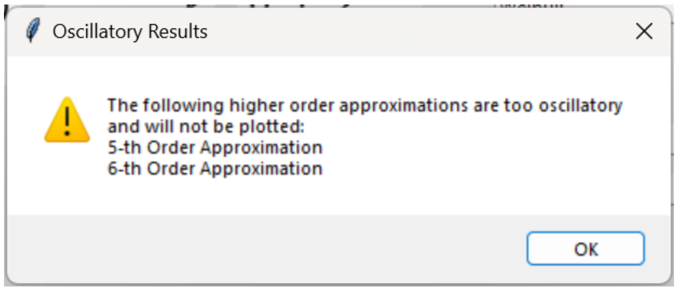
- **Error Handling and User Feedback**:
  - *Guided User Input*: Apart from the earlier discussed parameter validation error messages which feedback to guide users in entering valid input values, the interface has more warning messages to inform users of certain details. Moreover, for certain parameter values the approximations don't exist (symbolic evaluation of the expectation operator can be equal to ∞). The interface also communicates this to the user with specific warning messages, see Figure 5


Missing Approximations

Some higher order approximations don't exist for the chosen parameter values:
3-th Order Approximation
4-th Order Approximation
5-th Order Approximation
6-th Order Approximation

Fig. 5. Parameter Validation

  - *Clouded Plots*: Some higher-order approximations when using a Weibull severity distribution are very oscillatory. This is suspected to be due to the custom distribution functions and derivatives. The user will be warned about this (see Figure 6) and the approximations won't be plotted, to insure better visualisations.


Oscillatory Results

The following higher order approximations are too oscillatory and will not be plotted:
5-th Order Approximation
6-th Order Approximation

Fig. 6. Oscillatory Approximations Warning

G. *Benefits and Applications*

To summarize, the graphical user interface enhances the accessibility and usability of the higher-order approximation methods which were developed in our Python code. By offering an intuitive platform for user input, and approximation visualizations, the interface allows practitioners to have an easy-to-use and hands-on experience with the approximation techniques, without having to know all the mathematical details of said approximations. For applications like risk assessment, financial modeling, and actuarial science, where precise tail estimation is essential for making decisions, this capacity is crucial for practitioners.

In conclusion, the interface enhances the Python implementation and democratizes access to complex statistical methods, hence increasing the accessibility and usefulness of advanced tail distribution analysis for a wider range of users.

V. Further Improvements and Code Maintenance

The current methodology still has a certain amount of drawbacks. In this section, we will go over some of these drawbacks and some possible solutions. Moreover, we will also present how other developers can access the code and improve on said drawbacks.

A. Panjer Recursion

The Panjer Recursion [15] is a well-known algorithm to find the CDF of the compound sum distribution ($G(s)$). The reason why it is used is because it has a lower computational complexity ($O(n^2)$) than standard convolutions ($O(n^3)$), which is very handy when calculation the tail of compound sums. The goal of using the Panjer Recursion in my project was to plot the higher-order approximations as relative approximations compared to the value found for $\bar{G}(s)$ from the Panjer Recursion. This was tried in the following way:

The Panjer recursion algorithm for calculating the Cumulative Distribution Function (CDF) and Probability Density Function (PDF) for compound sum distribution is implemented by the `panjer_recursion_bounds` function.

The function precomputes severity distribution values and stores the results in dictionaries to increase efficiency. Compared to lists, this method enables faster lookups, especially when accessing intermediate items during the recursion process. The PDF is then iteratively computed using the Panjer recursion, and the CDF is obtained as the cumulative total of the PDF values.

The idea was to then divide all the approximations by the tail of the distribution computed with the panjer recursion, for the corresponding s values, stored in the dictionary. However, the Pareto and Frechet distributions don't start from the value 0, but rather from x_{min}, as chosen by the user. Unfortunately, I couldn't figure out how to address this issue and therefore opted not to use the Panjer Recursion as of now. Therefore, the approximations are not plotted as relative approximations.

Proposed Solution: To address this issue, future work should focus on adapting the Panjer recursion to handle distributions with a minimum value greater than zero. One possible approach is to modify the panjer recursion algorithm so that the minimum value aligns with the algorithm's requirements. As an alternative, a different numerical integration or convolution technique could be used to obtain a more precise estimate of the aggregate loss distribution.

B. Distribution Fitting

To use the code in its current stage, the user needs to already know the claim frequency and severity distributions, as well as its parameters. It would be more interesting to add functionalities which allow the user to upload its data and then let an algorithm fit (one or more) distribution(s) on the data to find the best approximations of the frequency and severity distributions.

C. Further Approximations

In our methodology, we've restricted ourselves to the use of standard higher-order approximations. As discussed in Albrecher et al. [1], more approximations exist. A further improvement to our methodology would thus be to include (some of) these approximations.

D. Code Maintenance

A public repository on GitHub contains all of the code used for the calculations and interface covered in this paper. Developers and researchers interested in exploring, utilizing, or enhancing the code can access it at the following link: https://github.com/evanhecke/StatsOrderExpansions.

The repository is structured to facilitate easy navigation and understanding of the codebase. Also, all the functions in the code are well-documented to avoid any confusion of their use. Developers are welcome to submit contributions via pull requests for modifications or enhancements to the project. By working together, we can make sure that the code is kept current and open to changes from the larger research and development community.

VI. Results

As my research question is not concerned with solving a specific problem, but rather with building a tool for practitioners to use, for my results I will present and discuss some of my plotted approximations for certain compound sum distributions.

The graphs presented in this section were all taken from the right-hand side pane of the interactive plotting window discussed in the Implementation section.

To assess the accuracy of the first-order and higher-order approximations, we first of all point to the findings of Abate et al. (1994) [16], which show that the first-order asymptotic formula often provides limited accuracy for tail probabilities in practical settings. This finding underscores the importance of using higher-order approximations. Unfortunately, because of the unresolved issue with regard to the Panjer Recursion, we are unable to assess the (under/over) performance of the approximations. For this, we point the interested reader towards Abate et al. (1994) [16] and Albrecher et al. (2010) [1]. We will provide a summary of the findings of both these papers at the end of this section.

However, we are still able to inspect the rate of convergence to 0 for the tails of the compound sums $\bar{G}(s)$. For this we will turn to figures 7, 8 and 9. The horizontal axes in the plots are shown on a logarithmic scale. This corresponds to tail (or ruin) probabilities for s values ranging from 0 (or x_{min}) to 1.000.000. This range encompasses the practical values of s of interest. In Figure 7 we find that all approximations converge to 0 for low s value. This is due to the large value for α. Whereas for the Weibull and Lognormal distributions (Figures 8 and 9, respectively) the tails of the compound sum only start converging for larger values for s. The convergence is of course highly dependent on the scale and shape parameters of the severity distributions. A persistent finding in all the figures is that the higher the order of the approximation, the slower it converges.

A. Findings from the Literature

According to the results in Abate et al. (1994) [16], in real-world scenarios, tail probabilities are frequently only partially accurately predicted by the first-order asymptotic

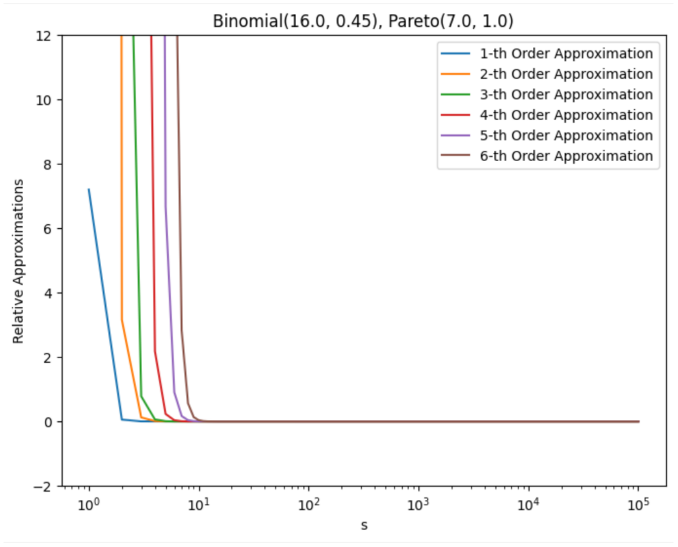


Fig. 7. $\bar{G}(s)$ for $N \sim Bin(n = 16, p = 0.45)$, $X \sim Par(\alpha = 7, x_m = 1)$

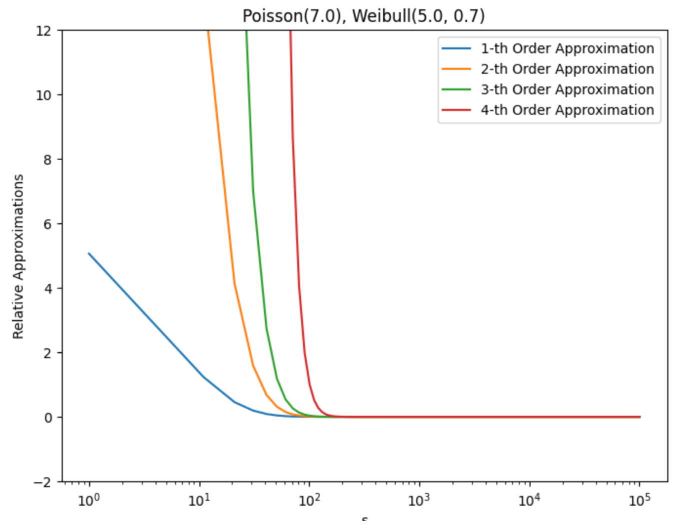


Fig. 8. $\bar{G}(s)$ for $N \sim Poi(\lambda = 7)$, $X \sim Weib(\lambda = 5, k = 0.7)$

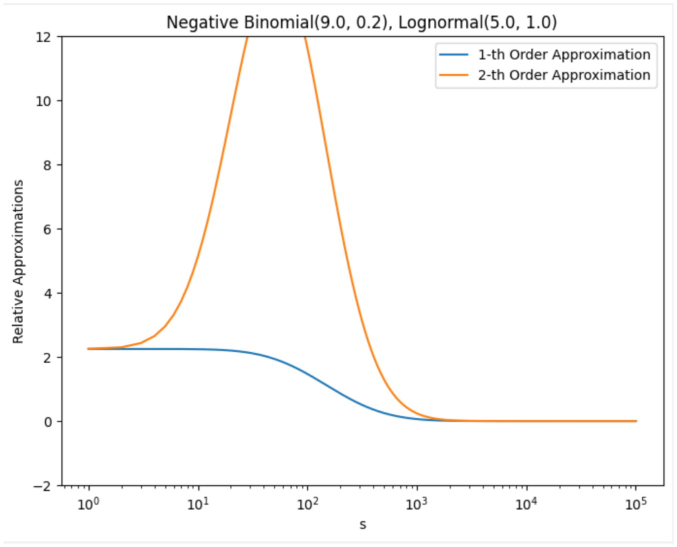


Fig. 9. $\bar{G}(s)$ for $N \sim NBin(r = 9, p = 0.2)$, $X \sim LN(\mu = 5, \sigma = 1)$

formula. This result emphasizes how crucial it is to use higher-order approximations. According to the results in Albrecher et al. (2010) [1], there exist situations where the higher-order approximations outperform the first-order approximation significantly. In some situations, however, there may be little to no improvement over the first-order approximation from the higher-order approximations.

It thus seems that the effectiveness of the higher-order approximations varies among different claim frequency and/or severity distributions specifications. In some case, they can provide notable improvements to first- and second-order approximations, but in others, their benefits might be limited or even negligible. This implies that the marginal gains might not always outweigh the additional computational effort in certain situations.

In summary, this section demonstrates the practical application of higher-order approximations and their comparative performance.

VII. Conclusion

In this project we have used higher-order approximations to tackle the computationally difficult problem of estimating the tail distribution $\bar{G}(s)$ of compound sums $S_N = \sum_{i=1}^{N} X_i$. The making of a Python-based implementation of these approximations, together with the user-friendly interface, can help in risk assessment, especially for applications using heavy-tailed distributions.

A. *Summary of Contributions*

The first-order to higher-order approximations used in this project only have been made easy to use for practitioners, while at the same time optimizing for computational efficiency and accuracy. Key contributions include:

- **Comprehensive Approximation Techniques**:Estimates of tail probabilities can be made with increasing accuracy by using first-order to sixth-order approximations. Understanding extreme event risks across a variety of disciplines, including environmental science, insurance, and finance, can be of great help.
- **Advanced Implementation Framework**: Our implementation is accurate and efficient since it integrates custom PDF and CDF functions together with JIT compilation from `numba`. The complexity of higher-order calculations is efficiently handled by the combination of symbolic and numerical approaches, where the symbolic evaluations have been even further optimized using Monte Carlo simulations.
- **User-Friendly Interface**: The user interface makes the use of the sophisticated statistical approximation methods accessible to a wider audience, and is also easy to use thanks to the graphical interface. The interactive feedback

with the user when declaring distributions, to interactive visualizations of the approximations, it helps users close the gap between intricate mathematical models and real-world applications.
- **Implications for Risk Assessment**: By providing accurate estimates of tail probabilities, our approach enhances risk assessment capabilities, particularly for rare but significant events. This improvement is vital for effective decision-making in sectors where extreme events have substantial financial or environmental impacts.

B. *Future Work and Recommendations*

During this project several points have been identified to improve our current framework even more:

- **Panjer Recursion Adaptation**: Distributions with a minimum value larger than zero, like the Pareto and Fréchet distributions, are not supported by the current Panjer recursion implementation. Future research should concentrate on modifying the Panjer recursion to handle these situations or investigating different numerical techniques to approximate the aggregate loss distribution with more accuracy.
- **Distribution Fitting Capabilities**: The usefulness of the approximation methods would be greatly increased by adding features which fit the best distribution approximation to user data. This way, the user could be able to upload his own data, and the algorithm would automatically fit distributions on the empirical user data. As a whole, this streamlines the process of parameter estimation, and also improves the practical abilities of our framework.
- **Incorporating Additional Approximations**: Albrecher et al. (2010) [1] provide us with more approximation methods, such as asymptotic expansions with a shifted argument, known to be used in practice. By adding such approximation methods, users would have access to an even greater variety of tools for studying the tail of compound sum distributions.
- **Optimizing Computational Efficiency**: More work is required to make the implementation as efficient as possible, especially when working with symbolic evaluations in higher-order approximations. Although the Monte Carlo simulations were a good temporary solution, they are not extremely precise, and when working with tails of extreme events, precision is of utmost importance.

In conclusion, this project offers a reliable and useful tool for calculating tail distributions of compound sums. The interface allows practitioners with different degrees of understanding of complex statistical methods to use the higher-order approximations to make useful risk-assessment decisions. From the further recommendations, it is clear that the current framework still has some important shortcomings, emphasizing the importance of further improving and broadening the framework and its methods. Regarding the usefulness of the use of higher-order approximation methods for the practitioners, we cannot draw firm conclusions. In certain situations, the first-order approximation is significantly improved by higher-order approximations, whereas in other situations these higher-order approximations provide little to no improvement. This raises doubts on the worth of additional computational effort. The practitioner will have to decide for him-or herself what is deemed optimal.

References

[1] Albrecher, H., Hipp, C., & Kortschak, D. (2010). Higher-order expansions for compound distributions and ruin probabilities with subexponential claims. Scandinavian Actuarial Journal, 2010(2), 105–135. https://doi.org/10.1080/03461230902722726

[2] Greenwood, P. (1973). Asymptotics of randomly stopped sequences with independent increments. *Annals of Probability*, 1, 317–321.

[3] Teugels, J. & Veraverbeke, N. (1973). Crame´r-type estimates for the probability of ruin. *C.O.R.E. Discussion Paper No. 7316*.

[4] von Bahr, B. (1975). Asymptotic ruin probabilities when exponential moments do not exist. *Scandinavian Actuarial Journal*, 6–10.

[5] Embrechts, P. & Veraverbeke, N. (1982). Estimates for the probability of ruin with special emphasis on the possibility of large claims. *Insurance Mathematics and Economics*, 1(1), 55–72.

[6] Grübel, R. (1987). On subordinated distributions and generalized renewal measures. *Annals of Probability*, 15(1), 394–415.

[7] Omey, E. & Willekens, E. (1987). Second-order behaviour of distributions subordinate to a distribution with finite mean. *Communications in Statistics – Stochastic Models*, 3(3), 311–342.

[8] Willekens, E. (1989). Asymptotic approximations of compound distributions and some applications. *Bulletin de la Société Mathématique de Belgique, Série B*, 41(1), 55–61.

[9] Baltrunas, A. (1999b). Second order behaviour of ruin probabilities. *Scandinavian Actuarial Journal*, (2), 120–133.

[10] Mikosch, T. & Nagaev, A. (2001). Rates in approximations to ruin probabilities for heavy-tailed distributions. *Extremes*, 4(1), 67–78.

[11] Geluk, J. L., Peng, L. & de Vries, C. G. (2000). Convolutions of heavy-tailed random variables and applications to portfolio diversification and MA(1) time series. *Advances in Applied Probability*, 32(4), 1011–1026.

[12] Borovkov, A. A. & Borovkov, K. A. (2002). On probabilities of large deviations for random walks. I. Regularly varying distribution tails. *Theory of Probability and its Applications*, 46(2), 193–213.

[13] Barbe, P. & McCormick, W. P. (2009). Asymptotic expansions for infinite weighted convolutions of heavy tail distributions and applications. *Memoirs of the American Mathematical Society*, 197.

[14] Barbe, P., McCormick, W. P. & Zhang, C. (2007). Asymptotic expansions for distributions of compound sums of random variables with rapidly varying subexponential distribution. *Journal of Applied Probability*, 44(3), 670–684.

[15] Panjer, H. H. (1981). Recursive evaluation of a family of compound distributions. ASTIN Bulletin: The Journal of the IAA, 12(1), 22-26.

[16] Abate, J., Choudhury, G. L. & Whitt, W. (1994). Waiting-time tail probabilities in queues with long-tail servicetime distributions. Queueing Systems Theory and Applications 16 (3-4), 311-338.

Appendix

I will shortly go over which helper-tool I've use, and why/what for.

A. *ChatGPT, GPT-4o mini*

I used ChatGPT to help build my interface. There exist many different functions in the `tkinter` package, each in turn having different parameters, which it was easy to get lost in. Working with ChatGPT helped me understand many of these functions and corresponding parameters way faster and also provided me with suggestions for them.

Similarly, to help create the README.md file in the Github repository, I used the help of ChatGPT.