STAT906: Rare Event Simulation with Heavy Tail Distribution Project

Ambrose Emmett-Iwaniw 20669099

Due: Dec 18th, 2020

Introduction:

In general the problem is using simulation to estimate small probabilities and in this project the problem we have to solve is ways to estimate $G(x) = \Pr(S_n > x)$ by simulation where S_n is the sum of n positive independent and identically random variables $X_1, \cdots X_n$ that are Heavy tailed in the special case of when x is large which implies G(x) is small. The reason for using simulation in this case is because standard algorithms for light tailed distributions fails due to the fact some distributions do not have exponential moments. Applications: Telecommunications in the case of service times that are distributed as an heavy tailed distribution in a Queueing model which was discussed as an example in the paper (1) that had improvements over standard approaches. Insurance Risk in the case of a ruin probability with a given reserve in the case of heavy tailed distributed resulting from the results of a claim, Financial Mathematics in for example Operational risk modeling such as Compound Poisson sums with independent summation terms, etc. In these application areas solutions which are easily computable or even explicit are only present in the light tailed distribution case.

In this project the algorithms used are mostly from the paper (1) which covered how to solve the problem of calculating rare event probabilities using methods that were concerned with using Variance Reduction Techniques of Conditional Monte Carlo and Importance Sampling in the case where n is replaced by N a positive random variable independent of the Y_i's.
For the numerical study of M/G/1 queue where n is replaced by a geometric r.v. N with the X_i's following either Weibull or Pareto distributions. They proved polynomial time estimators under Weibull and Pareto distributions. They also explored further improvements in CMC in random N case by incorporating Control Variates and Stratication Variance Reduction Techniques
The resulting simulations investigated showed Conditional Monte Carlo conditioning on $X_1, \cdots X_{n-1}$ is the best estimator for solving rare event probabilities under a heavy tailed distribution.

What I will being doing in this project is looking at using Variance Reduction Techniques I learned in class and some of the variations of the Variance Reduction Techniques referenced in the paper (1) with respect to a numerical study of the problem of Compound Poisson Process as was discussed as an application in Financial Mathematics and the numerical experiments to be performed would be on an example stated in the Paper (1) but wasn't investigated is the Compound Poisson Process:

$$X(t) = \sum_{i=1}^{N(t)} X_i$$

Where X_i's are independent and identically distributed and $N(t)$ is a Poisson Process (i.e. $N(t) \sim \text{Poisson}(\lambda t)$) Where we want to simulate using X_i's as either Weibull or Pareto distributions where we want to compute the following probability as efficient as possible:

$$\Pr(X(t) > x)$$

What are the Heavy-Tailed Distributions?

As was described in both the papers (1) and (2) the tail of pareto distribution (a regularly varying distribution as referenced in (1) and (2)) with parameter α is:

$$\bar{F}(x) = (1+x)^{-\alpha}$$

the tail of Weibull distribution with parameter β is:

$$\bar{H}(x) = e^{-x^{\beta}}$$

where $F(x) = 1 - \frac{1}{(1+x)^{\alpha}}$ is the CDF of a Pareto with parameter α
where $H(x) = 1 - e^{-x^{\beta}}$ is the CDF of a Weibull with parameter β

The Simulation Setup For Compound Poisson Process

For the Compound Poisson Process the number of random variables added together (aka jumps) follows a Poisson(λt) and so in this case we can hold $\lambda = 1$ since the Poisson parameter is λt and so depends on the values of both λ and t so changing t and holding constant λ at 1 will have a similar effect of changing λt and changing the value of x the reserve we will call it has on the probabilities. Since the goal is to calculate rare events under a heavy-tailed distribution we can simulate rare events by taking $t = 5, 10, 15$ which models in these cases summing over 5, 10 and 15 random variables on average since Poisson's mean is equal to the parameter and $x = 50, 75, 100$ which were reserves I found to give low probabilities for the event $X(t) > x$. For Pareto will take $\alpha = 1.5, 1.75$ since for $\alpha \leq 2$ we have for a given Pareto random variable its variance is infinite and for $\alpha > 1$ we have finite mean and the reason for choosing α this way is that looking at an α less than or equal to 1 we would have a mean infinite which when simulating made me choose reserves much bigger than the ones I am using in this project. And will use $n = 10^5, 10^6$ to see how the effect of number of simulations n has on the result.

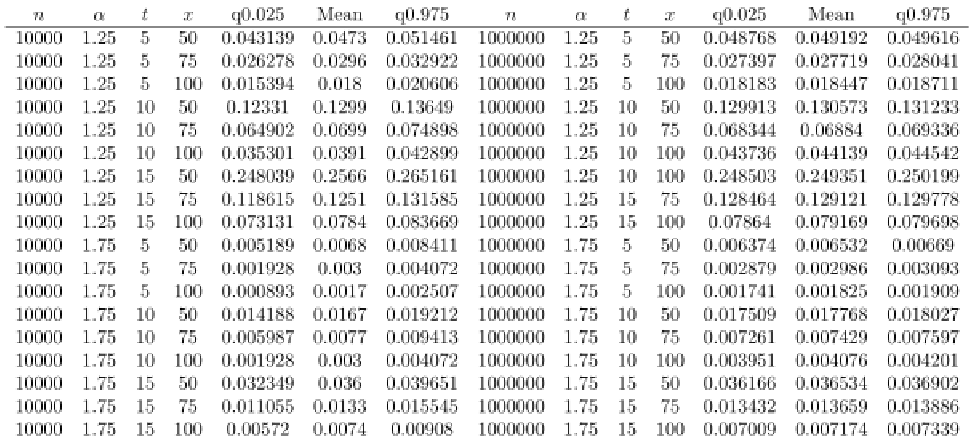
In the paper (1) for their example they used M/G/1 queuing model which they used a fact for extreme values approximation to have the correct reserve to have a small probability but in this case can't find or think of a similar trick to find proper reserves so will use Naive Monte Carlo as a base to use to compare how well other algorithms are compared to Naive Monte Carlo.

Naive MC Simulation - Setup Of Simulation

The Naive MC simulation estimator of $G(x)$ for n simulations is:

$$\hat{\mu}_{MC} = \frac{1}{n} \sum_{i=1}^{n} I(X_i(t) > x)$$

where $X_i(t) = \sum_{j=1}^{N_i(t)} X_{ij}$ and $N_i(t)$ $\sim$Poisson(t) are independent and identically distributed for $i = 1, \cdots n$ and X_{ij} are independent and identically distributed either Pareto or Weibull distributed.



n	α	t	x	q0.025	Mean	q0.975	n	α	t	x	q0.025	Mean	q0.975
10000	1.25	5	50	0.043139	0.0473	0.051461	1000000	1.25	5	50	0.048768	0.049192	0.049616
10000	1.25	5	75	0.026278	0.0296	0.032922	1000000	1.25	5	75	0.027397	0.027719	0.028041
10000	1.25	5	100	0.015394	0.018	0.020606	1000000	1.25	5	100	0.018183	0.018447	0.018711
10000	1.25	10	50	0.12331	0.1299	0.13649	1000000	1.25	10	50	0.129913	0.130573	0.131233
10000	1.25	10	75	0.064902	0.0699	0.074898	1000000	1.25	10	75	0.068344	0.06884	0.069336
10000	1.25	10	100	0.035301	0.0391	0.042899	1000000	1.25	10	100	0.043736	0.044139	0.044542
10000	1.25	15	50	0.248039	0.2566	0.265161	1000000	1.25	10	100	0.248503	0.249351	0.250199
10000	1.25	15	75	0.118615	0.1251	0.131585	1000000	1.25	15	75	0.128464	0.129121	0.129778
10000	1.25	15	100	0.073131	0.0784	0.083669	1000000	1.25	15	100	0.07864	0.079169	0.079698
10000	1.75	5	50	0.005189	0.0068	0.008411	1000000	1.75	5	50	0.006374	0.006532	0.00669
10000	1.75	5	75	0.001928	0.003	0.004072	1000000	1.75	5	75	0.002879	0.002986	0.003093
10000	1.75	5	100	0.000893	0.0017	0.002507	1000000	1.75	5	100	0.001741	0.001825	0.001909
10000	1.75	10	50	0.014188	0.0167	0.019212	1000000	1.75	10	50	0.017509	0.017768	0.018027
10000	1.75	10	75	0.005987	0.0077	0.009413	1000000	1.75	10	75	0.007261	0.007429	0.007597
10000	1.75	10	100	0.001928	0.003	0.004072	1000000	1.75	10	100	0.003951	0.004076	0.004201
10000	1.75	15	50	0.032349	0.036	0.039651	1000000	1.75	15	50	0.036166	0.036534	0.036902
10000	1.75	15	75	0.011055	0.0133	0.015545	1000000	1.75	15	75	0.013432	0.013659	0.013886
10000	1.75	15	100	0.00572	0.0074	0.00908	1000000	1.75	15	100	0.007009	0.007174	0.007339

Figure 1: Naive MC for Pareto distribution 95% CI

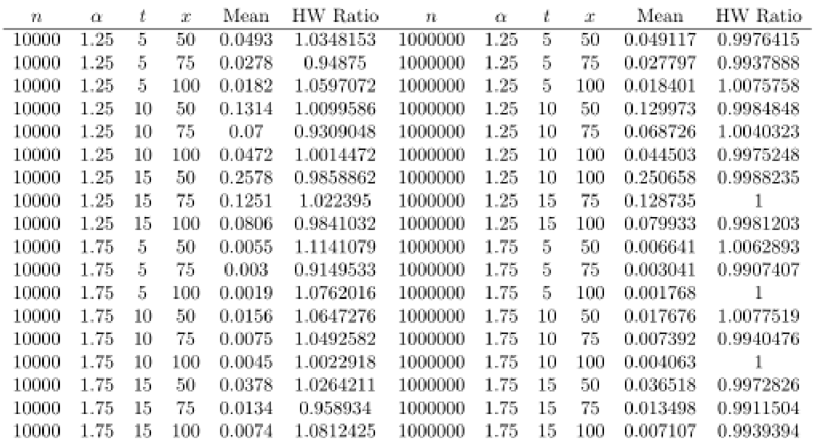
As was referenced in paper (1) the Naive Monte Carlo simulation of $G(x)$ becomes ever worse as $G(x)$ becomes smaller making the simulation a problem that needs Variance Reduction Techniques to improve efficiency as can be seen with the next simulation experiment.

Antithetic Variates - First Standard VRT To Try

The Antivariate Variates simulation estimator of $G(x)$ for n simulations is:

$$\hat{\mu}_{ant} = \frac{1}{n/2} \sum_{i=1}^{n/2} \frac{Y_i + \tilde{Y}_i}{2}$$

where $Y_i = I(X_i(t) > x), \tilde{Y}_i = I(\tilde{X}_i(t) > x)$, $X_i(t) = \sum_{j=1}^{N_i(t)} X_{ij}$ and $N_i(t)$ ~Poisson(t) are independent and identically distributed and X_{ij} are independent and identically distributed. for Pareto distribution. but $\tilde{Y}_i$ are generated by the same uniform random variable in the form 1 - the uniforms. This Variation Reduction Technique can't be used since in the i^{th} case the number of dimensions of a uniform random variable is random and will be different for Y_i and $\tilde{Y}_i$ so can't necessarily create that dependency in each of the dimensions, which is an assumption this VRT is based on. So Antivariate Variates is bad in this case. To get around this issue will hold Poisson random variables constant and see how Antivariate Variates does in improving the reduction which the dependency we are adding should give an improvement.



n	α	t	x	Mean	HW Ratio	n	α	t	x	Mean	HW Ratio
10000	1.25	5	50	0.0493	1.0348153	1000000	1.25	5	50	0.049117	0.9976415
10000	1.25	5	75	0.0278	0.94875	1000000	1.25	5	75	0.027797	0.9937888
10000	1.25	5	100	0.0182	1.0597072	1000000	1.25	5	100	0.018401	1.0075758
10000	1.25	10	50	0.1314	1.0099586	1000000	1.25	10	50	0.129973	0.9984848
10000	1.25	10	75	0.07	0.9309048	1000000	1.25	10	75	0.068726	1.0040323
10000	1.25	10	100	0.0472	1.0014472	1000000	1.25	10	100	0.044503	0.9975248
10000	1.25	15	50	0.2578	0.9858862	1000000	1.25	10	100	0.250658	0.9988235
10000	1.25	15	75	0.1251	1.022395	1000000	1.25	15	75	0.128735	1
10000	1.25	15	100	0.0806	0.9841032	1000000	1.25	15	100	0.079933	0.9981203
10000	1.75	5	50	0.0055	1.1141079	1000000	1.75	5	50	0.006641	1.0062893
10000	1.75	5	75	0.003	0.9149533	1000000	1.75	5	75	0.003041	0.9907407
10000	1.75	5	100	0.0019	1.0762016	1000000	1.75	5	100	0.001768	1
10000	1.75	10	50	0.0156	1.0647276	1000000	1.75	10	50	0.017676	1.0077519
10000	1.75	10	75	0.0075	1.0492582	1000000	1.75	10	75	0.007392	0.9940476
10000	1.75	10	100	0.0045	1.0022918	1000000	1.75	10	100	0.004063	1
10000	1.75	15	50	0.0378	1.0264211	1000000	1.75	15	50	0.036518	0.9972826
10000	1.75	15	75	0.0134	0.958934	1000000	1.75	15	75	0.013498	0.9911504
10000	1.75	15	100	0.0074	1.0812425	1000000	1.75	15	100	0.007107	0.9939394

Figure 2: AV for Pareto Ratio = MC HW / AV HW

As was discussed some improvement over Monte Carlo but overall looks like no real significant improvement most likely due to the Theorem in notes where $I(X(t) > x)$ isn't a monotone function in each of it's arguments so the Covariance between $I(X(t) > x)$ and $I(\tilde{X}(t) > x)$ isn't negative.

Conditional MC Using Order Statistics - Setup Of Simulation

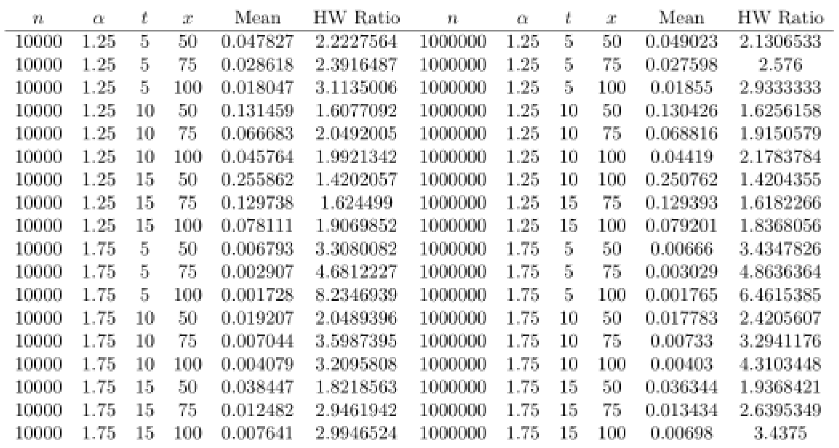
The Conditional Monte Carlo conditioned on the order statistics simulation estimator of $G(x)$ for n simulations as described in paper (1) is:

$$\hat{\mu}_{CMC} = \frac{1}{n}\sum_{i=1}^{n} \Pr(X_i(t) > x | X_{i,(1)} \cdots , X_{i,(N_i(t)-1)})$$

where $X_i(t) = \sum_{j=1}^{N_i(t)} X_{i,j}$ and $N_i(t)$ ∼Poisson(t) are independent and identically distributed for $i = 1, \cdots , n$ and $X_{i,j}$ are independent and identically distributed Pareto distributed. Idea: Generate $N_i(t)$ random variables form the order statistics and remove the largest random variables also can write this estimator as follows:

$$\hat{\mu}_{CMC} = \frac{1}{n}\sum_{i=1}^{n} \frac{\bar{F}(\max\{x - X_i(t)^*, X_{i,(N_i(t)-1)}\})}{\bar{F}(X_{i,(N_i(t)-1)})}$$

Where $X_i(t)^* = X_{i,(1)} + \cdots + X_{i,(N_i(t)-1)} = X_i(t) - X_{i,(N_i(t))}$



n	α	t	x	Mean	HW Ratio	n	α	t	x	Mean	HW Ratio
10000	1.25	5	50	0.047827	2.2227564	1000000	1.25	5	50	0.049023	2.1306533
10000	1.25	5	75	0.028618	2.3916487	1000000	1.25	5	75	0.027598	2.576
10000	1.25	5	100	0.018047	3.1135006	1000000	1.25	5	100	0.01855	2.9333333
10000	1.25	10	50	0.131459	1.6077092	1000000	1.25	10	50	0.130426	1.6256158
10000	1.25	10	75	0.066683	2.0492005	1000000	1.25	10	75	0.068816	1.9150579
10000	1.25	10	100	0.045764	1.9921342	1000000	1.25	10	100	0.04419	2.1783784
10000	1.25	15	50	0.255862	1.4202057	1000000	1.25	10	100	0.250762	1.4204355
10000	1.25	15	75	0.129738	1.624499	1000000	1.25	15	75	0.129393	1.6182266
10000	1.25	15	100	0.078111	1.9069852	1000000	1.25	15	100	0.079201	1.8368056
10000	1.75	5	50	0.006793	3.3080082	1000000	1.75	5	50	0.00666	3.4347826
10000	1.75	5	75	0.002907	4.6812227	1000000	1.75	5	75	0.003029	4.8636364
10000	1.75	5	100	0.001728	8.2346939	1000000	1.75	5	100	0.001765	6.4615385
10000	1.75	10	50	0.019207	2.0489396	1000000	1.75	10	50	0.017783	2.4205607
10000	1.75	10	75	0.007044	3.5987395	1000000	1.75	10	75	0.00733	3.2941176
10000	1.75	10	100	0.004079	3.2095808	1000000	1.75	10	100	0.00403	4.3103448
10000	1.75	15	50	0.038447	1.8218563	1000000	1.75	15	50	0.036344	1.9368421
10000	1.75	15	75	0.012482	2.9461942	1000000	1.75	15	75	0.013434	2.6395349
10000	1.75	15	100	0.007641	2.9946524	1000000	1.75	15	100	0.00698	3.4375

Figure 3: CMC for Pareto Ratio = MC HW / CMC with RQMC HW

From what can be seen this estimator produces a good reduction in the variance compared to Monte Carlo by about a factor of 2 which at this point is better than what Antithetic variances has done. This is expected given that the Conditional Monte Carlo conditioned on the order statistics in the literature is regarded as was proclaimed in paper (1) to be the first polynomial time algorithm to solve these rare event probabilities.

Conditional MC Using Most Efficient - Setup Of Simulation

This algorithm as referenced in paper (1) was supposed to be for their example and proved to be both efficient and give the largest reduction in variance but in the case of fixed value to be the number of random variables added together and the algorithm comes from the intuition that as was described in paper (2) on page 4 "the only way the sum can get large is by one of the summands getting large"
As given in paper (1) we have the identity:

$$\Pr(X_i(t) > x) = N_i(t) \Pr(X_i(t) > x, X_{i,(N_i(t))} = X_{i,N_i(t)})$$

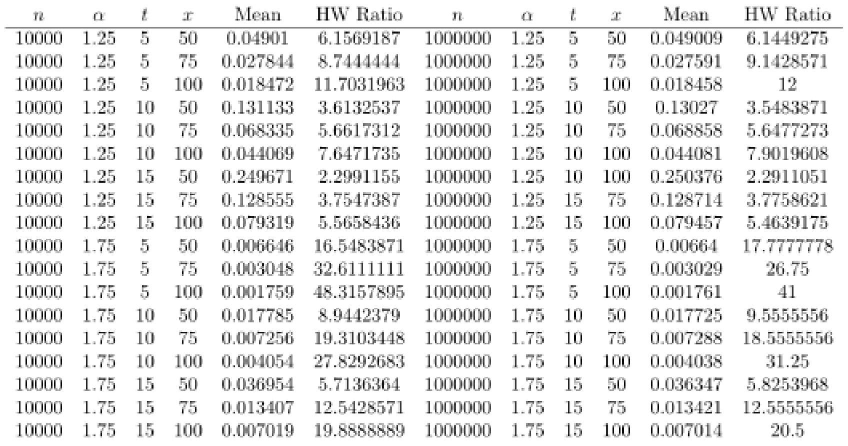
The Conditional MC simulation most efficient estimator of $G(x)$ for n simulations as described in (1) is:

$$\hat{\mu}_{CMC} = \frac{1}{n} \sum_{i=1}^{n} N_i(t) \Pr(X_i(t) > x, X_{i,(N_i(t))} = X_{i,N_i(t)} | X_1 \cdots X_{N_i(t)-1})$$

where $X_i(t) = \sum_{j=1}^{N_i(t)} X_{i,j}$ and $N_i(t)$ ~Poisson(t) are independent and identically distributed for $i = 1, \cdots, n$ and $X_{i,j}$ are independent and identically distributed Pareto distributed. Also can write this estimator as follows:

$$\hat{\mu}_{CMC} = \frac{1}{n} \sum_{i=1}^{n} N_i(t) \bar{F}(\max\{X_{i,(N_i(t)-1)}, x - S_{N_i(t)-1}\})$$

Where $S_{N_i(t)-1} = X_{i,1} + \cdots + X_{i,N_i(t)-1} = X_i(t) - X_{i,N_i(t)}$ and $X_{i,(N_i(t)-1)} = \max\{X_{i,1}, \cdots, X_{i,N_i(t)-1}\}$



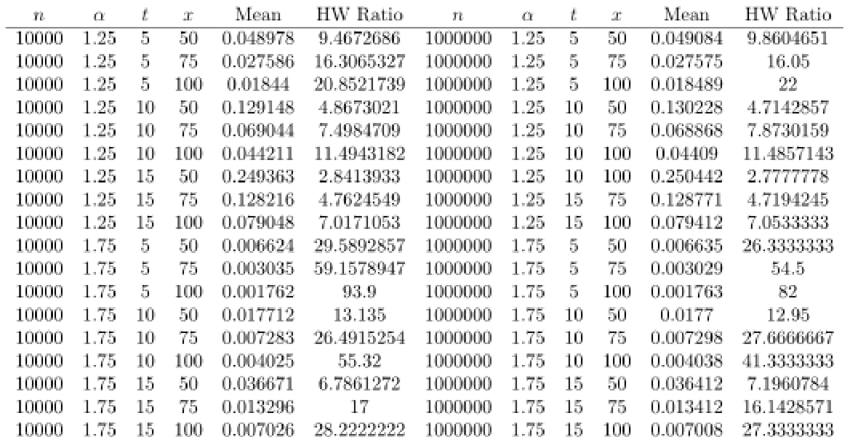
n	α	t	x	Mean	HW Ratio	n	α	t	x	Mean	HW Ratio
10000	1.25	5	50	0.04901	6.1569187	1000000	1.25	5	50	0.049009	6.1449275
10000	1.25	5	75	0.027844	8.7444444	1000000	1.25	5	75	0.027591	9.1428571
10000	1.25	5	100	0.018472	11.7031963	1000000	1.25	5	100	0.018458	12
10000	1.25	10	50	0.131133	3.6132537	1000000	1.25	10	50	0.13027	3.5483871
10000	1.25	10	75	0.068335	5.6617312	1000000	1.25	10	75	0.068858	5.6477273
10000	1.25	10	100	0.044069	7.6471735	1000000	1.25	10	100	0.044081	7.9019608
10000	1.25	15	50	0.249671	2.2991155	1000000	1.25	10	100	0.250376	2.2911051
10000	1.25	15	75	0.128555	3.7547387	1000000	1.25	15	75	0.128714	3.7758621
10000	1.25	15	100	0.079319	5.5658436	1000000	1.25	15	100	0.079457	5.4639175
10000	1.75	5	50	0.006646	16.5483871	1000000	1.75	5	50	0.00664	17.7777778
10000	1.75	5	75	0.003048	32.6111111	1000000	1.75	5	75	0.003029	26.75
10000	1.75	5	100	0.001759	48.3157895	1000000	1.75	5	100	0.001761	41
10000	1.75	10	50	0.017785	8.9442379	1000000	1.75	10	50	0.017725	9.5555556
10000	1.75	10	75	0.007256	19.3103448	1000000	1.75	10	75	0.007288	18.5555556
10000	1.75	10	100	0.004054	27.8292683	1000000	1.75	10	100	0.004038	31.25
10000	1.75	15	50	0.036954	5.7136364	1000000	1.75	15	50	0.036347	5.8253968
10000	1.75	15	75	0.013407	12.5428571	1000000	1.75	15	75	0.013421	12.5555556
10000	1.75	15	100	0.007019	19.8888889	1000000	1.75	15	100	0.007014	20.5

Figure 4: CMC Efficient for Pareto Ratio = MC HW / CMC HW

As can be seen this algorithm does very well compared to the even standard one for polynomial time in the literature. This algorithm as was described before supposedly work better under a fixed value to be the number of random variables added together, let us see if for the random variable N that it is true by combining this with Stratification will further lower variance.

Efficient CMC Using Stratification - Setup Of Simulation

To remove the variability produced by $N(t)$ we will use Stratication with proportional allocation and for $t = 5$ will divide into 8 strata $N_i = i$ for $i = 0, \cdots 7$, and $N_8 > 7$ and $t = 10$ will divide into 15 strata $N_i = i$ for $i = 0, \cdots 14$, and $N_{15} > 15$ and $t = 15$ will divide into 21 strata $N_i = i$ for $i = 0, \cdots 20$, and $N_{21} > 20$. These values were chosen so to evenly cover most of the domain so that the last strate has about a probability of 8 percent.



n	α	t	x	Mean	HW Ratio	n	α	t	x	Mean	HW Ratio
10000	1.25	5	50	0.048978	9.4672686	1000000	1.25	5	50	0.049084	9.8604651
10000	1.25	5	75	0.027586	16.3065327	1000000	1.25	5	75	0.027575	16.05
10000	1.25	5	100	0.01844	20.8521739	1000000	1.25	5	100	0.018489	22
10000	1.25	10	50	0.129148	4.8673021	1000000	1.25	10	50	0.130228	4.7142857
10000	1.25	10	75	0.069044	7.4984709	1000000	1.25	10	75	0.068868	7.8730159
10000	1.25	10	100	0.044211	11.4943182	1000000	1.25	10	100	0.04409	11.4857143
10000	1.25	15	50	0.249363	2.8413933	1000000	1.25	10	100	0.250442	2.7777778
10000	1.25	15	75	0.128216	4.7624549	1000000	1.25	15	75	0.128771	4.7194245
10000	1.25	15	100	0.079048	7.0171053	1000000	1.25	15	100	0.079412	7.0533333
10000	1.75	5	50	0.006624	29.5892857	1000000	1.75	5	50	0.006635	26.3333333
10000	1.75	5	75	0.003035	59.1578947	1000000	1.75	5	75	0.003029	54.5
10000	1.75	5	100	0.001762	93.9	1000000	1.75	5	100	0.001763	82
10000	1.75	10	50	0.017712	13.135	1000000	1.75	10	50	0.0177	12.95
10000	1.75	10	75	0.007283	26.4915254	1000000	1.75	10	75	0.007298	27.6666667
10000	1.75	10	100	0.004025	55.32	1000000	1.75	10	100	0.004038	41.3333333
10000	1.75	15	50	0.036671	6.7861272	1000000	1.75	15	50	0.036412	7.1960784
10000	1.75	15	75	0.013296	17	1000000	1.75	15	75	0.013412	16.1428571
10000	1.75	15	100	0.007026	28.2222222	1000000	1.75	15	100	0.007008	27.3333333

Figure 5: CMC combined with STR for Pareto Ratio = MC HW / CMC with STR HW

Which from the resulting table the Stratification Variance Reduction Technique improves upon the Conditional Monte Carlo Efficient version. Would expect a similar result for Conditional Monte Carlo conditioned on the order statistics to prove. This makes sense from the paper (1)'s authors' reasoning since in this case we are summing over a random number of random variables and using Stratification to reduce the variability in the Poisson random variable and seeing the improvement indicates a significant part of the variability was caused by the Poisson random variable.

Efficient CMC Using Control Variates - Setup Of Simulation

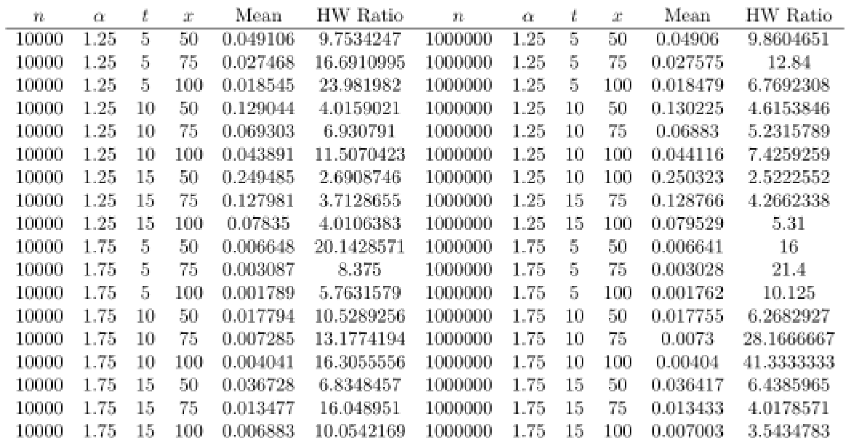
To remove the variability produced by $N(t)$ we will use Control Variates by having N(t) be our control variate since it is correlated to the poisson compound process because the sum depends on a Poisson random variable and $E[N(t)] = t$ which therefore is known. So our Control Variate estimator is:

$$\hat{\mu}_{CV} = \frac{1}{n}\sum_{i=1}^{n}(Y_i + \beta(\mu_{N_i(t)} - N_i(t))$$

where $Y_i = N_i(t)\bar{F}(\max\{X_{i,(N_i(t)-1)}, x - S_{N_i(t)-1}\})$
$S_{N_i(t)-1} = X_{i,1} + \cdots + X_{i,N_i(t)-1} = X_i(t) - X_{i,N_i(t)}$ and $X_{i,(N_i(t)-1)} = \max\{X_{i,1}, \cdots, X_{i,N_i(t)-1}\}$
as was the estimator of Conditional Monte Carlo efficient version and $X_i(t) = \sum_{j=1}^{N_i(t)} X_{i,j}$ and $N_i(t)$ ~Poisson(t) are independent and identically distributed for $i = 1, \cdots, n$ and $X_{i,j}$ are independent and identically distributed Pareto distributed. And the β coefficient is estimated as was described in class notes on Control Variates slide 4 with 1000 pilot runs as follows:

$$\hat{\beta} = \frac{\sum_{i=1}^{n} Y_i N_i(t) - n(\hat{\mu_{mc}\mu_{N_i(t)}})}{(n-1)\sigma^2_{N_i(t)}}$$

where $\sigma^2_{N_i(t)} = Var(N_i(t)) = t$



n	α	t	x	Mean	HW Ratio	n	α	t	x	Mean	HW Ratio
10000	1.25	5	50	0.049106	9.7534247	1000000	1.25	5	50	0.04906	9.8604651
10000	1.25	5	75	0.027468	16.6910995	1000000	1.25	5	75	0.027575	12.84
10000	1.25	5	100	0.018545	23.981982	1000000	1.25	5	100	0.018479	6.7692308
10000	1.25	10	50	0.129044	4.0159021	1000000	1.25	10	50	0.130225	4.6153846
10000	1.25	10	75	0.069303	6.930791	1000000	1.25	10	75	0.06883	5.2315789
10000	1.25	10	100	0.043891	11.5070423	1000000	1.25	10	100	0.044116	7.4259259
10000	1.25	15	50	0.249485	2.6908746	1000000	1.25	10	100	0.250323	2.5222552
10000	1.25	15	75	0.127981	3.7128655	1000000	1.25	15	75	0.128766	4.2662338
10000	1.25	15	100	0.07835	4.0106383	1000000	1.25	15	100	0.079529	5.31
10000	1.75	5	50	0.006648	20.1428571	1000000	1.75	5	50	0.006641	16
10000	1.75	5	75	0.003087	8.375	1000000	1.75	5	75	0.003028	21.4
10000	1.75	5	100	0.001789	5.7631579	1000000	1.75	5	100	0.001762	10.125
10000	1.75	10	50	0.017794	10.5289256	1000000	1.75	10	50	0.017755	6.2682927
10000	1.75	10	75	0.007285	13.1774194	1000000	1.75	10	75	0.0073	28.1666667
10000	1.75	10	100	0.004041	16.3055556	1000000	1.75	10	100	0.00404	41.3333333
10000	1.75	15	50	0.036728	6.8348457	1000000	1.75	15	50	0.036417	6.4385965
10000	1.75	15	75	0.013477	16.048951	1000000	1.75	15	75	0.013433	4.0178571
10000	1.75	15	100	0.006883	10.0542169	1000000	1.75	15	100	0.007003	3.5434783

Figure 6: CMC combined with CV for Pareto Ratio = MC HW / CMC with CV HW

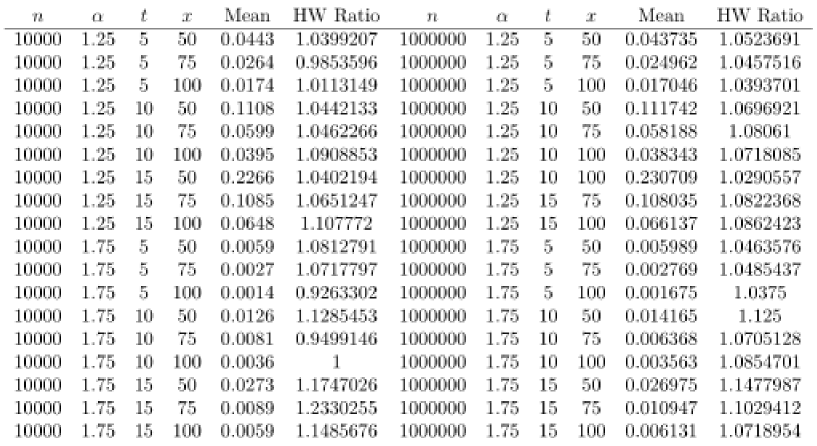
Which from the resulting table the Control Variate Variance Reduction Technique improves upon the Conditional Monte Carlo Efficient version. Would expect a similar result for Conditional Monte Carlo conditioned on the order statistics to hold. This makes sense from the paper (1)'s authors' reasoning since in this case we take into account the correlation between Y_i and $N_i(t)$ to reduce the variability in the Poisson random variable and seeing the improvement indicates a significant part of the variability was caused by the Poisson random variable. Also from these experiments found stratification and Control variates with Efficient Conditional Monte Carlo produce similar reduction results to each other which is what the authors of paper (1) was also drawn.

RQMC - Simulation Setup

The Randomized Quasi Monte Carlo simulation estimator of $G(x)$ for n simulations is:

$$\hat{\mu}_{MC} = \frac{1}{n}\sum_{i=1}^{n} I(X_i(t) > x)$$

where $X_i(t) = \sum_{j=1}^{N_i(t)} X_{ij}$ and $N_i(t)$ $\sim$Poisson(t) are independent and identically distributed for $i = 1, \cdots n$ and X_{ij} are independent and identically distributed Pareto distributed but using Sobol with digital-shift to have low-discrepancy in the uniform variables used to generate these random variables to add dependencies to lower the variance. As can be seen The Randomized Quasi Monte



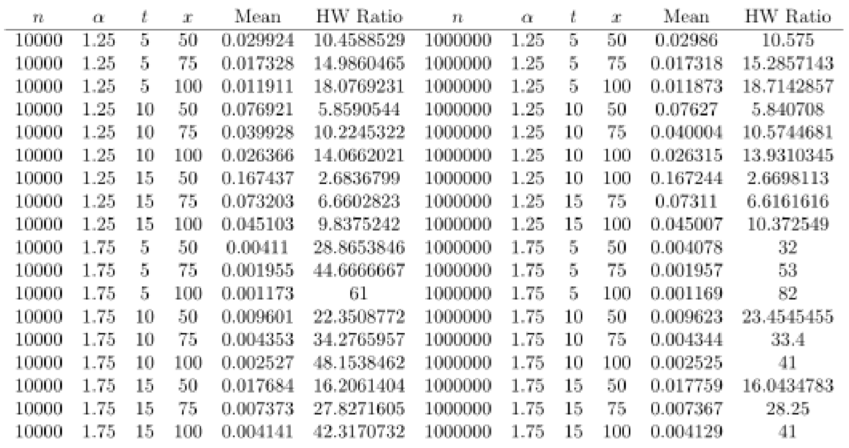
n	α	t	x	Mean	HW Ratio	n	α	t	x	Mean	HW Ratio
10000	1.25	5	50	0.0443	1.0399207	1000000	1.25	5	50	0.043735	1.0523691
10000	1.25	5	75	0.0264	0.9853596	1000000	1.25	5	75	0.024962	1.0457516
10000	1.25	5	100	0.0174	1.0113149	1000000	1.25	5	100	0.017046	1.0393701
10000	1.25	10	50	0.1108	1.0442133	1000000	1.25	10	50	0.111742	1.0696921
10000	1.25	10	75	0.0599	1.0462266	1000000	1.25	10	75	0.058188	1.08061
10000	1.25	10	100	0.0395	1.0908853	1000000	1.25	10	100	0.038343	1.0718085
10000	1.25	15	50	0.2266	1.0402194	1000000	1.25	10	100	0.230709	1.0290557
10000	1.25	15	75	0.1085	1.0651247	1000000	1.25	15	75	0.108035	1.0822368
10000	1.25	15	100	0.0648	1.107772	1000000	1.25	15	100	0.066137	1.0862423
10000	1.75	5	50	0.0059	1.0812791	1000000	1.75	5	50	0.005989	1.0463576
10000	1.75	5	75	0.0027	1.0717797	1000000	1.75	5	75	0.002769	1.0485437
10000	1.75	5	100	0.0014	0.9263302	1000000	1.75	5	100	0.001675	1.0375
10000	1.75	10	50	0.0126	1.1285453	1000000	1.75	10	50	0.014165	1.125
10000	1.75	10	75	0.0081	0.9499146	1000000	1.75	10	75	0.006368	1.0705128
10000	1.75	10	100	0.0036	1	1000000	1.75	10	100	0.003563	1.0854701
10000	1.75	15	50	0.0273	1.1747026	1000000	1.75	15	50	0.026975	1.1477987
10000	1.75	15	75	0.0089	1.2330255	1000000	1.75	15	75	0.010947	1.1029412
10000	1.75	15	100	0.0059	1.1485676	1000000	1.75	15	100	0.006131	1.0718954

Figure 7: RQMC for Pareto Ratio = MC HW / RQMC HW

Carlo estimator produces a better reduction over Monte Carlo but when running I noticed it took long to compute which is a huge drawback over Monte Carlo most likely caused to be inefficient because of Sobol sequences being generated are relatively less efficient than sampling straight from the random variable itself, also couldn't use the m repetitions approach seen in class since running time was to long but would expect it would create a slightly better improvement over Monte Carlo. Can still combined this version with other Variance Reduction Techniques to see the improvement like as follows next combining with Conditional Monte Carlo to see whether there is a significant improvement.

CMC with order statistics combined with RQMC

From previous implementation of Randomized Quasi Monte Carlo I want to see whether this can improve upon the standard Variance Reduction Technique for simulation rare events using heavy tailed distributions, Conditional Monte Carlo conditioning on the order statistics.



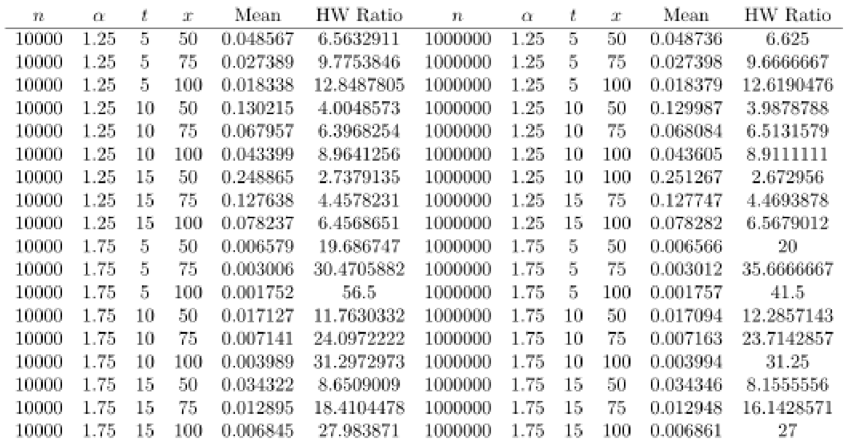
n	α	t	x	Mean	HW Ratio	n	α	t	x	Mean	HW Ratio
10000	1.25	5	50	0.029924	10.4588529	1000000	1.25	5	50	0.02986	10.575
10000	1.25	5	75	0.017328	14.9860465	1000000	1.25	5	75	0.017318	15.2857143
10000	1.25	5	100	0.011911	18.0769231	1000000	1.25	5	100	0.011873	18.7142857
10000	1.25	10	50	0.076921	5.8590544	1000000	1.25	10	50	0.07627	5.840708
10000	1.25	10	75	0.039928	10.2245322	1000000	1.25	10	75	0.040004	10.5744681
10000	1.25	10	100	0.026366	14.0662021	1000000	1.25	10	100	0.026315	13.9310345
10000	1.25	15	50	0.167437	2.6836799	1000000	1.25	10	100	0.167244	2.6698113
10000	1.25	15	75	0.073203	6.6602823	1000000	1.25	15	75	0.07311	6.6161616
10000	1.25	15	100	0.045103	9.8375242	1000000	1.25	15	100	0.045007	10.372549
10000	1.75	5	50	0.00411	28.8653846	1000000	1.75	5	50	0.004078	32
10000	1.75	5	75	0.001955	44.6666667	1000000	1.75	5	75	0.001957	53
10000	1.75	5	100	0.001173	61	1000000	1.75	5	100	0.001169	82
10000	1.75	10	50	0.009601	22.3508772	1000000	1.75	10	50	0.009623	23.4545455
10000	1.75	10	75	0.004353	34.2765957	1000000	1.75	10	75	0.004344	33.4
10000	1.75	10	100	0.002527	48.1538462	1000000	1.75	10	100	0.002525	41
10000	1.75	15	50	0.017684	16.2061404	1000000	1.75	15	50	0.017759	16.0434783
10000	1.75	15	75	0.007373	27.8271605	1000000	1.75	15	75	0.007367	28.25
10000	1.75	15	100	0.004141	42.3170732	1000000	1.75	15	100	0.004129	41

Figure 8: CMC with RQMC for Pareto Ratio = MC HW / CMC with RQMC HW

As can be seen combining the Conditional Monte Carlo method conditioned on Order Statistics with Randomized Quasi Monte Carlo creates a huge reduction in the variance compared to the Monte Carlo. For even one it is 61 times better! this is because as can be seen by other values caused by the fact for low probabilities the Monte Carlo method Half width becomes large. This comes at no surprise considering how well Conditional Monte Carlo method conditioned on Order Statistics and Randomized Quasi Monte Carlo did on their own as was expected to happen.

CMC Efficient combined with RQMC

From previous implementation of Randomized Quasi Monte Carlo I want to see whether this can improve upon the Variance Reduction Technique for simulation rare events using heavy tailed distributions, Conditional Monte Carlo efficient version.



n	α	t	x	Mean	HW Ratio	n	α	t	x	Mean	HW Ratio
10000	1.25	5	50	0.048567	6.5632911	1000000	1.25	5	50	0.048736	6.625
10000	1.25	5	75	0.027389	9.7753846	1000000	1.25	5	75	0.027398	9.6666667
10000	1.25	5	100	0.018338	12.8487805	1000000	1.25	5	100	0.018379	12.6190476
10000	1.25	10	50	0.130215	4.0048573	1000000	1.25	10	50	0.129987	3.9878788
10000	1.25	10	75	0.067957	6.3968254	1000000	1.25	10	75	0.068084	6.5131579
10000	1.25	10	100	0.043399	8.9641256	1000000	1.25	10	100	0.043605	8.9111111
10000	1.25	15	50	0.248865	2.7379135	1000000	1.25	10	100	0.251267	2.672956
10000	1.25	15	75	0.127638	4.4578231	1000000	1.25	15	75	0.127747	4.4693878
10000	1.25	15	100	0.078237	6.4568651	1000000	1.25	15	100	0.078282	6.5679012
10000	1.75	5	50	0.006579	19.686747	1000000	1.75	5	50	0.006566	20
10000	1.75	5	75	0.003006	30.4705882	1000000	1.75	5	75	0.003012	35.6666667
10000	1.75	5	100	0.001752	56.5	1000000	1.75	5	100	0.001757	41.5
10000	1.75	10	50	0.017127	11.7630332	1000000	1.75	10	50	0.017094	12.2857143
10000	1.75	10	75	0.007141	24.0972222	1000000	1.75	10	75	0.007163	23.7142857
10000	1.75	10	100	0.003989	31.2972973	1000000	1.75	10	100	0.003994	31.25
10000	1.75	15	50	0.034322	8.6509009	1000000	1.75	15	50	0.034346	8.1555556
10000	1.75	15	75	0.012895	18.4104478	1000000	1.75	15	75	0.012948	16.1428571
10000	1.75	15	100	0.006845	27.983871	1000000	1.75	15	100	0.006861	27

Figure 9: Efficient CMC with RQMC for Pareto Ratio = MC HW / CMC with RQMC HW

As can be seen combining the Efficient Conditional Monte Carlo method with Randomized Quasi Monte Carlo creates a huge reduction in the variance compared to the Monte Carlo. For even one it is 61 times better! this is because as can be seen by other values caused by the fact for low probabilities the Monte Carlo method Half width becomes large. This comes at no surprise considering how well the efficient Conditional Monte Carlo method and Randomized Quasi Monte Carlo did on their own as was expected to happen.

IS Using Hazard Rate Twisting as in (3) - Setup Of Simulation

Importance Sampling is very established for calculating rare event simulations but in the heavy tail case we run into problems using basic exponential change of measure since the needed exponential moments do not exist. Based on the paper (3) page 8-9 we have the following Importance Sampling Hazard Rate Twisting. So we have the hazard rate twisted probability density function:

$$f_\theta(x) = (1-\theta)\lambda(x)e^{-(1-\theta)\Lambda(x)}, x \geq 0$$

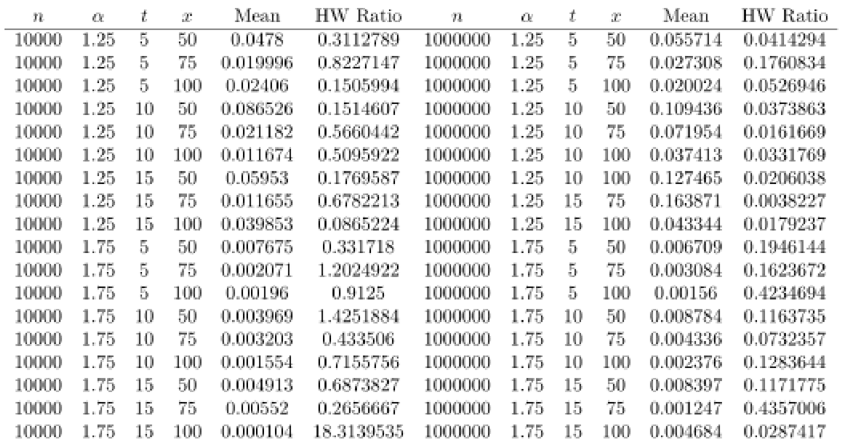
where $\lambda(x) = \frac{f(x)}{\bar{F}(x)}, \Lambda(x) = \int_0^x \lambda(y)\,dy, \theta$ some constant. We have for the Pareto case we have $f(x) = \frac{\alpha}{(1+x)^{\alpha+1}}, \bar{F}(x) = (1+x)^{-\alpha}, \lambda(x) = \frac{\alpha}{(1+x)}, \Lambda(x) = \alpha\log(1+x)$, so $f(x) = \lambda(x)e^{-\Lambda(x)}$ also from Thm 3.2 in (3) page 9 it states setting $\theta = 1 - b/\Lambda(u)$ where b is any constant is the asymptotically efficient value for estimating $\hat{\mu}_{IS}$ for reserve x that is:

$$\hat{\mu}_{IS} = \frac{1}{n}\sum_{i=1}^{n}\prod_{j=1}^{N_i(t)}\frac{f(X_{ij})}{f_\theta(X_{ij})}I(X_i(t) > x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{(1-\theta)^{N_i(t)}}e^{-\theta\sum_{j=1}^{N_i(t)}\Lambda(X_{ij})}I(X_i(t) > x)$$

where $X_i(t) = \sum_{j=1}^{N_i(t)} X_{ij}$ and $N_i(t)$ ~Poisson(t) are independent and identically distributed for $i = 1, \cdots n$ and X_{ij} are independent and identically distributed sampled from distribution of $f_\theta(.)$. Take $b = 1$ then find $F_\theta(.)$ so to generate from $f_\theta(.)$ by inversion:

$$F_\theta(x) = \int_0^x (1-\theta)\lambda(y)e^{-(1-\theta)\Lambda(y)}\,dy = 1 - e^{-(1-\theta)\Lambda(x)}$$

so using inversion $F_\theta(x) = U$, where $U \sim UNIF(0,1)$ we get $\frac{-\log(1-U)}{(1-\theta)} = \Lambda(x)$ and since $\Lambda(x) = \alpha\log(1+x)$ we therefore get: $x = e^{\frac{-\log(1-U)}{\alpha(1-\theta)}} - 1$



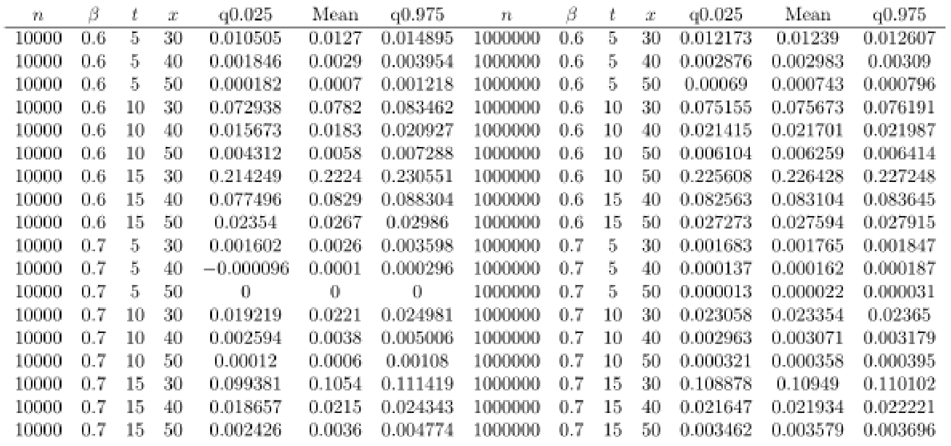
n	α	t	x	Mean	HW Ratio	n	α	t	x	Mean	HW Ratio
10000	1.25	5	50	0.0478	0.3112789	1000000	1.25	5	50	0.055714	0.0414294
10000	1.25	5	75	0.019996	0.8227147	1000000	1.25	5	75	0.027308	0.1760834
10000	1.25	5	100	0.02406	0.1505994	1000000	1.25	5	100	0.020024	0.0526946
10000	1.25	10	50	0.086526	0.1514607	1000000	1.25	10	50	0.109436	0.0373863
10000	1.25	10	75	0.021182	0.5660442	1000000	1.25	10	75	0.071954	0.0161669
10000	1.25	10	100	0.011674	0.5095922	1000000	1.25	10	100	0.037413	0.0331769
10000	1.25	15	50	0.05953	0.1769587	1000000	1.25	10	100	0.127465	0.0206038
10000	1.25	15	75	0.011655	0.6782213	1000000	1.25	15	75	0.163871	0.0038227
10000	1.25	15	100	0.039853	0.0865224	1000000	1.25	15	100	0.043344	0.0179237
10000	1.75	5	50	0.007675	0.331718	1000000	1.75	5	50	0.006709	0.1946144
10000	1.75	5	75	0.002071	1.2024922	1000000	1.75	5	75	0.003084	0.1623672
10000	1.75	5	100	0.00196	0.9125	1000000	1.75	5	100	0.00156	0.4234694
10000	1.75	10	50	0.003969	1.4251884	1000000	1.75	10	50	0.008784	0.1163735
10000	1.75	10	75	0.003203	0.433506	1000000	1.75	10	75	0.004336	0.0732357
10000	1.75	10	100	0.001554	0.7155756	1000000	1.75	10	100	0.002376	0.1283644
10000	1.75	15	50	0.004913	0.6873827	1000000	1.75	15	50	0.008397	0.1171775
10000	1.75	15	75	0.00552	0.2656667	1000000	1.75	15	75	0.001247	0.4357006
10000	1.75	15	100	0.000104	18.3139535	1000000	1.75	15	100	0.004684	0.0287417

Figure 10: IS Hazard Rate Twisting for Pareto

As can be seen using this version of Importance Sampling doesn't due well most likely due to a coding error that I made.

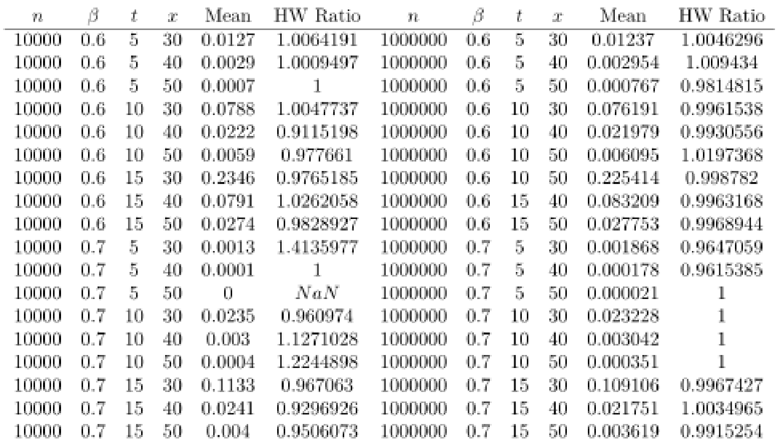
Running These algorithms on Weibull

we can simulate rare events for by taking $t = 5, 10, 15$ which models in these cases summing over 5, 10 and 15 random variables on average since Poisson's mean is equal to the parameter and $x = 30, 40, 50$ which were reserves I found to give low probabilities for the event $X(t) > x$. For Pareto will take $\beta = 0.6, 0.7$ since for $\beta < 1$ we have for a given Weibull random variable we have a decreasing failure rate over time. And will use $n = 10^5, 10^6$ to see how the effect of number of simulations n has on the result.



n	β	t	x	q0.025	Mean	q0.975	n	β	t	x	q0.025	Mean	q0.975
10000	0.6	5	30	0.010505	0.0127	0.014895	1000000	0.6	5	30	0.012173	0.01239	0.012607
10000	0.6	5	40	0.001846	0.0029	0.003954	1000000	0.6	5	40	0.002876	0.002983	0.00309
10000	0.6	5	50	0.000182	0.0007	0.001218	1000000	0.6	5	50	0.00069	0.000743	0.000796
10000	0.6	10	30	0.072938	0.0782	0.083462	1000000	0.6	10	30	0.075155	0.075673	0.076191
10000	0.6	10	40	0.015673	0.0183	0.020927	1000000	0.6	10	40	0.021415	0.021701	0.021987
10000	0.6	10	50	0.004312	0.0058	0.007288	1000000	0.6	10	50	0.006104	0.006259	0.006414
10000	0.6	15	30	0.214249	0.2224	0.230551	1000000	0.6	10	50	0.225608	0.226428	0.227248
10000	0.6	15	40	0.077496	0.0829	0.088304	1000000	0.6	15	40	0.082563	0.083104	0.083645
10000	0.6	15	50	0.02354	0.0267	0.02986	1000000	0.6	15	50	0.027273	0.027594	0.027915
10000	0.7	5	30	0.001602	0.0026	0.003598	1000000	0.7	5	30	0.001683	0.001765	0.001847
10000	0.7	5	40	−0.000096	0.0001	0.000296	1000000	0.7	5	40	0.000137	0.000162	0.000187
10000	0.7	5	50	0	0	0	1000000	0.7	5	50	0.000013	0.000022	0.000031
10000	0.7	10	30	0.019219	0.0221	0.024981	1000000	0.7	10	30	0.023058	0.023354	0.02365
10000	0.7	10	40	0.002594	0.0038	0.005006	1000000	0.7	10	40	0.002963	0.003071	0.003179
10000	0.7	10	50	0.00012	0.0006	0.00108	1000000	0.7	10	50	0.000321	0.000358	0.000395
10000	0.7	15	30	0.099381	0.1054	0.111419	1000000	0.7	15	30	0.108878	0.10949	0.110102
10000	0.7	15	40	0.018657	0.0215	0.024343	1000000	0.7	15	40	0.021647	0.021934	0.022221
10000	0.7	15	50	0.002426	0.0036	0.004774	1000000	0.7	15	50	0.003462	0.003579	0.003696

Figure 11: Naive MC for Weibull 95% CI



n	β	t	x	Mean	HW Ratio	n	β	t	x	Mean	HW Ratio
10000	0.6	5	30	0.0127	1.0064191	1000000	0.6	5	30	0.01237	1.0046296
10000	0.6	5	40	0.0029	1.0009497	1000000	0.6	5	40	0.002954	1.009434
10000	0.6	5	50	0.0007	1	1000000	0.6	5	50	0.000767	0.9814815
10000	0.6	10	30	0.0788	1.0047737	1000000	0.6	10	30	0.076191	0.9961538
10000	0.6	10	40	0.0222	0.9115198	1000000	0.6	10	40	0.021979	0.9930556
10000	0.6	10	50	0.0059	0.977661	1000000	0.6	10	50	0.006095	1.0197368
10000	0.6	15	30	0.2346	0.9765185	1000000	0.6	10	50	0.225414	0.998782
10000	0.6	15	40	0.0791	1.0262058	1000000	0.6	15	40	0.083209	0.9963168
10000	0.6	15	50	0.0274	0.9828927	1000000	0.6	15	50	0.027753	0.9968944
10000	0.7	5	30	0.0013	1.4135977	1000000	0.7	5	30	0.001868	0.9647059
10000	0.7	5	40	0.0001	1	1000000	0.7	5	40	0.000178	0.9615385
10000	0.7	5	50	0	NaN	1000000	0.7	5	50	0.000021	1
10000	0.7	10	30	0.0235	0.960974	1000000	0.7	10	30	0.023228	1
10000	0.7	10	40	0.003	1.1271028	1000000	0.7	10	40	0.003042	1
10000	0.7	10	50	0.0004	1.2244898	1000000	0.7	10	50	0.000351	1
10000	0.7	15	30	0.1133	0.967063	1000000	0.7	15	30	0.109106	0.9967427
10000	0.7	15	40	0.0241	0.9296926	1000000	0.7	15	40	0.021751	1.0034965
10000	0.7	15	50	0.004	0.9506073	1000000	0.7	15	50	0.003619	0.9915254

Figure 12: AV for Weibull Ratio = MC HW / AV HW

n	β	t	x	Mean	HW Ratio	n	β	t	x	Mean	HW Ratio
10000	0.6	5	30	0.012622	1.6553544	1000000	0.6	5	30	0.012427	1.6315789
10000	0.6	5	40	0.002767	2.1510204	1000000	0.6	5	40	0.002923	1.877193
10000	0.6	5	50	0.000777	1.738255	1000000	0.6	5	50	0.000753	2.0384615
10000	0.6	10	30	0.075858	1.3324892	1000000	0.6	10	30	0.076169	1.295
10000	0.6	10	40	0.020727	1.3308004	1000000	0.6	10	40	0.021716	1.4019608
10000	0.6	10	50	0.005632	1.5897436	1000000	0.6	10	50	0.006141	1.5346535
10000	0.6	15	30	0.228095	1.1878461	1000000	0.6	10	50	0.226345	1.1953353
10000	0.6	15	40	0.084719	1.2267877	1000000	0.6	15	40	0.083148	1.2436782
10000	0.6	15	50	0.028548	1.2495057	1000000	0.6	15	50	0.027677	1.3155738
10000	0.7	5	30	0.001835	1.8937381	1000000	0.7	5	30	0.001837	1.5769231
10000	0.7	5	40	0.000091	10.8888889	1000000	0.7	5	40	0.000171	1.7857143
10000	0.7	5	50	0.000008	0	1000000	0.7	5	50	0.000013	4.5
10000	0.7	10	30	0.022188	1.2942498	1000000	0.7	10	30	0.023401	1.2813853
10000	0.7	10	40	0.003364	1.4689403	1000000	0.7	10	40	0.003097	1.4025974
10000	0.7	10	50	0.000247	4.6601942	1000000	0.7	10	50	0.000367	1.5416667
10000	0.7	15	30	0.110162	1.1508604	1000000	0.7	15	30	0.109471	1.1746641
10000	0.7	15	40	0.022951	1.1990721	1000000	0.7	15	40	0.021871	1.2317597
10000	0.7	15	50	0.004056	1.2583065	1000000	0.7	15	50	0.003642	1.3

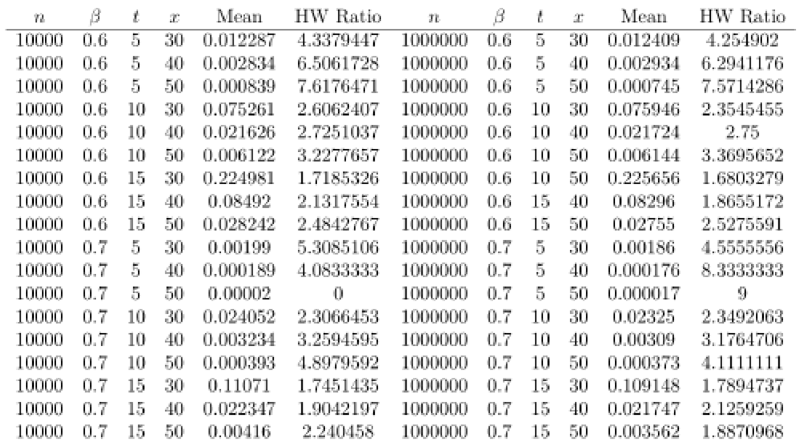
Figure 13: CMC Order Stats for Weibull Ratio = CMC with MC HW / RQMC HW

n	β	t	x	Mean	HW Ratio	n	β	t	x	Mean	HW Ratio
10000	0.6	5	30	0.012418	4.0573013	1000000	0.6	5	30	0.012406	4.0185185
10000	0.6	5	40	0.002964	6.0228571	1000000	0.6	5	40	0.002926	6.2941176
10000	0.6	5	50	0.000732	10.36	1000000	0.6	5	50	0.000748	8.8333333
10000	0.6	10	30	0.07778	2.0724695	1000000	0.6	10	30	0.075795	2.1316872
10000	0.6	10	40	0.021897	2.6428571	1000000	0.6	10	40	0.021764	2.86
10000	0.6	10	50	0.006267	3.5260664	1000000	0.6	10	50	0.00611	3.974359
10000	0.6	15	30	0.227407	1.5030426	1000000	0.6	10	50	0.22574	1.4909091
10000	0.6	15	40	0.083426	1.9075185	1000000	0.6	15	40	0.083	1.9116608
10000	0.6	15	50	0.026904	2.4901497	1000000	0.6	15	50	0.027655	2.4318182
10000	0.7	5	30	0.001691	5.9053254	1000000	0.7	5	30	0.001842	4.5555556
10000	0.7	5	40	0.000184	5.025641	1000000	0.7	5	40	0.000173	8.3333333
10000	0.7	5	50	0.000015	0	1000000	0.7	5	50	0.000017	9
10000	0.7	10	30	0.023504	2.1483967	1000000	0.7	10	30	0.023213	2.2424242
10000	0.7	10	40	0.00309	3.3041096	1000000	0.7	10	40	0.003108	3.1764706
10000	0.7	10	50	0.000337	7.0588235	1000000	0.7	10	50	0.00037	4.625
10000	0.7	15	30	0.108251	1.5948596	1000000	0.7	15	30	0.108944	1.5493671
10000	0.7	15	40	0.021532	2.0766983	1000000	0.7	15	40	0.021613	2.0647482
10000	0.7	15	50	0.00339	3.2164384	1000000	0.7	15	50	0.003549	2.7857143

Figure 14: CMC Efficient for Weibull Ratio = MC HW / CMC HW

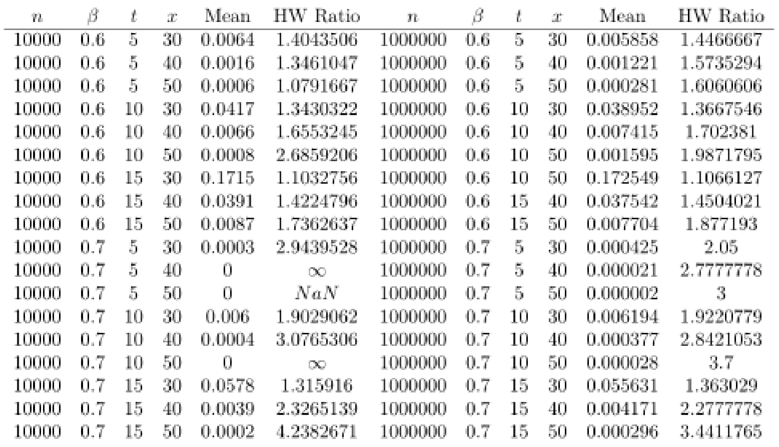
n	β	t	x	Mean	HW Ratio	n	β	t	x	Mean	HW Ratio
10000	0.6	5	30	0.012741	4.3039216	1000000	0.6	5	30	0.012441	4.5208333
10000	0.6	5	40	0.002906	6.8	1000000	0.6	5	40	0.002942	6.6875
10000	0.6	5	50	0.000748	10.1568627	1000000	0.6	5	50	0.000745	10.6
10000	0.6	10	30	0.075002	2.6336336	1000000	0.6	10	30	0.076163	2.5268293
10000	0.6	10	40	0.021414	2.9683616	1000000	0.6	10	40	0.021692	3.2134831
10000	0.6	10	50	0.006586	3.9055118	1000000	0.6	10	50	0.00613	4.1891892
10000	0.6	15	30	0.225014	1.7435294	1000000	0.6	10	50	0.226466	1.7596567
10000	0.6	15	40	0.082935	2.1914031	1000000	0.6	15	40	0.082821	2.2263374
10000	0.6	15	50	0.026153	2.8779599	1000000	0.6	15	50	0.027731	2.675
10000	0.7	5	30	0.001723	6.4387097	1000000	0.7	5	30	0.001844	4.5555556
10000	0.7	5	40	0.000146	11.5294118	1000000	0.7	5	40	0.000174	8.3333333
10000	0.7	5	50	0.000014	0	1000000	0.7	5	50	0.000017	9
10000	0.7	10	30	0.023357	2.3849338	1000000	0.7	10	30	0.023184	2.5299145
10000	0.7	10	40	0.003201	3.4261364	1000000	0.7	10	40	0.0031	3.375
10000	0.7	10	50	0.000353	8	1000000	0.7	10	50	0.000377	4.625
10000	0.7	15	30	0.110415	1.7395954	1000000	0.7	15	30	0.108982	1.8214286
10000	0.7	15	40	0.021932	2.0812592	1000000	0.7	15	40	0.021765	2.2421875
10000	0.7	15	50	0.003369	3.0572917	1000000	0.7	15	50	0.003558	2.8536585

Figure 15: CMC combined with STR for Weibull Ratio = MC HW / CMC with STR HW



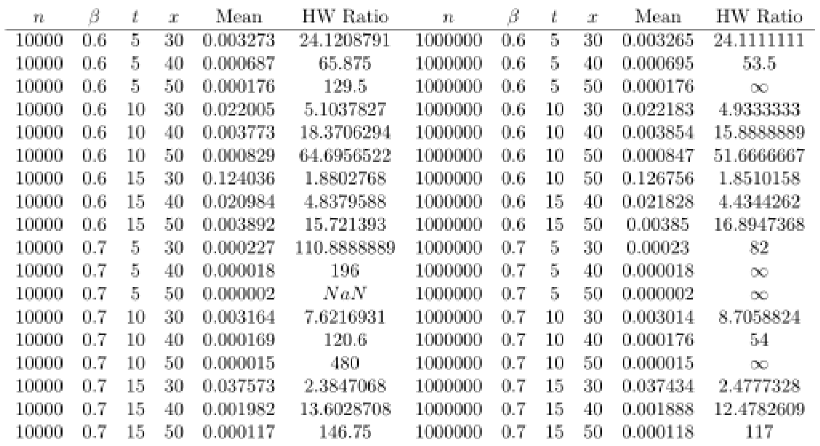
n	β	t	x	Mean	HW Ratio	n	β	t	x	Mean	HW Ratio
10000	0.6	5	30	0.012287	4.3379447	1000000	0.6	5	30	0.012409	4.254902
10000	0.6	5	40	0.002834	6.5061728	1000000	0.6	5	40	0.002934	6.2941176
10000	0.6	5	50	0.000839	7.6176471	1000000	0.6	5	50	0.000745	7.5714286
10000	0.6	10	30	0.075261	2.6062407	1000000	0.6	10	30	0.075946	2.3545455
10000	0.6	10	40	0.021626	2.7251037	1000000	0.6	10	40	0.021724	2.75
10000	0.6	10	50	0.006122	3.2277657	1000000	0.6	10	50	0.006144	3.3695652
10000	0.6	15	30	0.224981	1.7185326	1000000	0.6	10	50	0.225656	1.6803279
10000	0.6	15	40	0.08492	2.1317554	1000000	0.6	15	40	0.08296	1.8655172
10000	0.6	15	50	0.028242	2.4842767	1000000	0.6	15	50	0.02755	2.5275591
10000	0.7	5	30	0.00199	5.3085106	1000000	0.7	5	30	0.00186	4.5555556
10000	0.7	5	40	0.000189	4.0833333	1000000	0.7	5	40	0.000176	8.3333333
10000	0.7	5	50	0.00002	0	1000000	0.7	5	50	0.000017	9
10000	0.7	10	30	0.024052	2.3066453	1000000	0.7	10	30	0.02325	2.3492063
10000	0.7	10	40	0.003234	3.2594595	1000000	0.7	10	40	0.00309	3.1764706
10000	0.7	10	50	0.000393	4.8979592	1000000	0.7	10	50	0.000373	4.1111111
10000	0.7	15	30	0.11071	1.7451435	1000000	0.7	15	30	0.109148	1.7894737
10000	0.7	15	40	0.022347	1.9042197	1000000	0.7	15	40	0.021747	2.1259259
10000	0.7	15	50	0.00416	2.240458	1000000	0.7	15	50	0.003562	1.8870968

Figure 16: CMC combined with CV for Weibull Ratio = MC HW / CMC with CV HW



n	β	t	x	Mean	HW Ratio	n	β	t	x	Mean	HW Ratio
10000	0.6	5	30	0.0064	1.4043506	1000000	0.6	5	30	0.005858	1.4466667
10000	0.6	5	40	0.0016	1.3461047	1000000	0.6	5	40	0.001221	1.5735294
10000	0.6	5	50	0.0006	1.0791667	1000000	0.6	5	50	0.000281	1.6060606
10000	0.6	10	30	0.0417	1.3430322	1000000	0.6	10	30	0.038952	1.3667546
10000	0.6	10	40	0.0066	1.6553245	1000000	0.6	10	40	0.007415	1.702381
10000	0.6	10	50	0.0008	2.6859206	1000000	0.6	10	50	0.001595	1.9871795
10000	0.6	15	30	0.1715	1.1032756	1000000	0.6	10	50	0.172549	1.1066127
10000	0.6	15	40	0.0391	1.4224796	1000000	0.6	15	40	0.037542	1.4504021
10000	0.6	15	50	0.0087	1.7362637	1000000	0.6	15	50	0.007704	1.877193
10000	0.7	5	30	0.0003	2.9439528	1000000	0.7	5	30	0.000425	2.05
10000	0.7	5	40	0	∞	1000000	0.7	5	40	0.000021	2.7777778
10000	0.7	5	50	0	NaN	1000000	0.7	5	50	0.000002	3
10000	0.7	10	30	0.006	1.9029062	1000000	0.7	10	30	0.006194	1.9220779
10000	0.7	10	40	0.0004	3.0765306	1000000	0.7	10	40	0.000377	2.8421053
10000	0.7	10	50	0	∞	1000000	0.7	10	50	0.000028	3.7
10000	0.7	15	30	0.0578	1.315916	1000000	0.7	15	30	0.055631	1.363029
10000	0.7	15	40	0.0039	2.3265139	1000000	0.7	15	40	0.004171	2.2777778
10000	0.7	15	50	0.0002	4.2382671	1000000	0.7	15	50	0.000296	3.4411765

Figure 17: RQMC for Weibull Ratio = MC HW / RQMC HW



n	β	t	x	Mean	HW Ratio	n	β	t	x	Mean	HW Ratio
10000	0.6	5	30	0.003273	24.1208791	1000000	0.6	5	30	0.003265	24.1111111
10000	0.6	5	40	0.000687	65.875	1000000	0.6	5	40	0.000695	53.5
10000	0.6	5	50	0.000176	129.5	1000000	0.6	5	50	0.000176	∞
10000	0.6	10	30	0.022005	5.1037827	1000000	0.6	10	30	0.022183	4.9333333
10000	0.6	10	40	0.003773	18.3706294	1000000	0.6	10	40	0.003854	15.8888889
10000	0.6	10	50	0.000829	64.6956522	1000000	0.6	10	50	0.000847	51.6666667
10000	0.6	15	30	0.124036	1.8802768	1000000	0.6	10	50	0.126756	1.8510158
10000	0.6	15	40	0.020984	4.8379588	1000000	0.6	15	40	0.021828	4.4344262
10000	0.6	15	50	0.003892	15.721393	1000000	0.6	15	50	0.00385	16.8947368
10000	0.7	5	30	0.000227	110.8888889	1000000	0.7	5	30	0.00023	82
10000	0.7	5	40	0.000018	196	1000000	0.7	5	40	0.000018	∞
10000	0.7	5	50	0.000002	NaN	1000000	0.7	5	50	0.000002	∞
10000	0.7	10	30	0.003164	7.6216931	1000000	0.7	10	30	0.003014	8.7058824
10000	0.7	10	40	0.000169	120.6	1000000	0.7	10	40	0.000176	54
10000	0.7	10	50	0.000015	480	1000000	0.7	10	50	0.000015	∞
10000	0.7	15	30	0.037573	2.3847068	1000000	0.7	15	30	0.037434	2.4777328
10000	0.7	15	40	0.001982	13.6028708	1000000	0.7	15	40	0.001888	12.4782609
10000	0.7	15	50	0.000117	146.75	1000000	0.7	15	50	0.000118	117

Figure 18: CMC with RQMC for Weibull Ratio = MC HW / CMC with RQMC HW

n	β	t	x	Mean	HW Ratio	n	β	t	x	Mean	HW Ratio
10000	0.6	5	30	0.010307	6.5718563	1000000	0.6	5	30	0.010436	6.3823529
10000	0.6	5	40	0.002276	12.1149425	1000000	0.6	5	40	0.002282	11.8888889
10000	0.6	5	50	0.000563	19.9230769	1000000	0.6	5	50	0.000562	26.5
10000	0.6	10	30	0.067547	2.8864509	1000000	0.6	10	30	0.067445	2.9101124
10000	0.6	10	40	0.015737	5.032567	1000000	0.6	10	40	0.015867	5.2
10000	0.6	10	50	0.003911	8.9638554	1000000	0.6	10	50	0.003818	9.6875
10000	0.6	15	30	0.216685	1.9384067	1000000	0.6	10	50	0.219517	1.9385343
10000	0.6	15	40	0.068177	2.8487085	1000000	0.6	15	40	0.068368	2.8473684
10000	0.6	15	50	0.017941	4.8540707	1000000	0.6	15	50	0.018063	4.7910448
10000	0.7	5	30	0.001121	15.1212121	1000000	0.7	5	30	0.001141	11.7142857
10000	0.7	5	40	0.000093	21.7777778	1000000	0.7	5	40	0.00009	25
10000	0.7	5	50	0.000008	0	1000000	0.7	5	50	0.000008	∞
10000	0.7	10	30	0.014893	4.1935953	1000000	0.7	10	30	0.015073	4.1111111
10000	0.7	10	40	0.001295	12.5625	1000000	0.7	10	40	0.001298	10.8
10000	0.7	10	50	0.00011	48	1000000	0.7	10	50	0.000111	37
10000	0.7	15	30	0.089587	2.1496429	1000000	0.7	15	30	0.089452	2.2014388
10000	0.7	15	40	0.011639	4.2687688	1000000	0.7	15	40	0.011332	4.4153846
10000	0.7	15	50	0.001079	10.7706422	1000000	0.7	15	50	0.001081	11.7

Figure 19: Efficient CMC with RQMC for Weibull Ratio = MC HW / CMC with RQMC HW

As can be seen in Fig 11 At n = 10000, beta=0.7, t= 5, x = 50 has a problem of truncation to 0 so at this point Naive Monte Carlo is not a very good estimator to use to calculate this probability.
As can be seen in Fig 12 AV estimator has little to no affect on reduction as the reason was explained for Pareto.
As can be seen in Figure 13 Conditional Monte Carlo conditioned on Order Statistics improves upon Monte Carlo but not as much as for Pareto as is expected given this estimators works better for Regularly Varying distributions as discussed in (2).
As can be seen in Fig 14 the efficient Conditional Monte Carlo does the best like in the Pareto case.
As can be seen in Fig 15 efficient Conditional Monte Carlo combining with Stratification has a greater reduction in variance than efficient Conditional Monte Carlo on its own but not as significant as for Pareto case.
As can be seen in Fig 16 efficient Conditional Monte Carlo combining with Control Variates has a greater reduction in variance than efficient Conditional Monte Carlo on its own but not as significant as for Pareto case combining with Stratification as seen in Fig 15.
As can be seen in Fig 17, Randomized Quasi Monte Carlo as lower variance compared to Monte Carlo as expected and a greater effect as n increases.
As can be seen in Fig 18, Conditional Monte Carlo conditioned on Order Statistics combining with Randomized Quasi Monte Carlo reduces the Variance half width by a larger margin than any other estimator.
As can be seen in Fig 19 efficient Conditional Monte Carlo combining with Randomized Quasi Monte Carlo reduces the Variance half width by a larger margin than any other estimator seen before.

Conclusion

From looking at these experiments for both Pareto and Weibull distributions we have seen many Variance Reduction Techniques implemented but one that was show by example to be efficient in paper (1) was seen here to be the estimator that gave the greatest decrease in variance and combining it with RQMC, Control Variates, or Stratification saw a greater reduction in variance. As was also seen the standard approach to tail probabilities of using Importance Sampling wasn't very good compared to either of the Conditional Monte Carlo estimators

References

1 Asmussen, S. and Kroese, D (2006) Improved algorithms for rare event simulation with heavy tails. Advances in Appied Probability,38(2), 545-555
2 Asmussen, S., Binswanger, K. and HOjaard, B. (2000). Rare events simulation for heavy-tailed distributions. Bernoulli 6, 303-322
3. SANDEEP JUNEJA and PERWEZ SHAHABUDDIN. Simulating Heavy Tailed Processes Using Delayed Hazard Rate Twisting
4. Lognormal random walks
5. Pareto random variable Basic Facts
6. Levy Distribution to model price changes