Risk Based Approach to Calculate General Motor Insurance Reserve using High Performance Computing

Nikhil Rai[#1], Akhilesh Pandey[*2], Karam Rai[#3] , Pallav Kumar Baruah[#4], Satya Sai Mudigonda[#5] , Phani Krishna Kandala[#6]
[#]*Department of Mathematics and Computer Science*
Sri Sathya Sai Institute of Higher Learning, Puttaparthi, India

Abstract

Reserving calculation is a significant step in the strategic view of an insurance company. It is preformed periodically in order to show the realistic view of the future liabilities. Time required to calculate the reserves would grow exponentially depending on the size of input. Computation of reserves might need to be done several times by taking different factors into consideration. Hence the computation becomes even more costly in terms of time.

We applied HPC to calculate reserves using Risk based approach which is a combination of calculating a best estimate and risk margin surrounding this best estimate. Using GPUs we showed an improvement of 430X speed up compared to the serial execution for Risk based Inflation adjusted Chain Ladder Method

Keywords — *Accident year cohort, Chain ladder method (CLM), CUDA, Development year, Graphical Processing Unit (GPU), ,Risk based Inflation Adjusted CLM, Reserve.*

I. INTRODUCTION

Reserving is an important exercise to understand the realistic view of future liabilities. It would help us design the strategy for future business. It helps companies to ensure that they do not overstate and understate company's liabilities. This ensures transparency to various categories of stakeholders such as principals, agents, controllers, advisors and others including media and general public. Reserves help to achieve asset liability matching. In order to determine appropriate reserves, we have different actuarial methods mentioned below.

There are different methods for calculating motor in-surance reserves like Chain-Ladder method, Inflation adjusted Chain Ladder method, Bornhuetter-Ferguson method etc. which calculate the future liabilities.

The aforementioned methods does not take into account the risk margin for the future reserve of an insurance company. Risk Margin can be defined as the compensation required for transferring liability to another party. Under the European Union's Solvency II directive, risk margin represents the potential costs of transferring insurance obligation to a third party should an insurer fail. It is equal to an insurer's baseline solvency capital requirement for unhedgeable risks multiplied by the cost of capital at 6%.

In the recent years, the Cost of Capital Method (CoC) has gained popularity as a method to determine the value of so-called 'unhedgeable' risks. Unheadgeable risks are those risks that cannot be fully hedged with instruments traded in an active market. This is the case for various risks borne by insurers and pension funds, such as Longevity, Mortality, etc. The Risk Margin[1] according to the CoC method is generally calculated by the following steps:

1) Project the SCR, the Solvency Capital Requirement in all future periods of risk exposure.
2) Multiply the SCR by the Cost-of-Capital rate in each period.
3) Discount the amounts calculated under(2) using the risk free rate.

A. Motivation

Uncertainty involved in calculation of reserves and the amount of time it takes to calculate is a real world challenging scenario for most of the insurers. In order to achieve efficiency, High Performance Computing is utilized. It would address the above mentioned concerns by:

1) Performing parallelism for the time efficiency.

2) Producing a distribution of results, reducing the volatility in the results obtained.

It is mentioned in [2], HiPC provides us the Scalability and time and cost efficient use of the resources in reserving. During the recent years, HiPC has been applied in diverse fields of finance. Josh[3] priced Asian options and achieved a speed up of 150X. Nguyen[4] parallelized Cox-Ross-Rubinstein

pricing model on GPUs and showed a speed up of 30X. In 2012, Tucker and Bull[5] have explored HiPC to insurance solvency calculations and achieved a substantial improvement in performance over commercial software. Understanding the benefits of HiPC in calculating the reserve, we have proposed a new way of parallelizing the Chain-Ladder Method and Inflation Adjusted Chain Ladder Method for a best estimate calculation and cost of capital approach for calculation of risk margin.

The rest of the paper is organized into the following sections: Section 2 gives an idea about the Risk based Chain-Ladder and Risk based Inflation-Adjusted Methods. Section 3 talks about the proposed method of parallelization. Section 4 talks about the results and performance gain. Section 5 talks about the conclusion.

II. METHOD DESCRIPTION

In this method we give brief idea of the Risk Based Chain-Ladder Method and Risk Based Inflation Adjusted Chain-Ladder Method.

A. Risk Based Chain-Ladder Method

The Chain-Ladder Method is considered to be the most popular and simple model for estimation of outstanding claims and future reserve, both in theory and in practice [6]. It is used to estimate the incurred but not reported claims and project ultimate loss amounts [7]. Before going into the methodology of this model, let us try to understand some of the terms that are frequently used in these methods.

B. Run-Off Triangle

Run-off triangle (or delay triangles) are an important topic in the practical work of actuaries working in general insurance. It is used to forecast claim numbers and amounts. It usually arises in types of insurance (particularly non-life insurance) where it may take some time after a loss until the full extent of the claims which have to be paid is known. It is important that the claims are attributed to the year in which the policy was written. The insurance company needs to know how much it is liable to pay in claims so that it can calculate how much surplus it has made. There are many causes of the delays in the claims being finalized.

Let us think that, for every accident year, there are n numbers of claims which occurred during the period, but only x claims (x <= n) were reported to the insurer. The unreported claims are known as IBNR claims which we are trying to estimate using the risk-based chain ladder method and inflation-adjusted method. These methods use the historical or past data to obtain the future reported claims. Mathematically, the run-off triangle is expressed as follows:

Each entry Cij represents the incremental claims. It can be expressed as:

$$C_{ij} = r_j * s_i * x_{i+j} + e_{ij} \quad (1)$$

where r_j is the development factor for year j, representing the proportion of claim payments in development year j. Each r_j is independent of the origin year i. s_i is a parameter varying by origin year i, representing the exposure. x_{i+j} is a parameter varying by the calendar year. e_{ij} is an error term.

The run-off triangle is shown in Figure 1.

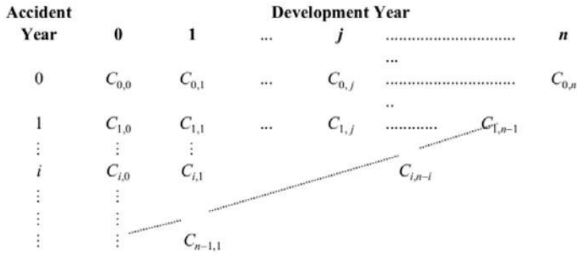

Figure 1. Run-off triangle

C. Chain-Ladder Method

Chain Ladder [8] as mentioned above, is a traditional method based on statistics and used for estimating the ultimate value of a set of development data. The main assumption and idea of the method are that present claims will approximately develop like past claims. This will be used to estimate the total reserve. The input for the chain ladder method will be an upper triangle and it predicts the lower triangle, which indicates the future claims. The input and output figure is shown in the following figure 2:

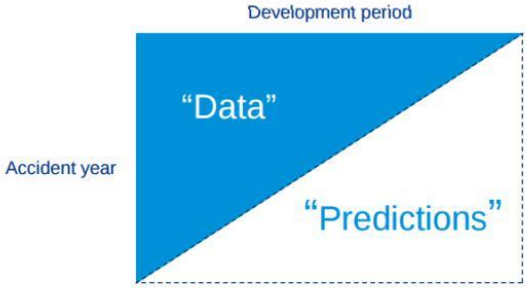

Figure 2.Input and output of Chain ladder method.

This chain ladder method works on the following assumptions:

1) The patterns of claims loss settlement observed in the past will continue in the future.
2) The estimates for settlement amounts in the future will be more accurate if all of the available data is used in the estimation.

Once the future reserves are calculated, it is used to calculate the risk margin taking into account of a discount rate. In previous work [2] a part of this method was already done. In this proposed work, we extend the work done in [2] to include the calculation of risk margin, ultimate, and reserve. We have also

proposed a novel method for implementing the algorithm parallely. We have used a discount rate of 3% and cost of capital approach for calculation of risk margin.

The Risk-Based Chain Ladder method is captured in Algorithm 1:

Algorithm 1: Risk Based Chain Ladder Method

Input: Input triangle in incremental form

Output: Reserve estimate, Ultimate, Risk Margin and Present value of Risk Margin

1 Step 1: (Find the cumulative sum):

2 The given incremental triangle is converted into cumulative triangle
3 Step 2: (Calculate the Development)
4 Find the development factors $_1$, $_2$,..., $_{n-1}$

5 Step 3:(Estimate the reserve):
6 Use the development factors calculated above and used it to calculate the reserve.
7 Step 4: (Find the Ultimate, Risk Margin and Present Value of Risk Margin)
8 From the above triangle calculate the Ultimate, Risk Margin and Present Value of Risk Margin
9 Step 5: (Output the Reserve Estimate, Ultimate, Risk Margin and Present Value of Risk Margin)

D. Risk Based Inflation Adjusted Chain Ladder Method

This method is similar to the above-mentioned method but it uses the historical inflation rate before calculating or predicting the future claims. The incremental run-off triangle is adjusted with the inflation rates, keeping one as inflation index. This method requires an appropriate inflation index to be available for the business being considered. The choice of index is a key to the success of reserving using this methods. This method takes into consideration the fact that claims inflation will affect the payments. Once the inflation rates are taken into consideration, it follows the same step of chain ladder method with an additional step after predicting the future reserves.

The Risk-Based inflation adjusted chain ladder method is captured in Algorithm2:

Input: Input triangle in incremental form, inflation factor

Output: Reserve estimate, Ultimate, Risk Margin and Present value of Risk Margin

1 Step 1: (Make adjustment to inflation):

2 The given input run-off triangle is adjusted to inflation by applying the inflation factors to non-cumulative data in order to get all the claims data into monetary terms of the recent accident year.

3 Step 2:(Find the cumulative run-off triangle)

4 Calculate the cumulative sum of the given reserves obtained after step 1. Step 3: (Calculate the Development)
5 Find the development factors $_1$, $_2$,..., $_{n-1}$

6 Step 4: (Estimate the future reserve):
7 Use the development factors calculated above and used it to calculate the reserve.
8 Step 5: (Calculate new incremental reserve)

9 Dis-accumulate the data to make it incremental.

10 Step 6:(Adjust to inflation):

11 By applying inflation assumptions made for future, the outstanding claim payments are converted into amounts corresponding to future years
12 Step 7:(Find the new cumulative reserve):

13 Accumulate the data
14 Step 8:(Find the Ultimate, Risk Margin and Present Value of Risk Margin)

15 From the above triangle calculate the Ultimate, Risk Margin and Present Value of Risk Margin
16 Step 9: (Output the Reserve Estimate, Ultimate, Risk Margin and Present Value of Risk Margin)

III. PROPOSE METHOD OF PARALLELIZATION

For our parallelization of above methods, we have used the benefits that are being provided by the GPU's structure. GPU provides the platform for computing the program parallely by executing different number of threads at the same time. For executing the program in GPU, it uses threads, blocks, and grids.

A thread can be defined as the smallest unit of program execution. A group of threads forms a block and the group of blocks forms a Grid as depicted in the Figure 3:

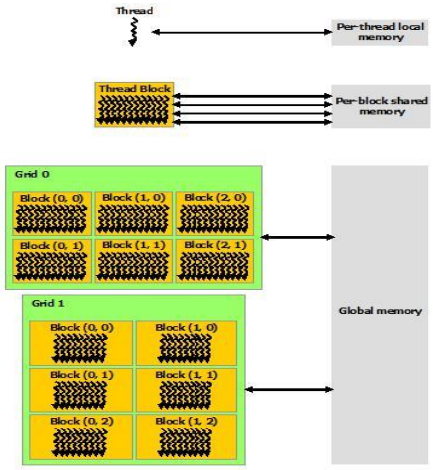

Figure 3.Thread, Block and Grid

A. Methodology

Let us visit the problem once again. Given the upper triangular matrix of size NxN as shown in Figure 4, where the cells marked blue, are non-zero, we wish to compute the lower triangular matrix (i.e. the cells marked green) using chain ladder method. In the previous sections a sequential method to solve the problem was described. However, the time taken by a sequential program is an exponential function of the input size N: So for a larger value of N time taken to compute the lower triangle is very large.

In order to complete the lower triangle, as an intermediate step, we need to compute the development factor (given by equation) using the columns of the upper triangular matrix. We find that each element of the development factor can be computed independently. Taking this as a motivation we designed a method to solve the problem using CUDA. Apart from computing the lower triangular matrix, we also computed the reserve and ultimate using CUDA.

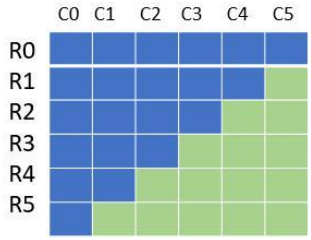

Figure 4.Given Matrix

1) Compute development factor: We use two temporary arrays to store the intermediate values, let us call them array1 and array2. Note that the size of array1 and array2 is N-1. The array1[i] stores the sum of all the elements in column i till the diagonal element whereas, array2[i] stores the sum of all the elements in the ith column like array1 but not including the diagonal element. We assign a thread to each element of array1 and array2. For example, in the above figure, the size of the matrix is 6x6. Thread 0(note that thread number starts from 0 and not 1) computes the sum of elements in column 0 i.e sum= (R0,C0) + (R1,C0) + (R2, C0) + (R3,C0) + (R4,C0), sum1= (R0,C0) + (R1,C0)

+ (R2, C0) + (R3,C0) + (R4,C0) + (R5, C0) and stores it to the arrays i.e. array1[0]= sum and array2[0]= sum1. Each thread from 0 to N-2 does the computation simultaneously except the last thread N-1 which only compute array2[N-1]= (R0, CN-1). We wait for each thread to complete its part using a CUDA keyword __syncthreads():

In order to compute the development factor, each thread 0 to N-2 (note that thread N-1 does not participate in this) compute its share using arrat1 and array2. In general, thread i (0 < i < N-2) compute dev[i]= array2[i+1]/array[i]. The idea is illustrated in the Figure 5:

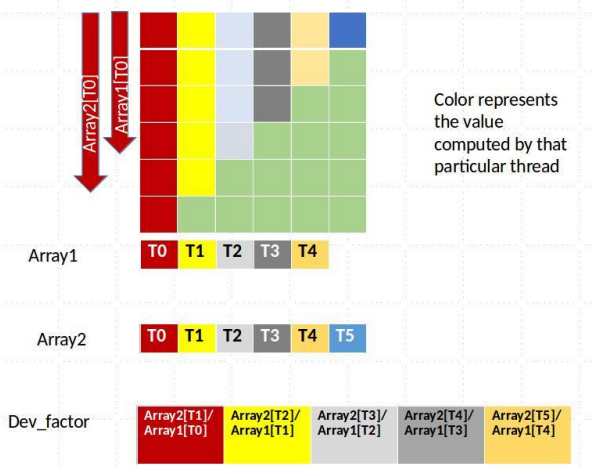

Figure 5. Computing the development factor using threads

2) Compute the lower triangular matrix: Having computed the development factor, our next step is to compute the lower triangular matrix along with the reserve, risk-margin and ultimate. We observe that each element in the row depends only on its previous element in that row and the development factor for that row. Therefore, each row can be assigned to one single thread. All the threads can execute on their part independently. The method is illustrated in the figure 6:

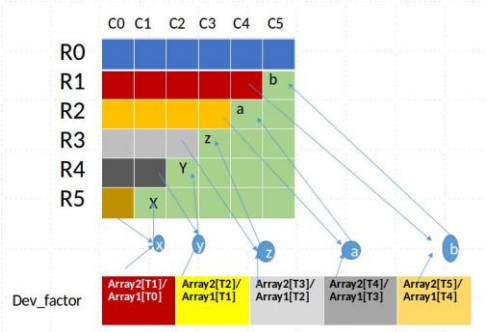

Figure 6. Computation of lower triangle using threads

3) Compute the Ultimate and Risk Margin: To compute the ultimate, each thread assigns its last element in its row into the ultimate array. To compute the reserve, each thread computes copies the diagonal element of its assigned row into the reserve array. To compute the risk margin, each thread takes its diagonal element and multiplies it with the risk-factor and stores in the risk-margin array. This is demonstrated in Figure 7.

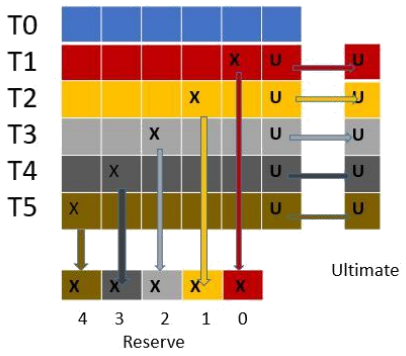

Figure 7. Calculation of reserve and ultimate using multiple threads

IV. RESULTS

In order the see the performance and implementation of these parallelized methods, same available data[2] has been used. This data was validated by experts from the field of actuaries. The input to all the methods are matrices, called input triangles. The lower triangle of these matrices excluding the diagonal elements are all zeros. The methods mentioned in the previous sections have been used to calculate the elements of the lower triangle. These elements represent the future reserves that an insurance company has to keep with them in order to meet the future liabilities.

A. System and GPU Details

We have used the local system for the serial implementation of the code and the CUDA implementation has been tested on NVIDIA GeForce TITAN X GPUs. The specifications of the system and GPU used are given in Table I and II respectively.

Processor	Intel(R) Core(TM) i5-4670 CPU @ 3.40GHz
Cores	4
Cache Size	6144 KB
CPU max MHz	3.40 GHz
Memory Size	16 GB

Table I: Cpu Details

GPU Model	Nvidia GeForce TITAN X
Cudacores	3072
Clock speed	1000 MHz
TFLOPS	6.144
Effective Memory speed	7012
Memory bus	384 bits
RAM	12 GB GDDR5
Memory BW	366GB/s
Single Precision	7 TFLOPS
Double Precision	0.2 TFLOPS
CUDA toolkit	CUDA v7.5

Table II: Gpu Details

B. Results and Graphs

1) Serial Code Execution Details: We have implemented the serial code of inflation-adjusted chain ladder method using the c-language and got the following timings for the different sizes of the matrix. The serial code was executed in the local system whose details are given in table III.

The graph shown in Figure 8 shows that as the size of the matrix increases, the time taken by the code increases too. Moreover time taken is an exponential function of the input size. This motivates us to use the accelerators such as GPUs.

Size (in 1000s)	Time (secs)
3x3	21
5x5	89
10x10	711
15x15	2430.17
20x20	5578.88
25x25	10857.8
30x30	18764.33

Table III: Serial Execution Results

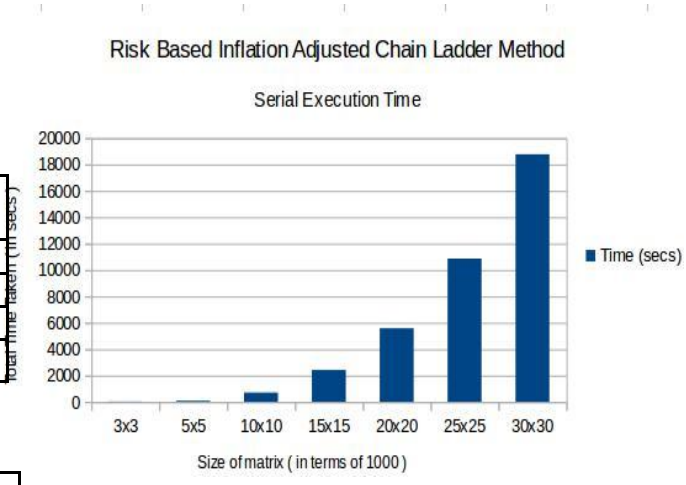

Figure 8. Execution Time of Serial Risk Based Inflation adjusted CLM

2) Parallel Code execution Details: Seeing the need and importance of the parallel implementation of the Risk-based chain ladder method, we have written the parallel code using the CUDA programming language to run on the GPU. We have written separate kernels for all the steps that are parallelizable. The following table and graph show the result of the execution of the parallel code.

Size (in 1000s)	Time (secs)
3x3	0.5377
5x5	1.1151
10x10	3.9302
15x15	8.3403
20x20	14.8859
25x25	26.8831
30x30	43.24844

Clearly, the Figure 9 shows that the parallel code took very less time compared to the serial code. This is due to the use of threads and blocks. The strategies of using them are already mentioned in the above sections.

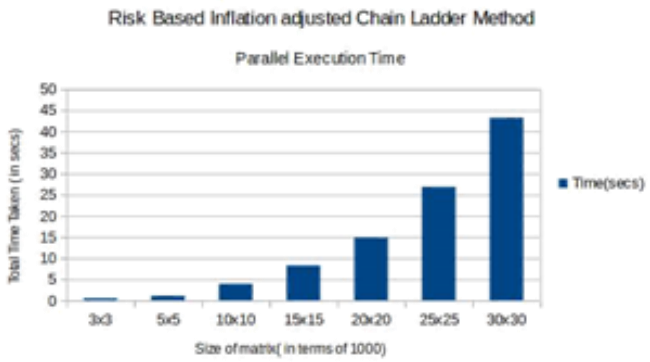

Figure 9: Execution Time of Parallel Risk Based Inflation adjusted CLM

Clearly, the Figure 9 shows that the parallel code took very less time compared to the serial code. This is due to the use of threads and blocks. The strategies of using them are already mentioned in the above sections.

3) Comparison: Now we have seen both the serial and parallel execution timings. The following figure shows the comparison between the two results.

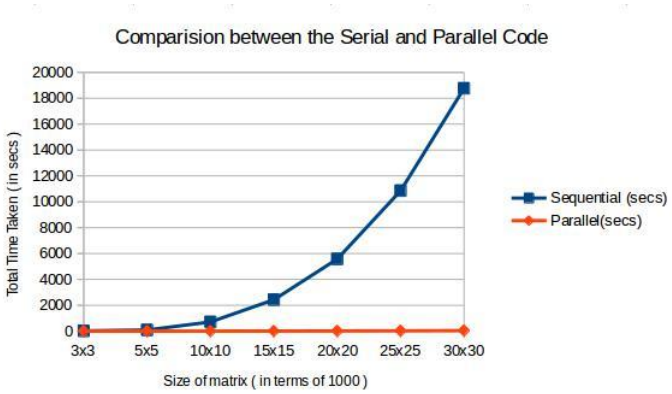

Figure 10: Comparison of Sequential and Parallel Execution Time

From the Figure 10, it is clearly evident that the time taken by the parallel code is much much lesser compared to the serial. This is even true for the larger values of input size.

4) Performance Improvement: From Figure 10, it is clearly seen that the our parallel Risk-Based Inflation Adjusted method out performs the serial code in terms of time taken. The following figure shows the speed up the obtained for different sizes of the matrix:

Size (in 1000s)	Speed Up
3x3	39
5x5	80
10x10	181
15x15	293
20x20	375
25x25	404
30x30	434

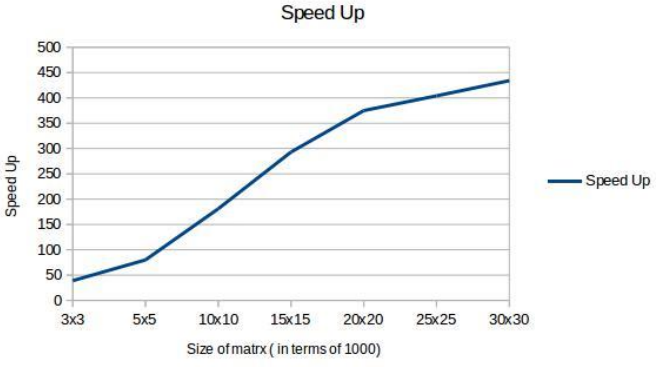

Figure 11.Speed Up

Figure 11 shows that as the input size increases the speed up too increases and for input size 30 the speed up is about 440X.

V. CONCLUSION

We have developed a method to compute the reserves, ultimate, risk margin and present value of risk margin. However, to reduce the time taken for computation where input size is very large, we have developed the parallelized version for the same. Using the high computational power of the GPU, we have gained about 430X speed up compared to the sequential method.

ACKNOWLEDGMENT

This work is dedicated to our Founder Chancellor Bha-gawan Sri Sathya Sai Baba. Our sincere and heartfelt thanks to Department of Mathematics and Computer Science (DMACS).

REFERENCES

[1] Hans Waszink, Waszink, Considerations on the Discount Rate in the Cost of Capital Method for the Risk Margin, www.actuaries.org/ASTIN/Colloquia, 2013.

[2] J Bhanu Teja , Pallav Kumar Baruah , Satya Sai Mudigonda , and Phani Krishna Kandala, Application of High Perfor-mance Computing for Calculation of Reserves for a Com-pany,International Journal of Scientific and Engineering Re-search Volume 9,2018

[3] Joshi, M.S., Graphical Asian Options,The University Of Mel-bourne

[4] Jauvion, G. and Nguyen, T.,arallelized Tri-nomial Option Pricing Model On GPU With CUDA, www.arbitragisresearch.com/cuda-in-computational-finance.

[5] Mark Tucker and J. Mark Bull,Application of High Performance Computing to Solvency and Profitability Calculations for Life Assurance Contracts

[6] Wütrich, M.V. and Merz, M. Stochastic Claims Reservig in Insurance Wiley,2008

[7] https://web.archive.org./web/20140327110448/http://www.soa.org/file/pd/health/hspring07005bk.pdf

[8] Peter D England and Richard J Verrall,Stochastic claims re-serving in general insurance, British Actuarial Journal, vol. 8, no. 3, pp. 443–518 2002