# Application of High-Performance Computing for Calculating Reserves using the Cape Cod Method

**S. R. Pranav Sai, Ajay Singh Pawar, Satya Sai Mudigonda , Pallav Kumar Baruah**

***Abstract:*** ***Calculation of reserves is one significant step in the strategic perspective for an insurance company. This is done on a periodic manner in order to have a better understanding regarding the future liabilities of the company. The time required for these calculations will grow quadratically as the size of the input increases. These computations are done several times by taking different factors in view. Thus, these computations become costly with regard to time. One of the many actuarial techniques for calculating reserves namely the Cape Cod method was used to calculate the reserves. HPC was applied for the same in order to reduce the time required to calculate a company's reserves. We applied HPC to calculate Projected Ultimate Claims based on Cape Cod method. Method takes year wise input of reported claims, incurred claims and earned premiums and outputs Projected Ultimate claims for each year. Making use of the independence which exists in computation of output across years, we parallelized the computations across years. The parallelization strategy proposed gives the speed up to 66194X compared to the serial implementation of the same.***

***Keywords : Reserves, Insurance, Development triangle, Cape Cod method, Actuarial analysis, Development year, Accident year, Claims incurred, Claims pair, GPU, CUDA.***

## I. INTRODUCTION

Reserving is a significant practice to get to know the realistic view of the liabilities of the future. This practice aids in the designing of the strategies for future business. This helps a company from avoiding the overstating or understating the future liabilities. This provides a better transparency for the various stakeholders of the company such as the agents, controllers, advisors, debtors and many more. Reserves help in the asset-liability matching of the company. There are many actuarial methods which are used in order to calculate those reserves for the company. Some ways to calculate the reserves for motor insurance business are the basic chain- ladder method, average cost per claim method, Bornhuetter-Ferguson method, Cape Cod method and so on. The development triangles are one of the common tools that actuaries use to organize historical data it is also used to identify the patterns and for analyzing the same. Development triangles are used by the actuaries to quantify the historical development. These development patterns deduced from the past is a significant input in many techniques to estimate the unpaid claims. Actuaries learn how to build these development triangles for case outstanding, paid claims, reported claims, and reported claims counts and hence use the information to predict the future claims to be incurred. In this work we proposed a framework for the parallelization of one such method of reserving known as the cape cod method. Since the computations in this method across the years are independent, there is scope for performing the computations in parallel across years using the Graphics Processing Unit (GPU).

| | Reported Claims as of (months) | | | |
|---|---|---|---|---|
| Accident Year | 12 | 24 | 36 | 48 |
| 20X5 | 1,500 | 2,420 | 2,720 | 3,020 |
| 20X6 | 1,150 | 1,840 | 2,070 | |
| 20X7 | 1,650 | 2,640 | | |
| 20X8 | 1,740 | | | |

**Fig. 1. Reported Claims**

This work is divided into seven different sections. Section one talks about the development triangle and explains its dimensions. Section three introduces the Cape Cod methodology for calculating the reserves. It also gives a step by step algorithm for calculating the same. Section four explains how HPC was used i.e., the parallelization of the Cape Cod method. Section five puts forth the results obtained for both the serial and the parallel code and compares the results from the two. Section six talks about the conclusions drawn from this work. Section shows the references which were used for this work.

## II. DEVELOPMENT TRIANGLE

A development triangle has three important dimensions: rows, diagonals, and columns. The exhibit (Fig 1) contains a simple reported claims triangle. A row in a development triangle denote a single accident year. Organizing data by accident year (AY) groups all the claims occured by the year of the accident (the accident date or AY). Thus, the data is grouped into accident years and in the development triangle, every row has a fixed number of groups of claims. In the example, we have a development triangle for reported claims.

**Revised Manuscript Received on October 20, 2019.**

***** Correspondence Author

**S.R.Pranav Sai,** Doctorial Research Scholar, Department of Mathematics and Computer Science, Sri Sathya Sai Institute of Higher Learning, Puttaparthi,, India Email: srpranavsai@sssihl.edu.in

**Ajay Singh Pawar**, Master's Student, Department of Mathematics and Computer science, Sri Sathya Sai Institute of Higher Learning, Puttaparthi India Email: ajay257223@gmail.com

**Satya Sai Mudigonda,** Senior Tech Actuarial Consultant, Hon. Professor in Department of Mathematics and Computer Science, Sri Sathya Sai Institute of Higher Learning Puttaparthi, India Email:satyasaibabamudigonda@sssihl.edu.in

**Dr.Pallav Kumar Baruah,** Associate Professor,Department of Mathematics and Computer Science, Sri Sathya Sai Institute of Higher Learning, Puttaparthi, India Email:pkbaruah@sssihl.edu.in

These reported claims are for AY 20X5 through 20X8. The triangle has four rows. The first to the fourth row represents the accident years from 20X5 to 20X8 respectively.

Successive valuation dates are represented by the diagonals. The triangle shown in Fig 2 has four diagonals.

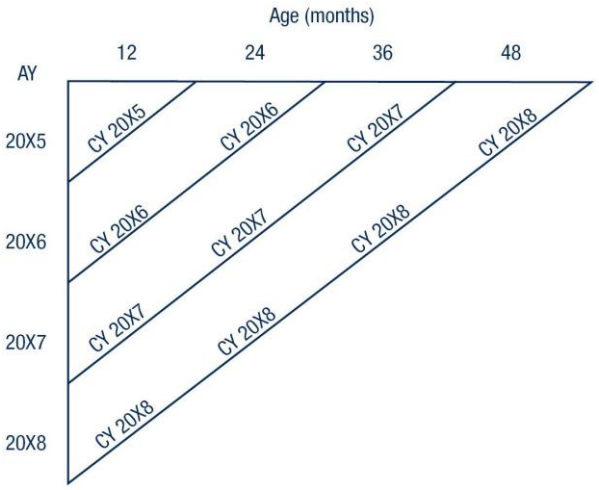


**Fig. 2. The Diagonals**

The first diagonal which is a single point over here represents the valuation date of 31st December 20X5

The next diagonal is for the valuation date of 31st December 20X6 for the accident years of 20X5 and 20X6

The third diagonal is for the valuation date of 31st December 20X7 for the accident years of 20X5, 20X6 and 20X7.

The last diagonal is for the valuation date of 31st December 20X8 for the accident years 20X5 through 20X8

Fig 2 shows the pictorial representation for the same.
Here the valuation period for a particular year is from 1st January till 31st December.
Consider the last diagonal of the triangle, where the valuation date is 31st December 20X8. Thus, the accident year 20X5 present in this diagonal would be forty-eight months old, since the date of evaluation is at the end of 20X8. Similarly, the accident year of 20X6 is thirty-six months old. Thus, this holds for all the other accident years in all the other diagonals as well. The columns in the claims development triangle represent the maturity. It is directly related to both accident year (represented by the rows) and the valuation date (represented by the columns) which are used to form the triangle. In the above-mentioned example, the data for the accident years is represented using the annual evaluations. Thus, the ages in the columns of the triangle are twelve, twenty-four, thirty-six and forty-eight months from left to right in the Fig 1.

## III. THE CAPE COD METHOD

This method is one of the many actuarial techniques in which the future unknown or the unreported claims are predicted using the development triangles. In this method two such triangles are used:

- The development triangle for the incurred claims.
- The development triangle for the reported claims

The key assumption: Unreported claims development will be based on the expected claims, which are computed using reported (or paid) claims and earned premium.
Re insurers often use the Cape Cod technique. The technique can used: with reported claims and paid claims, for all lines of insurance including short-tail lines a factor and long-tail lines.

**Ultimate Claims = Actual Reported Claims + Expected Unreported Claims**

**Algorithm 1 lists out the steps in steps of Cape Cod Method.**

**Algorithm 1: Cape Cod Algorithm**

**Input: The Development triangle for incurred and reported claims and earned premium.**

**Output: Ultimate Projected Claims**

1) Obtain the Upper claims triangles for the paid and reported claims.
2) Obtain the corresponding year wise earned premium.
3) Convert the incremental triangles to cumulative triangles.
4) Using the development factors calculate the year wise cumulative development factors (CDF).
5) Compute year wise Used Up Premium as Earned Premium / CDF.
6) Compute year wise Estimated Claim Ratio as Reported Claims/ Used Up Premiums.
7) Calculate average estimated claim ratio.
8) Compute year wise Estimated Expected Claims as Average estimated claim ratio x Earned Premium.
9) Compute Projected Ultimate Claims as Reported Claims + (1+1/CDF) * Estimated Expected Claim.
10) Reserves is the sum of the year wise difference of the projected ultimate claims and the total claims paid.

## IV. PARALLELISM

For the parallelization of above method, we have used the benefits provided by the GPU's structure. GPU provides the platform for computing the program parallelly by executing different number of threads parallelly. We parallelized the algorithm using CUDA. A Thread can be defined as the smallest unit of program execution. In CUDA, the threads are partitioned into various thread blocks. The thread blocks are further represented as elements of a bigger grid.

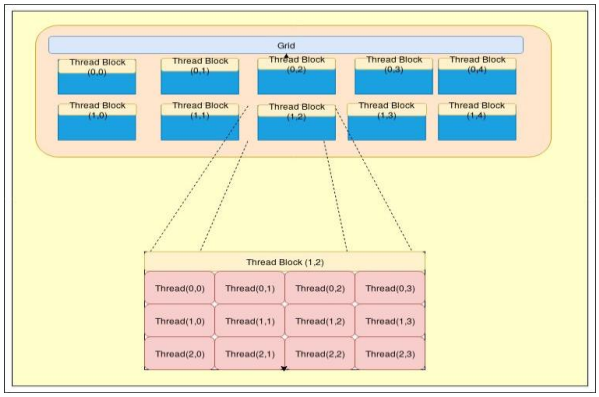


**Fig. 3. Organization of threads, blocks and grid**

Let us revisit the problem. Given the upper triangular matrix of NxN. We wish to compute the last column of the matrix. The last column represents the ultimate reserve for each year which insurance company must hold to meet the future claims. In the previous section an algorithm to solve the problem was described. The complexity of the algorithm is $O(N^2)$. For large *N,* the method could be time consuming. We propose a strategy to parallelize the algorithm to reduce the computation time drastically. In order to compute the final column of the matrix, as an intermediate step we need to compute cumulative upper triangular matrix, year wise development factor, year wise cumulative development factor, year wise used up premium, year wise estimated claim ratio, average estimated claim ratio and year wise estimated expected claim. It can be observed that all the intermediate computations are done year wise except for computing the average estimated claim ratio which is computed across years. There exists a clear year wise independence in the computations to be performed. Also the computation of the average estimated claim ratio can be performed efficiently using atomicAdd(). There can be at most of 1024 threads in a thread block (in the GPU used – Table I), for better division we fixed the number of threads in a thread block to be 1000 and fixed number of thread blocks to be. For the case where $N < 1000$ we have just one thread block with *N* threads. In the kernel we assign a unique *threadIdx* (*tid*) to each thread by using the formula *tid=theadIdx.x+blockIdx.x * blockDim.x*. Now we have *N* threads in the kernel, we assign the thread with *tid i to* perform the computation of the $i^{th}$ year where $0 \leq i < N$ . This facilitates the computation of Ultimate claims of each year in parallel. In the next section we present the speed up obtained to the parallelization proposed.

## V. RESULTS

In order to validate the performance promised by parallelization proposed we ran both the parallel and serial code on synthetic data generated in form of the upper triangular matrices of different sizes. However, the correctness of the algorithm was established using a real-life data and existing results. We now present the details of the CPU and GPU (Table I and Table II) used for executing the serial and parallel code respectively. The execution time of the serial and parallel implementation is presented in Table III and Table IV respectively. Graph in fig 4 and fig 5 presents the variation in run time of the serial and parallel implementation with respect to the increase in matrix dimension. The speed up obtained is presented in table V. Graph in fig 6 presents the execution time of serial and parallel code with respect to matrices of various sizes. Fig 7 depicts the relationship between matrix size and the speed up obtained. It can be observed that speed up obtained is directly proportional to matrix size.

**TABLE I NVIDIA K40 GPU DETAILS**

| GPU Model | Nvidia Tesla K40 |
|---|---|
| Number of Cuda Cores | 2880 |
| Clock Speed | 745 MHz |
| Single Precision TFlops | 4.29 |
| Double Precision TFlops | 1.43 |
| Memory Clock | 6 GHz GDDR5 |
| Memory Bus | 384-bit |
| RAM | 12 GB |
| Architecture | Kepler |

**TABLE II CPU DETAILS**

| CPU | Intel(R) Core(TM) i5-4770 CPU @ 3.40GHz |
|---|---|
| Cores | 4 |
| Cache size | 6144 KB |
| CPU max speed | 3.40 GHz |
| RAM | 16GB |

**Table III Execution Time By Serial Implementation**

| Size of the Matrix (in 000) | Time Taken (seconds) |
|---|---|
| 10 x 10 | 1.66 |
| 20 x 20 | 8 |
| 25 x 25 | 12.40 |
| 30 x 30 | 18.40 |
| 35 x 35 | 25.13 |

**Table Iv Execution Time By Parallel Implementation**

| Size of the Matrix (in 000) | Time Taken (seconds) |
|---|---|
| 10 x 10 | 0.000216 |
| 20 x 20 | 0.000254 |
| 25 x 25 | 0.000268 |
| 30 x 30 | 0.000313 |
| 35 x 35 | 0.000380 |

**Table V Speed Up Obtained**

| Size of the Matrix (in 000) | Speed Up |
|---|---|
| 10 x 10 | 7865 |
| 20 x 20 | 31496 |
| 25 x 25 | 47089 |
| 30 x 30 | 58805 |
| 35 x 35 | 66194 |

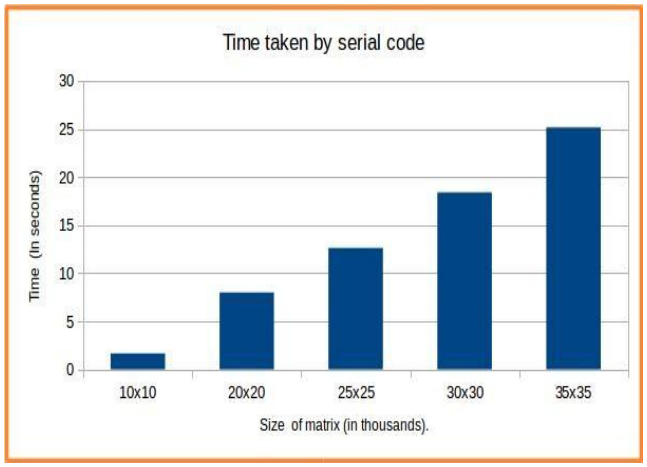


**Figure 4: Time taken by the serial code**

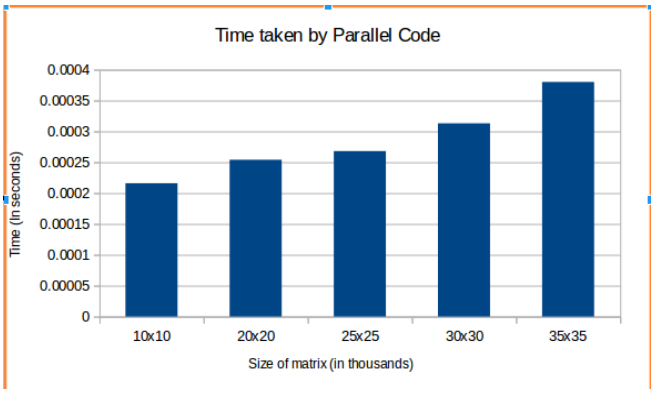


**Figure 5: Time taken by the parallel code**

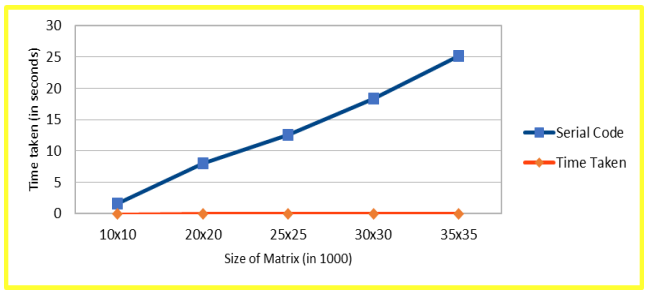


**Figure 6: The time comparisom between the serial and the parallel code**

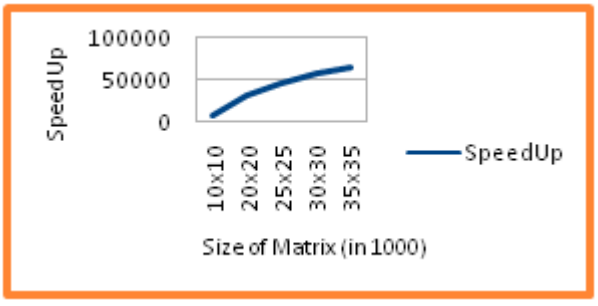


**Figure 7: The speed up achieved**

## VI. CONCLUSION

Thus, we developed an algorithm for the Cape Cod method for calculating the future liabilities and the reserves for the insurance company. We also developed the parallelized version for the same in order to reduce the time taken for the computations for calculating the reserves. Using the high computational power of the GPU we gained a highspeed ups as compared to the sequential way of calculating reserves.

## 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 Performance Computing for Calculation of Reserves for a Company*,International Journal of Scientific and Engineering Research Volume 9,2018.0
3. Joshi, M.S., *Graphical Asian Options*,The University Of Melbourne.
4. Jauvion, G. and Nguyen, T.,*arallelized Trinomial Option Pricing Model On GPU With CUDA*,www.arbitragisresearch.com/cuda-in-computationalfinance.
5. Mark Tucker and J. Mark Bull,*Application of High Performance Computing to Solvency and Profitability Calculationsfor 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/files/pd/health/hspring07-005bk.pdf
8. Peter D England and Richard J Verrall,*Stochastic claims reserving in general insurance*, British Actuarial Journal, vol. 8,no. 3, pp. 443–518 2002


## AUTHORS PROFILE

**S.R.Pranav Sai –** He is a doctoral research scholar in Sri Sathya Sai Institute of Higher Learning in the field of Actuarial sciences. His area of research includes Enterprise Risk Management, Operational Efficiency of insurance companies and IFRS 17. He is an Actuarial data science expert and has published research papers in international journals.

**Ajay Singh Pawar** – He is currently involved in research projects as part of the Master of technology in computer science in Sri Sathya Sai Institute of Higher Learning. He has a master's in science in Mathematics to his credit. His areas of research are deep learning and computer vision applied to sports and financial services.

**Satya Sai Mudigonda** - A Senior Tech Actuarial Consultant providing services in USA and India. With a wide skill set, he managed numerous multi-million-dollar international assignments for major insurance companies across the globe. He is an honorary professor in Sri Sathya Sai Institute of Higher Learning. He has published about fifteen papers in the field of Actuarial data science and has presented in several international conferences.

**Dr. Pallav Kumar Baruah** – He is an Associate Professor and the former HOD of the Department of Mathematics and Computer Science of Sri Sathya Sai Institute of Higher Learning. He has guided several research scholars in mathematics and computer science. He has numerous research publications to his credit and has presented in several national and international conferences.