Development of GPU-based Strategies for Finite Element Simulation of Elastoplastic Problems

Synopsis Report
In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

by

Utpal Kiran
Roll No. 176103028

DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

June 2023

1 Introduction

Elastoplasticity is a physical phenomenon in which materials deform elastically up to a certain threshold limit and plastically afterwards. The elastic deformation is temporary and can be recovered, but plastic deformation remains even after the load is removed. The elastoplastic behavior is commonly observed in materials of practical interest like metals, soil, rocks, polymers, biological tissues, etc. which are often subjected to high loads and pressure during service, often leading to yielding. A carefully performed elastoplastic analysis plays a key role in the design and manufacturing of engineering components, allowing for judicious decision about strength of solids so that an optimum use of resources can be made. In certain industrial applications, like metal forming, plastic deformation is intentionally introduced to obtain desired material properties and build products of various shapes. An efficient elastoplastic analysis can help researchers better understand natural occurrences like earthquakes, where very limited experiments could be performed. Some other application areas where elastoplasic analysis plays a decisive role are crashworthiness, fracture mechanics, and geophysical applications, among many others.

Finite element method (FEM) is the standard numerical technique for simulating elastoplastic problems (de Souza Neto et al. 2008). An efficient finite element analysis of elastoplastic problems is of great importance because it permits reliable estimation of the strength and deformation of solids and structures under various loading conditions. However, realistic simulation of various physical processes like metal forming or geophysical problems requires complex large-scale three dimensional (3D) models. The finite element computation for large-scale 3D models having millions or billions of degrees of freedom (DOFs) can be very expensive and may lead to huge computational time. In addition, the nonlinear nature of elastoplastic problems makes it necessary to use an incremental-iterative solution approach where FEM is used iteratively. In such a scenario, the processing time of a large-scale 3D elastoplastic simulation may be too large to be useful for practical purposes. The large computational time associated with elastoplastic simulation is reduced by employing a large number of compute resources in the form of parallel computing (Bhardwaj et al. 2002, Adams et al. 2004).

Graphics processing unit (GPU)-based computing has become popular in recent times for parallel computing across various disciplines. The primary reasons for its rising popularity are massive parallelism, high memory bandwidth, high performance-to-cost ratio and improvements in programmability. In addition, GPUs are the answer to modern day challenges of increasing energy demands, as they deliver more performance-per-watt than a CPU. Originally developed for gaming applications, modern advancements in GPU technology have led to its applicability in a wide range of sectors like image and video processing, machine learning and artificial intelligence (AI), high performance computing (HPC), etc. In the field of scientific computation, GPUs have evolved to take a prominent place as a co-processor or accelerator to speed up compute intensive parts of scientific codes. The popular numerical methods like FEM, computational fluid dynamics (CFD), molecular dynamics (MD), etc. have already seen great speedups due to GPU acceleration. However, running an application on a GPU is not as straightforward as a CPU. It requires architecture dependent parallel programming strategies along with fine tuning of several parameters to reach peak performance of the hardware. The GPUs have been found more suitable for tasks that involve high computation and less memory usage. This has led to a renewal of interest in many numerical algorithms that had been considered obsolete due to their high computational requirements.

Despite the existence of numerous efforts aimed at achieving parallel implementation of elastoplastic analysis on computing clusters, very few of them make use of GPU hardware. A comparative analysis of iterative solvers for elastoplasticity over a GPU cluster is presented by Khalevitsky et al. (2016), where performance achieved over six GPUs was found equivalent to hundred CPUs. He et al. (2017) proposed a GPU-based strategy for elastoplastic reanalysis, performing computation of elemental matrices on the GPU and assembly of the global tangent matrix on the CPU. A significant amount of speedup with respect to the CPU is reported, even when an expensive CPU-GPU data transfer step is involved. Recently, GPU implementation of elastoplastic simulation for perfectly plastic material is presented by Prabhune & Suresh (2020) for additive manufacturing applications with a matrix-free approach. Although very few previous works are found for elastoplastic analysis, there is an adequate amount of literature discussing GPU implementation of FEM in the context of various applications. However, elastoplasticity brings additional challenges to GPU implementation.

Due to its nonlinear nature, an incremental-iterative procedure is required that involves repeated use of the FEM procedure. This requires GPU implementation of the complete pipeline of FEM, which is again less discussed in the literature. Apart from the solution of linear system of equations, the computation of elemental matrices and their assembly into a global tangent matrix, computation of elemental force vectors and computation of stress and strain constitute important stages in elastoplastic analysis. In addition, elastoplastic analysis suffers from a well-known branching issue in parallel implementation due to the presence of both elastic and plastic states during the simulation process.

Matrix-free iterative solvers with GPU acceleration provide additional opportunities for reduction of simulation timings of elastoplatic analysis. In a matrix-free iterative solver, the computation of sparse matrix vector multiplication (SpMV) with the global tangent matrix is replaced by the matrix-free computation of SpMV using constituent elemental matrices. The elemental matrices are small, dense and provide better memory access than sparse storage formats in SpMV. The matrix-free SpMV has a distinct advantage over the assembly-based SpMV for voxel-based mesh. For problems with simple geometry in 3D, a voxel-based mesh is a kind of structured mesh that consists of linear cubic elements having the same size and orientation, generating the elemental tangent matrices of the same value in FEM. This property allows matrix-free SpMV computation for linear elastic problems with only one elemental tangent matrix, dramatically reducing the memory requirement to a minimum. On the other hand, the assembly-based implementation still needs to construct the global tangent matrix and therefore can not take full advantage of the voxel-based mesh. In the literature, matrix-free solvers with GPU acceleration have been effectively used to reduce the execution timings of FEM-based engineering simulations (Pikle et al. 2018). However, there is not much in the literature that discusses the application of matrix-free solvers to elastoplasticity.

In this work, a GPU-based framework to perform end-to-end computation in elastoplastic analysis is presented. The incremental-iterative solution approach to elastoplastic analysis is implemented in the form of a CPU-based loop that launches multiple compute kernels on the GPU to perform all the computations. The expensive CPU-GPU data transfer is completely avoided inside the computational loop, as all the computations are performed on the GPU. Wherever applicable, the proposed parallel strategies use a coloring method to avoid data race conditions. In order to achieve the best performance on the GPU, the thread divergence issue is prevented in all the proposed strategies. The objectives of the thesis can be summarized as follows:

- Development of a GPU-based parallel framework for elastoplastic analysis. It entails development of parallel strategies for all expensive steps in elastoplatic analysis, like computation of elemental matrices and their assembly, computation of internal force vectors and their assembly, and computation of stress using radial-return method (Simo & Hughes 1998).
- Development of matrix-free CG iterative solver for elastoplasticity applicable to all-hexahedral unstructured mesh.
- Development of matrix-free CG iterative solver for elastoplasticity applicable to voxel-based structured mesh.
- Performance evaluation of the proposed strategies over large-scale benchmark problems of elastoplasticity. The performance results are evaluated over problems involving associated flow rule and isotropic linear strain hardening with von Mises yield criteria.

In the following discussions, the proposed strategies and highlights of the associated outcomes are presented.

2 GPU acceleration of assembly-based elastoplasticity solver

The current work presents a novel GPU-based framework for elastoplastic analysis using FEM. The primary goal is to minimize the wall-clock timings for elastoplastic simulation by implementing all expensive steps on the GPU. The main computation is performed inside the Newton-Raphson iteration which is implemented using a CPU-based loop. The compute kernels for GPU are launched for every expensive step that appears inside the Newton iteration. The user-defined CUDA kernels are developed to implement steps like computation of elemental matrices and their assembly, computation of internal force vectors and their assembly and computation of stress on the GPU. The other simpler steps like computation of unbalanced force, updating the external force vector, evaluation of termination criteria, etc. are implemented on the GPU using algebraic operations provided in THRUST library. The solution of linear system of equations is done by CUSP and Ginkgo (Anzt et al. 2022) library, though any other suitable library may also be used. It is noted that there is no expensive data transfer with the CPU inside the Newton iteration.

The computation of the elemental tangent matrix is done by associating one thread with one element of the mesh. Each thread makes extensive use of the local memory space for storage of intermediate variables and computation results. For a linear hexahedral element, the computation is done inside a loop over all the Gauss points, and elemental tangent matrix is accumulated into the local memory. Considering the limited memory available with a GPU, the proposed strategy does not store the elemental matrices into global memory. It rather assembles them into a global tangent matrix. The assembly is done directly into a CSR sparse storage format in the GPU memory that can be readily used with any linear solver. The issue of data race conditions during parallel assembly is handled by the coloring method. The strategy for computation of elemental tangent matrix is extended to compute the internal force vectors and its assembly into a global vector. The current work also proposes a GPU-based strategy for the computation of stresses using the radial-return method. The comparison of execution timings with sequential CPU implementation reveals speedups in the range 20.4×–69.7× for computation of elemental matrices and assembly, 47.2×–66.1× for computation of stresses using the radial-return method and 53.7×–67.3× for computation of internal force vectors and their assembly. In wall-clock timings, speedups in the range 1.4× to 7.2× are obtained with respect to sequential CPU implementation using GPU-based linear solver. Finally, as a result of the proposed strategies, the proportion of the iterative linear solver timings in wall-clock timings of the GPU implementation reaches up to 98.9% for the finest mesh consisting of 5.1 million DOFs. As shown in Fig. 1, the collective time spent in other steps is marginally smaller than that of the linear solver for all three examples. This contrasts with the CPU implementation, where time consumed in steps other than the linear solver is also substantial.

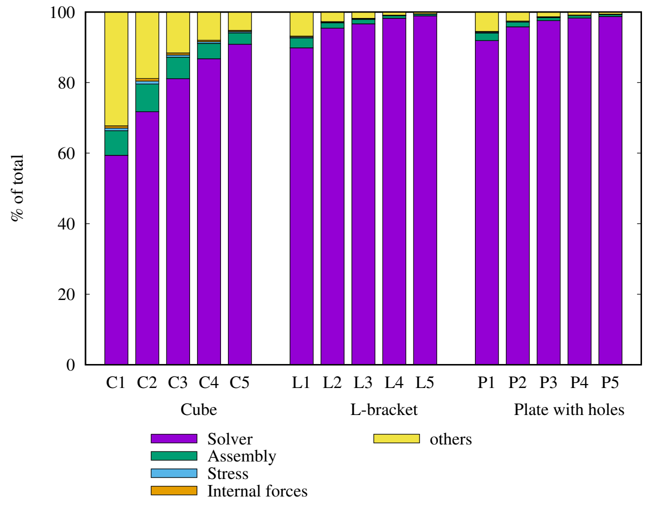
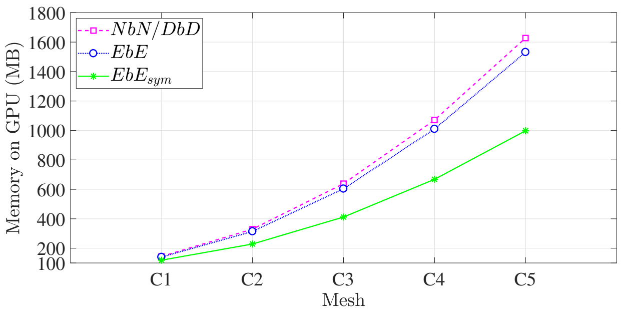


Figure 1: Execution timings breakup of the proposed assembly-based solver for cube, L-bracket and plate with holes examples. Here C1, L1 and P1 are the coarsest mesh, whereas C5, L5 and P5 are the finest mesh.

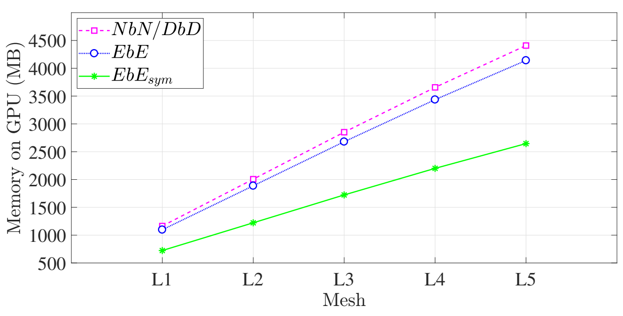
3 GPU-based matrix-free strategies for unstructured mesh

The wall-clock timings of the assembly-based elastoplasticity solver are found to be bounded by the performance of iterative linear solver. The execution timings of an iterative solver is largely determined by the performance of the SpMV implementation. A SpMV operation performs multiplication of a sparse matrix stored in specialized storage formats (CSR, COO, ELL, etc.) with a vector. These sparse storage formats reduce the storage requirement of a sparse matrix, but introduce an irregular memory access pattern that degrades performance on the GPU. In the present work, various methods for matrix-free computation of SpMV on GPU are explored, and a matrix-free CG solver for elastoplasticity is developed.

The matrix-free SpMV is implemented on the GPU using node-by-node (NbN), DOFs-by-DOFs (DbD) and element-by-element (EbE) strategies, and their performances are studied. The NbN, DbD and EbE strategies are taken from the literature and modified according to proposed optimizations for best performance on the GPU. In the present work, an EbE_{sym} strategy that works with only symmetric part of elemental tangent matrices is proposed. For elasticity problems using quadrilateral elements, approximately 5$\times$ speedup was observed over the NbN, 2.8$\times$ over the DbD and 1.4$\times$



(a) 2D cantilever beam.



(b) 2D L-bracket.

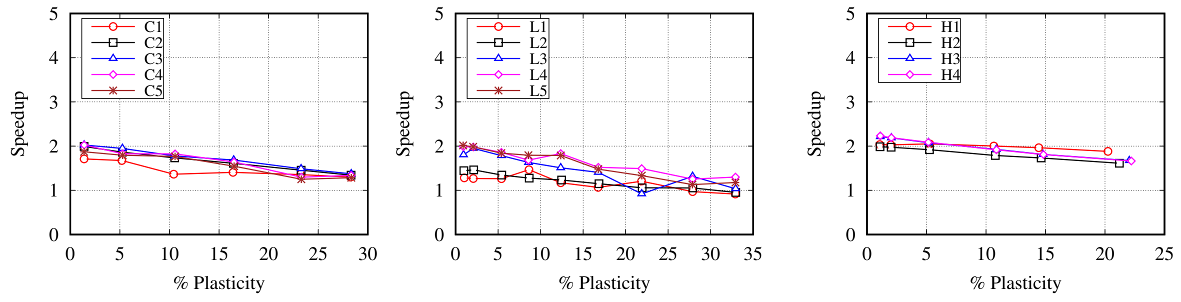
Figure 2: GPU memory utilization by various matrix-free strategies. Here, C1 and L1 are the coarsest mesh, and C5, L5 are the finest mesh.

over the EbE matrix-free strategy by the proposed EbE_{sym} strategy. The EbE_{sym} strategy is found to have the most efficicent memory access. As a consequence of using the symmetric part of the elemental matrices, the overall memory footprint of the proposed EbE_{sym} strategy is reduced by 1.5× as compared to the EbE strategy. Figure 2 shows the variation of memory consumption with mesh refinement for two benchmark examples. It is noted that the performance of all matrix-free strategies is found limited by memory access due to the large number of elemental tangent matrices associated with unstructured mesh.

4 GPU-based matrix-free strategies for structured mesh

Several engineering problems consist of relatively simple geometry that can be discretized with a structured mesh. In many cases, a structured mesh can be obtained with linear cubic elements having the same shape and size, referred to as voxel-based mesh. In such cases, the elemental tangent matrix remains the same for all the elements of the voxel-based mesh. The computation of SpMV for a linear elastic problem using the voxel mesh can be done with only one elemental tangent matrix. However, the presence of both elastic and plastic states in elastoplastic analysis makes the situation challenging. The determination of the plastic state is dependent on the local values of internal variables. This leads to unique elemental tangent matrices for elements belonging to the plastic zone. In addition, the presence of two types of states introduces branching issues in parallel SpMV computation.

In the present work, two strategies for SpMV computation in elastoplasticity, namely the single kernel strategy and the improved split kernel strategy, are proposed. Both the proposed strategies avoid thread divergence by restructuring the computation according to elements in elastic and plastic zones. To achieve efficient memory access, one common elemental tangent matrix is used for all the elements in the elastic zone, and individual elemental matrices are used for all the elements in the plastic zone. For effective implementation on GPU, both the proposed strategies are implemented on GPU using *NbN*, *DbD* and *EbE* strategies and their performance is compared. Since standard *NbN*, *DbD* and *EbE* strategies cannot be used directly for elastoplasticity, modifications are proposed to improve the performance so that they can be used with the proposed strategies. The performance evaluations are done over multiple large-scale benchmark examples from elastoplasticity over different levels of mesh refinement and different amounts of plasticity. The single kernel *NbN* strategy is found to be up to 2.25× faster than the GPU-based strategy in the literature for low percentage of plasticity. For moderate to high amounts of plasticity, the proposed single kernel *DbD* strategy is found achieving up to 2.6× speedups over strategy from the literature. Further, *EbE* strategies are found better than *NbN* and *DbD* strategies in both the single kernel and improved split kernel strategy. As shown in Fig. 3, the proposed matrix-free solver is found to be up to 2× faster than the GPU optimized assembly-based elastoplasticity solver in terms of wall-clock timings.



(a) Cantilever beam (b) L-bracket (c) Plate with square holes

Figure 3: Speedups by the proposed matrix-free elastoplasticity solver over assembly-based elastoplasticity solver.

5 Organization of the Thesis

The thesis is organized in six chapters as follows.

- **Chapter 1** presents a brief introduction of elastoplastic simulation and associated challenges, followed by survey of literature. Then, the research gaps are discussed followed by the objectives of the thesis.

- **Chapter 2** provides the background for theory of elastoplasticity and its numerical solution using FEM. The numerical procedure for the estimation of stresses in finite element framework is also discussed. This is followed by description of sequential CPU implementation.

- **Chapter 3** discusses the proposed GPU framework for assembly-based elastoplatic solver. GPU-based strategies are presented for the computation of elemental tangent matrices and their assembly, computation of internal force vectors and their assembly, and computation of stresses using the radial return method. The performance of the proposed framework is discussed for both structured as well as unstructured meshes by applying it over various 3D benchmark examples.

- **Chapter 4** presents the matrix-free SpMV strategies for elastoplastic problems with unstructured mesh. A novel strategy is proposed that exploits the symmetry of elemental tangent matrices to reduce memory access and improve the performance.

- **Chapter 5** discusses two novel matrix-free SpMV strategies for elastoplastic problems, namely single kernel strategy and improved split kernel strategy for efficient usage of voxel-based structured mesh. Both the proposed strategies are implemented on GPU a using node-based and element-based parallelization strategies and performance is compared for various benchmark examples.

- **Chapter 6** concludes the thesis with a note on scope for future research.

References

Adams, M. F., Bayraktar, H. H., Keaveny, T. M. & Papadopoulos, P. (2004), Ultra-scalable implicit finite element analyses in solid mechanics with over a half a billion degrees of freedom, *in* 'SC'04: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing', IEEE, pp. 34–34.

Anzt, H., Cojean, T., Flegar, G., Göbel, F., Grützmacher, T., Nayak, P., Ribizel, T., Tsai, Y. M. & Quintana-Ortí, E. S. (2022), 'Ginkgo: A modern linear operator algebra framework for high performance computing', *ACM Transactions on Mathematical Software* **48**(1), Article no. 2.

Bhardwaj, M., Pierson, K., Reese, G., Walsh, T., Day, D., Alvin, K., Peery, J., Farhat, C. & Lesoinne, M. (2002), Salinas: A scalable software for high-performance structural and solid mechanics simulations, *in* 'SC'02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing', IEEE, pp. 35–35.

de Souza Neto, E. A., Perić, D. & Owen, D. R. J. (2008), *Computational Methods for Plasticity: Theory and Applications*, John Wiley & Sons, Ltd.

He, G., Wang, H., Huang, G., Liu, H. & Li, G. (2017), 'A parallel elastoplastic reanalysis based on GPU platform', *International Journal of Computational Methods* **14**(05), Article no. 1750051.

Khalevitsky, Y. V., Burmasheva, N. V., Konovalov, A. V. & Partin, A. S. (2016), 'Comparative study of Krylov subspace method implementations for a GPU cluster in elastoplastic problems', *AIP Conference Proceedings* **1785**(1), Article no. 040024.

Pikle, N. K., Sathe, S. R. & Vyavhare, A. Y. (2018), 'GPGPU-based parallel computing applied in the FEM using the conjugate gradient algorithm: a review', *Sādhanā* **43**(7), Article no. 111.

Prabhune, B. C. & Suresh, K. (2020), 'A fast matrix-free elasto-plastic solver for predicting residual stresses in additive manufacturing', *Computer-Aided Design* **123**, Article no. 102829.

Simo, J. C. & Hughes, T. J. R. (1998), *Computational Inelasticity*, Springer-Verlag, New York.

Journal Publications

- Kiran, U., Sharma, D. & Gautam, S. S. (2024), 'An efficient framework for matrix-free SpMV computation on GPU for elastoplastic problems', *Mathematics and Computers in Simulation* 216, 318–346.

- Kiran, U., Sharma, D. & Gautam, S. S. (2023), 'A GPU-based framework for finite element analysis of elastoplastic problems', *Computing* 105, 1673–1696.

- Kiran, U., Gautam, S. S. & Sharma, D. (2020), 'GPU-based matrix-free finite element solver exploiting symmetry of elemental matrices', *Computing* 102(9), 1941–1965.

- Kiran, U., Sharma, D. & Gautam, S. S. (2023), 'Development of GPU-based matrix-free strategies for large-scale elastoplasticity analysis using conjugate gradient solver', *International Journal of Numerical Methods in Engineering* (Under review).

Conference Publications

- Kiran, U., Gautam, S. S. & Sharma, D. (2022), 'Accelerating finite element assembly on a GPU', International Conference on Future Learning Aspects of Mechanical Engineering (FLAME - 2022), Amity University, Noida, 3–5th August.

- Kiran, U., Sanfui, S., Ratnakar, S. K., Gautam, S. S. & Sharma, D. (2019), 'Comparative analysis of GPU-based solver libraries for a sparse linear system of equations', In *Advances in Computational Methods in Manufacturing: Select Papers from ICCMM 2019*, Springer, Singapore, pp. 889-897.

Other Publication

- Ratnakar, S. K., Kiran, U., & Sharma, D. (2022), 'Acceleration of structural topology optimization using symmetric element-by-element strategy for unstructured meshes on GPU', *Engineering Computations* 39(10), 3354-3375.