GPU-based Strategies for Accelerating Topology Optimization of 3D Continuum Structures using Unstructured Mesh

Synopsis Report

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

by

Shashi Kant Ratnakar

146103004

Under Supervision of

Dr. Deepak Sharma

to the

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

June, 2022

1 Introduction

Various industries and engineering disciplines require lightweight structures that can be manufactured with the lesser amount of material while still fulfilling their strength and structural performance requirements [1, 2, 3]. One of the viable methods to develop such structures is structural topology optimization. It is an application of optimization that enables the design engineers to optimally distribute the material within a design space, taking into consideration the given loads, boundary conditions, and other performance constraints. The ultimate result is a topology that is sufficiently robust while making efficient use of the limited material available, striking a balance between desired strength and material costs.

Structural topology optimization is a well established field and various methods have been developed in the literature. One of the most popular methods is solid isotropic material with penalization (SIMP) [4]. The computational steps of a typical SIMP-based structural topology optimization include meshing, finite element analysis (FEA), objective function computation, sensitivity analysis, mesh-independency filter, and design variable update. Among them, FEA is the most computationally expensive process, and other computational steps require significantly less time than FEA. As a result, a majority of contemporary research in the literature prioritizes accelerating FEA solver by leveraging the massively parallel architecture of graphics processing units (GPUs) for reducing the overall computational time [5, 6, 7]. In some studies, the entire structural topology optimization has been performed in parallel [8, 9]. GPU has been the preferred hardware because it offers a low-cost solution for accelerating data-parallel applications such as topology optimization.

From the literature, it is found that GPU can speedup topology optimization by many-folds. However, performing various computational steps of topology optimization on GPU brings a new set of challenges. The first major challenge is the development of efficient kernel that can harness the massive parallelism of GPU. This is achieved by ensuring optimal load distribution among the GPU threads, and avoiding situations such as thread divergence, thread idling, and race-condition. The other challenge involves the efficient use of GPU memories, such as using shared memory for inter-thread communications, minimizing global memory read operations etc. It is also observed that majority of the studies devoted to GPU-based acceleration of topology optimization use structured meshes. Structured meshes are efficient for discritizing the structures with regular domain geometry. GPU implementation are also easier to handle when structured meshes are used. However, the domain geometry of many applications can be complex, irregular, and can have curved boundaries. In such cases unstructured meshes are used for better representation of design domain. Unstructured meshes pose a couple of additional computational challenges on GPU, such as a huge memory space requirement for storing elemental and connectivity data, and proper load balancing among GPU threads. Although a number of studies has used unstructured meshes, there are several research gaps in the area, such as developing efficient GPU-based strategies for optimal thread allocation and reduced memory transactions to speedup the FEA solver on GPU using unstructured meshes. In this thesis, some of these major challenges of GPU-based acceleration of FEA solver for structural topology optimization are addressed.

The aim of this thesis is to develop an efficient GPU-based FEA solver for topology optimization of large-scale 3D continuum structures using unstructured meshes. The preconditioned conjugate gradient (PCG) FEA solver used in this thesis adopts a matrix-free approach where the system of linear equations is solved at the elemental level by GPU threads. The thesis later focuses on enhancing the computational performance of the GPU-based matrix-free PCG solver by tackling the challenges posed by unstructured meshes on GPU. This is accomplished through creating efficient thread allocation strategies that ensure a balanced computational load among GPU threads, as well as GPU-specific data storage formats that optimize data accesses on the GPU while reducing CPU-GPU data transfer. The following are the objectives of the thesis.

1. Develop a GPU-based matrix-free FEA solver for structural topology optimization using unstructured meshes.
2. Develop efficient thread allocation strategies for matrix-free FEA solver.
3. Develop novel data storage and data access patterns for matrix-free FEA solver.
4. Performance analysis of the proposed GPU-based matrix-free FEA solver over benchmark problems and comparing with the state-of-the-art.

In the following sections, the strategies proposed to achieve the objectives of the thesis are discussed in brief.

2 GPU-based Matrix-Free FEA Solver for Structural Topology Optimization using Unstructured Mesh

The first objective of this thesis is to develop a GPU-based matrix-free PCG solver tailored to efficiently handle the unstructured meshes on the GPU. The PCG solver is then used in a SIMP method-based structural topology optimization framework for large-scale 3D continuum structures that are discretized using unstructured meshes. The proposed framework takes a generalized approach by considering unstructured mesh regardless of domain geometry. The meshing is done using the ANSYS R16.1 APDL module using $8-$noded hexahedral elements for all examples considered in the thesis, and the mesh data is imported into the framework. The focus is on executing entire FEA efficiently on GPU. The matrix-free PCG FEA solver is used in this thesis that consists of two types of operations: sparse matrix-vector multiplications

(SpMV) and vector arithmetic operations. Initial experiments show that SpMV operations on GPU are more complex and consume $80\% - 99\%$ of the solver's execution time, as compared to other vector arithmetic operations. In the literature, the following two types of GPU-based SpMV strategies can be found.

1. **element-by-element** (***ebe***) [10]: One compute thread of GPU is assigned to each element of FE mesh [6]. However, this strategy suffers with race-condition when more than one thread try to access and modify the same memory location, ultimately producing inconsistent results. The coloring method or atomic operation can be used for alleviating this issue.

2. **node-by-node** (***nbn***) [5]: A single thread computes the state for all degree-of-freedoms (DoF) of a node. This strategy requires access to data of neighboring elements. Since each node performs its computation independently, there is no race-condition observed [11].

An ***ebe***$-$strategy is proposed for the matrix-free PCG solver. Since the proposed strategy is developed for unstructured meshes, SpMV requires the elemental stiffness matrices of all finite elements. They are stored in the GPU global memory along with the connectivity data. The thread assigned to an element reads its connectivity matrix from the global memory, performs the multiplication, and finally writes back the result into the output vector. The atomics feature of CUDA is used to handle the inherent problem of race-condition that arises during write operations. The connectivity data of the elements is stored in the GPU's local memory, thereby reducing the number of global memory read operations. When tested over a variety of benchmark problems, the proposed GPU-based ***ebe***$-$strategy shows speedup of $3\times - 5\times$ over the CPU-based matrix-free solver.

Next, an ***nbn***$-$strategy is also proposed in which a customized nodal connectivity storage format is developed to enable the efficient access of neighborhood data by the compute thread. This customized connectivity is referred to as the 'reverse - connectivity matrix,' and is created by performing a search operation through the connectivity matrix for each node. Although conducting a search operation for each node consumes a lot of time, it is only required once at the beginning of the optimization process. The computations at the node level are independent of one another and therefore, no race-condition is observed. The GPU-based ***nbn***$-$strategy shows speedup of $2.5\times - 4\times$ over its CPU-based matrix-free version.

It is observed that the ***nbn***$-$ strategy requires significantly higher execution time than the ***ebe***$-$ strategy. One of the reason is the additional time required in creating the reverse - connectivity matrix. Each node can be associated with different number of neighboring elements in unstructured mesh. For the ***nbn***$-$ strategy, this causes an imbalanced computational load across GPU threads, diminishing overall computational performance on GPU. The GPU memory requirement of ***nbn***$-$ strategy is also found significantly higher than the ***ebe***$-$ strategy.

The execution time of the other computational steps of topology optimization is minuscule compared to FEA solver. The findings of the first objective show that the ***ebe−*** strategy is a better suited SpMV strategy for unstructured mesh in terms of computational performance, as long as the race condition issue is handled adequately.

3 Efficient Thread Allocation Strategies for Matrix-Free FEA Solver

According to the literature and the results from previous section, SpMV operations of matrix-free PCG solver using unstructured meshes involve a large number of floating-point operations as well as access to connectivity and elemental data. As a result, it is essential to execute SpMV efficiently in order to enhance the computational performance of the PCG solver. In the second objective of the thesis we focus on developing appropriate thread allocation schemes for the ***ebe−*** strategy to ensure better load distribution among GPU threads.

There are 24×24 entries in the elemental stiffness matrix (K_e) of an 8− noded hexahedral elements. The standard ***ebe−*** strategy allocates a single GPU thread to perform multiplication for all the entries of K_e with their corresponding vector elements, as discussed in the previous section. We develop fine-grained SpMV strategies that use various levels of granularity present in K_e of an element. The proposed strategies reduce the computational load of a compute thread by allocating more number of threads to an element. The following three fine-grained SpMV strategies are proposed.

1. ***ebe8* :** The strategy allocates 8− GPU threads to each finite element. Three successive rows of K_e are assigned to each of these 8 threads for matrix-vector multiplication.

2. ***ebe24* :** The strategy allocates 24− GPU threads to each finite element. Multiplications for a single row entries are handled by a single thread.

3. ***ebe64* :** The strategy allocates 64− GPU threads to each finite element. A single thread is assigned to a tile of 3×3 entries.

Using the proposed GPU-based SpMV strategies a matrix-free PCG solver is developed. Their computational performance is compared with the standard ***ebe*** over a number of benchmark examples using various mesh sizes. The ***ebe8***, ***ebe24***, and ***ebe64*** strategies show maximum speedups of $4\times - 8\times$, $3\times - 4\times$, and $6\times - 7\times$, respectively. The ***ebe64*** strategy appears to perform best for smaller meshes. For larger meshes the ***ebe8−*** strategy outperforms all other strategies. Although the ***ebe24*** strategy is $3\times - 4\times$ faster than the ***ebe−*** strategy, it is found inferior than both ***ebe8*** and ***ebe64−*** strategies. Thread divergence and multiple global memory transactions issues have been observed for inferior performance of the ***ebe24*** and ***ebe64−***

strategies as compared with the ***ebe*8** strategy. The ***ebe*8−** strategy is found to be the best performing SpMV strategy for large-scale unstructured meshes using 8−noded hexhedral finite elements.

4 Novel Data Storage and Data Access Patterns for Matrix-Free FEA Solver

The fine-grained SpMV strategies discussed in previous section stores the connectivity data in shared memory of GPU, thereby reducing number of global memory read operations. To access the elemental stiffness matrices a large number of global memory read operations is still needed. Additionally, during each optimization iteration, the data must be copied from CPU to GPU. Both the challenges degrade the performance of the matrix-free PCG solver. These challenges and their proposed solutions are discussed in this section.

The third objective of the thesis aims to develop efficient data storage and data access patterns on GPU to further enhance the performance of the proposed matrix-free PCG solver for topology optimization. First, we focus on reducing the amount of data transferred between the CPU and GPU. In elasticity, the elemental stiffness matrices are symmetric. Using this property, SpMV can be performed using only the symmetric half of the elemental stiffness matrices. The idea of storing and using only the symmetric half entries has been used by Zegard and Paulino [8] to perform the assembly for 2D unstructured meshes. Duarte et al. [6] used this idea for 2D and 3D polygonal meshes. We develop a novel GPU-strategy (***ebeSym***) which performs SpMV of matrix-free PCG solver by utilizing only a symmetric half of elemental stiffness matrices. Instead of using 576 entries per element we use only 300 entries that significantly reduces the amount of data copied to GPU.

We develop two novel data storage formats that ensure coalesced read and write operations in order to optimize the data accesses on GPU. The entries from the symmetric half of K_e are divided into 'diagonal' and 'off-diagonal' parts that are stored separately in two arrays. The storage is performed in such a manner that when consecutive threads are assigned to consecutive elements, threads make coalesced access to entries in the respective arrays. The ***ebeSym−*** strategy allocates 8−GPU threads to each element. The eight threads first multiply the entries of the diagonal group with the corresponding entries of the vector. Thereafter, the same operation is performed by each thread for off-diagonal entries.

The proposed ***ebeSym−*** strategy along with the novel data storage patterns are incorporated in the matrix-free PCG solver. The computational performance is tested over a number of benchmark topology optimization examples with various mesh sizes. When compared to the 8− thread per element SpMV strategy (***ebe*8**) that uses full K_e, the ***ebeSym−*** strategy shows $1.8\times - 3.7\times$ speedup while using $1.8\times$ lesser GPU memory. With the proposed ***ebeSym−*** strategy speedup of $17\times - 26\times$ is observed with respect to the ***ebe−*** strategy discussed in

Section 2.

5 Organization of Thesis

The thesis is organised in six chapters. The details are as follows.

- **Chapter 1** presents the problem statement, major challenges, and a brief discussion over the existing methods addressing these challenges, leading to the motivation and the objectives of the thesis.

- **Chapter 2** discusses the theoretical and implementational aspects of the structural topology optimization. The fundamentals of GPU computing are discussed, followed by a literature review on using GPUs to accelerate density-based structural topology optimization methods.

- **Chapter 3** presents the GPU-based matrix-free FEA solver for structural topology optimization for large-scale 3D continuum structures using unstructured meshes. The popular strategies for accelerating matrix-free FEA solver on GPU, their implementation, and performance over benchmark examples are discussed in this chapter.

- In **Chapter 4** fine-grained SpMV strategies to accelerate the matrix-free FEA solver on GPU are discussed. The proposed three SpMV strategies are tested over benchmark problems and their computational performance is compared with the standard strategy form the literature.

- **Chapter 5** presents the ***ebeSym−*** strategy that uses the symmetric half of the elemental stiffness matrices, along with two novel data storage formats to ensure optimized data accesses on GPU. The results of the proposed strategy are presented and compared with the standard ***ebe−*** strategy.

- **Chapter 6** presents the conclusions drawn from the thesis along with a note on future work.

References

[1] Alok Sutradhar, Glaucio H Paulino, Michael J Miller, and Tam H Nguyen. Topological optimization for designing patient-specific large craniofacial segmental bone replacements. *Proceedings of the National Academy of Sciences*, 107(30):13222–13227, 2010.

[2] Deepak Sharma, Kalyanmoy Deb, and NN Kishore. Customized evolutionary optimization procedure for generating minimum weight compliant mechanisms. *Engineering Optimization*, 46(1):39–60, 2014.

[3] David J Munk and Jonathan D Miller. Topology optimization of aircraft components for increased sustainability. *AIAA Journal*, 60(1):1–16, 2021.

[4] Martin P Bendsøe. Optimal shape design as a material distribution problem. *Structural optimization*, 1(4):193–202, 1989.

[5] Stephan Schmidt and Volker Schulz. A 2589 line topology optimization code written for the graphics card. *Computing and Visualization in Science*, 14(6):249–256, 2011.

[6] Leonardo S Duarte, Waldemar Celes, Anderson Pereira, Ivan FM Menezes, and Glaucio H Paulino. Polytop++: an efficient alternative for serial and parallel topology optimization on cpus & gpus. *Structural and Multidisciplinary Optimization*, 52(5):845–859, 2015.

[7] Jesús Martínez-Frutos and David Herrero-Pérez. Large-scale robust topology optimization using multi-gpu systems. *Computer Methods in Applied Mechanics and Engineering*, 311:393–414, 2016.

[8] Tomás Zegard and Glaucio H Paulino. Toward GPU accelerated topology optimization on unstructured meshes. *Structural and multidisciplinary optimization*, 48(3):473–485, 2013.

[9] Jesús Martínez-Frutos, Pedro J Martínez-Castejón, and David Herrero-Pérez. Efficient topology optimization using gpu computing with multilevel granularity. *Advances in Engineering Software*, 106:47–62, 2017.

[10] Cris Cecka, Adrian J Lew, and Eric Darve. Assembly of finite element methods on graphics processors. *International journal for numerical methods in engineering*, 85(5):640–669, 2011.

[11] Utpal Kiran, Sachin Singh Gautam, and Deepak Sharma. Gpu-based matrix-free finite element solver exploiting symmetry of elemental matrices. *Computing*, 102(9):1941–1965, 2020.

List of Publications

Journal Publications

- Shashi Kant Ratnakar, Subhajit Sanfui and Deepak Sharma, 2022, “Graphics Processing Unit-Based Element-by-Element Strategies for Accelerating Topology Optimization of Three-Dimensional Continuum Structures Using Unstructured All-Hexahedral Mesh”, ASME Journal of Computing and Information Science in Engineering, 22(2), 1–11. https://doi.org/10.1115/1.4052892

- Shashi Kant Ratnakar, Utpal Kiran and Deepak Sharma, 2022, “Acceleration of Structural Topology Optimization using Symmetric Element-by-Element Strategy for Unstructured Meshes on GPU”, Engineering Computations, **(Resubmitted after revision).**

Conference Publications

- Shashi Kant Ratnakar, Subhajit Sanfui and Deepak Sharma, 2021, “SIMP-Based Structural Topology Optimization Using Unstructured Mesh on GPU”, *in* N. Kumar, S. Tibor, R. Sindhwani, J. Lee and P. Srivastava, eds, ‘Advances in Interdisciplinary Engineering: Select Proceedings of FLAME 2020’, Springer, pp 1–10, https://doi.org/10.1007/978 − 981 − 15 − 9956 − 9_1.

- Shashi Kant Ratnakar, Subhajit Sanfui and Deepak Sharma, 2021, “GPU-Based Topology Optimization Using Matrix-Free Conjugate Gradient Finite Element Solver with Customized Nodal Connectivity Storage”, *in* N. Kumar, S. Tibor, R. Sindhwani, J. Lee and P. Srivastava, eds, ‘Advances in Interdisciplinary Engineering. : Select Proceedings of FLAME 2020’, Springer, pp 87−97, https://doi.org/10.1007/978−981−15−9956−9_9.

Other Publications

- Utpal Kiran, Subhajit Sanfui, Shashi Kant Ratnakar, Sachin Singh Gautam, and Deepak Sharma, “Comparative Analysis of GPU-based Solver Libraries for A Sparse Linear System of Equations”, *in* 2nd International Conference on Computational Methods in Manufacturing (ICCMM), 8 − 9 March 2019, IIT Guwahati, India.