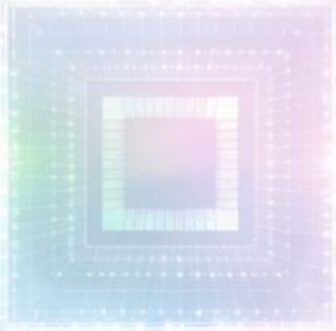
iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

CGAD



High-Performance Graphics Computing

PRACTICAL PROJECT

HIGH-PERFORMANCE GPU-ACCELERATED **FINITE ELEMENT ANALYSIS**

António Cruz (140129), **Bruno Santos** (140586), **Pedro Miranda** (129268), **Ricardo Kayseller** (95813)

Table of Contents

1. Introduction

The continuous increase in model complexity and fidelity in engineering and scientific simulations has made computational efficiency a central concern in modern numerical analysis. As simulation models grow in size and resolution, the computational cost associated with their numerical solution becomes a limiting factor, particularly when large parametric studies or high-resolution meshes are required.

Among numerical techniques for solving boundary-value problems governed by partial differential equations (PDEs), the Finite Element Method (FEM) stands out as one of the most widely adopted and versatile approaches. FEM enables the approximation of field variables over complex geometries while supporting heterogeneous material properties and mixed boundary conditions. It is extensively used across engineering and applied science domains, including structural analysis, heat transfer, diffusion processes, electrostatics, and fluid-related potential problems.

Despite its flexibility and robustness, FEM is computationally demanding. As mesh resolution increases, the number of degrees of freedom grows rapidly, leading to large sparse systems of algebraic equations. In such scenarios, traditional CPU-based implementations face scalability challenges, particularly during the assembly of global sparse matrices and the iterative solution of linear systems. These limitations motivate the exploration of parallel execution models capable of exploiting fine-grained data parallelism more effectively.

Recent advances in Graphics Processing Units (GPUs) have significantly expanded the capabilities of high-performance computing. GPUs offer massive data-parallel throughput and high memory bandwidth, characteristics that align well with the linear algebra workloads dominating FEM solvers. Furthermore, modern Python-based acceleration frameworks allow heterogeneous CPU–GPU execution while preserving developer productivity, making GPU acceleration accessible in a scientific computing context.

In this context, the present project proposes the design and implementation of a GPU-accelerated Finite Element Analysis solver, developed progressively from a functional CPU baseline to a GPU-enabled implementation. From a practical standpoint, the project will be developed using a Python-based scientific computing stack, leveraging widely adopted numerical and GPU-acceleration frameworks.

A functional CPU baseline and an initial GPU-enabled execution path have already been validated using this stack, providing a solid starting point for systematic experimentation, optimisation, and performance evaluation throughout the project. The work aims to investigate parallelisation strategies, execution models, and performance evaluation techniques, in alignment with the objectives of the *High Performance Graphical Computing* curricular unit.

2. The Finite Element Method (FEM)

2.1 Definition and Core Principles

The Finite Element Method is a numerical discretisation technique used to approximate the solution of partial differential equations over complex domains. The continuous domain is

subdivided into a finite number of smaller subdomains, known as elements, which are interconnected at nodes. Within each element, the unknown field variables are approximated using polynomial shape functions.

By applying a variational (weak) formulation, the governing PDE is transformed into a global system of algebraic equations. For a wide class of linear problems, this process leads to a sparse linear system of the form

$$\mathbf{Ku} = \mathbf{f},$$

where ($\mathbf{K}$) is the global stiffness matrix, ($\mathbf{u}$) is the vector of nodal unknowns, and ($\mathbf{f}$) represents the global load vector. The stiffness matrix is typically sparse, symmetric, and positive definite, properties that strongly influence the choice of numerical solvers and optimisation strategies.

2.2 Importance and Applications of FEM

FEM is a cornerstone of modern engineering and scientific analysis due to its ability to handle complex geometries, irregular domains, and realistic boundary conditions. It is widely applied in civil, mechanical, aerospace, and energy engineering, as well as in physics-based simulations such as heat conduction, mass diffusion, electrostatics, and potential flow problems.

The ability to perform accurate simulations prior to physical testing reduces development costs and improves safety and reliability. However, these benefits come at the cost of increased computational demand, particularly for large-scale or high-resolution problems, making FEM an ideal candidate for parallel and GPU-accelerated computing techniques.

2.3 Types of Problems Addressed

The Finite Element Method supports a broad range of physical problems, including:

- Linear static structural analysis.
- Heat conduction and diffusion problems.
- Electrostatic and potential field problems.
- Steady-state fluid potential flow.
- Coupled and multi-physics formulations.

Within the scope of this project, the initial focus is placed on linear elliptic PDEs, which provide a mathematically well-posed and computationally controlled setting for evaluating parallel execution strategies. These formulations are representative of several physical phenomena and allow the solver architecture to be extended to related problems with minimal structural changes.

3. Computational Motivation for Parallelism and GPU Acceleration

From a computational perspective, FEM workloads are dominated by linear algebra operations, particularly sparse matrix assembly, sparse matrix-vector products, and vector updates performed repeatedly within iterative solvers such as the Conjugate Gradient method. These operations exhibit a high degree of data parallelism, as identical arithmetic operations are applied independently across large data sets.

GPUs follow a Single Instruction, Multiple Threads (SIMT) execution model, enabling thousands of lightweight threads to execute concurrently on different data elements. This execution paradigm makes GPUs particularly effective for accelerating the core computational kernels of FEM solvers. The FEM computational pipeline exhibits natural parallelism at multiple stages: element-level operations can be processed independently, sparse matrix assembly involves accumulating local contributions that can be parallelized with appropriate synchronization, and iterative solver operations such as matrix-vector products are inherently data-parallel.

Based on the data-parallel nature of FEM assembly and the memory-bound characteristics of sparse iterative solvers, GPU acceleration is expected to provide significant performance improvements. Element-level assembly operations are anticipated to benefit substantially from parallel execution due to their embarrassingly parallel structure, while iterative solvers may achieve more moderate speedups constrained by memory bandwidth and sparse access patterns. Overall workflow acceleration is expected to increase with problem size, making large-scale simulations particularly well suited for GPU execution.

By analysing the FEM computational pipeline and identifying performance-critical stages, this project aims to assess how GPU acceleration can be leveraged to improve execution time, scalability, and overall efficiency, while maintaining numerical correctness.

4. Project Objectives

4.1 General Objective

The primary objective of this project is to design, implement, and evaluate a GPU-accelerated Finite Element Analysis solver, assessing performance improvements and scalability relative to a CPU-based baseline.

4.2 Specific Objectives

The specific objectives of the project are:

1. To implement a correct and complete CPU baseline FEM solver using Python and established numerical libraries, serving as a reference for correctness and performance comparison.
2. To analyse the FEM computational pipeline and identify performance bottlenecks, with particular emphasis on sparse matrix assembly and iterative solvers.
3. To explore CPU-level optimisation techniques, including just-in-time compilation and parallel execution, where appropriate.
4. To develop a GPU-accelerated implementation targeting the most computationally intensive stages of the FEM workflow.
5. To perform a systematic performance evaluation comparing CPU and GPU executions, focusing on execution time, scalability, and efficiency.
6. To design an integrated application that enables accurate inspection of numerical results and performance metrics through interactive visualisation.

5. Technologies and Execution Models

The project will be implemented using a Python-based high-performance computing stack that supports both CPU and GPU execution, enabling a consistent development workflow across heterogeneous architectures.

The baseline CPU implementation will rely on established numerical libraries for array manipulation and sparse linear algebra, providing a reference for correctness and performance. GPU acceleration will be explored using Python frameworks that expose CUDA-enabled execution models while preserving a high-level programming interface. These frameworks allow the use of vectorised operations, custom GPU kernels, and GPU-resident sparse linear algebra routines.

The project will adopt a progressive execution model, subject to feasibility within the course timeline:

Implementation Stage	Technologies	Purpose
CPU baseline	NumPy, SciPy	Correctness and reference performance
Optimised CPU	Numba (JIT, parallel execution)	Incremental CPU acceleration
GPU-accelerated	CuPy, Numba CUDA	Data-parallel GPU execution

This selection reflects both the technologies covered in the curricular unit and those already validated as suitable for the problem class under consideration. The chosen stack allows fine-grained control over memory placement, kernel execution, and performance measurement, which is essential for meaningful analysis in a high-performance computing context.

6. Planned Performance Evaluation

The project will include a structured performance evaluation, focusing on metrics that reflect both computational efficiency and scalability. Performance will be assessed through standard high-performance computing metrics including execution time, speedup relative to baseline, and time per degree of freedom. Scalability will be evaluated across multiple problem sizes to assess how performance characteristics change with increasing mesh resolution.

The evaluation will systematically compare CPU and GPU implementations across representative problem sizes, documenting execution times for key computational stages (assembly, solving, post-processing) and overall workflow performance. Results will be presented through comparative visualizations including speedup charts, scaling plots, and per-stage performance breakdowns.

7. Case Study and Problem Definition

To evaluate the proposed solver, a two-dimensional elliptic PDE problem will be adopted as a benchmark. Such problems arise naturally in potential flow, heat conduction, diffusion, and electrostatics, and are well suited for FEM discretisation and performance analysis.

Example scenario: Two-dimensional potential flow governed by Laplace's equation, discretized using 8-node quadrilateral (Quad-8) elements over meshes with complex geometry and mixed

boundary conditions. Problem sizes will range from tens of thousands to potentially over one million degrees of freedom, spanning from typical desktop-scale simulations to workstation-class problems suitable for evaluating GPU acceleration benefits.

This benchmark is intended to provide a realistic and computationally demanding test case while remaining mathematically well-posed and reproducible.

8. Application and Visualisation Strategy

The project will include an integrated application that presents both numerical results and performance metrics in a coherent and interactive manner. The application will be implemented as either an interactive Jupyter Notebook with visualization widgets or a web-based interface using Python visualization frameworks, depending on deployment requirements.

The application will allow users to:

- Configure simulation parameters and select execution modes (CPU or GPU).
- Inspect scalar and vector fields produced by the FEM solver.
- Visualize convergence behavior and solution accuracy.
- Compare results across execution modes.
- Analyse performance metrics through interactive charts and tables.

The visual component is intended to support accurate interpretation and validation of numerical data and performance behavior, while providing a clear and accessible interface for assessing computational results.

9. Conclusion and Expected Contributions

This project proposes a systematic investigation into accelerating the Finite Element Method using GPU-based parallel execution. By focusing on a mathematically well-defined class of problems and progressing from a CPU baseline to a GPU-accelerated implementation, the project aims to provide clear insights into the performance characteristics of heterogeneous computing architectures.

Expected contributions include:

- A validated CPU and GPU FEM solver implementation.
- Quantitative performance analysis demonstrating the benefits and limitations of GPU acceleration.
- An integrated application supporting result inspection and performance evaluation.
- A structured discussion of design decisions, optimisation strategies, and observed trade-offs.

This project aims to directly address the learning objectives of the curricular unit through practical application of GPU acceleration techniques, systematic performance evaluation, and development of technical proficiency in modern heterogeneous computing frameworks.