An Investigation of Decoupling Single Element Solutions from the Mesh in the Finite Element Method on Nvidia GPUs

MSc. High Performance Computing

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Author: Alex Keating
Student ID: 17307230
Supervisor: Jose Refojo & Prof. Kirk Soodhalter

Department of Mathematics
Faculty of Engineering, Mathematics and Science
Trinity College Dublin, University of Dublin

August 17, 2019

Declaration

Alex Keating

August 17, 2019

Acknowledgements

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden. Ich versichere, dass die eingereichte elektronische Fassung der eingereichten Druckfassung vollständig entspricht.

Abstract

[Abstract goes here (max. 1 page)]

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, with the advent of modern computing, there has been a huge push towards taking advantage of these resources. In the scientific world, there is a constant drive towards building optimised libraries for performing operations, spanning areas such as, basic linear algebra [CITE], fast Fourier transformations [CITE] or genetic algorithms for machine learning [CITE]. All of these libraries have been written with the intentions of exhausting as much processing power, memory and parallelisation as possible. These libraries are available across all kinds of architectures, Intel's MKL for example built for Intel CPUs, Nvidia's CUDA SDK written for their own GPUs or MAGMA, a 3rd party library written for heterogeneous architectures. These advancements in scientific computing have not come from nowhere, but rather are becoming duly necessary as most modern problems in science become impossible to solve without taking advantage of modern computing resources. A clear example of this was seen last year when the first image of a black hole was rendered, using a novel image cleaning machine learning algorithm and 5 petabytes of data - clearly something that cannot be done without some form of distributed memory architecture.

Nvidia have been among the leaders of this charge from both a hardware and software point...

The need for taking advantage of parallelism in modern science now clear, one problem that crops up in a vast array of areas is solving partial differential equations, or PDEs. These are relationships between variables and their partial derivatives and can be seen in areas such as fluid dynamics with the Navier-Stokes equations, in finance with the Black-Scholes equation or even in engineering when modelling stress of a structure - an important one for the topic at hand in this paper. Unfortunately, while analytic solutions exist for theoretical problems of this nature, studied in undergraduate courses, in the real world most of these PDEs are not of this convenient solvable nature and thus require numerical methods to solve. There are many variants of numerical methods which one can use to solve PDEs, some more simple than others, such as the finite difference method which decomposes the domain into a simple grid and approximates the derivatives, and some more complex but robust, like meshfree methods which create a collection of Voronoi cells on the domain instead of a basic grid - allowing for more complex structures. Certain methods land somewhere in a middle ground of both complexity and adaptability such as the finite volume method and the finite element method - the later of which will form the basis of this study.

The finite element method [CITE STRANG] is a numerical method developed originally for use in engineering for modelling stress on structures and has since quickly expanded to use in all branches of science such as electrostatics, something and something. Its engineering foundations will become clear throughout the paper as much of the terminology has remained unchanged. This paper will investigate current approaches to applying the finite element method on Nvidia GPUs and attempt at isolating and pinpointing certain bottlenecks for parallelism upon which may be improved. The importance of this is clear, as PDE-related problems get larger and more complex, the need for more computing power is evident to get approximate solutions in reasonable amounts of time.

1.1 Preliminaries

Maybe ???

Chapter 2

Finite Element Method

This paper focuses its mathematics on implementing the finite element method to solve PDEs. This chapter will go through the necessary mathematics behind each of the steps behind the finite element method such as variational calculus, approximation theory and the finite elements themselves. The chapter also goes through the actual approach itself, alongside the worked example for the case of this study of the Laplace equation.

2.1 General Problem

Before the nuts and bolts of the finite element method are discussed, let us first consider an n-dimensional, general n$^{\text{th}}$ order PDE of the form,

$$f(\mathbf{x}; u(\mathbf{x}), \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_n}; \frac{\partial^2 u}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u}{\partial x_1 \partial x_n}; \ldots) = 0, \tag{2.1}$$

where $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$. For a case of a 2-dimensional, 2$^{\text{nd}}$ order PDE, we end up with the equation,

$$a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial x \partial y} + c\frac{\partial^2 u}{\partial y^2} + F\left(x, y; u; \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right) = f(x, y). \tag{2.2}$$

This can be rewritten, treating the right-hand side as a linear operator, leaving the equation,

$$\mathcal{L}u = f, \tag{2.3}$$

which is going to be our entire basis for this study - attempting to approximate the function u. This problem can then be bounded by three types of boundary conditions in order to have a unique/non-trivial solution:

1. Dirichlet condition: $u = g(x, y)$.
2. Neumann condition: $\frac{\partial u}{\partial \mathbf{n}} = g(x, y)$.
3. Robin condition: $\frac{\partial u}{\partial \mathbf{n}} + \sigma(s)u = g(x, y)$.

2.2 Approximation Theory

...

2.3 Variational Calculus

...

2.4 Finite Elements

...

2.5 Implemented Example

Chapter 3

GPU Implementations

3.1 Current Approaches

3.2 FEM-SES

3.3 Issues???

In this research we follow

PowerTAC is an example of a multiagent competitive gaming platform

Chapter 4

Results

4.1 Test-bed Architecture

Chapter 5

Conclusion

Appendix A

Appendix

...