Discontinuous Galerkin FEMs for Radiation Transport Problems

Richard Widdowson

A thesis submitted to the University of Nottingham for the degree of Doctor of Philosophy

June 2023

Abstract

The linear Boltzmann transport equation (LBTE), a high-dimensional partial integro-differential equation, is used to model radiation transport. Radiation transport is an area of physics that is concerned with the propagation and distribution of radiative particles, such as photons and electrons within a material medium.

In this thesis, we present a high-order discontinuous Galerkin finite element method (DGFEM) discretisations of the steady state linear Boltzmann transport equation in the spatial, angular, and energy domains. Comparisons between the tensor discontinuous Galerkin finite element method and discrete ordinates method show that the former is higher order than the latter. A method of block diagonalising the resulting matrix into a sequence of transport equations coupled by the right hand side while retaining high order convergence, is demonstrated for both the angular and energy domains. This new method offers the arbitrary order convergence rates of the discontinuous Galerkin finite element method, but it can be implemented in an almost identical form to standard multigroup discrete ordinates methods.

The assembly of the matrix for the resulting transport equations for a variety of different type of elements is discussed. The generation of meshes formed of general polytopes is discussed, and a comparison between the time to solve the transport equation on meshes formed of the different element types follows.

An efficient implementation of the discontinuous Galerkin finite element method for transport equations is then presented. This algorithm exploits the fixed wind direction of the transport problems resulting from the discontinuous Galerkin finite element method of the LBTE, to solve the transport problem while never forming the matrix. We then compare this algorithm to a direct matrix solver for both convex and non-convex polytopes.

An a posteriori error bound for the discontinuous Galerkin finite element method of the LBTE and the transport problem is then derived. We use this error bound to develop an adaptive framework for the LBTE. Three different adaptive algorithms for the LBTE are then presented and compared. The h-refinement algorithm, which marks the element with the error of each tensor element it is part of, shows a clear advantage over the other methods.

Acknowledgements

I would first like to express my gratitude to my supervisors, Matthew Hubbard and Paul Houston, for their guidance, insight, and support throughout the course of my PhD. I would like to extend my thanks to the rest of the School of Mathematical Sciences at the University of Nottingham, especially the Scientific Computation research group.

I would like to thank Oliver Sutton for providing insight and mentorship, and being a supervisor in spirit (if not in pay). I also thank Fred Currell and Balder Villagomez-Bernabe for many inspiring ideas about how our work could be used and the physics underpinning the model. To Thomas Radley I would like to express my gratitude for his constant support, insight, and for the many hours of conversation, a few of which may have even been on topic!

I would like to thank all of my friends, for their constant support and lifting of my spirits. Their friendship and companionship has been the highlight of my life. Finally, I would like to thank my family for their support, and motivation throughout the years – the many, many years – I have spent at university.

Contents

Chapter 1

Introduction

In the UK alone, around 1,000 cases of cancer are diagnosed every day [66]. Radiotherapy is an integral part of treatment plans for combatting cancer, with 4 out of every 10 cancer cures including radiotherapy [39]. A system that provides a rapid, accurate, and robust prediction of the dosage of radiation that the tumour needs to be exposed to, is invaluable. This is complicated in part due to each individual requiring their own customised treatment plan. Another complication is that radiation does not distinguish between the cancerous cells and the cells of the surrounding tissue and organs, which will also be damaged by the exposure to radiation. The treatment plan, therefore, must balance the need to deliver enough radiation to denature the DNA of the cancerous cells with the need to preserve the healthy surrounding cells.

The current "gold standard" of this dose calculation procedure is a statistical method called Monte Carlo. This models the motion of each individual radiative particle through the patient. The distance each particle travels between interactions, as well as the deflection angle, and associated energy loss after each interaction, are randomly sampled using given cross-sectional data which contains relevant material data. Each cross-section is defined by the energy of the incident particle and to the kind of interaction it undergoes. This data must be supplied for each material present.

The particles can interact in three ways. Absorption removes the particle from the system, due to loss of energy. The other interaction is scattering, where a particle collides with an atom in the patient's body, transferring some energy and/or changing the direction of travel. Both of these interactions can produce secondary particles. If the secondary particles produced are charged, the interaction is said to be ionising. It is this ionising effect that denatures the DNA in the cell, preventing the cells from growing and dividing; thus, causing them to die. The particle can also be removed from the system by exiting the domain of interest, i.e. the particle passed out of the patient's body.

Deterministic methods have started to be proposed as alternative methods to Monte Carlo. Most of these methods are based on solving coupled Boltzmann transport

equations. To account for the different types of particles that the scattering can produce, deterministic methods separate out the equation for photons and electrons. The linear Boltzmann transport equation (LBTE) is used for photons, and the electron transport equation [32] or the Boltzmann Fokker-Plank transport equation [46], are solved for electrons. Deterministic methods have been compared favourably to Monte Carlo methods [28]. In this thesis, we focus on the linear Boltzmann transport equation.

The linear Boltzmann transport equation is a 7-dimensional partial integro-differential equation. The dimensions are split into four domains, space $\Omega \in \mathbb{R}^3$, the position in 3D space, angle $\boldsymbol{\mu} \in \mathbb{S}$, the direction of travel in 3D space parametrised on to the surface of the unit sphere, energy $E \in \mathbb{E} = [0, \infty)$, and time $t \in \mathbb{T} = [0, \infty)$. The solution to the LBTE gives the fluence of radiative particles in the patient as a function of their positions, directions of travel, energies, and the time since the simulation was started. The dosage the patient receives can then be calculated from the fluence.

The discretisation of the energy domain is most commonly done using the multigroup method [50]. This subdivides the energy domain into a number of so-called energy groups. The multigroup methodology results in a collection of mono-energetic problems, one for each energy group. The solutions of these mono-energetic problems are then multiplied by an energy group-specific energetic function, to yield an approximation of the exact angular fluence.

Discretising the angular domain has a long and rich history of study. There have been several different discretisations proposed over the years. Spherical harmonic approximations are constructed utilising a basis of typically hig$h$-order smooth spherical harmonic functions defined globally on the sphere [17]. The emphasis of such schemes is to simplify the implementation of the scattering operator. Such schemes offer a natural variational setting for their analysis, but the global nature of the basis functions makes local adaptivity a challenging task and Gibbs'-type oscillations may be expected around sharp variations in the solution [26].

Discrete ordinates [48] is generally the most popular option, whereby the angular domain is discretised using a collocation at a discrete set of angular quadrature points. The advantage of this approach is that the LBTE can be solved as a set of independent linear transport problems in the spatial domain with fixed wind directions.

One deterministic method for approximating the solution on the spatial domain is the finite element method. The classical finite element method, also known as continuous Galerkin finite element method (CGFEM), seeks a continuous piecewise-polynomial approximation of the solution discretised on an underlying mesh, formed of basic element shapes, such as simplices, quadrilaterals etc. The discontinuous Galerkin finite element method (DGFEM), on the other hand, is capable of employing meshes consisting of more general element geometries, and does not require the approximation of the solution to be continuous. DGFEM was originally developed to solve this very problem [56], but

has gone on to be used on a variety of different problems, such as hyperbolic [63, 38], parabolic [59, 5], and elliptic [7, 58] partial differential equations (PDEs) as well as ordinary differential equations (ODEs) [57].

The use of arbitrary polytopic elements in the mesh allows the underlying geometry of a problem to be preserved more accurately. This is especially useful in this application as the changes in the medium's electron density will result in different scattering cross-sections being used. Polytopic elements can also reduce the size of the matrix that needs to be inverted to find a solution.

The accuracy of DGFEM relies on mesh refinement (h-refinement) and increasing the polynomial order of the approximation (p-refinement). For the transport problem, the order of convergence of the error in the L^2 norm is given as $O(h^{p+1})$ [40] for most meshes [53]. Naturally, p-refinement is very advantageous in situations where a smooth solution is to be approximated [14]. hp-adaptive DGFEMs have been employed on a variety of problems, as they provide accurate estimation of the solution while reducing the degrees of freedom (DOFs) of the system [**?**].

1.1 Thesis Outline

The thesis is structured as follows. In Chapter 2, we first provide the formulation and overview of the linear Boltzmann transport equation (LBTE). We provide an insight into the common forms of discretising the LBTE, particularly the angular component. We show how the discrete ordinates angular discretisation can be devolved into transport equations which are coupled by their right hand side. Motivated by this, we discuss the transport equation and discontinuous Galerkin finite element (DGFEM) discretisation as applied to the transport equation.

In Chapter 3 we discuss applying the discontinuous Galerkin finite element method to the time independent linear Boltzmann transport equation. We show the DGFEM discretisation of the energy, angle, and space components of the LBTE. We then compare DGFEM angular discretisation to discrete ordinates, showing that DGFEM achieves arbitrarily high order convergence of the error in the L^2 norm compared to the fixed order of discrete ordinates, at the cost of increased computation time and complexity. We then introduce a new discretisation, first introduced in our paper [33], known as Discrete Ordinate Galerkin (DOG). This discretisation retains the high order convergence while still devolving the PDE into coupled transport equations.

Motivated by a desire to create an efficient transport equation solver for our DOG discretisation, we look at the structure of the matrix formed by DGFEM on a transport equation in Chapter 4. We consider a number of different spatial element types, such as simplices, convex polytopes, and arbitrarily shaped polytopes. We compare the density of the resulting matrices and then the time to solve the matrix equation.

Continuing to examine the transport solver, we introduce a new matrix free sweep solver in Chapter 5. This solver exploits the fixed wind direction of the transport problems that result from the DOG discretisation, to be efficiently solved without having to store the matrix. We compare the time taken to solve transport equations with our different element types. This also shows the effect of element convexity on the sweep solver.

Finally, in Chapter 6, we introduce an *a posteriori* error estimator, and dual weighted residuals to calculate it. We use this error estimation to guide our mesh refinements for both the transport equation and the LBTE. When the standard h-refinement algorithm under-performs due to the tensor structure of the LBTE, we introduce two alternative versions of a h-refinement algorithm that perform more satisfactorily.

Chapter 2

Background

2.1 Formulation of the linear Boltzmann transport equation

Radiation transport is an area of physics that is concerned with the interactions of particles in some material medium. These particles, photons, neutrons, and electrons, interact by transferring some of their energy from collisions to the material, irradiating the material. Radiation transport has two main areas of study, nuclear reactor theory [12, 62], and radiotherapy treatments [11]. In this thesis, we focus on radiotherapy applications, which are dictated by the linear Boltzmann transport equation (LBTE).

In radiotherapy, the principle function of concern is the angular flux, denoted by $u(\mathbf{x}, \boldsymbol{\mu}, E, t)$. This function relates the distribution of particles in physical space $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, as a function of time $t \in \mathbb{R}^+$, position $\mathbf{x} \in \Omega$, energy $E > 0$, and the angle the particles are travelling in direction, $\boldsymbol{\mu} \in S^{d-1}$, where S^{d-1} denotes the surface of the unit sphere in $\mathbb{R}^d$.

The linear Boltzmann transport equation (LBTE) is a seven-dimensional partial integro-differential equation for the angular flux $u(\mathbf{x}, \boldsymbol{\mu}, E, t)$ [12, 50, 62] given by:

$$\begin{aligned}
\frac{1}{v}\frac{\partial u}{\partial t}(\mathbf{x}, \boldsymbol{\mu}, E, t) + \boldsymbol{\mu} \cdot \nabla_{\mathbf{x}} u(\mathbf{x}, \boldsymbol{\mu}, E, t) + (\alpha(\mathbf{x}, \boldsymbol{\mu}, E, t) + \beta(\mathbf{x}, \boldsymbol{\mu}, E, t))u(\mathbf{x}, \boldsymbol{\mu}, E, t) \\
= \int_{\mathbb{E}} \int_{\mathbb{S}} \theta(\mathbf{x}, \boldsymbol{\mu}' \to \boldsymbol{\mu}, E' \to E, t) u(\mathbf{x}, \boldsymbol{\mu}', E', t) \; d\boldsymbol{\mu}' \, dE' + f(\mathbf{x}, \boldsymbol{\mu}, E, t) \\
u(\mathbf{x}, \boldsymbol{\mu}, E, 0) = u_0(\mathbf{x}, \boldsymbol{\mu}, E) \\
u(\mathbf{x}, \boldsymbol{\mu}, E, t) = g(\mathbf{x}, \boldsymbol{\mu}, E, t) \quad \text{on } \Gamma_- \\
u(\mathbf{x}, \boldsymbol{\mu}, E, t) = \int_{\mathbb{E}} \int_{\mathbb{S}} \kappa(\mathbf{x}, \boldsymbol{\mu}' \to \boldsymbol{\mu}, E' \to E, t) u(\mathbf{x}, \boldsymbol{\mu}', E', t) \, d\boldsymbol{\mu}' \, dE' \quad \text{on } \Gamma_-.
\end{aligned}$$

Where

- $\mathbf{x}$ - position in $\Omega \subset \mathbb{R}^d$.

- $\boldsymbol{\mu}$, $\boldsymbol{\mu}'$ - angle of travel in $\mathbb{S}$.
- E, E' - energies in $\mathbb{E} = [0, \infty)$.
- t - time in $\mathbb{R}^+$.
- $u(\mathbf{x}, \boldsymbol{\mu}, E, t)$ - angular flux at position $\mathbf{x}$ and time t for particles with energy E travelling in direction $\boldsymbol{\mu}$.
- $v(E)$ - particle speed.
- $\theta(\mathbf{x}, \boldsymbol{\mu}' \to \boldsymbol{\mu}, E' \to E, t)$ - differential scattering cross-section. This represents the particles at position $\mathbf{x}$ and time t initially travelling in a direction $\boldsymbol{\mu}'$ with energy E' which then interact with the medium, scattering in direction $\boldsymbol{\mu}$ with energy E.
- $\alpha(\mathbf{x}, \boldsymbol{\mu}, E, t)$ - macroscopic absorption cross-section of the medium. This describes the rate of absorption in the system, with particles of energy E travelling in direction $\boldsymbol{\mu}$ at position $\mathbf{x}$ and time t leaving the system.
- $\beta(\mathbf{x}, \boldsymbol{\mu}, E, t)$ - macroscopic scattering cross-section of the medium. This describes the rate of scattering in the system, with particles of energy E travelling in direction $\boldsymbol{\mu}$ at position $\mathbf{x}$ and time t leaving the system.
- $f(\mathbf{x}, \boldsymbol{\mu}, E, t)$ - source term from outside the space domain Ω.
- $u_0(\mathbf{x}, \boldsymbol{\mu}, E)$ - the initial condition.
- $g(\mathbf{x}, \boldsymbol{\mu}, E, t)$ - Dirichlet boundary condition.
- $\Gamma_- = \{(\mathbf{x}, \boldsymbol{\mu}, E, t) \in \Omega \times \mathbb{S} \times \mathbb{E} \times [0, \infty) : \boldsymbol{\mu} \cdot \mathbf{n}(\mathbf{x}) < 0\}$ - inflow boundary set where $\mathbf{n}(\mathbf{x})$ denotes the outward unit normal to Ω on the boundary $\partial\Omega$. $\int_{\mathbb{E}} \int_{\mathbb{S}} \kappa(\mathbf{x}, \boldsymbol{\mu}' \to \boldsymbol{\mu}, E' \to E, t) u(\mathbf{x}, \boldsymbol{\mu}', E', t)\, d\boldsymbol{\mu}'\, dE'$ - scattering boundary condition. The function $\kappa(\mathbf{x}, \boldsymbol{\mu}' \to \boldsymbol{\mu}, E' \to E, t)$ is an albedo function that describes a particle travelling in a direction $\boldsymbol{\mu}'$ which is deflected along $\boldsymbol{\mu}$ upon interacting with the boundary.

The macroscopic scattering cross-section of the medium can be calculated from the differential scattering cross-section

$$\beta(\mathbf{x}, \boldsymbol{\mu}, E, t) = \int_{\mathbb{E}} \int_{\mathbb{S}} \theta(\mathbf{x}, \boldsymbol{\mu} \to \boldsymbol{\mu}', E \to E', t)\ d\boldsymbol{\mu}'\, dE'.$$

For many practical applications, the angular flux is often not a quantity of interest, but rather another quantity called the scalar flux [62], defined by

$$\phi(\mathbf{x}, E, t) = \int_{\mathbb{S}} u(\mathbf{x}, \boldsymbol{\mu}, E, t)\ d\boldsymbol{\mu}.$$

The scalar flux represents the distribution of particles at a given spatial position $\mathbf{x}$ with energy E at time t moving in any direction.

2.2 Angular discretisation

The first approach we will examine is called the discrete ordinates method. This method approximates the integral over the angular domain by forming a quadrature scheme on the surface of the sphere [8]. This results in a system of coupled transport equations that can then be solved. For the remainder of this report we shall only be concerned with discretisation, other presented here are for completeness sake. Naturally, there is a lot of literature on forming quadrature schemes on the surface of the sphere. We shall detail two key different versions.

2.2.1 Level-symmetric quadrature methods

S_N, or Level-symmetric quadrature methods, involve inserting quadrature points in each octant of the unit sphere, with N being an even positive integer. In these quadratures, the points are arranged on $\frac{N}{2}$ horizontal levels relative to each vertex of the first octant of a unit sphere centred at the origin. The number of points on each ith level is equal to $\frac{N}{2}-i+1$, where $1 \leq i \leq \frac{N}{2}$. The order of a S_N quadrature, N, represents the number of different direction cosines for every axis. In the S_N quadrature, the total number of directions per octant is $\frac{N(N+2)}{8}$, while the total number of directions is $N(N+2)$. Figure 2.1 shows an example of a level-symmetric S_8 quadrature from [60].

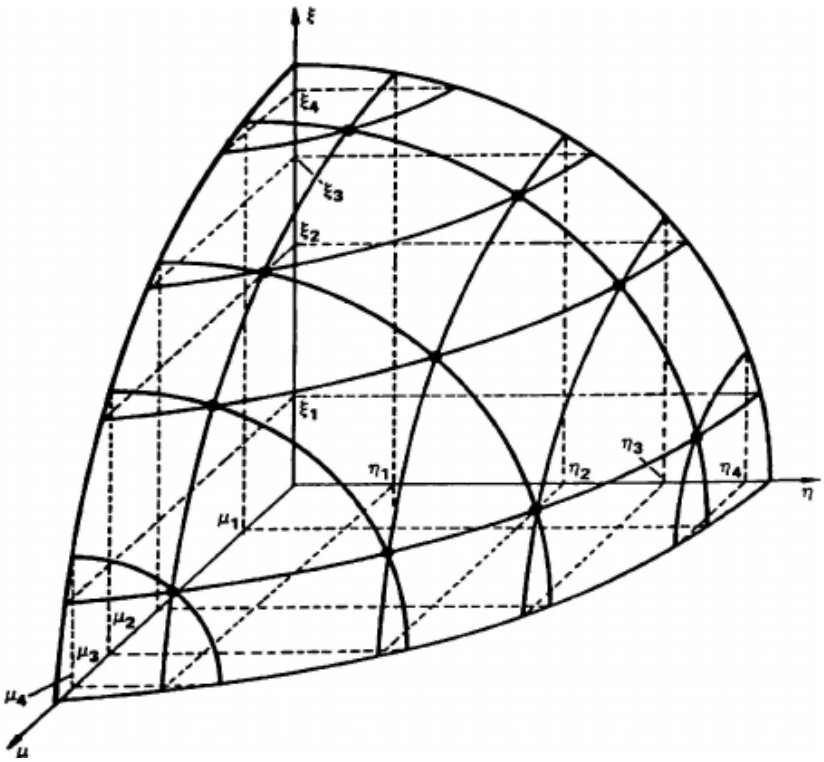

Figure 2.1: S_8 quadrature [60]

2.2.2 T_N quadrature

T_N quadrature is a similar style quadrature scheme. Each octant is treated as the unit triangle. The unit triangle is then subtessellated with N equally sized triangles. The centroid of every small triangle is projected from the origin onto the surface of the sphere, and that is the quadrature point. It is generally accepted that T_N is less efficient than S_N, but it has two interesting features. Firstly, the weights are always positive so any selection of N is viable. Furthermore, it exactly corresponds to the zeroth order discontinuous Galerkin discretisation in angle.

2.2.3 Spherical harmonics

Spherical harmonics are another way to discretise the LBTE. This method is based on truncating the expansion of the angular flux in terms of an orthogonal basis of the unit sphere in d dimensions[62]. The basis in the three-dimensional case, $\mathbb{S} \in \mathbb{R}^3$, may be written in terms of parameters $(\psi, \varphi) \in (0, \pi) \times [0, 2\pi)$

$$\boldsymbol{\mu} = (\sin\psi\cos\varphi, \sin\psi\sin\varphi, \cos\psi).$$

We then express u in terms of $\mathbf{x}$ and our basis, in order that we can then have a spherical harmonic decomposition [62]

$$u(\mathbf{x}, \boldsymbol{\mu}) = u(\mathbf{x}, \psi, \varphi) = \sum_{l=0}^{\infty} \frac{2l+1}{4\pi} \sum_{m=-l}^{l} \phi_{l,m}(\mathbf{x}) Y_{l,m}(\psi, \varphi).$$

The spherical harmonic functions $Y_{l,m}(\psi, \varphi)$ are given as

$$Y_{l,m}(\psi, \varphi) = \sqrt{\frac{(l-m)!}{(l+m)!}} P_l^m(\cos\psi) e^{im\varphi},$$

where each function $P_l^m(\cdot)$ is an associated Legendre function:

$$P_l^m(x) = \begin{cases} (1-x^2)^{\frac{m}{2}} \frac{d^m P_l(x)}{dx^m} & m \geq 0, \\ (-1)^{|m|} P_l^{|m|}(x) & m < 0, \end{cases}$$

and $P_l(\cdot)$ is the l^{th} Legendre polynomial, defined recursively by

$$\begin{aligned} P_0(x) &= 1, \\ P_1(x) &= x, \\ (2l+1)xP_l(x) &= (l+1)P_{l+1}(x) + lP_{l-1}(x) \quad \text{for } l \geq 2. \end{aligned}$$

The spherical harmonic method truncates the series expansion for some non-negative integer N, yielding the approximation:

$$u(\mathbf{x}, \boldsymbol{\mu}) \approx \sum_{i=0}^{N} \frac{2i+1}{4\pi} \sum_{j=-i}^{i} \phi_{i,j}(\mathbf{x}) Y_{i,j}(\boldsymbol{\mu}).$$

The source term and scattering kernel can also be expressed in the same way:

$$f(\mathbf{x}, \boldsymbol{\mu}) \approx \sum_{l'=0}^{N} \frac{2l'+1}{4\pi} \sum_{m'=-l'}^{l'} f_{l',m'}(\mathbf{x}) Y_{l',m'}(\boldsymbol{\mu}),$$
$$\theta(\mathbf{x}, \boldsymbol{\mu} \cdot \boldsymbol{\mu}') \approx \sum_{l'=0}^{N} \frac{2l'+1}{4\pi} \theta_{l'}(\mathbf{x}) P_{l'}(\boldsymbol{\mu} \cdot \boldsymbol{\mu}').$$

By substituting the truncated expansion of $u(\mathbf{x}, \boldsymbol{\mu})$, multiplying by $Y^*_{l,m}(\boldsymbol{\mu})$ and integrating over $\mathbb{S}$ with respect to $\boldsymbol{\mu}$, this gives a set of spherical harmonic moments

$$\{\phi_{l,m}(\mathbf{x}) : 0 \leq l \leq N, -l \leq m \leq l\}$$

that satisfy the following system of first-order PDEs for $0 \leq l \leq N$ and $-l \leq m \leq l$ [62]:

$$\begin{aligned}
&\frac{1}{2l+1}\Bigg[\frac{1}{2}\sqrt{(l+m+2)(l+m+1)}\left(-\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right)\phi_{l+1,m+1}(\mathbf{x}) \\
&\quad + \frac{1}{2}\sqrt{(l-m+2)(l-m+1)}\left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right)\phi_{l+1,m-1}(\mathbf{x}) \\
&\quad + \frac{1}{2}\sqrt{(l-m-1)(l-m)}\left(\frac{\partial}{\partial x} + i\frac{\partial}{\partial y}\right)\phi_{l-1,m+1}(\mathbf{x}) \\
&\quad + \frac{1}{2}\sqrt{(l+m-1)(l+m)}\left(-\frac{\partial}{\partial x} + i\frac{\partial}{\partial y}\right)\phi_{l-1,m-1}(\mathbf{x}) \\
&\quad + \sqrt{(l+m+1)(l-m+1)}\frac{\partial \phi_{l+1,m}(\mathbf{x})}{\partial z} \\
&\quad + \sqrt{(l+m)(l-m)}\frac{\partial \phi_{l-1,m}(\mathbf{x})}{\partial z}\Bigg] + ((\mathbf{x}) + \beta(\mathbf{x}))\phi_{l,m}(\mathbf{x}) \\
&= \theta_l(\mathbf{x})\phi_{l,m}(\mathbf{x}) + f_{l,m}(\mathbf{x}).
\end{aligned} \tag{2.2.1}$$

Setting $\phi_{i,j}(\mathbf{x}) = 0$ for $(i,j) \notin \{(l,m) : 0 \leq l \leq N, -l \leq m \leq l\}$.

2.3 Transport Problems

The transport equation is a linear hyperbolic partial differential equation (PDE), that describes the concentration of some substance flowing in and out of a domain. It has a large number of applications in applied mathematics, physics, and engineering, particularly in fluid dynamics, where it is known as the advection-reaction equation.

Given a bounded open domain Ω of $\mathbb{R}^d, d \geq 1$, with boundary $\partial\Omega$, let $\mathcal{L}$ be the given differential operator and $G(\mathcal{L}, \Omega) = \{v \in L^2(\Omega) : \mathcal{L}v \in L^2(\Omega)\}$ be the graph space

associated with the differential operator over the domain. Then the transport problem is given as:
Find $u \in G(\mathcal{L}, \Omega)$ such that

$$\begin{aligned} \mathcal{L}u = \nabla \cdot (\boldsymbol{a}u) + bu = f \quad & \text{in } \Omega \\ u = g \quad & \text{on the inflow boundary} \end{aligned} \tag{2.3.1}$$

Where $f \in L^2(\Omega)$, $b \in L^\infty(\Omega)$ and $\boldsymbol{a} = \{a_i\}_{i=1}^d$ where a_i are Lipschitz continuous real functions on Ω [35]. As $\mathcal{L}$ is hyperbolic, the inflow and outflow boundaries of the domain are given by

$$\begin{aligned} \partial_- \Omega &= \{x \in \partial\Omega : \boldsymbol{a}(x) \cdot \boldsymbol{n}(x) < 0\} \text{ inflow}, \\ \partial_+ \Omega &= \{x \in \partial\Omega : \boldsymbol{a}(x) \cdot \boldsymbol{n}(x) \geq 0\} \text{ outflow}, \end{aligned} \tag{2.3.2}$$

where $\boldsymbol{n}(x) = \{n_i(x)\}_{i=1}^d$ are the outward facing unit norm on Ω. We introduce the standard hypothesis [35] that there exists some $\gamma > 0$ such that

$$b - \nabla \cdot \boldsymbol{a} \geq \gamma \quad \forall x \in \Omega. \tag{2.3.3}$$

Therefore there exists a c such that

$$c^2 = b - \frac{1}{2}\nabla \ \cdot \boldsymbol{a}. \tag{2.3.4}$$

As this PDE is hyperbolic, the solution is likely to exhibit localised phenomena, such as propagating discontinuities and sharp transition layers. Therefore, classical numerical techniques cannot be used as the schemes lacks sufficient stability. Due to this, we need to introduce some stabilised methods to get a numerically significant approximation. There are a few common approaches, one is to use the streamline upwind Petrov-Galerkin scheme (SUPG) FEM [15], another is Finite Volume, however we shall use the discontinuous Galerkin finite element method (DGFEM). One of the other biggest advantages of DGFEM is its simple handling of non-standard element shapes, such as arbitrary polytopes. As the basis functions are not required to be continuous across elements, they can be defined without reference to the element's shape. Polytopic elements allow for greater flexibility and allow for efficiently designed meshes, especially where complex underlying geometries are present in the problem.

2.3.1 Discontinuous Galerkin finite element method

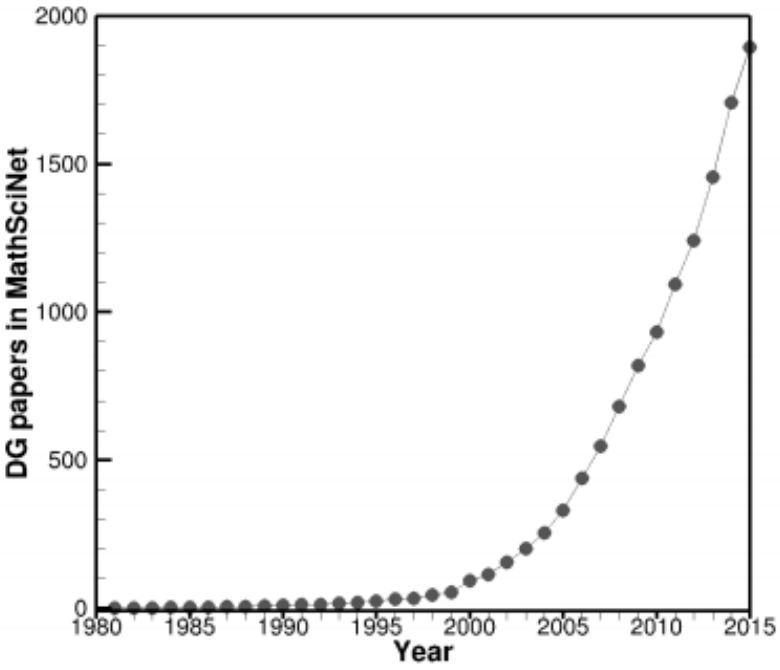

Figure 2.2: The number of papers in MathSciNet, cumulatively, that have discontinuous Galerkin in their title [18].

The discontinuous Galerkin finite element method, referred to as the DGFEM from here on, was introduced in 1973 by W.H. Reed and T.R. Hill [56, 19] as a numerical method to solve the neutron transport equation. This method was first analysed by P. LaSaint and P-A. Raviart [47] in 1974, and then again by C. Johnson and J. Pitkäranta [40] in 1986. Since B. Cockburn and C-W. Shu published [20] in 1989, it has become an increasingly popular area of study, as evident in Figure 2.2, and has been used to solve hyperbolic [63], parabolic [59], and elliptic [7] partial differential equations (PDEs) as well as ordinary differential equations (ODEs) [57].

Let $\mathcal{T}_\Omega$ be a subdivision of the spatial domain Ω into non-overlapping open polytopic elements κ_Ω, such that $\overline{\Omega} = \cup\overline{\kappa_\Omega}$. This is our mesh formed by elements of arbitrary size and shaped polytopes. On each element, we have a set of faces $\mathcal{F}_{\kappa_\Omega} = \{f_i\}_{i=1}^{N_{f\kappa_\Omega}}$, which are defined as the $(d-1)$-dimensional planar facets of the element κ_Ω. Each element will have $N_{f\kappa_\Omega}$ faces, and the mesh's faces are given by $\mathcal{F} = \bigcup \mathcal{F}_{\kappa_\Omega}$. On each of these elements, we need to define a set of basis functions to form a finite element space. Unlike classical finite elements, we do not require these basis functions to have continuity across the elements. Given $\kappa_\Omega \in \mathcal{T}_\Omega$, we denote by $p_{\kappa_\Omega} \geq 0$ the polynomial degree of the basis on κ_Ω, and define the vector $\mathbf{p} := (p_{\kappa_\Omega} : \kappa_\Omega \in \mathcal{T}_\Omega)$. Using this polynomial degree, we can define our polynomial basis functions $\mathbb{H}_{p_{\kappa_\Omega}}(\kappa_\Omega)$. We will consider two versions of the

basis functions: $\mathbb{H}_{p_{\kappa_\Omega}}(\kappa_\Omega) = \mathbb{P}_{p_{\kappa_\Omega}}(\kappa_\Omega)$ denotes the set of polynomial functions with the maximum total degree less than or equal to p_{κ_Ω} on κ_Ω. $\mathbb{H}_{p_{\kappa_\Omega}}(\kappa_\Omega) = \mathbb{Q}_{p_{\kappa_\Omega}}(\kappa_\Omega)$ denotes the set of polynomial functions of maximal degree p_{κ_Ω} in each independent variable on κ_Ω. These basis functions result in different degrees of freedom (DOFs) on each element, with $\mathbb{Q}_{p_{\kappa_\Omega}}$ giving $(p_{\kappa\Omega}+1)^d$ DOFs, and $\mathbb{P}_{p_{\kappa_\Omega}}$ giving $\prod_{i=1}^{d} \frac{p_{\kappa_\Omega}+i}{i}$ DOFs for each element.

We can then define the finite element space, formed of our mesh and basis functions on each element of said mesh:

$$\mathbb{V}_\Omega^{\mathbf{p}} = \{v \in L_2(\Omega) : v|_{\kappa_\Omega} \in \mathbb{H}_{p_{\kappa_\Omega}}(\kappa_\Omega) \text{ for all } \kappa_\Omega \in \mathcal{T}_\Omega\}.$$

The boundary of each element is defined by $\partial\kappa_\Omega$, the union of $(d-1)$–dimensional open faces of the element κ_Ω. Then we can define the inflow and outflow boundary of each element with respect to a

$$\begin{aligned}\partial_-\kappa_\Omega &= \{\mathbf{x} \in \partial\kappa_\Omega : \ \boldsymbol{a} \cdot \boldsymbol{n}_{\kappa_\Omega}(\mathbf{x}) < 0\},\\ \partial_+\kappa_\Omega &= \{\mathbf{x} \in \partial\kappa_\Omega : \ \boldsymbol{a} \cdot \boldsymbol{n}_{\kappa_\Omega}(\mathbf{x}) \geq 0\},\end{aligned}$$

respectively, where $\boldsymbol{n}_{\kappa_\Omega}(\mathbf{x})$ denotes the unit outward normal vector to $\partial\kappa_\Omega$ at $\mathbf{x} \in \partial\kappa_\Omega$. For some $\kappa_\Omega \in \mathcal{T}_\Omega$, the trace of some sufficiently smooth function v on $\partial_-\kappa_\Omega$ from κ_Ω is denoted by $v^+_{\kappa_\Omega}$. Further, if $\partial_-\kappa_\Omega \backslash \partial\Omega$ is non-empty, then for $\mathbf{x} \in \partial_-\kappa_\Omega \backslash \partial\Omega$ there exists a unique $\kappa'_\Omega \in \mathcal{T}_\Omega$ such that $\mathbf{x} \in \partial_+\kappa'_\Omega$; with this notation, we denote by $v^-_{\kappa_\Omega}$ the trace of $v|_{\kappa'_\Omega}$ on $\partial_-\kappa_\Omega \backslash \partial\Omega$. Hence, the upwind jump of the function v across a face $F \subset \partial_-\kappa_\Omega \backslash \partial\Omega$ is denoted by

$$\lfloor v \rfloor := v^+_{\kappa_\Omega} - v^-_{\kappa_\Omega}.$$

We can then find the weak (variational) formulation of the transport problem by integrating by parts we get:

$$\begin{aligned}&\sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \left(\int_{\kappa_\Omega} -\boldsymbol{a} u_h \cdot \nabla v_h + b u_h v_h \ d\mathbf{x} + \int_{\partial\kappa_\Omega} \boldsymbol{a} \cdot \boldsymbol{n}_{\kappa_\Omega} uv \ ds \right)\\ &= \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\kappa_\Omega} f v_h \ d\mathbf{x} \quad \forall v_h \in \mathbb{V}_\Omega^{\mathbf{p}}.\end{aligned}$$

Introducing a numerical flux $H(u_h^+, u_h^-, \mathbf{n}_{\kappa_\Omega})$ gives the weak formulation:
Find $u_h \in \mathbb{V}_\Omega^{\mathbf{p}}$ such that:

$$\begin{aligned}&\sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \left(\int_{\kappa_\Omega} -a u_h \cdot \nabla v_h + b u_h v_h \ d\mathbf{x} + \int_{\partial\kappa_\Omega} H(u_h^+, u_h^-, \mathbf{n}_{\kappa_\Omega}) v^+ \ ds \right)\\ &= \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\kappa_\Omega} f v_h \ d\mathbf{x} \quad \forall v_h \in \mathbb{V}_\Omega^{\mathbf{p}}.\end{aligned}$$

Here, $H(\cdot,\cdot,\cdot)$ denotes some numerical flux. We typically require the following properties

[16] for the numerical flux.

- Consistency: $H(u, u, \boldsymbol{n}_{\kappa\Omega})|_{\partial\kappa\Omega} = (au) \cdot \boldsymbol{n}_{\kappa\Omega}$.
- Conservation: for two neighbouring elements κ_Ω and $\kappa'_\Omega \in \mathcal{T}_h$ at each point $x \in \partial_{\kappa\Omega} \cap \partial_{\kappa'_\Omega} \neq \emptyset$, and as $\boldsymbol{n}_{\kappa\Omega} = -\boldsymbol{n}_{\kappa'_\Omega}$, then $H(u, v, \boldsymbol{n}_{\kappa\Omega}) = -H(u, v, \boldsymbol{n}_{\kappa\Omega})$.

The upwind flux is a common choice for the numerical flux. For some $\kappa_\Omega \in \mathcal{T}_\Omega$

$$H(u^+, u^-, \mathbf{n}_{\kappa\Omega})|_{\partial\kappa\Omega} = \begin{cases} a \cdot \mathbf{n}_{\kappa\Omega}\ u^+ & \mathbf{x} \in \partial_+\kappa\Omega, \\ a \cdot \mathbf{n}_{\kappa\Omega}\ u^- & \mathbf{x} \in \partial_-\kappa\Omega \setminus \partial_-\Omega, \\ a \cdot \mathbf{n}_{\kappa\Omega}\ g & \mathbf{x} \in \partial_-\kappa\Omega \cap \partial_-\Omega. \end{cases}$$

Putting this upwind flux into our weak formulation, we get the DGFEM for the transport problem:
Find $u_h \in \mathbb{V}_\Omega^{\mathbf{p}}$

$$B(u_h, v_h) = \ell(v_h) \quad \forall v_h \in \mathbb{V}_\Omega^{\mathbf{p}},$$

where

$$B(u_h, v_h) = \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\kappa_\Omega} -au_h \cdot \nabla v_h + bu_h v_h \ d\mathbf{x} \tag{2.3.5}$$

$$+ \int_{\partial_+\kappa_\Omega} \boldsymbol{a} \cdot \boldsymbol{n}_{\kappa\Omega} u_h^+ v_h^+ \, ds + \int_{\partial_-\kappa_\Omega \setminus \partial_-\Omega} \boldsymbol{a} \cdot \boldsymbol{n}_{\kappa\Omega} u_h^- v_h^+ \, ds \tag{2.3.6}$$

$$\ell(v_h) = \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\kappa_\Omega} f v_h \ d\mathbf{x} + \int_{\partial_-\kappa_\Omega \cap \partial_-\Omega} \boldsymbol{a} \cdot \boldsymbol{n}_{\kappa\Omega} g v_h^+ \, ds. \tag{2.3.7}$$

It is often beneficial, especially in terms of implementation, to rewrite the DGFEM formulation in terms of faces and elements, rather than just elements.

Let $\mathcal{F}_\Omega$ denote the collection of element faces in $\mathcal{T}_\Omega$. We can divide these faces into two sets, ${\mathcal{F}_\Omega}^\partial$ are the faces that only have one element connected to them, that is to say the face is on the boundary of the domain. ${\mathcal{F}_\Omega}^{int}$ are the set of faces that have two elements connected to them, they are referred to as the internal faces. Obviously ${\mathcal{F}_\Omega}^\partial \cup {\mathcal{F}_\Omega}^{int} = \mathcal{F}_\Omega$. We can define the boundary faces as either inflow or outflow boundary faces for a given a,

$${\mathcal{F}_\Omega}^+ = \{{\mathcal{F}_\Omega}^\partial \cup \partial_+\Omega\}, \tag{2.3.8}$$

$${\mathcal{F}_\Omega}^- = \{{\mathcal{F}_\Omega}^\partial \cup \partial_-\Omega\}. \tag{2.3.9}$$

So that ${\mathcal{F}_\Omega}^\partial = {\mathcal{F}_\Omega}^+ \cup {\mathcal{F}_\Omega}^-$.

$$\begin{aligned}
&\sum_{\kappa_\Omega} \int_{\kappa_\Omega} -a u_h \cdot \nabla v_h + b u_h v_h \, d\mathbf{x} \\
&+ \sum_{\mathcal{F}_\Omega} \left(\int_{{\mathcal{F}_\Omega}^-} a \boldsymbol{n}_{\mathcal{F}_\Omega} u_h^+ v_h^+ - \int_{{\mathcal{F}_\Omega}^{int}} a \boldsymbol{n}_{\mathcal{F}_\Omega} \lfloor u_h \rfloor v_h^+ \right) \\
&= \sum_{\kappa_\Omega} \int_{\kappa_\Omega} f v_h \, d\mathbf{x} + \sum_{\mathcal{F}} \int_{{\mathcal{F}_\Omega}^+} a \boldsymbol{n}_{\mathcal{F}_\Omega} g \, dS.
\end{aligned} \tag{2.3.10}$$

Where $\boldsymbol{n}_{\mathcal{F}_\Omega}$ is the unit outward normal vector of one of the elements whose union forms that face, which one is picked is arbitrary, but the choice must be consistent across every face.

2.3.2 Properties of discontinuous Galerkin finite element method

DGFEM has many useful properties that give advantages over many other numerical methods. One of these properties is Galerkin Orthogonality As we have

$$B(u, v) = \ell(v) \quad \forall v \in G(\mathcal{L}, \Omega) \quad \text{and} \quad B(u_h, v_h) = \ell(v_h) \quad \forall v_h \in \mathbb{V}_\Omega^{\mathbf{p}},$$

and u is sufficiently smooth, we get

$$B(u, v_h) = \ell(v_h) \quad \forall v_h \in \mathbb{V}_\Omega^{\mathbf{p}}.$$

Therefore,

$$\begin{aligned}
B(u - u_h, v_h) &= B(u, v_h) - B(u_h, v_h) \\
&= \ell(v_h) - \ell(v_h) \\
&= 0 \quad \forall v_h \in \mathbb{V}_\Omega^{\mathbf{p}}.
\end{aligned}$$

One of the advantages of the DGFEM is that we can show that Galerkin Orthogonality holds element-wise as well.

$$B(u, v) = \ell(v) \quad \forall v \in G(\mathcal{L}, \kappa_\Omega) \quad \text{and} \quad B(u_h, v_h) = \ell(v_h) \quad \forall v_h \in \mathbb{V}_\Omega^{\mathbf{p}}|_{\kappa_\Omega}.$$

By the same arguments as before, we get

$$B(u - u_h, v_h)|_{\kappa_\Omega} = 0 \quad \forall v_h \in \mathbb{V}_\Omega^{\mathbf{p}}|_{\kappa_\Omega}.$$

DGFEM was shown to be stable for transport problems in [34]. Using our standard hypothesis (2.3.3) and (2.3.4) we can show well-posedness of the problem. Then u_h is

constrained by this bound

$$\sum_{\kappa_\Omega}\left(\|cu_h\|^2_{L^2(\kappa_\Omega)}+\|u_h^+-u_h^-\|^2_{\partial_-\kappa_\Omega\setminus\partial\Omega}+\|u_h^+\|^2_{\partial_+\kappa_\Omega\cap\partial\Omega}+\frac{1}{2}\|u_h^+\|^2_{\partial_-\kappa_\Omega\cap\partial_-\Omega}\right)$$
$$\leq\sum_{\kappa_\Omega}\left(\|c^{-1}f\|^2_{L^2(\kappa_\Omega)}+2\|g\|^2_{\partial_-\kappa_\Omega\cap\partial_-\Omega}\right).$$

Where

$$\|u\|_\tau=\left(\int_\tau a\cdot\boldsymbol{n}_{\kappa_\Omega}u^2dS\right)^{\frac{1}{2}}_\tau \text{ for some } \tau\subset\partial\kappa_\Omega.$$

This bound implies the existence and uniqueness of the solution to DGFEM for a transport problem [35].

We can now introduce the discontinuous Galerkin (DG) norm [35].

$$\|w\|^2_{DG}=\sum_{\kappa_\Omega}\left(\|cw\|^2_{L^2(\kappa_\Omega)}+\frac{1}{2}\|w^+\|^2_{\partial_-\kappa_\Omega\cap\partial_-\Omega}+\frac{1}{2}\|w^+-w^-\|^2_{\partial_-\kappa_\Omega\setminus\partial\Omega}+\frac{1}{2}\|w^+\|^2_{\partial_+\kappa_\Omega\cap\partial\Omega}\right). \tag{2.3.11}$$

Using this, we can get an a-priori error bound from [35]:
If γ defined above exists and

$$a\cdot\nabla_{\mathcal{T}}v_h\in\mathbb{V}^{\mathbf{p}}_\Omega\quad\forall v_h\in\mathbb{V}^{\mathbf{p}}_\Omega,$$

where $\nabla_{\mathcal{T}}u$ is the broken gradient of

$$u$$

, $\nabla_{\mathcal{T}}u|_{\kappa_\Omega}=\nabla(u|_{\kappa_\Omega})$, and that solution u exists in some Sobolov space $H^k(\kappa_\Omega)$ for some $k\geq 1$ then for a uniform $\boldsymbol{p}$

$$\|u-u_h\|_{DG}\leq C\left(\frac{h}{\boldsymbol{p}+1}\right)^{s-\frac{1}{2}}|u|_{s,\mathcal{T}_h}. \tag{2.3.12}$$

Where $1\leq s\geq\min(\boldsymbol{p}+1,k)$.

Hence, the above hp-bound is optimal in h and suboptimal in $\boldsymbol{p}$ by $\boldsymbol{p}^{\frac{1}{2}}$. This implies the rate of convergence for the error in the DG norm is $O(h^{\boldsymbol{p}+\frac{1}{2}})$. The error in the L^2 norm is similarly suboptimal on certain meshes [52] but on most meshes is said to be $O(h^{\boldsymbol{p}+1})$ [49]. Using a simple example of the transport problem on $\Omega=(0,1)^2$,

$$\begin{aligned}\nabla\cdot(au)+u&=2\pi\cos(2\pi x)\sin(2\pi y)+2\pi\cos(2\pi y)\sin(2\pi x)+\sin(2\pi x)\sin(2\pi y)\quad\text{in }\Omega\\ u&=0\quad\text{on }\partial_-\Omega.\end{aligned} \tag{2.3.13}$$

Where $a=(1,1)^T$. This PDE was chose as it has an analytical solution $u=\sin(2\pi x)\sin(2\pi y)$.

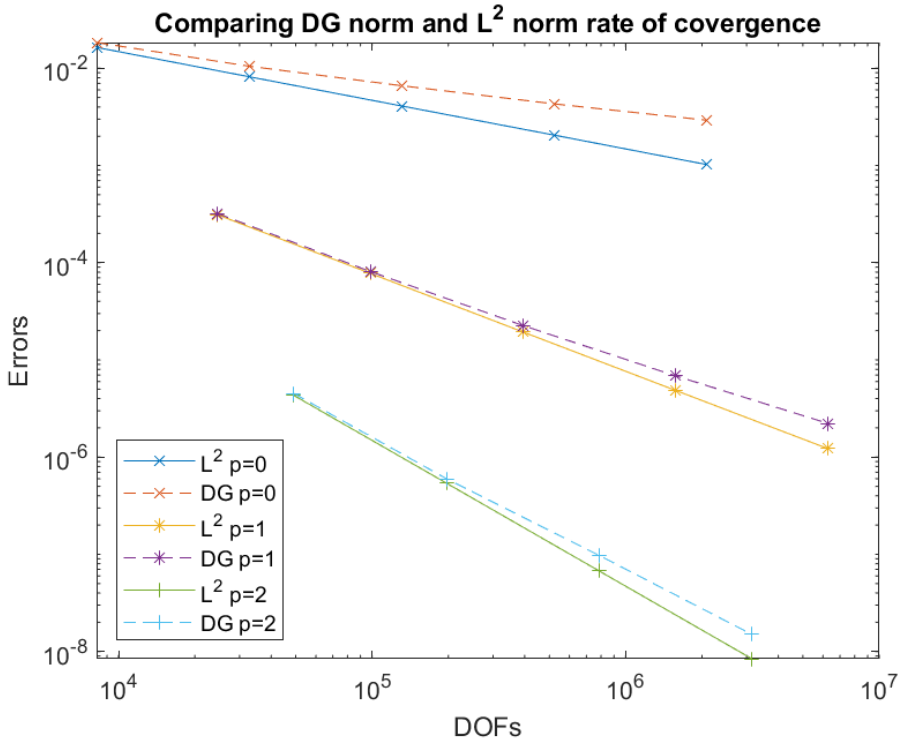

Figure 2.3: Comparing DG norm and L^2 norm rate of covergence

Figure 2.3 shows the different rate of convergence of the errors in the L^2 and DG norms.

Chapter 3

Discontinuous Galerkin finite element method for the time independent linear Boltzmann transport equation

In this chapter, we discuss how we plan to discretise each of the domains (space, angle, and energy) of the time independent LBTE. Having now introduced the notion of DGFEM, we can apply it in the case of the LBTE. For simplicity, we will deal exclusively with the steady state (time-independent) LBTE, reducing it to six dimensions. The spatial domain Ω, describes the location of the particles, and has three dimensions. The angular domain $\mathbb{S}$ gives the direction of travel of the particles in 3D but is described by polar and azimuthal angles, making it 2D only. The energy domain $\mathbb{E}$ describes the amount of energy the particles possess, and is 1D. We then compare the DGFEM discretisation of the angular domain to the commonly used discrete ordinates as described in the previous Chapter 2.2. Having done so, we introduce our new angular discretisation, DOG, which retains the simple matrix structure of discrete ordinates with the high order DG.

3.1 Time-independent linear Boltzmann transport equation

Given an open bounded polyhedral spatial domain $\Omega \subset \mathbb{R}^d$ for $d = 2$ or 3, with $\partial\Omega$ signifying the union of its $(d-1)$-dimensional open faces, let $\mathcal{D} = \Omega \times \mathbb{S} \times \mathbb{E}$, where $\mathbb{S} = \{\boldsymbol{\mu} \in \mathbb{R}^d : |\boldsymbol{\mu}|_2 = 1\}$ denotes the surface of the d-dimensional unit sphere and $\mathbb{E} = \{E \in \mathbb{R} : E \geq 0\}$ is the real half line.

Then, the time independent linear Boltzmann transport problem is given by:
Find $u : \mathcal{D} \to \mathbb{R}$ such that

$$\begin{aligned} \boldsymbol{\mu} \cdot \nabla_{\mathbf{x}} u(\mathbf{x}, \boldsymbol{\mu}, E) + (\alpha(\mathbf{x}, \boldsymbol{\mu}, E) + \beta(\mathbf{x}, \boldsymbol{\mu}, E)) u(\mathbf{x}, \boldsymbol{\mu}, E) &= \mathcal{S}[u](\mathbf{x}, \boldsymbol{\mu}, E) \\ &\quad + f(\mathbf{x}, \boldsymbol{\mu}, E) \text{ in } \mathcal{D}, \\ u(\mathbf{x}, \boldsymbol{\mu}, E) &= g_{\mathrm{D}}(\mathbf{x}, \boldsymbol{\mu}, E) \text{ on } \Gamma_{\mathrm{in}}, \end{aligned} \tag{3.1.1}$$

where $f, g, \alpha : \mathcal{D} \to \mathbb{R}$ are problem specific data terms, $\nabla_{\mathbf{x}}$ is the spatial gradient operator, and $\Gamma_{\mathrm{in}} = \{(\mathbf{x}, \boldsymbol{\mu}, E) \in \bar{\mathcal{D}} : \mathbf{x} \in \partial\Omega \text{ and } \boldsymbol{\mu} \cdot \boldsymbol{n} < 0\}$ denotes the inflow boundary of $\mathcal{D}$, where $\boldsymbol{n}$ denotes the unit outward normal vector on the boundary $\partial\Omega$. $\mathcal{S}[u]$ denotes the scattering operator on u and is given by:

$$\mathcal{S}[u](\mathbf{x}, \boldsymbol{\mu}, E) = \int_{\mathbb{E}} \int_{\mathbb{S}} \theta(\mathbf{x}, \boldsymbol{\eta} \to \boldsymbol{\mu}, E' \to E) u(\mathbf{x}, \boldsymbol{\eta}, E') \, d\boldsymbol{\eta} \, dE',$$

where θ is the scattering kernel given by the problem. The θ usually defined by the type of particle interaction in the material present. $\boldsymbol{\eta} \to \boldsymbol{\mu}$ is a shorthand for the incoming and outgoing angular direction of the interaction often given as a scattering angle $\cos(\vartheta) = \boldsymbol{\eta} \cdot \boldsymbol{\mu}$. Simiarily the incoming and outgoing energy is shown as $E' \to E$. The total scattering data term β can be calculated as:

$$\beta(\mathbf{x}, \boldsymbol{\mu}, E) = \int_{\mathbb{E}} \int_{\mathbb{S}} \theta(\mathbf{x}, \boldsymbol{\mu} \to \boldsymbol{\eta}, E \to E') \, d\boldsymbol{\eta} \, dE'.$$

From the physical interpretation of the LBTE we know that particles can only lose energy during a scattering interaction and thus $\theta(\mathbf{x}, \boldsymbol{\eta} \to \boldsymbol{\mu}, E' \to E) = 0$ for $E < E'$. The other interaction that the particles can go through is absorption, denoted by α. Similarly, we presume the problem is angularly isotropic, so $\alpha(\mathbf{x}, \boldsymbol{\mu}, E) \geq 0$ for all $\mathbf{x} \in \Omega$ and $E \in \mathbb{E}$. Finally, we assume there exists some constant c_0 such that

$$c = \alpha(\mathbf{x}, \boldsymbol{\mu}, E) + \frac{1}{2}(\beta(\mathbf{x}, \boldsymbol{\mu}, E) - \gamma(\mathbf{x}, \boldsymbol{\mu}, E)) \geq c_0 > 0, \tag{3.1.2}$$

where

$$\gamma(\mathbf{x}, \boldsymbol{\mu}, E) = \int_{\mathbb{E}} \int_{\mathbb{S}} \theta(\mathbf{x}, \boldsymbol{\eta} \to \boldsymbol{\mu}, E' \to E) \, d\boldsymbol{\eta} \, dE'$$

3.2 Energy discretisation

3.2.1 Multigroup

Discretising the energy domain of the LBTE has historically been done by a multi-group approximation [50]. This method first restricts the domain $\mathbb{E}$ to a finite interval

$(E_{\min}, E_{\max})$. Then it creates a series of sub-intervals between $(E_{\min}, E_{\max})$ such that

$$E_{\max} = E_0 > E_1 > \ldots > E_{N_{\mathbb{E}}-1} > E_{N_{\mathbb{E}}} = E_{\min}$$

for some $N_{\mathbb{E}} \in \mathbb{N}$. Each energy group g has a function u_g defined by

$$u_g(\mathbf{x}, \boldsymbol{\mu}) = \int_{E_g}^{E_{g-1}} u(\mathbf{x}, \boldsymbol{\mu}, E) dE,$$

and a weighting function ω_g such that

$$u(\mathbf{x}, \boldsymbol{\mu}, E) \approx \omega_g(E) u_g(\mathbf{x}, \boldsymbol{\mu}) \text{ for } E \in [E_g, E_{g-1}],$$

$$\int_{E_g}^{E_{g-1}} \omega_g(E) dE = 1.$$

The simplest example of $\omega_g(E)$ is

$$\omega_g(E) = \frac{1}{E_{g-1} - E_g}$$

We can then define data terms for each group $1 \leq g, g' \leq N_{\mathbb{E}}$ as follows:

$$\begin{aligned}
\alpha_g(\mathbf{x}, \boldsymbol{\mu}) &= \int_{E_g}^{E_{g-1}} \omega_g(E) \alpha(\mathbf{x}, \boldsymbol{\mu}, E)\, dE, \\
\beta_g(\mathbf{x}, \boldsymbol{\mu}) &= \int_{E_g}^{E_{g-1}} \omega_g(E) \beta(\mathbf{x}, \boldsymbol{\mu}, E)\, dE, \\
\theta_{g' \to g}(\mathbf{x}, \boldsymbol{\mu}' \to \boldsymbol{\mu}) &= \int_{E_g}^{E_{g-1}} \int_{E_{g'}}^{E_{g'-1}} w(E') \theta(\mathbf{x}, \boldsymbol{\mu}' \to \boldsymbol{\mu}, E' \to E)\, dE'\, dE, \\
f_g(\mathbf{x}, \boldsymbol{\mu}) &= \int_{E_g}^{E_{g-1}} \omega_g(E) f(\mathbf{x}, \boldsymbol{\mu}, E)\, dE.
\end{aligned}$$

Using these we get a coupled system of mono-energetic LBTE problems for $\{u_g\}_{g=1}^{N_{\mathbb{E}}}$

$$\begin{aligned}
&\boldsymbol{\mu} \cdot \nabla_{\mathbf{x}} u_g(\mathbf{x}, \boldsymbol{\mu}) + (\alpha_g(\mathbf{x}, \boldsymbol{\mu}) + \beta_g(\mathbf{x}, \boldsymbol{\mu}))\, u_g(\mathbf{x}, \boldsymbol{\mu}) \\
&\qquad = \sum_{g'=1}^{N_{\mathbb{E}}} \int_{\mathbb{S}} \theta_{g' \to g}(\mathbf{x}, \boldsymbol{\mu}' \to \boldsymbol{\mu}) u_{g'}(\mathbf{x}, \boldsymbol{\mu}')\, d\boldsymbol{\mu}' + f_g(\mathbf{x}, \boldsymbol{\mu}).
\end{aligned}$$

Thus, the multigroup approximation gives the poly-energetic LBTE as a system of coupled mono-energetic LBTEs. As we assume not particle can gain energy from an interaction, the ordering matters for the multigroup method, we must first solve for the fluence in the highest energy group, and then subsequently for each lower energy group in turn.

3.2.2 Discontinuous Galerkin (DG) in Energy

We can easily define a mesh using the same energy groups, as in multigroup

$$\mathcal{T}_{\mathbb{E}} = \{\kappa_g\}_{g=1}^{N_{\mathbb{E}}},$$

where the interval $\kappa_g = (E_g, E_{g-1})$ is called the energy group g, $g = 1, \ldots, N_{\mathbb{E}}$. For every energy group κ_g, $g = 1, \ldots, N_{\mathbb{E}}$, we give a polynomial degree $r_g \geq 0$. Defining $\mathbf{r} = (r_{\kappa_g})_{g=1}^{N_{\mathbb{E}}}$, we introduce the energy finite element space

$$\mathbb{V}_{\mathbb{E}}^{\mathbf{r}} = \{v \in L_2(E_{\min}, E_{\max}) : v|_{\kappa_g} \in \mathbb{P}_{r_{\kappa_g}}(\kappa_g) \text{ for all } \kappa_g \in \mathcal{T}_{\mathbb{E}}\}.$$

3.3 Angular discretisation

There are several different discretisations of the angular domain that are used in the literature. We will be focusing on two different methods, discrete ordinates, as mentioned in Chapter 2 and finite element methods, specifically the discontinuous Galerkin finite element method.

3.3.1 Discrete Ordinates (DO)

The discrete ordinates method involves dividing the angular domain into a finite set of discrete angles [8], denoted by $\boldsymbol{\mu}_n$ for $n = 1, ..., N$. The angles $\boldsymbol{\mu}_n$ act as quadrature points, used to integrate over the angular domain

$$\int_{\mathbb{S}} f(\mathbf{x}, \boldsymbol{\mu}) \approx \sum_{n=1}^{N} \omega_n f(\mathbf{x}, \boldsymbol{\mu}_n),$$

where ω_n are the associated quadrature weights. The radiation transport equation can then be discretised into a set of N transport equations:

$$\boldsymbol{\mu}_n \cdot \nabla_{\mathbf{x}} u(\mathbf{x}, \boldsymbol{\mu}_n) + (\alpha(\mathbf{x}, \boldsymbol{\mu}_n) + \beta(\mathbf{x}, \boldsymbol{\mu}_n)) u(\mathbf{x}, \boldsymbol{\mu}_n) = \sum_{n=1}^{N} \omega_n \theta(\mathbf{x}, \boldsymbol{\eta}_n \to \boldsymbol{\mu}_n) u(\mathbf{x}, \boldsymbol{\eta}_n) + f(\mathbf{x}, \boldsymbol{\mu}_n),$$

for $1 \leq n \leq N$. All N transport equations are coupled via the scattering operator. There are many choices of quadrature points and weights, but in general we want a quadrature scheme with the following properties [45]:

- Quadrature points should be located within the domain of integration, and the weights should all be positive.
- Any rotation of the arrangement of the quadrature points must have no effect.

- The principle of optical reciprocity

$$\theta(\mathbf{x}, \boldsymbol{\eta}_n \to \boldsymbol{\mu}_n) = \theta(\mathbf{x}, \boldsymbol{\mu}_n \to \boldsymbol{\eta}_n),$$

  must be maintained for the quadrature scheme.

- Quadrature points and weights must satisfy

$$\sum_{n=1}^{N} \omega_n \boldsymbol{\mu}_n = 0.$$

There are numerous quadrature schemes that satisfy these conditions. For the rest of this section, we will be focusing on a common type of discrete ordinates T_N, which we introduced in Section 2.2. This discrete ordinate system was chosen to allow easy comparison to DG in angle.

3.3.2 Discontinuous Galerkin (DG) in Angle

Similar to DG for the spatial domain, we need to define a mesh and finite element space over the angular domain. As the angular domain $\mathbb{S}$ is the surface of the unit circle in $\mathbb{R}^2$ and the unit sphere in $\mathbb{R}^3$, defining a mesh presents a challenge that is well documented in the literature [4, 22, 24]. Many mappings will become singular at the poles and force elements near to the poles to have degenerate faces. In this work, a cube-sphere mesh over the angular domain $\mathbb{S}$ is used. A cube-sphere mesh requires a mesh of the surface of the d-dimensional unit cube that is then mapped, using a smooth and invertible mapping, on to the unit sphere. We denote a mesh $\tilde{\mathcal{T}}_{\mathbb{S}} = \{\tilde{\kappa}_{\mathbb{S}}\}$ on an approximation of the surface of a sphere $\mathbb{S}_h$, i.e., $\mathbb{S}_h = \cup_{\tilde{\kappa}_{\mathbb{S}} \in \tilde{\mathcal{T}}_{\mathbb{S}}} \tilde{\kappa}_{\mathbb{S}}$. We introduce a smooth invertible mapping, $\phi_{\mathbb{S}} : \mathbb{S}_h \to \mathbb{S}$, assuming the surface is star-shaped with respect to the origin, defined as $\phi_{\mathbb{S}}(\tilde{\boldsymbol{\mu}}) = |\tilde{\boldsymbol{\mu}}|_2^{-1} \tilde{\boldsymbol{\mu}}$, where $|\cdot|_2$ denotes the l_2-norm. The mesh on $\mathbb{S}$ can then be defined as

$$\mathcal{T}_{\mathbb{S}} = \left\{ \kappa_{\mathbb{S}} : \kappa_{\mathbb{S}} = \phi_{\mathbb{S}}(\tilde{\kappa}_{\mathbb{S}}) \ \forall \tilde{\kappa}_{\mathbb{S}} \in \tilde{\mathcal{T}}_{\mathbb{S}} \right\}.$$

We assume that elements $\tilde{\kappa}_{\mathbb{S}} \in \tilde{\mathcal{T}}_{\mathbb{S}}$ are mapped to $\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}$, without any significant rescaling. Assuming $\phi_{\kappa_{\mathbb{S}}} : \hat{\kappa}_{\mathbb{S}} \to \tilde{\kappa}_{\mathbb{S}}$ for some reference element $\hat{\kappa}_{\mathbb{S}} \subset \mathbb{R}^{d-1}$ is affine, we can define $F_{\kappa_{\mathbb{S}}} : \hat{\kappa}_{\mathbb{S}} \to \kappa_{\mathbb{S}}$ by $F_{\kappa_{\mathbb{S}}} = \phi_{\mathbb{S}} \circ \phi_{\kappa_{\mathbb{S}}}$ for some reference element $\hat{\kappa}_{\mathbb{S}}$.
Then, for each $\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}$, we write $q_{\kappa_{\mathbb{S}}} \geq 0$ to denote the polynomial degree employed on $\kappa_{\mathbb{S}}$ and write $\mathbf{q} := (q_{\kappa_{\mathbb{S}}} : \kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}})$. The corresponding finite element space defined on the surface of the sphere $\mathbb{S}$ is given by

$$\mathbb{V}_{\mathbb{S}}^{\mathbf{q}} = \{v \in L_2(\mathbb{S}) : v|_{\kappa_{\mathbb{S}}} = \hat{v} \circ F_{\kappa_{\mathbb{S}}}^{-1}, \ \hat{v} \in \mathbb{H}_k(\hat{\kappa}_{\mathbb{S}}) \text{ for all } \kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}\},$$

here $\mathbb{H}_k(\hat{\kappa}_\mathbb{S})$ are the polynomial basis functions on the angular element. The basis functions are of the form $\mathbb{P}_k(\hat{\kappa}_\mathbb{S})$ or $\mathbb{Q}_k(\hat{\kappa}_\mathbb{S})$ as defined in Section 2.3.1.

3.4 DG-DG-DG

Employing the definitions introduced in the previous sections, we define the full space-angle-energy mesh by taking a tensor product of the meshes.

$$\mathcal{T} = \mathcal{T}_\Omega \times \mathcal{T}_\mathbb{S} \times \mathcal{T}_\mathbb{E} = \{\kappa : \kappa = \kappa_\Omega \times \kappa_\mathbb{S} \times \kappa_g,\ \kappa_\Omega \in \mathcal{T}_\Omega,\ \kappa_\mathbb{S} \in \mathcal{T}_\mathbb{S},\ \kappa_g \in \mathcal{T}_\mathbb{E}\}.$$

Likewise, we can obtain our space-angle-energy finite element space by taking the tensor product of the finite element space over each different domain.

$$\mathbb{V}^{\mathbf{p},\mathbf{q},\mathbf{r}}_{\Omega,\mathbb{S},\mathbb{E}} = \mathbb{V}^{\mathbf{p}}_{\Omega} \otimes \mathbb{V}^{\mathbf{q}}_{\mathbb{S}} \otimes \mathbb{V}^{\mathbf{r}}_{\mathbb{E}},$$

and, for any $\boldsymbol{\mu} \in \mathbb{S}$, let $\mathcal{G}_{\boldsymbol{\mu},h} = \{v \in L_2(\Omega) : \boldsymbol{\mu} \cdot \nabla_\mathbf{x} v|_{\kappa_\Omega} \in L_2(\kappa_\Omega) \text{ for all } \kappa_\Omega \in \mathcal{T}_\Omega\}$ denote the broken spatial graph space. Exploiting the tensor structure of our domain $\mathcal{D}$ and our finite element space $\mathbb{V}^{\mathbf{p},\mathbf{q},\mathbf{r}}_{\Omega,\mathbb{S},\mathbb{E}}$, allows us to finally introduce the DGFEM:
Find $u_h \in \mathbb{V}^{\mathbf{p},\mathbf{q},\mathbf{r}}_{\Omega,\mathbb{S},\mathbb{E}}$ such that

$$\begin{aligned}
&\sum_{\kappa_g \in \mathcal{T}_\mathbb{E}} \int_{\kappa_g} \sum_{\kappa_\mathbb{S} \in \mathcal{T}_\mathbb{S}} \int_{\kappa_\mathbb{S}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\kappa_\Omega} (\boldsymbol{\mu} \cdot \nabla_\mathbf{x} u_h v_h + (\alpha + \beta) u_h v_h)\, d\mathbf{x}\, d\boldsymbol{\mu}\, dE \\
&- \sum_{\kappa_g \in \mathcal{T}_\mathbb{E}} \int_{\kappa_g} \sum_{\kappa_\mathbb{S} \in \mathcal{T}_\mathbb{S}} \int_{\kappa_\mathbb{S}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\partial_- \kappa_\Omega \setminus \partial\Omega} (\boldsymbol{\mu} \cdot \boldsymbol{n}_{\kappa_\Omega})(u_h^+ - u_h^-) v_h^+\, ds\, d\boldsymbol{\mu}\, dE \\
&- \sum_{\kappa_g \in \mathcal{T}_\mathbb{E}} \int_{\kappa_g} \sum_{\kappa_\mathbb{S} \in \mathcal{T}_\mathbb{S}} \int_{\kappa_\mathbb{S}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\partial_- \kappa_\Omega \cap \partial\Omega} (\boldsymbol{\mu} \cdot \boldsymbol{n}_{\kappa_\Omega}) u_h^+ v_h^+\, ds\, d\boldsymbol{\mu}\, dE \\
&= \sum_{\kappa_g \in \mathcal{T}_\mathbb{E}} \int_{\kappa_g} \sum_{\kappa_\mathbb{S} \in \mathcal{T}_\mathbb{S}} \int_{\kappa_\mathbb{S}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\kappa_\Omega} \mathcal{S}[u_h](\mathbf{x}, \boldsymbol{\mu}, E) v_h\, d\mathbf{x}\, d\boldsymbol{\mu}\, dE \\
&+ \sum_{\kappa_g \in \mathcal{T}_\mathbb{E}} \int_{\kappa_g} \sum_{\kappa_\mathbb{S} \in \mathcal{T}_\mathbb{S}} \int_{\kappa_\mathbb{S}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\kappa_\Omega} f v_h\, d\mathbf{x}\, d\boldsymbol{\mu}\, dE \\
&- \sum_{\kappa_g \in \mathcal{T}_\mathbb{E}} \int_{\kappa_g} \sum_{\kappa_\mathbb{S} \in \mathcal{T}_\mathbb{S}} \int_{\kappa_\mathbb{S}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\partial_- \kappa_\Omega \cap \partial\Omega} (\boldsymbol{\mu} \cdot \boldsymbol{n}_{\kappa_\Omega}) g_\mathrm{D} v_h\, ds\, d\boldsymbol{\mu}\, dE \quad \forall v_h \in \mathbb{V}^{\mathbf{p},\mathbf{q},\mathbf{r}}_{\Omega,\mathbb{S},\mathbb{E}}.
\end{aligned}$$

This can be rewritten as:
Find $u_h \in \mathbb{V}^{\mathbf{p},\mathbf{q},\mathbf{r}}_{\Omega,\mathbb{S},\mathbb{E}}$ such that

$$b(u_h, v_h) \equiv a(u_h, v_h) - s(u_h, v_h) = \ell(v_h) \tag{3.4.1}$$

for all $v_h \in \mathbb{V}^{\mathbf{p},\mathbf{q},\mathbf{r}}_{\Omega,\mathbb{S},\mathbb{E}}$, where $a, s : \mathbb{V}^{\mathbf{p},\mathbf{q},\mathbf{r}}_{\Omega,\mathbb{S},\mathbb{E}} \times \mathbb{V}^{\mathbf{p},\mathbf{q},\mathbf{r}}_{\Omega,\mathbb{S},\mathbb{E}} \to \mathbb{R}$ and $\ell : \mathbb{V}^{\mathbf{p},\mathbf{q},\mathbf{r}}_{\Omega,\mathbb{S},\mathbb{E}} \to \mathbb{R}$ are given,

respectively, by

$$
\begin{aligned}
a(u_h, v_h) = &\sum_{\kappa_g \in \mathcal{T}_{\mathbb{E}}} \int_{\kappa_g} \sum_{\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}} \int_{\kappa_{\mathbb{S}}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\kappa_\Omega} (\boldsymbol{\mu} \cdot \nabla_{\mathbf{x}} u_h v_h + (\alpha + \beta) u_h v_h) \, d\mathbf{x} \, d\boldsymbol{\mu} \, dE \\
&- \sum_{\kappa_g \in \mathcal{T}_{\mathbb{E}}} \int_{\kappa_g} \sum_{\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}} \int_{\kappa_{\mathbb{S}}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\partial_- \kappa_\Omega \setminus \partial\Omega} (\boldsymbol{\mu} \cdot \boldsymbol{n}_{\kappa_\Omega}) \lfloor u_h \rfloor v_h^+ \, ds \, d\boldsymbol{\mu} \, dE \\
&- \sum_{\kappa_g \in \mathcal{T}_{\mathbb{E}}} \int_{\kappa_g} \sum_{\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}} \int_{\kappa_{\mathbb{S}}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\partial_- \kappa_\Omega \cap \partial\Omega} (\boldsymbol{\mu} \cdot \boldsymbol{n}_{\kappa_\Omega}) u_h^+ v_h^+ \, ds \, d\boldsymbol{\mu} \, dE
\end{aligned}
$$

$$
s(w_h, v_h) = \sum_{\kappa_g \in \mathcal{T}_{\mathbb{E}}} \int_{\kappa_g} \sum_{\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}} \int_{\kappa_{\mathbb{S}}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\kappa_\Omega} \mathcal{S}[u_h](\mathbf{x}, \boldsymbol{\mu}, E) v_h \, d\mathbf{x} \, d\boldsymbol{\mu} \, dE
$$

$$
\begin{aligned}
\ell(v_h) = &\sum_{\kappa_g \in \mathcal{T}_{\mathbb{E}}} \int_{\kappa_g} \sum_{\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}} \int_{\kappa_{\mathbb{S}}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\Omega} f v_h \, d\mathbf{x} \, d\boldsymbol{\mu} \, dE \\
&- \sum_{\kappa_g \in \mathcal{T}_{\mathbb{E}}} \int_{\kappa_g} \sum_{\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}} \int_{\kappa_{\mathbb{S}}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\partial_- \kappa_\Omega \cap \partial\Omega} (\boldsymbol{\mu} \cdot \boldsymbol{n}_{\kappa_\Omega}) g_{\mathrm{D}} v_h \, ds \, d\boldsymbol{\mu} \, dE
\end{aligned}
$$

We note that this scheme is consistent in the sense that if the analytical solution u to (3.1.1) is sufficiently smooth then

$$
b(u, v) = \ell(v)
$$

for all $v \in \mathbb{V}^{\mathbf{p},\mathbf{q},\mathbf{r}}_{\Omega,\mathbb{S},\mathbb{E}}$.

3.5 DG-DG

The monoenergetic LBTE is given as

$$
\begin{aligned}
\boldsymbol{\mu} \cdot \nabla_{\mathbf{x}} u(\mathbf{x}, \boldsymbol{\mu}) + (\alpha(\mathbf{x}, \boldsymbol{\mu}) + \beta(\mathbf{x}, \boldsymbol{\mu})) u(\mathbf{x}, \boldsymbol{\mu}) &= \mathcal{S}[u](\mathbf{x}, \boldsymbol{\mu}) \\
&\quad + f(\mathbf{x}, \boldsymbol{\mu}) \text{ in } \mathcal{D}, \\
u(\mathbf{x}, \boldsymbol{\mu}) &= g_{\mathrm{D}}(\mathbf{x}, \boldsymbol{\mu}) \text{ on } \Gamma_{\mathrm{in}}, \qquad (3.5.1)
\end{aligned}
$$

By the same arguments we can derive a DGFEM for the mono-energetic LBTE. Defining the space-angle mesh

$$
\mathcal{T}_{\Omega,\mathbb{S}} = \mathcal{T}_\Omega \times \mathcal{T}_{\mathbb{S}} = \{\kappa : \kappa = \kappa_\Omega \times \kappa_{\mathbb{S}}, \ \kappa_\Omega \in \mathcal{T}_\Omega, \ \kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}\}.
$$

Again, we can obtain our space-angle finite element space

$$
\mathbb{V}^{\mathbf{p},\mathbf{q}}_{\Omega,\mathbb{S}} = \mathbb{V}^{\mathbf{p}}_{\Omega} \otimes \mathbb{V}^{\mathbf{q}}_{\mathbb{S}},
$$

and, for any $\boldsymbol{\mu} \in \mathbb{S}$, let $\mathcal{G}_{\boldsymbol{\mu},h} = \{v \in L_2(\Omega) : \boldsymbol{\mu} \cdot \nabla_{\mathbf{x}} v|_{\kappa_\Omega} \in L_2(\kappa_\Omega) \text{ for all } \kappa_\Omega \in \mathcal{T}_\Omega\}$ denote the broken spatial graph space. Notice that we can define the space-angle finite element space as the subspace of $\mathbb{V}^{\mathbf{p},\mathbf{q},\mathbf{r}}_{\Omega,\mathbb{S},\mathbb{E}}$ where the space-angle-energy functions are constant in the energetic argument:

$$\mathbb{V}^{\mathbf{p},\mathbf{q}}_{\Omega,\mathbb{S}} = \{v_h \in \mathbb{V}^{\mathbf{p},\mathbf{q},\mathbf{r}}_{\Omega,\mathbb{S},\mathbb{E}} : v_h(\cdot,\cdot,E) = v_h(\cdot,\cdot,E') \quad \forall E, E' \in \mathbb{E}\}.$$

Likewise, energy independent data terms can be defined from the poly-energetic equivalents

$$\theta(\mathbf{x}, \boldsymbol{\eta} \to \boldsymbol{\mu}, E' \to E) = \frac{1}{|\mathbb{E}|}\theta(\mathbf{x}, \boldsymbol{\eta} \to \boldsymbol{\mu}),$$

$$\alpha(\mathbf{x}, \boldsymbol{\mu}, E) = \alpha(\mathbf{x}, \boldsymbol{\mu}),$$

$$f(\mathbf{x}, \boldsymbol{\mu}, E) = f(\mathbf{x}, \boldsymbol{\mu}),$$

$$g(\mathbf{x}, \boldsymbol{\mu}, E) = g(\mathbf{x}, \boldsymbol{\mu}).$$

The other data terms are given by

$$\beta(\mathbf{x}, \boldsymbol{\mu}) = \int_{\mathbb{S}} \theta(\mathbf{x}, \boldsymbol{\eta} \to \boldsymbol{\mu})\, d\boldsymbol{\eta},$$

$$\gamma(\mathbf{x}, \boldsymbol{\mu}) = \int_{\mathbb{S}} \theta(\mathbf{x}, \boldsymbol{\eta} \to \boldsymbol{\mu})\, d\boldsymbol{\eta},$$

the condition on α, (3.1.2), is reduced to

$$c = \alpha(\mathbf{x}, \boldsymbol{\mu}) \geq c_0 > 0.$$

The DG-DG scheme can be written as:
Find $u_h \in \mathbb{V}^{\mathbf{p},\mathbf{q}}_{\Omega,\mathbb{S}}$ such that

$$b(u_h, v_h) \equiv a(u_h, v_h) - s(u_h, v_h) = \ell(v_h), \tag{3.5.2}$$

for all $v_h \in \mathbb{V}^{\mathbf{p},\mathbf{q}}_{\Omega,\mathbb{S}}$, where $a, s : \mathbb{V}^{\mathbf{p},\mathbf{q}}_{\Omega,\mathbb{S}} \times \mathbb{V}^{\mathbf{p},\mathbf{q}}_{\Omega,\mathbb{S}} \to \mathbb{R}$ and $\ell : \mathbb{V}^{\mathbf{p},\mathbf{q}}_{\Omega,\mathbb{S}} \to \mathbb{R}$ are given, respectively, by

$$\begin{aligned} a(u_h, v_h) = &\sum_{\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}} \int_{\kappa_{\mathbb{S}}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\kappa_\Omega} (\boldsymbol{\mu} \cdot \nabla_{\mathbf{x}} u_h v_h + (\alpha + \beta) u_h v_h)\, d\mathbf{x}\, d\boldsymbol{\mu} \\ &- \sum_{\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}} \int_{\kappa_{\mathbb{S}}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\partial_- \kappa_\Omega \setminus \partial\Omega} (\boldsymbol{\mu} \cdot \boldsymbol{n}_{\kappa_\Omega}) \lfloor u_h \rfloor v_h^+\, ds\, d\boldsymbol{\mu} \\ &- \sum_{\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}} \int_{\kappa_{\mathbb{S}}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\partial_- \kappa_\Omega \cap \partial\Omega} (\boldsymbol{\mu} \cdot \boldsymbol{n}_{\kappa_\Omega}) u_h^+ v_h^+\, ds\, d\boldsymbol{\mu} \end{aligned}$$

$$s(w_h, v_h) = \sum_{\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}} \int_{\kappa_{\mathbb{S}}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\kappa_\Omega} \mathcal{S}[u_h](\mathbf{x}, \boldsymbol{\mu}, E) v_h \, d\mathbf{x} \, d\boldsymbol{\mu}$$

$$\begin{aligned} \ell(v_h) = &\sum_{\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}} \int_{\kappa_{\mathbb{S}}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\Omega} f v_h \, d\mathbf{x} \, d\boldsymbol{\mu} \\ &- \sum_{\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}} \int_{\kappa_{\mathbb{S}}} \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \int_{\partial_- \kappa_\Omega \cap \partial\Omega} (\boldsymbol{\mu} \cdot \boldsymbol{n}_{\kappa_\Omega}) g_{\mathrm{D}} v_h \, ds \, d\boldsymbol{\mu} \end{aligned}$$

We note that this scheme is consistent in the sense that if the analytical solution u to (3.5.1) is sufficiently smooth then

$$b(u, v) = \ell(v)$$

for all $v \in \mathbb{V}^{\mathbf{p},\mathbf{q}}_{\Omega,\mathbb{S}}$.

3.6 Source iteration

We may express the mono-energetic problem, as defined in Section 3.5, in matrix form: find the vector $U \in \mathbb{R}^N$ of coefficients with respect to a basis of $\mathbb{V}^{\mathbf{p},\mathbf{q}}_{\Omega,\mathbb{S}}$ such that

$$AU - SU = F \tag{3.6.1}$$

where $A, S \in \mathbb{R}^{N \times N}$ and $F \in \mathbb{R}^N$ denote the matrix representation of the streaming, A, and scattering, S, operators and load term, F, respectively. Rather than seeking to compute $u = (A - S)^{-1} F$ directly, which is prohibitively expensive due to the density of S; we employ a linear solver, so that the solution to the previous iteration can be used on the scattering turn. Source iteration is the technique of solving this linear system using the Richardson iteration:
Given $U^0 \in \mathbb{R}^N$, find $U^r \in \mathbb{R}^N$ such that

$$AU^r = SU^{r-1} + F, \tag{3.6.2}$$

for $r = 1, 2, \ldots$. It may be shown that this iteration converges to the solution of (3.6.1) if this scattering ratio,

$$\operatorname*{ess\,sup}_{\mathbf{x} \in \Omega} \frac{\beta(\mathbf{x})}{\alpha(\mathbf{x}) + \beta(\mathbf{x})} < 1,$$

holds [1].

The advantage of this approach is that it avoids inverting the scattering matrix, which is typically dense and highly coupled in angle. To investigate the structure of the matrix A, we introduce the following notation: for an angular element $\kappa_{\mathbb{S}}$, $\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}$, we define the local element basis by $\{\varphi^i_{\kappa_{\mathbb{S}}}\}_{i=1}^{|q_{\kappa_{\mathbb{S}}}|}$, where $|q_{\kappa_{\mathbb{S}}}|$ denotes the dimension of the

polynomial space defined on $\kappa_{\mathbb{S}}$. Furthermore, write $\mathbb{V}_{\Omega}^{\mathbf{p}} = \text{span}\{\varphi_{\Omega}^{i}\}_{i=1}^{N_{\Omega}}$, $N_{\Omega} = \dim(\mathbb{V}_{\Omega}^{\mathbf{p}})$. The matrix A has the nested block structure

$$A = \begin{bmatrix} D^1 & 0 & & & \\ 0 & D^2 & 0 & & \\ & 0 & \ddots & & \\ & & & \ddots & 0 \\ & & & 0 & D^{|\mathcal{T}_{\mathbb{S}}|} \end{bmatrix}, \quad \text{with} \quad D^n = \begin{bmatrix} D^n_{1,1} & \dots & D^n_{1,|q_{\kappa_{\mathbb{S}}}|} \\ \vdots & \ddots & \vdots \\ D^n_{|q_{\kappa_{\mathbb{S}}}|,1} & \dots & D^n_{|q_{\kappa_{\mathbb{S}}}|,|q_{\kappa_{\mathbb{S}}}|} \end{bmatrix},$$

where $|\mathcal{T}_{\mathbb{S}}| = \text{card}(\mathcal{T}_{\mathbb{S}})$ and, for $n = 1, 2, \dots, |\mathcal{T}_{\mathbb{S}}|$, $D^n_{i,j} = \int_{\kappa_{\mathbb{S}}} \varphi^i_{\kappa_{\mathbb{S}}}(\boldsymbol{\mu}) \varphi^j_{\kappa_{\mathbb{S}}}(\boldsymbol{\mu}) D_{\boldsymbol{\mu}} \, d\boldsymbol{\mu}$, $i, j = 1, 2, \dots, |q_{\kappa_{\mathbb{S}}}|$, where $D_{\boldsymbol{\mu}} \in \mathbb{R}^{N_{\Omega} \times N_{\Omega}}$, with $(D_{\boldsymbol{\mu}})_{i,j} = B_{\boldsymbol{\mu}}(\phi^j_{\Omega}, \phi^i_{\Omega})$, $i, j = 1, 2, \dots, N_{\Omega}$. Where $B_{\boldsymbol{\mu}}$ the bilinear form the DGFEM transport problem (2.3.5) with $\boldsymbol{a}$ selected to be direction vector $\boldsymbol{\mu}$. Solving (3.6.2) therefore requires inverting each diagonal block D^n, $n = 1, 2, \dots, |\mathcal{T}_{\mathbb{S}}|$, which corresponds to solving a coupled system of spatial transport problems on each angular element.

3.7 DG-DG vs DO mono-energetic LBTE

To compare these two methods of discretisation on a mono-energetic model problem is introduced. This problem is unrealistic and has has been picked due to it having an analytical solution.

Find $u \in \Omega \times \mathbb{S}$ such that

$$\begin{aligned} \boldsymbol{\mu} \cdot \nabla_{\mathbf{x}} u(\mathbf{x}, \boldsymbol{\mu}) + (\alpha(\mathbf{x}, \boldsymbol{\mu}) + \beta(\mathbf{x}, \boldsymbol{\mu})) u(\mathbf{x}, \boldsymbol{\mu}) &= \mathcal{S}[u](\mathbf{x}, \boldsymbol{\mu}) \\ &\quad + f(\mathbf{x}, \boldsymbol{\mu}) \text{ in } \mathcal{D}, \\ u(\mathbf{x}, \boldsymbol{\mu}) &= g_{\mathrm{D}}(\mathbf{x}, \boldsymbol{\mu}) \text{ on } \Gamma_{\mathrm{in}}. \end{aligned}$$

The macroscopic absorption cross-section is set at $\alpha = 1$, and the differential scattering cross-section $\mathcal{S}$ is selected to be

$$\mathcal{S}(\mathbf{x}, \boldsymbol{\mu} \cdot \boldsymbol{\mu}') = \frac{1}{4\pi},$$

over $\mathcal{D}$ therefore $\beta = 1$. Finally, the forcing term f and g are selected so that the analytical solution is given by

$$u(\mathbf{x}, \boldsymbol{\mu}) = \cos(4\phi)(x \cos y + y \sin x),$$

where the angular variable is parameterised by $\boldsymbol{\mu} = (\sin\phi\cos\varphi, \sin\phi\sin\varphi, \cos\phi)$ for $0 \le \phi \le \pi$ and $0 \le \varphi \le 2\pi$. $\alpha = 1$, $\beta = 1$, $\theta(x, \boldsymbol{\mu} \cdot \boldsymbol{\eta}) = \frac{1}{4\pi}$ over $\mathcal{D}$. And the terms f and

g are defined such that the analytical solution is

$$u(\mathbf{x}, \boldsymbol{\mu}) = \cos(4\phi)(x\cos y + y\sin x).$$

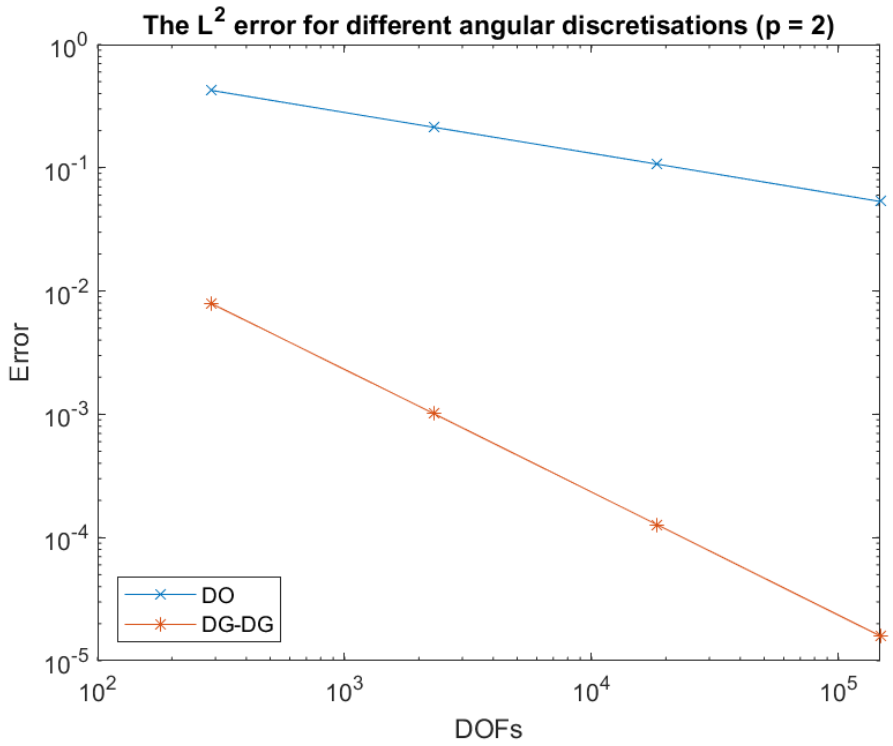

Figure 3.1: The L^2 error for DO and DG-DG $\boldsymbol{q} = \boldsymbol{p} = 2$.

Figure 3.1 shows the convergence of the DGFEM using meshes consisting of uniform squares in the spatial domain Ω for both the different angular discretisations. In this example, we observe that as we uniformly refine both space and angle at the same rate, the error in the L^2 norm does not behave in the same way for both angular discretisations. The convergence of error in the L^2 norm for DG-DG is of order $\|u - u_h\|_{L_2(\mathcal{D})} \sim O(h^3)$, but for DO, it is of order $\|u - u_h\|_{L_2(\mathcal{D})} \sim O(h)$.

We can show that this is due to the different angular discretisations. By integrating out the angle, we get the scalar flux:

$$\phi(\mathbf{x}) = \int_{\mathbb{S}} u(\mathbf{x}, \boldsymbol{\mu})\, d\boldsymbol{\mu}$$

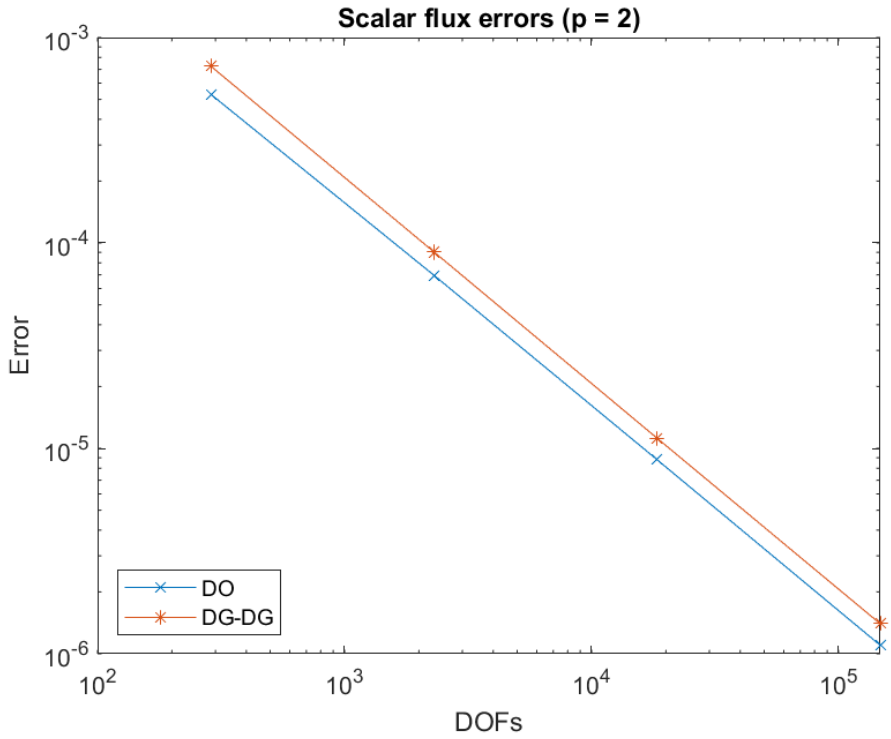

Figure 3.2: The scalar flux error for DO and DG-DG $\boldsymbol{q} = \boldsymbol{p} = 2$.

Figure 3.2 shows the error of the scalar flux $\|\phi - \phi_h\|_{L_2(\mathcal{D})} \sim O(h^3)$ for both of the angular discretisations. It demonstrates that the low order of the L^2 error of the discrete ordinates is due to the angular discretisations used. Another point of comparison between the two methods is the time taken to solve the mono-energetic LBTE.

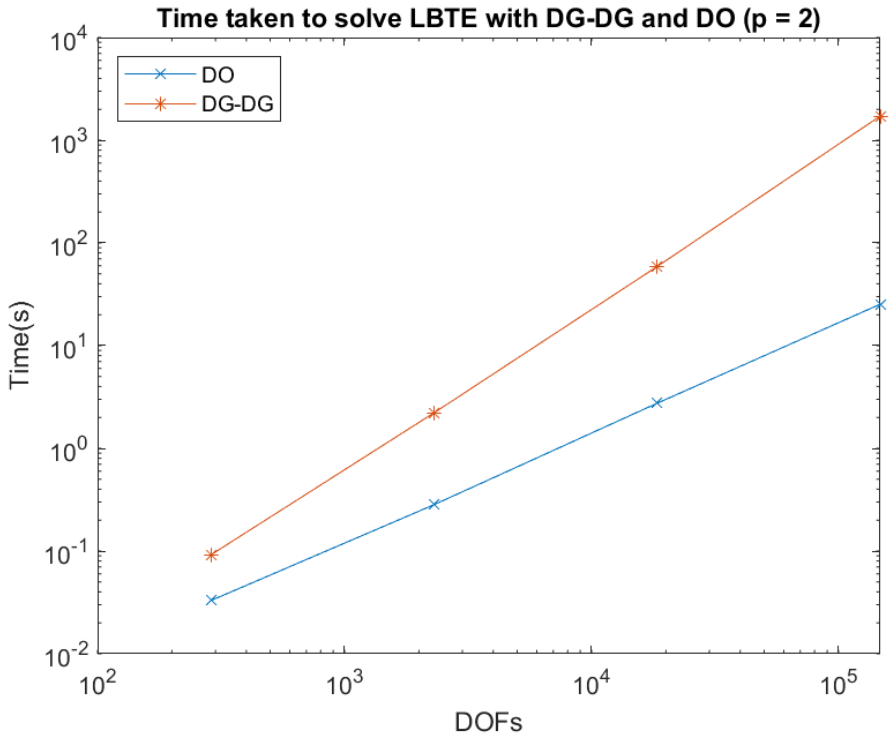

Figure 3.3: Time taken to solve LBTE with DG-DG and DO $\boldsymbol{q} = \boldsymbol{p} = 2$

Figure 3.3 shows that the DG-DG method takes significantly longer to solve than the DO method, while both are using the same optimised matrix solver MUMPS [65]. We can split the time taken to solve the LBTE into three different stages: forming the matrix; solving the matrix; and calculating the scattering term.

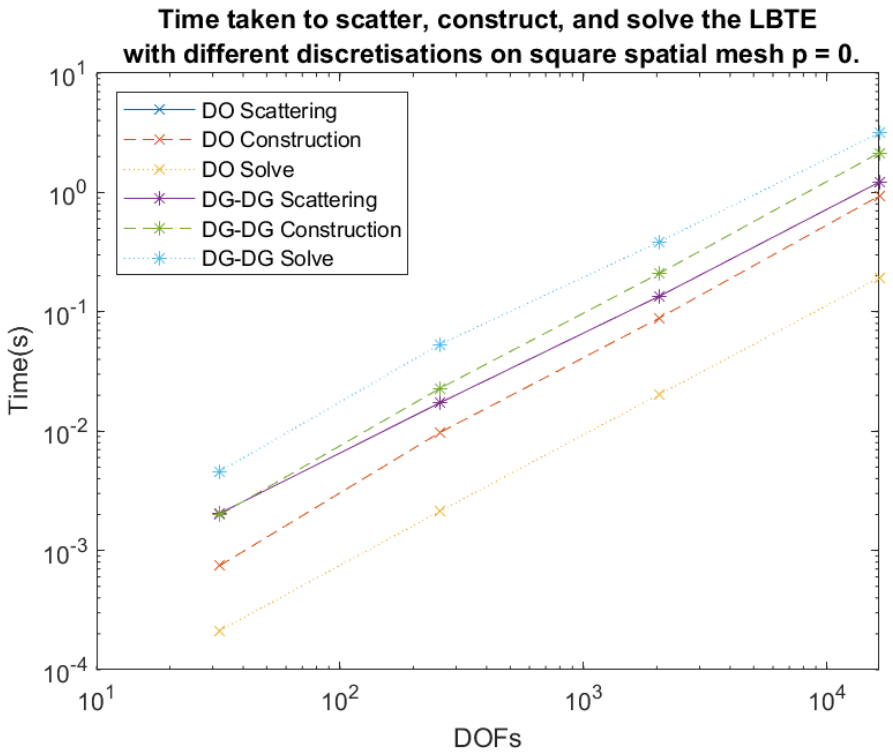

Figure 3.4: The time taken for each stage of solving the LBTE $\boldsymbol{q} = \boldsymbol{p} = 0$

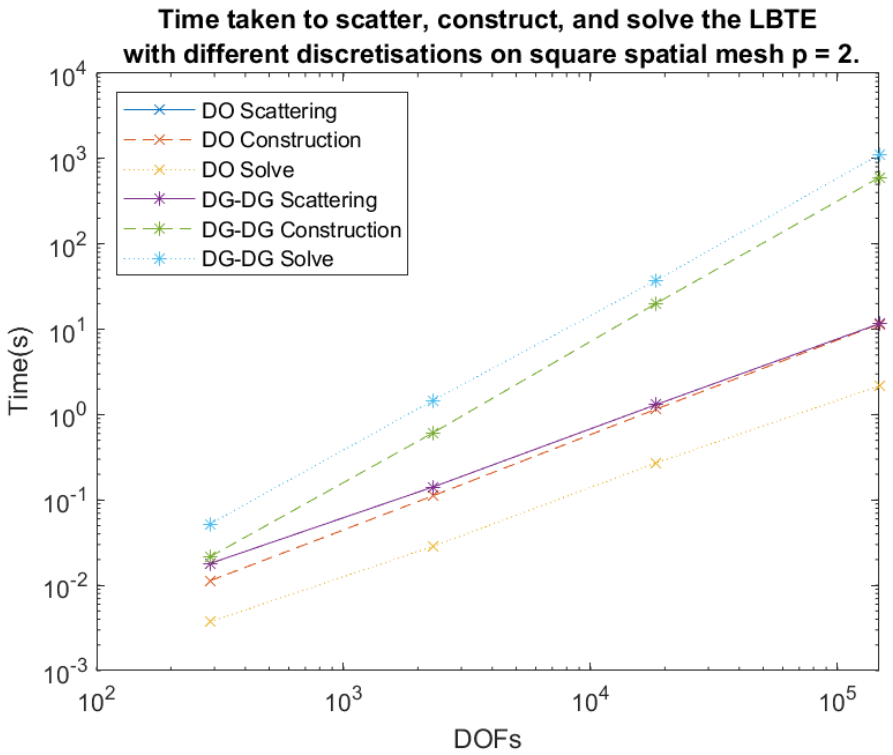

Figure 3.5: The time taken for each stage of solving the LBTE $\boldsymbol{q} = \boldsymbol{p} = 2$

From Figure 3.4 we can see that computing the scattering term takes roughly the same amount of time for each scheme. Figure 3.5 demonstrates that at higher orders, however, the time taken to construct and solve the matrix is significantly longer and

growing at a faster rate for the DG-DG method. This difference is due to the matrix structure of each of the schemes.

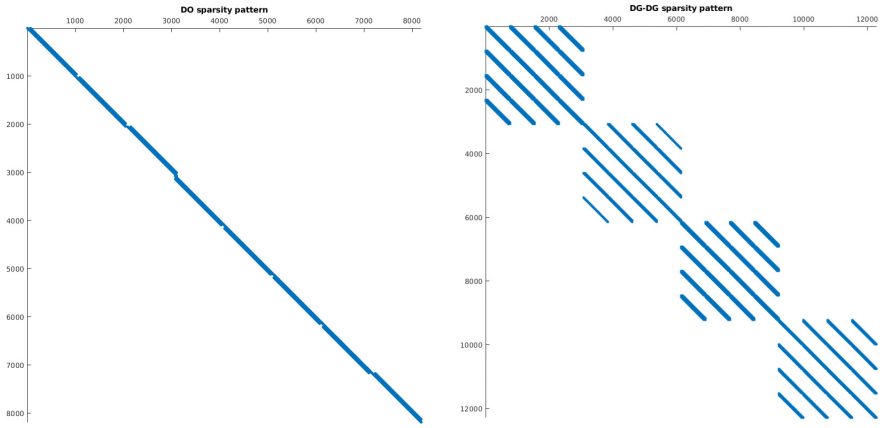

Figure 3.6: Matrix sparsity pattern for the DO and DG-DG schemes.

The discrete ordinates method results in a block diagonal matrix, with each block being a transport equation with the ordinate (quadrature point) as the advection field. This allows each transport problem to be solved separately. With DG-DG, however, the matrix formed is much denser and cannot easily be separated into transport problems. Figure 3.6 shows an example of the different matrix structures; please note with both of these, each entry is a block matrix itself. Section 3.6 shows the structure of the blocks produced by the DG-DG method.

3.8 Discrete Ordinates Galerkin (DOG)

From the DG-DG implementation we know that on each angular element $\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}$, the mono-energetic solution can be approximated in terms of an angular basis $\{\varphi^i_{\kappa_{\mathbb{S}}}\}_{i=1}^{|q_{\kappa_{\mathbb{S}}}|}$ on $\kappa_{\mathbb{S}}$

$$u_h(\mathbf{x},\boldsymbol{\mu})|_{\kappa_{\mathbb{S}}} = \sum_{i=1}^{|q_{\kappa_{\mathbb{S}}}|} u^i_{\kappa_{\mathbb{S}}}(\mathbf{x})\varphi^i_{\kappa_{\mathbb{S}}}(\boldsymbol{\mu}) \quad \forall u^i_{\kappa_{\mathbb{S}}} \in \mathbb{V}^{\mathbf{p}}_{\Omega}.$$

Which we can rewrite as

$$u_h(\mathbf{x},\boldsymbol{\mu}) = \sum_{\kappa_{\mathbb{S}}\in\mathcal{T}_{\mathbb{S}}} \sum_{i=1}^{|q_{\kappa_{\mathbb{S}}}|} u^i_{\kappa_{\mathbb{S}}}(\mathbf{x})\varphi^i_{\kappa_{\mathbb{S}}}(\boldsymbol{\mu}),$$

in terms of these basis functions.

Selecting test functions of the form $v_h = v_{\kappa_{\mathbb{S}}}\varphi^i_{\kappa_{\mathbb{S}}}, v_{\kappa_{\mathbb{S}}} \in \mathbb{V}^{\mathbf{p}}_{\Omega}$ the DG-DG problem can be expressed as:
$\forall \kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}$, find $\{u^i_{\kappa_{\mathbb{S}}}\}_{i=1}^{|q_{\kappa_{\mathbb{S}}}|} \in \mathbb{V}^{\mathbf{p}}_{\Omega}$ such that

$$\sum_{j=1}^{|q_{\kappa_{\mathbb{S}}}|} a(u^j_{\kappa_{\mathbb{S}}}\varphi^j_{\kappa_{\mathbb{S}}}, v_{\kappa_{\mathbb{S}}}\varphi^i_{\kappa_{\mathbb{S}}}) = \sum_{\kappa'_{\mathbb{S}}\in\mathcal{T}_{\mathbb{S}}} \sum_{j=1}^{|q_{\kappa_{\mathbb{S}}}|} s(u^j_{\kappa'_{\mathbb{S}}}\varphi^j_{\kappa'_{\mathbb{S}}}, v_{\kappa_{\mathbb{S}}}\varphi^i_{\kappa_{\mathbb{S}}}) + \ell(v_{\kappa_{\mathbb{S}}}\varphi^i_{\kappa_{\mathbb{S}}}) \quad \forall v_{\kappa_{\mathbb{S}}} \in \mathbb{V}^{\mathbf{p}}_{\Omega},$$

and $1 \leq i \leq |q_{\kappa_{\mathbb{S}}}|$.

For some given angular reference element $\hat{\kappa}_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}$, let $\{(\hat{\boldsymbol{\mu}}_q, \hat{\omega}_q)\}_{q=1}^{|q_{\kappa_{\mathbb{S}}}|}$ (where $|q_{\kappa_{\mathbb{S}}}| = (q_{\kappa_{\mathbb{S}}} + 1)^{d-1}$) denote the tensor-product Gauss-Legendre quadrature scheme with $q_{\kappa_{\mathbb{S}}} + 1$ points in each direction. Then, on the same reference element $\hat{\kappa}_{\mathbb{S}}$, let $\{\hat{\varphi}_i\}_{i=1}^{|q_{\kappa_{\mathbb{S}}}|}$ denote the Lagrangian basis for $\mathbb{Q}_{q_{\kappa_{\mathbb{S}}}}(\hat{\kappa}_{\mathbb{S}})$ constructed with respect to the Gauss-Legendre quadrature points $\hat{\boldsymbol{\mu}}_q$, $q = 1, 2, \ldots, |q_{\kappa_{\mathbb{S}}}|$, which satisfies

$$\hat{\varphi}_i(\hat{\boldsymbol{\mu}}_j) = \delta_{ij}, i, j = 1, 2, \ldots, |q_{\kappa_{\mathbb{S}}}|.$$

On each angular element, $\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}$, we map the local basis defined on the reference element to $\kappa_{\mathbb{S}}$ based on employing the mapping $F_{\kappa_{\mathbb{S}}}$; more precisely, this yields the local basis

$$\{\varphi^i_{\kappa_{\mathbb{S}}} = \hat{\varphi}_i \circ F^{-1}_{\kappa_{\mathbb{S}}}\}_{i=1}^{|q_{\kappa_{\mathbb{S}}}|},$$

on $\kappa_{\mathbb{S}}$. Furthermore, the quadrature scheme on $\kappa_{\mathbb{S}} \in \mathcal{T}_{\mathbb{S}}$, is given by $(\boldsymbol{\mu}^q_{\kappa_{\mathbb{S}}}, \omega^q_{\kappa_{\mathbb{S}}})_{q=1}^{|q_{\kappa_{\mathbb{S}}}|}$, where $\boldsymbol{\mu}^q_{\kappa_{\mathbb{S}}} = F_{\kappa_{\mathbb{S}}}(\hat{\boldsymbol{\mu}}_q)$, $\omega^q_{\kappa_{\mathbb{S}}} = \hat{\omega}_q \mathcal{J}_{\kappa_{\mathbb{S}}}(\hat{\boldsymbol{\mu}}_q)$, $q = 1, 2, \ldots, |q_{\kappa_{\mathbb{S}}}|$, and $\mathcal{J}_{\kappa_{\mathbb{S}}}$ denotes the square root of the determinant of the first fundamental form of the mapping $F_{\kappa_{\mathbb{S}}}$. Hence, the mapped basis retains the Lagrangian property of the reference basis.

$$\sum_{j=1}^{|q_{\kappa_{\mathbb{S}}}|} a(u^j_{\kappa_{\mathbb{S}}} \varphi^j_{\kappa_{\mathbb{S}}}, v_{\kappa_{\mathbb{S}}} \varphi^i_{\kappa_{\mathbb{S}}}) \approx \sum_{q=1}^{|q_{\kappa_{\mathbb{S}}}|} \omega^q_{\kappa_{\mathbb{S}}} \varphi^j_{\kappa_{\mathbb{S}}}(\boldsymbol{\mu}^q_{\kappa_{\mathbb{S}}}) \varphi^i_{\kappa_{\mathbb{S}}}(\boldsymbol{\mu}^q_{\kappa_{\mathbb{S}}}) a^q_{\kappa_{\mathbb{S}}} = \omega^i_{\kappa_{\mathbb{S}}} a^i_{\kappa_{\mathbb{S}}} \delta_{ij} \text{ for } i = 1, 2, \ldots |q_{\kappa_{\mathbb{S}}}|$$

Where $a^i_{\kappa_{\mathbb{S}}}$ is $a(u_h, v_h)$ evaluated at $\boldsymbol{\mu} = \boldsymbol{\mu}^i_{\kappa_{\mathbb{S}}}$, i.e. a transport problem with a fixed wind direction. This changes the structure of our matrix equations used for source iteration to

$$D^n \approx \begin{bmatrix} \omega_1 A_{\boldsymbol{\mu}_1} & 0 & & \\ 0 & \omega_2 A_{\boldsymbol{\mu}_2} & \ddots & \\ & \ddots & \ddots & 0 \\ & & 0 & \omega_{|q_{\kappa_{\mathbb{S}}}|} A_{\boldsymbol{\mu}_{N_{|q_{\kappa_{\mathbb{S}}}|}}} \end{bmatrix}.$$

A becomes a block diagonal matrix formed from block diagonal matrices where the individual blocks correspond to a single spatial transport problem, exactly like DO, as seen in Section 3.7, but retaining the DG variational framework. The DOG discretisations allow the mono-energetic LBTE to be treated as a system of transport equations coupled on the right side. We require the basis functions to be $\mathbb{Q}_{q_{\kappa_{\mathbb{S}}}}(\kappa_{\mathbb{S}})$ rather than offering the choice between that and $\mathbb{P}_{q_{\kappa_{\mathbb{S}}}}(\kappa_{\mathbb{S}})$ in order that the number of quadrature points used to evaluate the angular domain is high enough while keeping that:

$$\hat{\varphi}_i(\hat{\boldsymbol{\mu}}_j) = \delta_{ij}, i, j = 1, 2, \ldots, |q_{\kappa_{\mathbb{S}}}|.$$

3.8.1 DOG in energy

In our paper [33], we show that the same arguments can be applied to the DGFEM in energy domain to result in a system of coupled mono-energetic LBTEs. If we had perfect knowledge of the function

$$u^+(\mathbf{x}, \boldsymbol{\mu}, E) = \begin{cases} u(\mathbf{x}, \boldsymbol{\mu}, E) \text{ for } E > \hat{E}, \\ 0 \text{ otherwise,} \end{cases}$$

for some $\hat{E} > 0$, then the assumption that the scattering kernel satisfies $\theta(\mathbf{x}, \boldsymbol{\eta} \cdot \boldsymbol{\mu}, E' \to E) = 0$ for $E' < E$, would imply that $\hat{u}(\mathbf{x}, \boldsymbol{\mu}) \equiv u(\mathbf{x}, \boldsymbol{\mu}, \hat{E})$ satisfies the monoenergetic radiation transport problem:
Find $\hat{u} : \Omega \times \mathbb{S} \to \mathbb{R}$ such that

$$\begin{aligned} \boldsymbol{\mu} \cdot \nabla_{\mathbf{x}} \hat{u}(\mathbf{x}, \boldsymbol{\mu}) + (\alpha(\mathbf{x}, \boldsymbol{\mu}, \hat{E}) + \beta(\mathbf{x}, \boldsymbol{\mu}, \hat{E}))\hat{u}(\mathbf{x}, \boldsymbol{\mu}) &= \mathcal{S}[u^+](\mathbf{x}, \boldsymbol{\mu}, \hat{E}) \\ &\quad + f(\mathbf{x}, \boldsymbol{\mu}, \hat{E}) \text{ in } \mathcal{D}, \\ \hat{u}(\mathbf{x}, \boldsymbol{\mu}) &= g(\mathbf{x}, \boldsymbol{\mu}, \hat{E}) \text{ on } \Gamma_{\text{in}}. \end{aligned}$$

As with multigroup discretisation we must first solve in the higher energy groups before moving on to the lower energy groups. We then introduce the following family of energy cutoff functions:

$$u_g^+(\mathbf{x}, \boldsymbol{\mu}, E) = \begin{cases} u_h(\mathbf{x}, \boldsymbol{\mu}, E) \text{ for } E \geq E_{g-1}, \\ 0 \text{ otherwise,} \end{cases}$$

for all $1 \leq g \leq N_{\mathbb{E}}$, which represents the component of the discrete fluence which may be considered as pre-computed 'data' when solving for the fluence in group κ_g, and focus on solving the problem in a single energy group κ_g, $1 \leq g \leq N_{\mathbb{E}}$. We expand u_h in group κ_g in terms of energy basis functions as

$$u_h(\mathbf{x}, \boldsymbol{\mu}, E)|_{\kappa_g} \equiv u_g(\mathbf{x}, \boldsymbol{\mu}, E) = \sum_{j=1}^{r_{\kappa_g}+1} u_g^j(\mathbf{x}, \boldsymbol{\mu}) \varphi_g^j(E),$$

where $u_g^j \in \mathbb{V}_{\Omega,\mathbb{S}}^{\mathbf{p},\mathbf{q}}$, $j = 1, 2, \ldots, r_{\kappa_g} + 1$, and $\{\varphi_g^j\}_{j=1}^{r_{\kappa_g}+1}$ forms a basis of $\mathbb{P}_{r_{\kappa_g}}(\kappa_g)$ (which is only supported on κ_g).

Selecting $v_h = v_g \varphi_g^i \in \mathbb{V}_{\Omega,\mathbb{S},\mathbb{E}}^{\mathbf{p},\mathbf{q},\mathbf{r}}$, with $v_g \in \mathbb{V}_{\Omega,\mathbb{S}}^{\mathbf{p},\mathbf{q}}$, $i = 1, 2, \ldots, r_{\kappa_g} + 1$, the fluence in group κ_g may then be computed by solving:

Find $\left\{u_g^i\right\}_{i=1}^{r_{\kappa_g}+1} \in \mathbb{V}_{\Omega,\mathbb{S}}^{\mathbf{p},\mathbf{q}}$ such that

$$\sum_{j=1}^{r_{\kappa_g}+1}\left(\int_{\kappa_g}\int_{\mathbb{S}} a_{\boldsymbol{\mu}}^{E}(u_g^j, v_g)\varphi_g^j\varphi_g^i \, d\boldsymbol{\mu}\, dE - s(u_g^j\varphi_g^j, v_g\varphi_g^i)\right) = s(u_g^+, v_g\varphi_g^i) + \ell(v_g\varphi_g^i)$$

for all $v_g \in \mathbb{V}_{\Omega,\mathbb{S}}^{\mathbf{p},\mathbf{q}}$ and $i = 1, 2, \ldots, r_{\kappa_g}+1$. Where we have defined the upwind transport bilinear form $a_{\boldsymbol{\mu}}^{E} : \mathcal{G}_{\boldsymbol{\mu},h} \times \mathcal{G}_{\boldsymbol{\mu},h} \to \mathbb{R}$ as

$$\begin{aligned} a_{\boldsymbol{\mu}}^{E}(w_h, v_h) = &\sum_{\kappa_\Omega\in\mathcal{T}_\Omega}\int_{\kappa_\Omega}(\boldsymbol{\mu}\cdot\nabla_{\mathbf{x}} w_h v_h + (\alpha+\beta)w_h v_h)\, d\mathbf{x} \\ &- \sum_{\kappa_\Omega\in\mathcal{T}_\Omega}\int_{\partial_-\kappa_\Omega\setminus\partial\Omega}(\boldsymbol{\mu}\cdot\boldsymbol{n}_{\kappa_\Omega})\lfloor w_h\rfloor v_h^+\, ds \\ &- \sum_{\kappa_\Omega\in\mathcal{T}_\Omega}\int_{\partial_-\kappa_\Omega\cap\partial\Omega}(\boldsymbol{\mu}\cdot\boldsymbol{n}_{\kappa_\Omega})w_h^+ v_h^+\, ds. \end{aligned}$$

And $s(w_h, v_h)$ and $\ell(v_h)$ are the definitions given in Section 3.4,

$$s(w_h, v_h) = \sum_{\kappa_g\in\mathcal{T}_{\mathbb{E}}}\int_{\kappa_g}\sum_{\kappa_{\mathbb{S}}\in\mathcal{T}_{\mathbb{S}}}\int_{\kappa_{\mathbb{S}}}\sum_{\kappa_\Omega\in\mathcal{T}_\Omega}\int_{\kappa_\Omega}\mathcal{S}[u_h](\mathbf{x},\boldsymbol{\mu},E)v_h\, d\mathbf{x}\, d\boldsymbol{\mu}\, dE,$$

$$\begin{aligned} \ell(v_h) = &\sum_{\kappa_g\in\mathcal{T}_{\mathbb{E}}}\int_{\kappa_g}\sum_{\kappa_{\mathbb{S}}\in\mathcal{T}_{\mathbb{S}}}\int_{\kappa_{\mathbb{S}}}\sum_{\kappa_\Omega\in\mathcal{T}_\Omega}\int_{\Omega} f v_h\, d\mathbf{x}\, d\boldsymbol{\mu}\, dE \\ &- \sum_{\kappa_g\in\mathcal{T}_{\mathbb{E}}}\int_{\kappa_g}\sum_{\kappa_{\mathbb{S}}\in\mathcal{T}_{\mathbb{S}}}\int_{\kappa_{\mathbb{S}}}\sum_{\kappa_\Omega\in\mathcal{T}_\Omega}\int_{\partial_-\kappa_\Omega\cap\partial\Omega}(\boldsymbol{\mu}\cdot\boldsymbol{n}_{\kappa_\Omega})g_{\mathrm{D}}v_h\, ds\, d\boldsymbol{\mu}\, dE. \end{aligned}$$

This forms a fully coupled system of monoenergetic Boltzmann transport problems for the $r_{\kappa_g}+1$ unknowns within the energy group κ_g. To simplify this structure, let $\{E_g^q\}_{q=1}^{r_{\kappa_g}+1} \subset \kappa_g$ denote the $r_{\kappa_g}+1$ Gauss-Legendre quadrature points on κ_g with associated weights $\{\omega_g^q\}_{q=1}^{r_{\kappa_g}+1} \subset \mathbb{R}_{\geq 0}$. We then select the basis functions $\{\varphi_g^i\}_{i=1}^{r_{\kappa_g}+1}$ to be the unique set of polynomials which satisfy the Lagrangian property $\varphi_g^i(E_g^j) = \delta_{ij}$, $i, j = 1, 2, \ldots, r_{\kappa_g}+1$, where δ_{ij} denotes the Kronecker delta. This quadrature is exact for polynomials of degree $2r_{\kappa_g}+1$, and so we use it to evaluate the (energy) integrals present in the bilinear form $a_{\boldsymbol{\mu}}^{E}(\cdot,\cdot)$:
Find $\left\{u_g^j\right\}_{j=1}^{r_{\kappa_g}+1} \in \mathbb{V}_{\Omega,\mathbb{S}}^{\mathbf{p},\mathbf{q}}$ such that

$$\omega_g^i\int_{\mathbb{S}} a_{\boldsymbol{\mu}}^{E_g^i}(u_g^i, v_g)\, d\boldsymbol{\mu} - \sum_{j=1}^{r_{\kappa_g}+1} s(u_g^j\varphi_g^j, v_g\varphi_g^i) = s(u_g^+, v_g\varphi_g^i) + \ell(v_g\varphi_g^i) \tag{3.8.1}$$

for all $v_g \in \mathbb{V}_{\Omega,\mathbb{S}}^{\mathbf{p},\mathbf{q}}$ and $i = 1, 2, \ldots, r_{\kappa_g}+1$. Here, $a_{\boldsymbol{\mu}}^{E_g^i}(\cdot,\cdot)$ is defined analogously to $a_{\boldsymbol{\mu}}^{E}(\cdot,\cdot)$ with the coefficient data α and β evaluated at the energy quadrature point E_g^i, $i = 1, 2, \ldots, r_{\kappa_g}+1$. Furthermore, with a slight abuse of notation, we have written $\left\{u_g^i\right\}_{i=1}^{r_{\kappa_g}+1}$ to also denote the solution of the equation above. We have not applied

the above quadrature scheme in energy to the forcing and scattering terms, since in applications it is usually preferable to treat these terms separately. Instead, we express the scattering term in an alternative form. For $w, v \in \mathbb{V}_{\Omega,\mathbb{S}}^{\mathbf{p},\mathbf{q}}$, we define

$$s_{g',g}^{j,i}(w,v) = \int_{\mathbb{S}} \int_{\Omega} \int_{\mathbb{S}} \Theta_{g',g}^{j,i}(\mathbf{x}, \boldsymbol{\eta}\cdot\boldsymbol{\mu}) w(\mathbf{x},\boldsymbol{\eta}) v(\mathbf{x},\boldsymbol{\mu})\, d\boldsymbol{\eta}\, d\mathbf{x}\, d\boldsymbol{\mu},$$

where

$$\Theta_{g',g}^{j,i}(\mathbf{x}, \boldsymbol{\eta}\cdot\boldsymbol{\mu}) = \int_{\kappa_g} \int_{\kappa_{g'}} \theta(\mathbf{x}, \boldsymbol{\eta}\cdot\boldsymbol{\mu}, E' \to E) \varphi_g^i(E) \varphi_{g'}^j(E')\, dE'\, dE,$$

for $g, g' = 1, 2, \ldots, N_E$, $i = 1, 2, \ldots, r_{\kappa_g} + 1$, and $j = 1, 2, \ldots, r_{\kappa_{g'}} + 1$. With this, (3.8.1) may be rewritten in the following equivalent form:
Find $\left\{u_g^j\right\}_{j=1}^{r_{\kappa_g}+1} \in \mathbb{V}_{\Omega,\mathbb{S}}^{\mathbf{p},\mathbf{q}}$ satisfying the discrete monoenergetic radiation transport problem

$$\omega_g^i \int_{\mathbb{S}} a_{\boldsymbol{\mu}}^{E_g^i}(u_g^j, v_g)\, d\boldsymbol{\mu} - \sum_{j=1}^{r_{\kappa_g}+1} s_{g,g}^{j,i}(u_g^j, v_g) = \sum_{g'=1}^{g-1} \sum_{j=1}^{r_{\kappa_{g'}}+1} s_{g',g}^{j,i}(u_{g'}^j, v_g) + \ell(v_g \varphi_g^i)$$

for all $v_g \in \mathbb{V}_{\Omega,\mathbb{S}}^{\mathbf{p},\mathbf{q}}$ and $i = 1, 2, \ldots, r_{\kappa_g} + 1$. This yields a system of $r_{\kappa_g} + 1$ monoenergetic radiation transport problems to solve within each energy group, which are only coupled through the scattering operator. Moreover, as we assume particles never gain energy from their interactions, solutions within a given energy group depend only on the solutions within the same group and from higher energy groups.

3.9 DOG vs DG-DG vs DO

Using the same model problem as in Section 3.7, Figure 3.7 shows that the DOG scheme retains the high order approximation of the DG-DG scheme, $\|u - u_h\|_{L_2(\mathcal{D})} \sim O(h^3)$, rather than the DO scheme which is $\|u - u_h\|_{L_2(\mathcal{D})} \sim O(h)$.

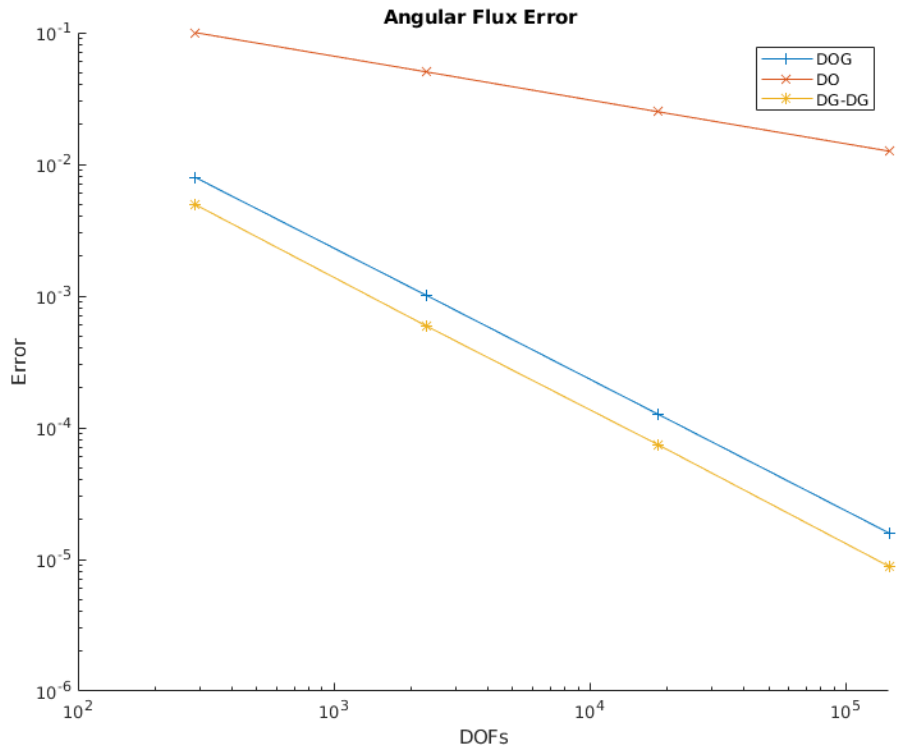

Figure 3.7: Angular flux error in the L^2 norm for DO, DOG and DG-DG $\boldsymbol{q} = \boldsymbol{p} = 2$.

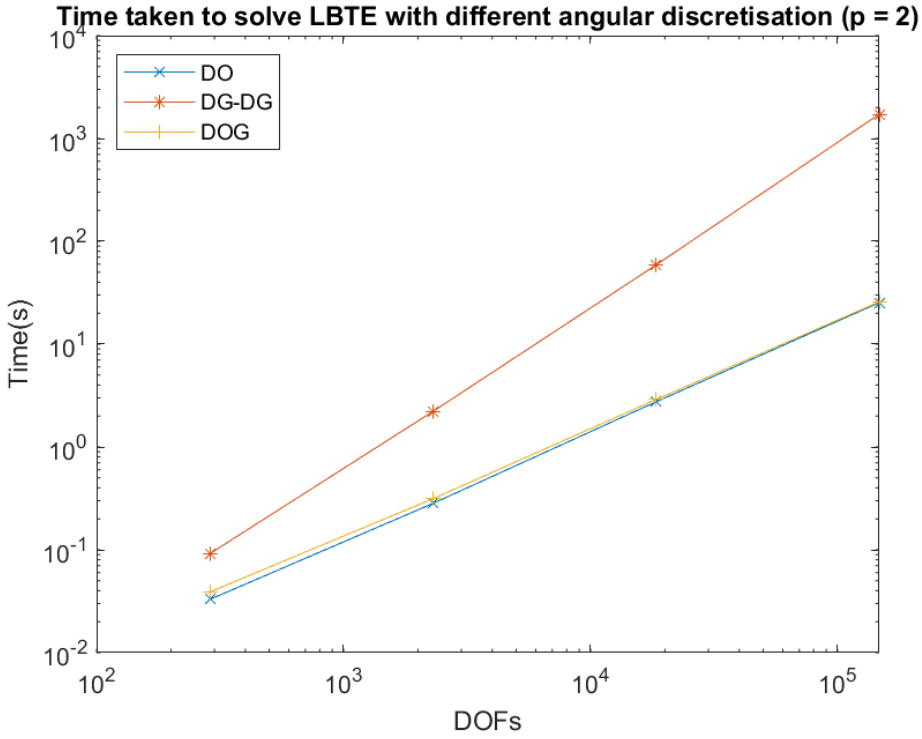

Figure 3.8: Time taken to solve LBTE with different angular discretisations $\boldsymbol{q} = \boldsymbol{p} = 2$.

Figure 3.8 shows that the DOG scheme has retained the shorter solve times of the DO scheme. This is as expected as the matrix has been block diagonalised in a similar way to the DO scheme. Figures 3.9 and 3.10 show that at higher orders, DOG construction

and solve times scale like the DO ones.

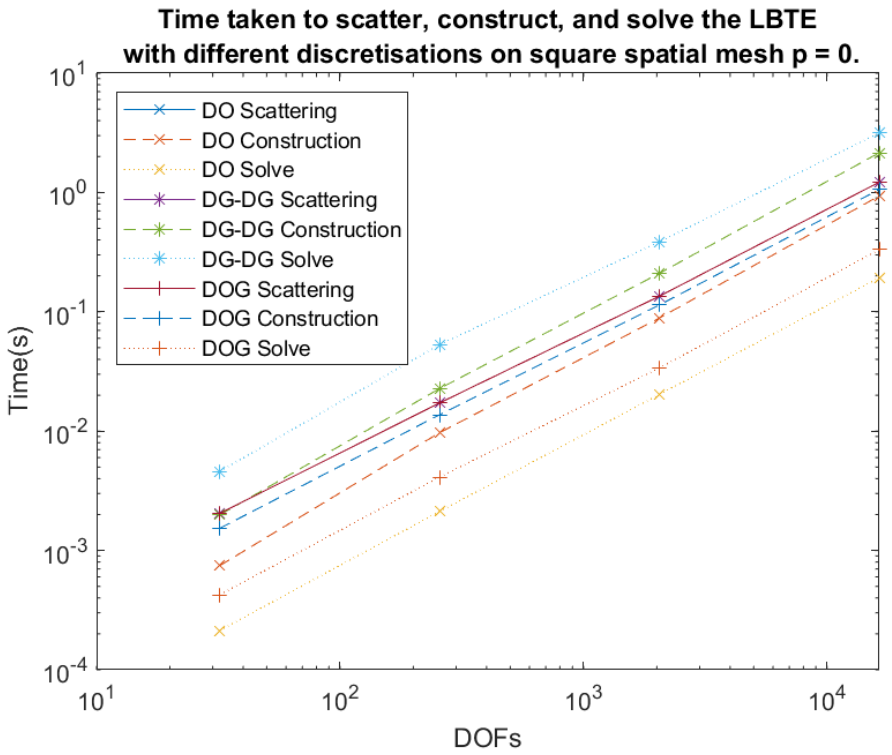

Figure 3.9: The time taken for each stage of solving the LBTE $\boldsymbol{q} = \boldsymbol{p} = 0$

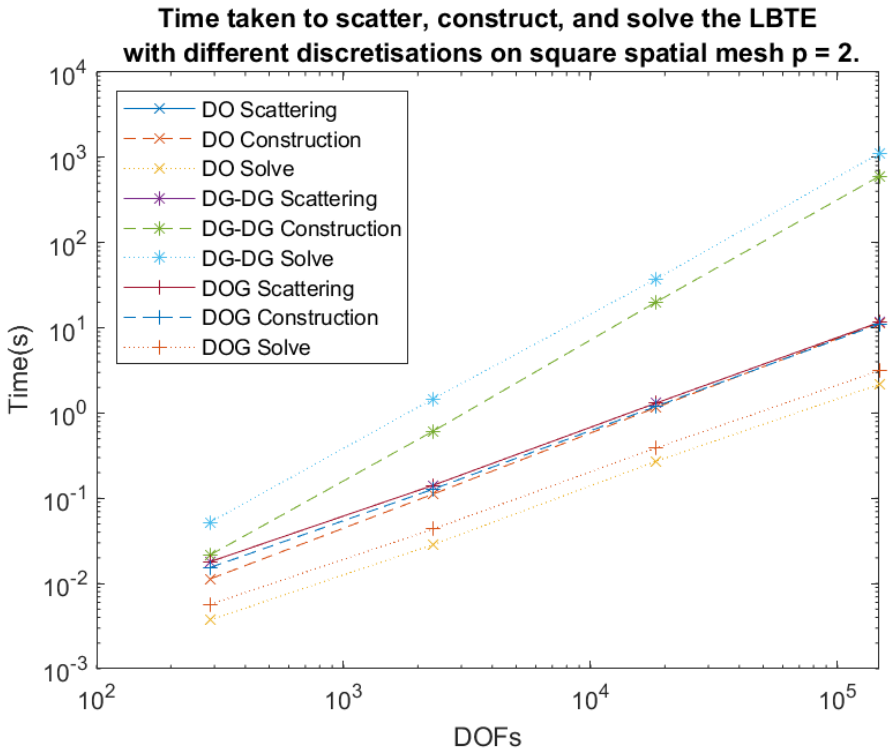

Figure 3.10: The time taken for each stage of solving the LBTE $\boldsymbol{q} = \boldsymbol{p} = 2$

3.10 Summary

In this chapter, we have shown how the domain $\mathcal{D}$ of the LBTE can be formed by the tensor product of three domains $\Omega, \mathbb{S}, \mathbb{E}$. This tensor structure can then be exploited to allow each domain to be discretised separately, such as multigroup in energy and discrete ordinates in angle. This simplicity, however, comes at the cost of them being low order methods. For high order methods, traditionally we have to treat and consider multiple domains at once, such as DG-DG-DG; however, the new discretisation we have presented, DOG, and its analogous results for the energy domain in [33], can be used to treat each domain separately while retaining the high order convergence from the discontinuous Galerkin methods.

The DOG scheme, Section 3.8, relies on using quadrilateral elements in the angular mesh, as other shaped elements cannot be integrated as accurately with a lower number of quadrature points entirely contained within their domain. Likewise, the basis functions on the angular elements must be $\mathbb{Q}_{q_{\kappa_{\mathbb{S}}}}$ with Lagrangian polynomials so that

$$\hat{\varphi}_i(\hat{\boldsymbol{\mu}}_j) = \delta_{ij}, i, j = 1, 2, \ldots, |q_{\kappa_{\mathbb{S}}}|.$$

Consequently, the DOG scheme results in a system of transport problems coupled only on the right hand side; thus, the matrix can be more efficiently constructed and solved than the coupled domain system of DG-DG. We have also seen that the scattering step of solve, takes the same amount of time, no matter the discretisation. While there are more efficient algorithms for performing this step, see [55], we will not be focussing on that stage for the rest of this thesis. We shall, instead, be focussing on optimising the construction and solve times of the DOG scheme, by developing an efficient solver for the transport problem.

Chapter 4

The structure of the matrix system resulting from the discrete ordinate Galerkin discretisation

As we have shown, the DOG discretisation allows us to solve a series of transport equations coupled on the right hand side. It is, therefore, worth examining how we solve the transport equation, as if we find an efficient way to solve the individual transport problems, we will be able to solve the LBTE much more efficiently.

Also, motivated by our desire to use real life MRI or CAT scan data, we need to consider the size of these problems. An average MRI image is formed of 16,777,216 voxels, though higher resolutions are becoming more popular, which means using the MRI voxels would require a lot of resources. In this chapter we, therefore, examine the effect of using different element types in our spatial mesh. We first explain how the different standard element shapes affect the sparsity of the resulting matrix, we then examine polytopic meshes, convex elements from Voronoi tessellation, and then agglomerating elements.

4.1 The linear system for the LBTE

Taking the weak form of the transport problem (2.3.5), we can find the finite element formulation to this problem:

Find $u_h \in V_h$ such that

$$B(u_h, v_h) = \ell(u_h) \quad \forall v_h \in V_h.$$

where $B : V_h \times V_h \to \mathbb{R}$ is a bilinear functional and $\ell : V_h \to \mathbb{R}$, gives a matrix system

$$Au = l$$

This matrix A and vector l have coefficients $A_{ij} = B(\phi_i, \phi_j)$, $l_i = \ell(\phi_i)$ as our basis functions ϕ are constructed such that their span equals the test space for our problem.

However, due to the scattering operator present in the LBTE, our bilinear functional A can be separated into two separate operators $A = T - S$

This gives a linear system

$$Tu = Su + l$$

where T is a matrix of coefficients corresponding to the transport part of the LBTE and S is a matrix of coefficient for the scattering part of the LBTE. When the problem is discretised, however, the structure of S is always a dense matrix formed of blocks corresponding to each combination of basis functions of the space and angular dimensions.

$$S = \begin{bmatrix} S_{11} & S_{12} & \dots & S_{1,N} \\ \vdots & \vdots & \ddots & \vdots \\ S_{N1} & S_{N2} & \dots & S_{NN}. \end{bmatrix}$$

The structure of T, however, is much more dependent on the choice of discretisation. As discussed previously, one of the main advantages derived from the DOG scheme, Section 3.8, is its block diagonal matrix structure. Not only does this structure dramatically reduce the computation required to solve the linear system, see Section 3.9, it also allows us to reduce polyenergetic LBTE to a sequence of monoenergetic LBTE systems [33]. Furthermore, it allows us to treat a monoenergetic LBTE as a sequence of transport equations. It is, therefore, these transport equations that form the blocks in the resulting block diagonal matrix T. In the DOG scheme we get a transport matrix

$$T = \begin{bmatrix} \omega_1^E \omega_1^A T_{11} & & & \\ & \omega_1^E \omega_2^A T_{12} & & \\ & & \ddots & \\ & & & \omega_M^E \omega_N^A T_{MN} \end{bmatrix},$$

Here, M, N are the number of quadrature points in the discrete ordinate scheme formed by the DOG scheme for energy and angle, and $\{\omega_i^E\}_{i=1}^M$ and $\{\omega_j^A\}_{j=1}^N$ are the weights associated with these quadrature points. Also $\{T_{ij}\}_{i=1,j=1}^{M,N}$ is a matrix that is formed from the unique transport problem formed for the quadrature point pairs. So it is worth discussing the structure of these transport block matrices T_{ij}.

4.2 Transport block structure

The matrix T_{ij} is formed by the transport bilinear form $B(\cdot,\cdot)$, as defined in (2.3.5) and has two distinct components, the element, and the face blocks. The element block is formed when the trial and test functions $(u_h, v_h) \in V_\Omega$ are wholly within a given element κ_Ω is

$$\begin{aligned} A_{\kappa_\Omega} &= B(u_h, v_h)|_{\kappa_\Omega} \\ &= \int_{\kappa_\Omega} -u_h a \cdot \nabla v_h + b u_h v_h dx, \\ A_{\kappa_\Omega}[i,j] &= \int_{\kappa_\Omega} -\phi_i a \cdot \nabla \phi_j + b \phi_i \phi_j dx, \end{aligned}$$

where $\phi_i \in \mathbb{H}_{\boldsymbol{p}_{\kappa_\Omega}}$, $i, j \in [1, \boldsymbol{p}_{\kappa_\Omega}]$ is our finite element basis as defined in 2.3.1 and $\boldsymbol{p}_{\kappa_\Omega}$ is the maximum polynomial degree in the finite element space on the element κ_Ω.

The resulting square block is of size $((\boldsymbol{p}_{\kappa_\Omega}+1)(\boldsymbol{p}_{\kappa_\Omega}+2)/2)^2$ for 2D and $((\boldsymbol{p}_{\kappa_\Omega}+1)(\boldsymbol{p}_{\kappa_\Omega}+2)(\boldsymbol{p}_{\kappa_\Omega}+3)/6)^2$ for 3D with a $\mathbb{H}_{\boldsymbol{p}_{\kappa_\Omega}} = \mathbb{P}_{\boldsymbol{p}_{\kappa_\Omega}}$ basis or $(\boldsymbol{p}_{\kappa_\Omega}+1)^4$ for 2D and $(\boldsymbol{p}_{\kappa_\Omega}+1)^6$ for 3D with $\mathbb{H}_{\boldsymbol{p}_{\kappa_\Omega}} = \mathbb{Q}_{\boldsymbol{p}_{\kappa_\Omega}}$. This block is placed in the matrix on the diagonal of our transport matrix T_{ij}.

There are two different types of face block, internal and external face blocks. Internal face blocks are formed when the test and trial functions are defined on different convex elements, $\kappa_\Omega^1, \kappa_\Omega^2$ which have an intersection known as a face $\mathcal{F}$ between them. These blocks for a given face $\mathcal{F} = \kappa_\Omega^1 \cap \kappa_\Omega^2$ are given by

$$\begin{aligned} F_{\kappa_\Omega^1,\kappa_\Omega^2} &= B(u_h, v_h)|_{\mathcal{F}} \\ &= \int_{\mathcal{F}} a \cdot n_{\kappa_\Omega}(u_h^+ - u_h^-) v_h^+ dS \\ F_{\kappa_\Omega^1,\kappa_\Omega^2}[i,j] &= \int_{\mathcal{F}} a \cdot n_{\kappa_\Omega}(\phi_i^+ - \phi_j^-) \psi_i^+ dS, \end{aligned}$$

where $\phi_i^+, \psi_i^+ \in \mathbb{H}_{\boldsymbol{p}_{\kappa_\Omega^1}}, \phi_j^- \in \mathbb{H}_{\boldsymbol{p}_{\kappa_\Omega^2}}, i \in [1, \boldsymbol{p}_{\kappa_\Omega^1}]$ and $j \in [1, \boldsymbol{p}_{\kappa_\Omega^2}]$. This matrix is a rectangular block $((\boldsymbol{p}_{\kappa_\Omega^1}+1)(\boldsymbol{p}_{\kappa_\Omega^1}+2)/2) \times ((\boldsymbol{p}_{\kappa_\Omega^2}+1)(\boldsymbol{p}_{\kappa_\Omega^2}+2)/2)$ for 2D and $((\boldsymbol{p}_{\kappa_\Omega^1}+1)(\boldsymbol{p}_{\kappa_\Omega^1}+2)(\boldsymbol{p}_{\kappa_\Omega^1}+3)/6) \times (\boldsymbol{p}_{\kappa_\Omega^2}+1)(\boldsymbol{p}_{\kappa_\Omega^2}+2)(\boldsymbol{p}_{\kappa_\Omega^2}+3)/6)$ for 3D if $\mathbb{H}_{\boldsymbol{p}_{\kappa_\Omega}} = \mathbb{P}_{\boldsymbol{p}_{\kappa_\Omega}}$. Similarly, $(\boldsymbol{p}_{\kappa_\Omega^1}+1)^2 \times (\boldsymbol{p}_{\kappa_\Omega^2}+1)^2$ for 2D and $(\boldsymbol{p}_{\kappa_\Omega^1}+1)^3 \times (\boldsymbol{p}_{\kappa_\Omega^2}+1)^3$ for 3D if $\mathbb{H}_{\boldsymbol{p}_{\kappa_\Omega}} = \mathbb{Q}_{\boldsymbol{p}_{\kappa_\Omega}}$.

This block is positioned in an off diagonal position in our transport matrix T_{ij}, aligned with the two element blocks of the two elements κ_Ω^1 and κ_Ω^2. The wind direction a will determine which of these blocks are formed. $F_{\kappa_\Omega^1,\kappa_\Omega^2}$ will be the face block formed with the wind coming from element κ_Ω^1 to element κ_Ω^2 and $F_{\kappa_\Omega^2,\kappa_\Omega^1}$ will be the face block formed by the wind going in the opposite direction across the face $\mathcal{F}$.

The external face block is formed by a boundary face with an element that is down wind from the face, consequently the test and trial functions are only sampled on a single

element.

$$\begin{aligned} F_K &= B(u_h, v_h)|_{\mathcal{F}\cap\partial_-\Omega} \\ &= \int_{\mathcal{F}\cap\partial_-\Omega} a \cdot n_{\kappa_\Omega} u_h^+ v_h^+ dS \\ F_K[i,j] &= \int_{\mathcal{F}\cap\partial_-\Omega} a \cdot n_{\kappa_\Omega} \phi_i^+ \psi_j^+ dS, \end{aligned}$$

where $\phi_i^+ \subseteq \mathbb{P}_{\boldsymbol{p}_{\kappa_\Omega}}$ and $i, j \in [1, \boldsymbol{p}_{\kappa_\Omega}]$. These blocks are positioned on the diagonal and are the same size and shape as our elemental blocks, so they can be added together to form our diagonal block

$$E_K[i,j] = A_{\kappa_\Omega}[i,j] + F_{\kappa_\Omega}[i,j].$$

As an illustrative example of how these blocks are placed within the matrix T_{ij}, we will use a mesh with 4 square elements, as shown in Figure 4.1, and wind direction of $a = [1,1]^T$. This gives the following block structure:

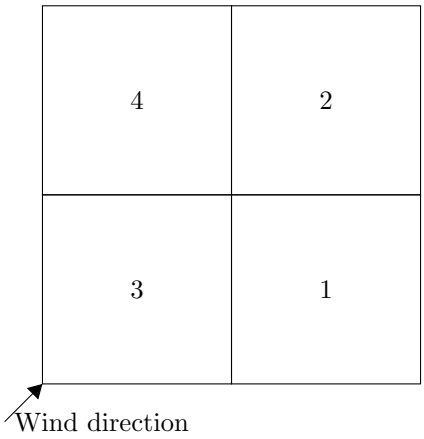

Figure 4.1: Square mesh with a north-east wind direction

$$T_{ij} = \begin{bmatrix} [E1] & & [F13] & \\ [F21] & [E2] & & [F24] \\ & & [E3] & \\ & & [F43] & [E4] \end{bmatrix}. \tag{4.2.1}$$

Obviously, this is a very small example of our transport block T_{ij}. It is worth noting that the matrix T_{ij} is square, and it will have DOFs^2 number of entries. So even this small example could result in a large matrix with a high enough $\boldsymbol{p}$. But the number of empty (or equal to zero) cells will be fixed by the number of elements and the number of faces between those elements. This leads us to consider the effect the type of the elements has on the matrix sparsity.

4.3 Matrix Sparsity

A sparse matrix is a matrix where most of the elements are zero valued. A matrix's sparsity is defined as the percentage of elements in the matrix which are zero valued. The example matrix above, (4.2.1) , has a block sparsity of 50%. Figure 4.2 shows the actual sparsity pattern formed by the mesh in Figure 4.1 with polynomial degree one i.e. $\boldsymbol{p} = 1$. This has a sparsity of 71.6%. Similarly, Figure 4.3 shows the sparsity pattern for a 3D cube mesh with eight elements, again with $\boldsymbol{p} = 1$, this has a sparsity of 89.7%.

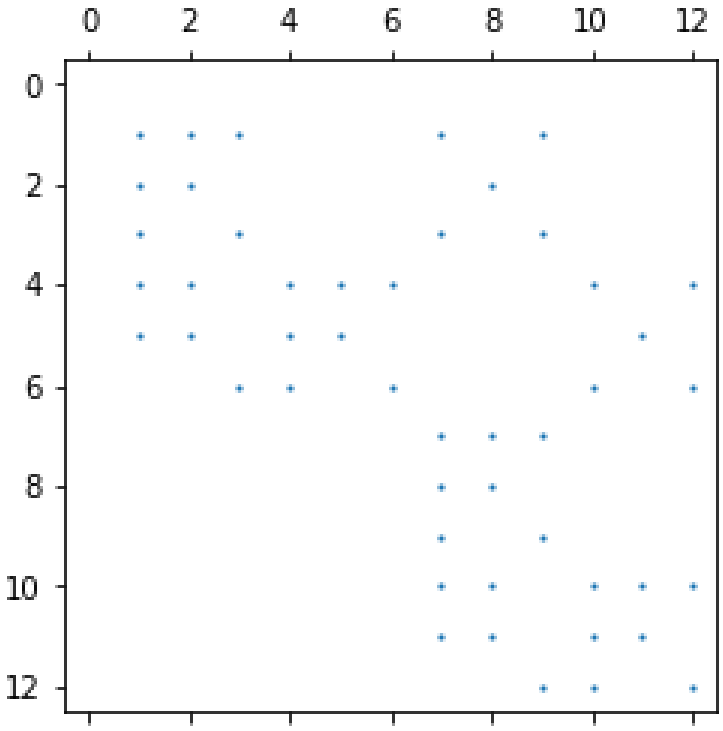

Figure 4.2: Sparsity pattern for 4 square elements, as shown in Figure 4.1, with $\boldsymbol{p} = 1$

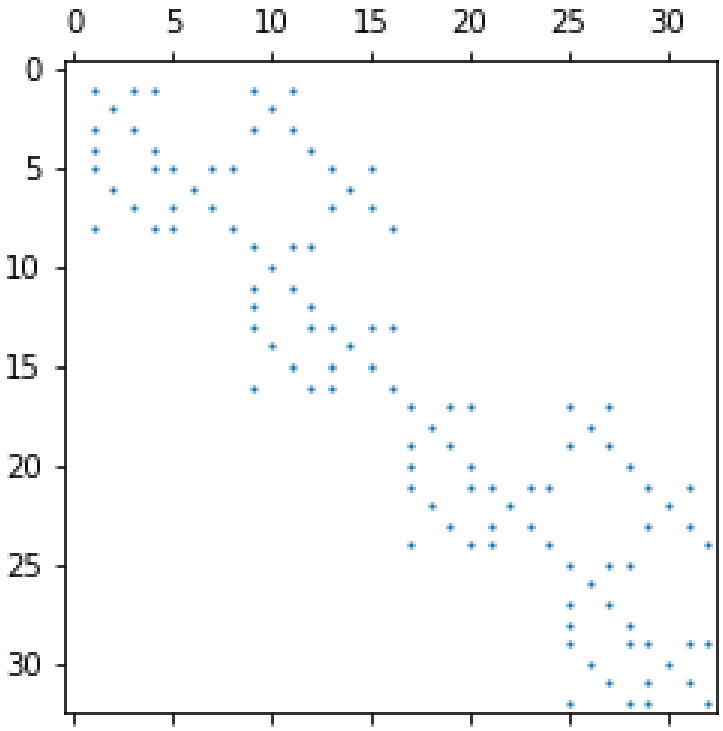

Figure 4.3: Sparsity pattern for 8 cube elements with $\boldsymbol{p} = 1$

The sparser the matrix, the less memory and time is required to solve the matrix equation; however, it is worth noting that this is a very basic example, and the sparsity and structure of the transport matrix block is obviously dependent on three things:

- The number of elements will define how many diagonal blocks there are.
- The maximum polynomial degree basis function defined on each element will define the block size for that element and its associated face blocks.
- The number of other elements that have at least one connected face between this element and them, will define the number of face blocks on that element's row and/or column (depending on wind direction across the face(s)).

The number of elements and the polynomial degree are chosen for accuracy, as the convergence rate of the error in the L^2 norm is $\mathcal{O}(h^{\boldsymbol{p}+1})$ [53], see Section 2.3.1. The number of faces between elements, however, has no direct bearing on the accuracy of the calculation, it is usually just a consequence of the element type and refinement strategy chosen. It will, however, have a large bearing on the sparsity of the matrix, which is directly linked to the time taken to solve the transport matrix equation.

In this section, we will be comparing three different element types: quadrilaterals; simplices; and arbitrarily shaped polytopic elements.

Quadrilaterals and simplices are both examples of meshes with a single defined element shape. Because of this, we know a priori how many faces every element should have. In 2D the quadrilaterals we will use are squares which have four faces, in 3D we

will use cubes which have six faces. The simplices, used in our examples, in 2D are right-angled triangles which have three faces and in 3D tetrahedra, which have four faces. Polytopic meshes, however, do not have a predetermined shape, so elements can have a different number of faces.

Unfortunately, if a quadrilateral or simplex mesh is not refined uniformly, there exist elements of different sizes in the grid, and hanging nodes can form. A hanging node is a node defined on the face of an element that is not a node used to define the shape of the element. For example, in Figure 4.4 we see a hanging node has been formed, giving a mesh with four smaller elements with four faces each and one larger element which has 5 faces. If a mesh has at least one hanging node, it is called nonconforming. Nonconforming meshes are no more complicated to use due to the discontinuous basis functions in DGFEM; however, they can cause the calculation to slow down due to the increased density of the matrix.

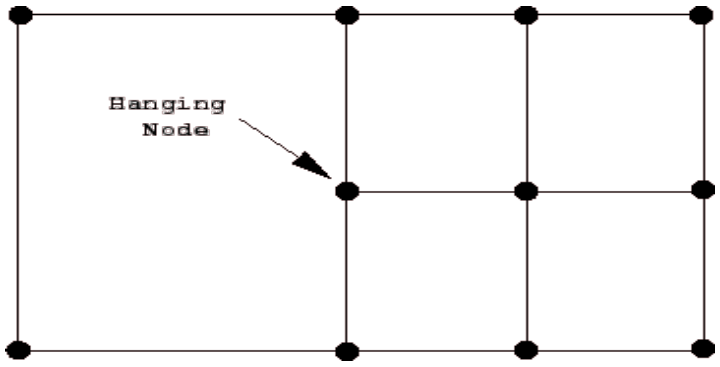

Figure 4.4: A hanging node formed by unevenly refining a square mesh

Figure 4.5 shows the sparsity pattern for a mesh formed of 8 simplices which has a sparsity of 86.9%. Comparing with Figure 4.2 we can see that the reduction in the number of faces has made the matrix 15.3% more sparse. In Figure 4.6 we see the matrix sparsity for a mesh of 48 tetrahedral elements has a sparsity of 96.8%, an increase in sparsity of 7.1%.

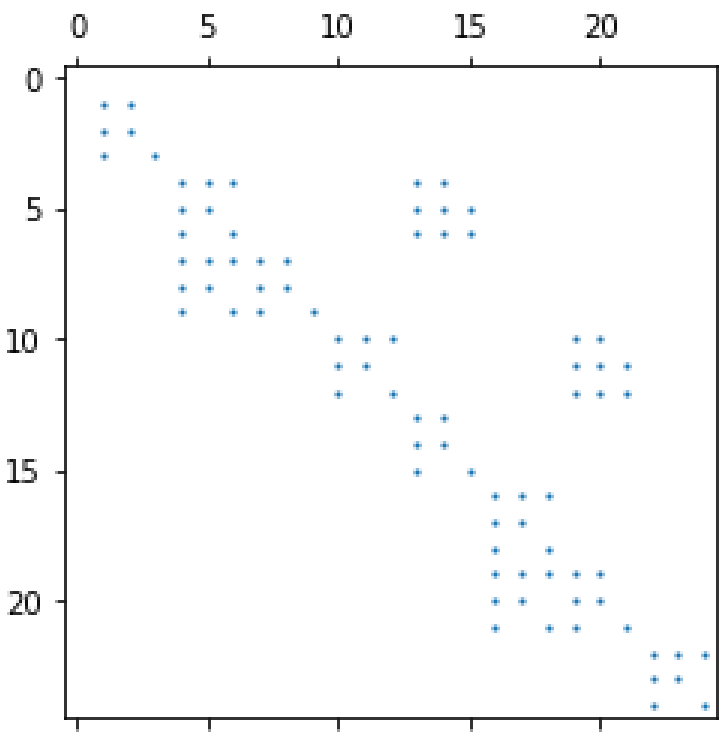

Figure 4.5: Sparsity pattern for 8 triangular elements with $\boldsymbol{p} = 1$.

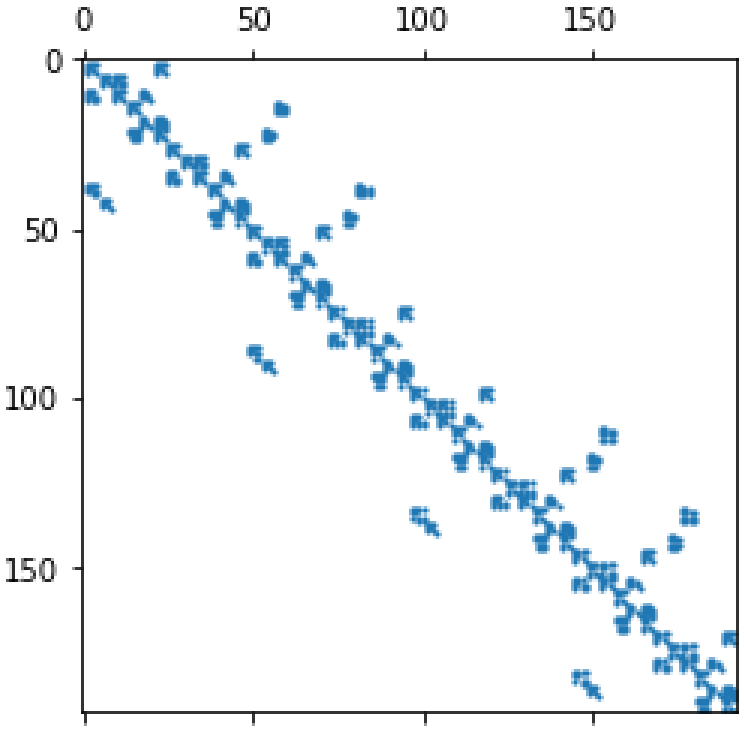

Figure 4.6: Sparsity pattern for 48 tetrahedral elements with $\boldsymbol{p} = 1$.

While simplex meshes do have a higher sparsity than quadrilaterals they also have quite a few more elements in them. This is due to two right-angled triangles forming a square and six tetrahedra forming a cube (in our implementation), as shown in Figure

4.7.

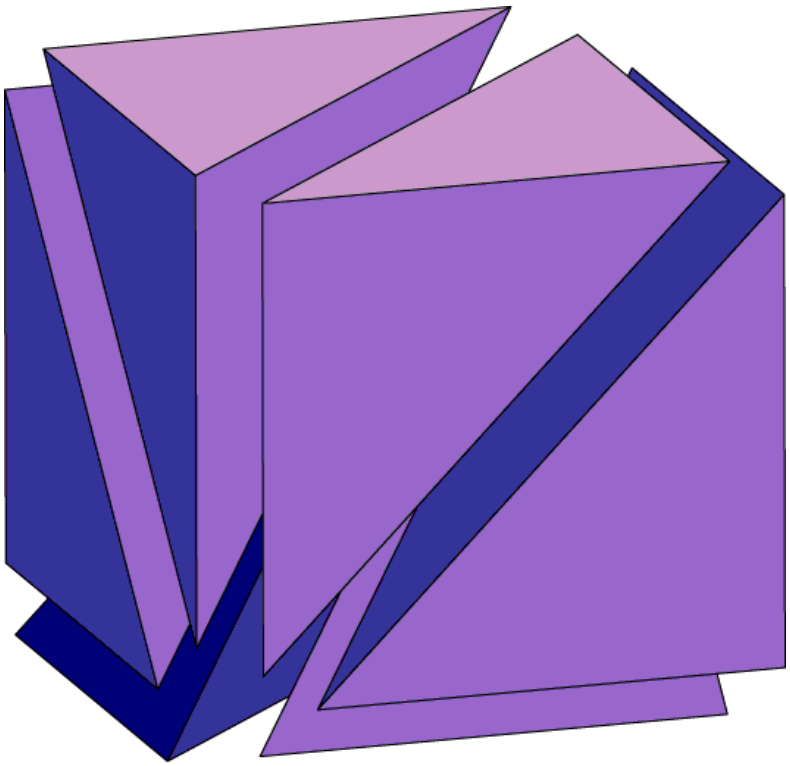

Figure 4.7: How six tetrahedrons form one cube [25].

Simplex elements are polytopic elements with the lowest number of faces possible, but a polytope element can have an arbitrarily high number of faces. This will result in a denser matrix, as shown in Figure 4.8 where the matrix formed from 4 polygonal elements has a sparsity of 36.1%, about half of the sparsity of the equivalent mesh with square elements.

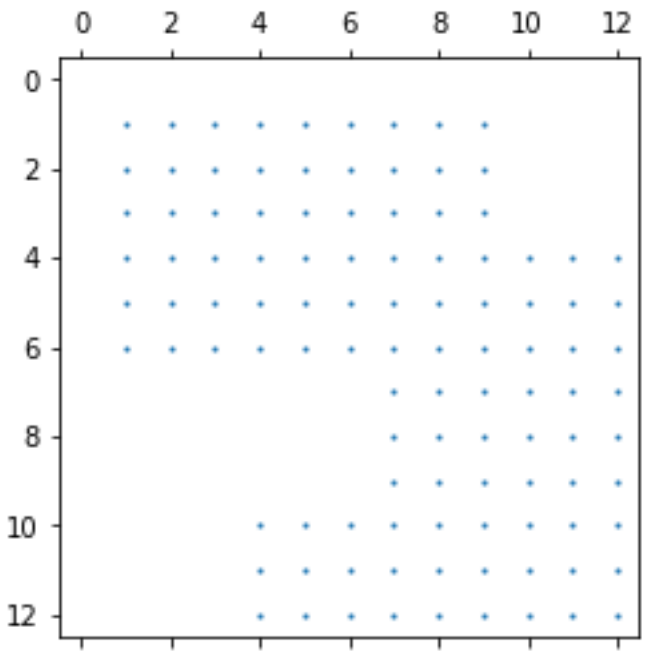

Figure 4.8: Sparsity pattern for 4 polygonal elements with $\boldsymbol{p} = 1$.

4.4 Polytopic mesh generation

Polytopic meshes do have advantages of their own. Polytopic elements can be created to fit the shape of the underlying geometry, rather than having to increase the number of elements used. Doing this will result in a smaller matrix, with fewer element blocks. Thus, a smaller denser matrix can give the same accuracy, in the L^2 norm, as the larger more sparse matrix of simplexes or quadrilaterals. Additionally, polytopic elements can model complicated geometries more accurately than using smaller standard elements to approximate the geometry.

As these polytopic elements are not as structured, they can be non-convex and have multiple different faces between two elements. In the latter case, to evaluate the test and trial functions across this face, we have to consider the subfaces. We define the subfaces as the boundary between two elements that do not form the intersection of the two elements. Consequently, the union of subfaces will form the face between two elements. Due to this the face blocks of our transport matrix become

$$F_{\kappa_\Omega^1,\kappa_\Omega^2}[i,j] = \sum_{l}^{N} sf \int_{\mathcal{F}_l} a \cdot n_{\kappa_\Omega}(\phi_i^+ - \phi_i^-)\psi_j^+ dS,$$

where $\phi_i^+, \psi_i^+ \in \mathbb{H}_{\boldsymbol{p}_{\kappa_\Omega^1}}, \phi_j^- \in \mathbb{H}_{\boldsymbol{p}_{\kappa_\Omega^2}}, i \in [1, \boldsymbol{p}_{\kappa_\Omega^1}]$ and $j \in [1, \boldsymbol{p}_{\kappa_\Omega^2}]$ and N_{sf} is the number of subfaces that form face F. This will not change the structure of the block, as the same basis functions are being evaluated for each face between the same elements. This will not affect the way the diagonal element blocks are calculated.

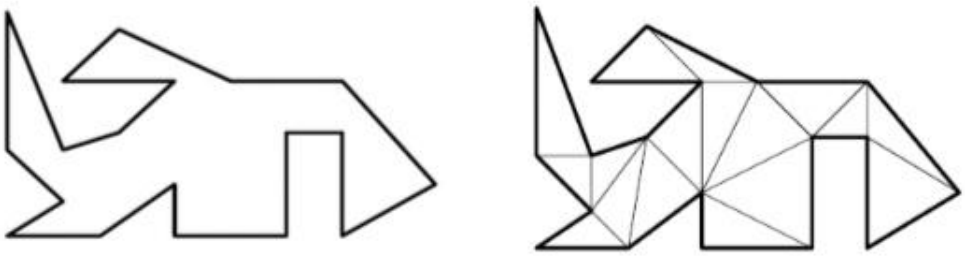

Figure 4.9: A polygonal element and the triangular element's representation of an underlying geometry [54].

Figure 4.9 shows how a polygonal element has to be split into seventeen triangular elements to accurately represent its shape, thus producing a larger matrix.

Polytopic meshes are extremely valuable for our application as the human body is full of complex geometries which have different material properties, i.e. reaction to radiation of different organs. Being able to form our meshes around such material properties avoids edge cases where a single element has different material properties within it.

Polytopic meshes can be generated in many ways, with different approaches producing meshes with very different element shapes. In this report, we will focus on two different forms, Voronoi tessellation and agglomeration.

4.4.1 Voronoi mesh generation

A Voronoi tessellation is defined by:
In space X with metric d with set of points K, I randomly generate them, and let $(P_k)_{k\in K}$ be a subset of X containing point k [29, 9]. The Voronoi region for a point k is then defined as the set

$$R_k = \{x \in X \mid d(x, P_k) \leq d(x, P_j) \text{ for all } j \neq k\}. \tag{4.4.1}$$

The Voronoi tessellation of space X is the set of $(R_{\kappa_\Omega})_{k\in K}$ such that $\bigcup_{\kappa_\Omega} R_{\kappa_\Omega} = X$. In Euclidean space, the Voronoi regions are convex polytopes, polygons in 2D and polyhedra in 3D. To form a Voronoi mesh, we need $(R_{\kappa_\Omega})_{k\in K} \bigcap \Omega$. Our boundary elements will be guaranteed to be convex if our domain Ω is itself convex, as the intersection of two convex sets is itself convex. In Figures 4.10, 4.11 and 4.12 we see examples of Voronoi meshes on $\Omega = [0, 1]^2$.

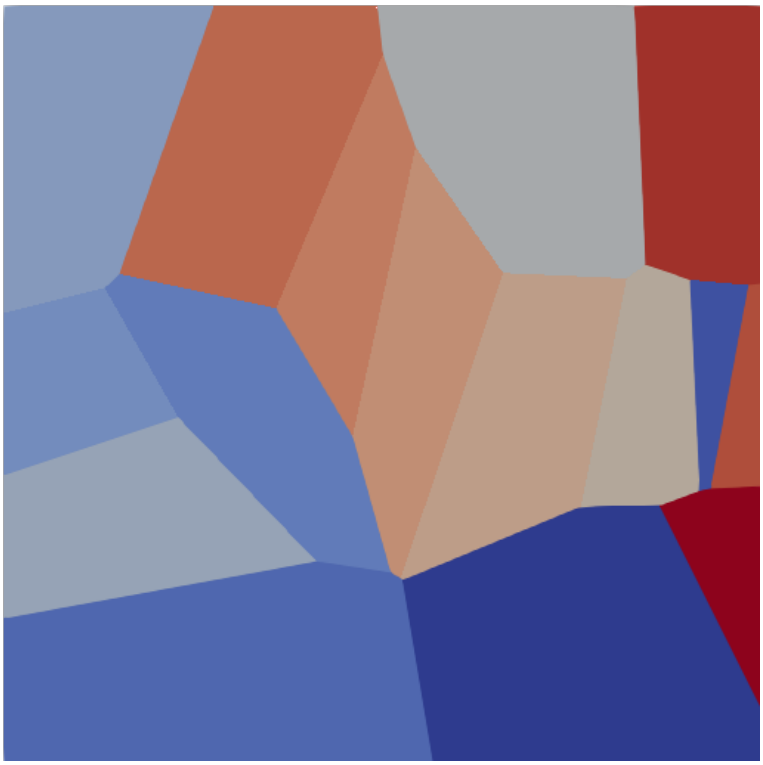

Figure 4.10: A Voronoi mesh with 16 elements.

Figure 4.11: A Voronoi mesh with 64 elements.

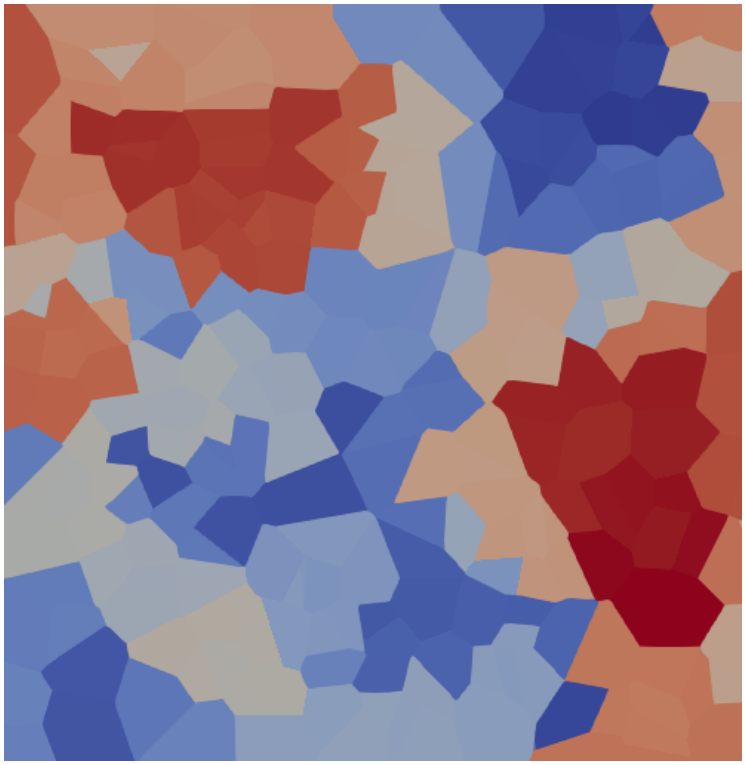

Figure 4.12: A Voronoi mesh with 256 elements.

The Voronoi mesh examples show how different in shape and size the resulting elements can be, with the number of faces for an element varying between three and eight.

4.4.2 Agglomerated meshes

An agglomerated mesh is a mesh whose elements are defined by the union of contiguous elements of an underlying fine mesh. The fine mesh is usually formed with standard elements, small enough to capture the underlying geometry. An algorithm then groups these fine elements into the coarse elements. There are a multitude of different algorithms that can be used to group the elements.

METIS is a graph partitioning software package [42, 41]. We can convert the mesh into a graph $G = (V, E)$ with elements as the vertices V and faces as the edges E, with $|V| = N_\Omega$. Using METIS to apply a multilevel graph partitioning algorithm to partition the graph into l subsets $V_1, V_2, ..., V_l$ such that $V_i \cap V_j = \varnothing$ for $i \neq j, |V_i| \approx \frac{N_\Omega}{l}$ and $\bigcup_i v_i = V$, and the number of edges connecting different V_i is reduced [43]. METIS will output l graph partitions, which we can convert back into polygonal coarse elements, formed from a roughly equal number of fine elements, with a minimised number of faces between coarse elements. Figures 4.13, 4.14, and 4.15 show examples of the coarse meshes produced by using METIS on a fine mesh of 8,192 triangles.

While the elements are the same size, their shape will present some problems. Due to there being no restriction on the element's shape, we get elements that have a lot

of faces. The resulting matrix will end up being relatively dense, with the number of faces for an element varying between three and thirteen in these examples. Another issue with the shape of these elements is the large number of subfaces, with many forming jagged edges that will result in cyclic dependences. A cyclic dependence, in a transport problem, is where an element is simultaneously up and down wind from another element, directly or indirectly, see Section 5.2.

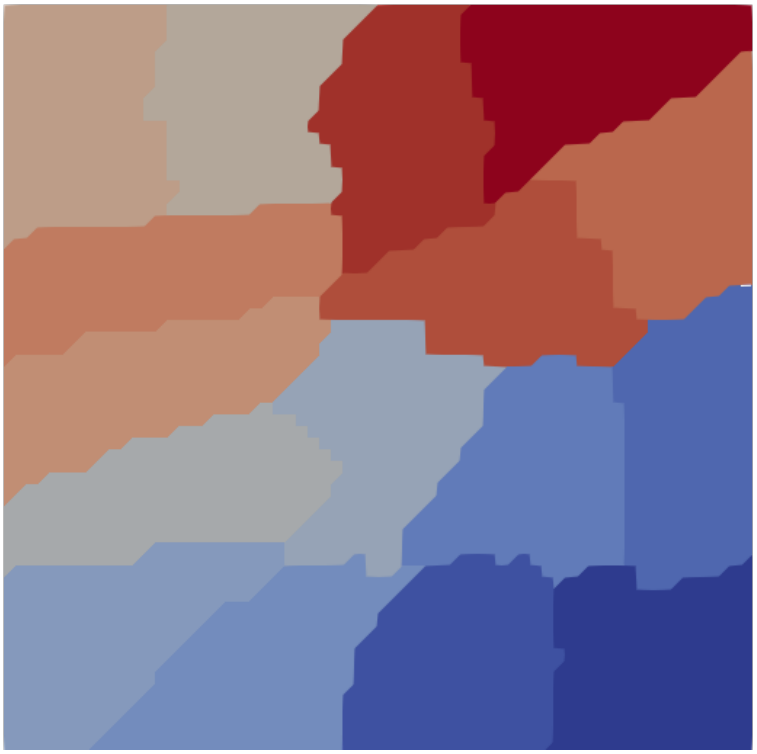

Figure 4.13: An agglomerated polygon mesh produced by METIS with 16 elements.

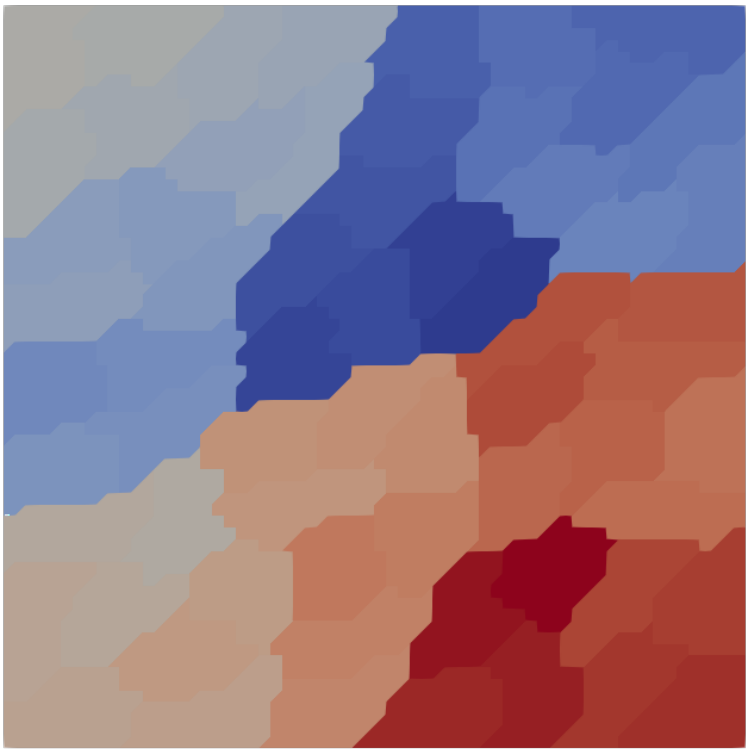

Figure 4.14: An agglomerated polygon mesh produced by METIS with 64 elements.

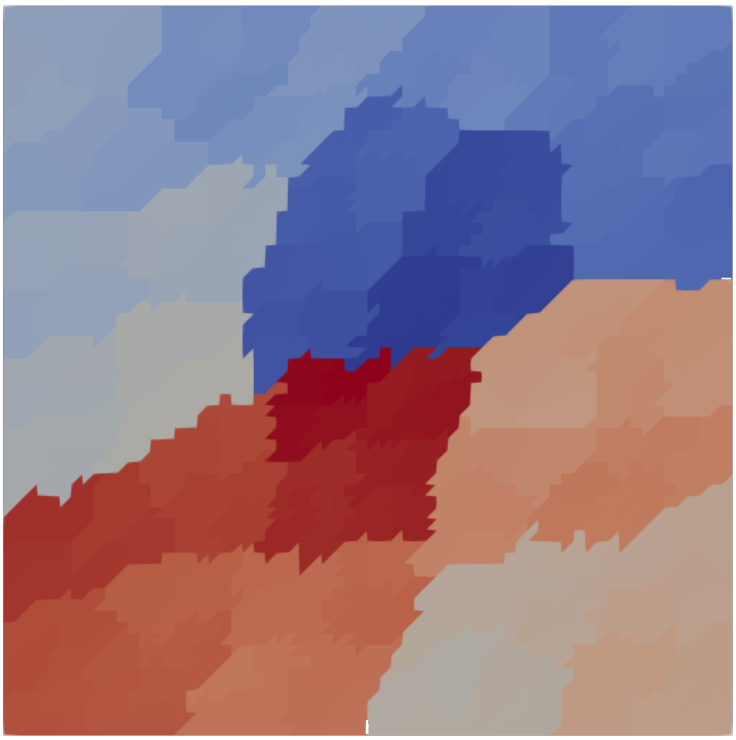

Figure 4.15: An agglomerated polygon mesh produced by METIS with 256 elements.

The agglomerated mesh examples show how different in shape and size the resulting elements can be, even compared to the Voronoi examples. Another way we can form an agglomerated mesh is by grouping the fine mesh into predefined shaped coarse elements.

To do this, we need a rubric that knows how to form this shape and how many fine elements are needed to form one coarse element. In our example, we agglomerate a quadrilateral fine mesh into a quadrilateral coarse mesh. This is a very simple example, but other examples could be made.

Having presented the different element shapes and discussed their expected properties, we will give some test examples of solving the transport equation with the different element types.

4.5 Timings for different element types

4.5.1 2D

We will be solving this transport problem introduced in Section 2.3.1: With $\Omega = (0,1)^2$

$$\begin{aligned} \nabla \cdot (au) + u &= 2\pi \cos(2\pi x)\sin(2\pi y) + 2\pi \cos(2\pi y)\sin(2\pi x) + \sin(2\pi x)\sin(2\pi y) \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial_- \Omega. \end{aligned} \tag{4.5.1}$$

Where $a = (1,1)^T$. This has an analytical solution $u = \sin(2\pi x)\sin(2\pi y)$. We shall be using a direct sparse matrix solver, MUMPS, to solve these equations.

The triangular, convex polygonal, and agglomerated polygonal meshes have been made with the same number of degrees of freedom (DOFs), so we can see the effects of using the different element types on the time to solve this system.

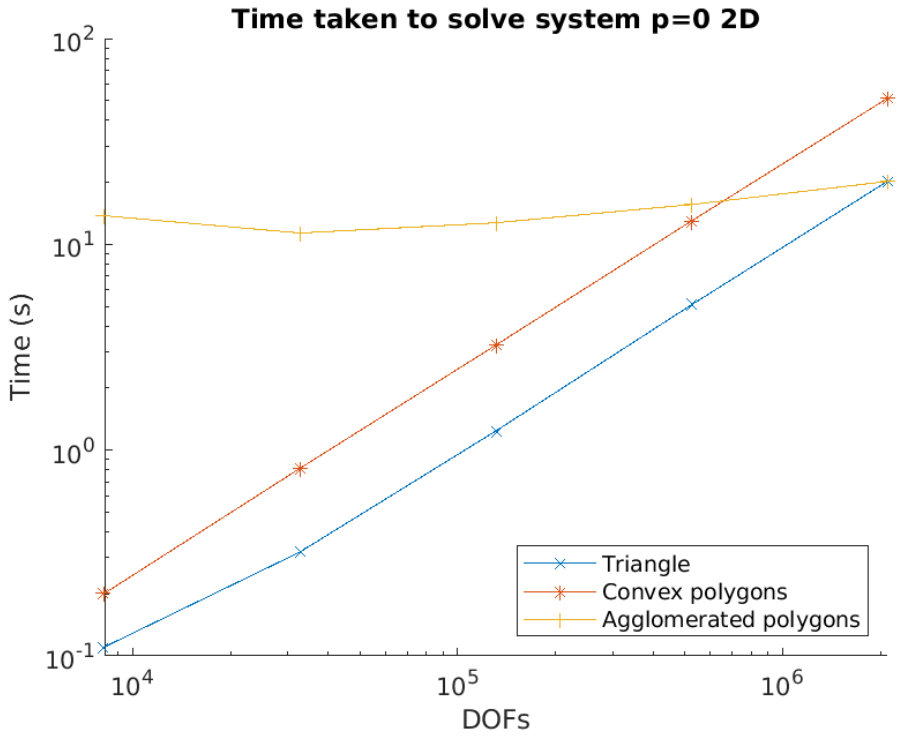

Figure 4.16: Time taken to solve the system with different element types with $\boldsymbol{p} = 0$

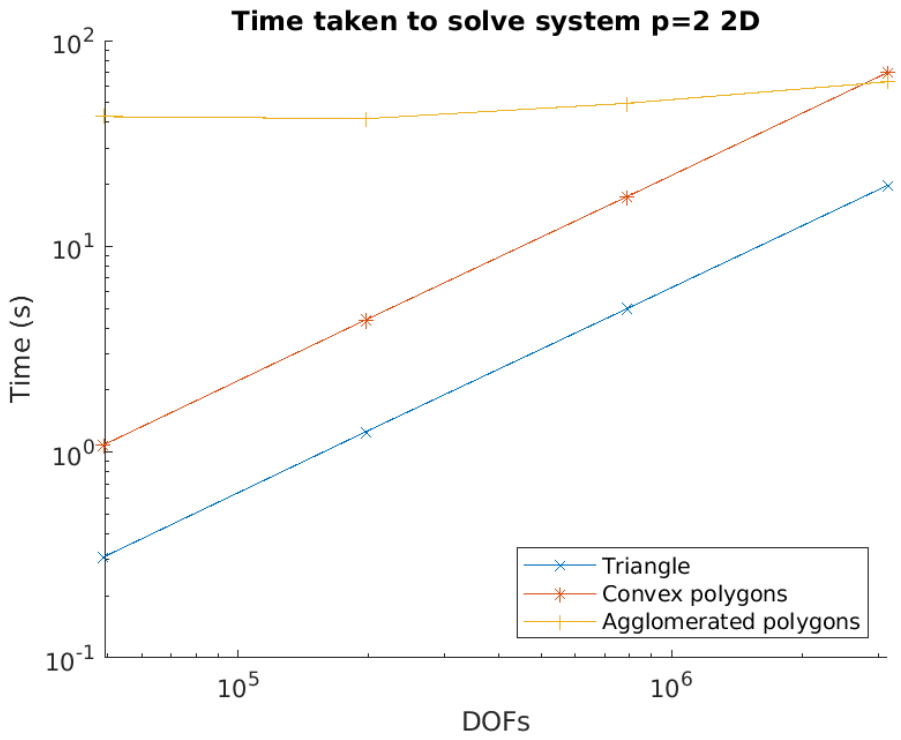

Figure 4.17: Time taken to solve the system with different element types with $\boldsymbol{p} = 2$

From Figures 4.16 and 4.17 we can see that the triangular elements are the fastest to solve for the same number of DOFs, as expected. The convex polygons created by our Voronoi tessellation take slightly longer than the triangular elements. The agglomerated elements' matrix takes considerably longer to solve the system than the other elements, this isn't due to the matrix structure but because they are agglomerated elements. To evaluate the weak form on the agglomerated elements, we must evaluate the weak form on each of the underlying fine mesh elements. We are evaluating the weak form on the same 2,097,152 triangular elements that were used to form the agglomerated partition. For $\boldsymbol{p} = 0$ it is clear that evaluating on the fine mesh is dominating any effect of the matrix density in the timing. This can be seen in the final run, where both matrices are the same size, i.e. the fine and coarse meshes have the same number of elements, and take comparable times.

For, $\boldsymbol{p} = 2$ we can see that agglomerated polygons take considerably longer to solve. As both algorithms are evaluating the same number of elements and have matrices the same size, this difference must be due to the increase in the density of the agglomerated polygons matrix.

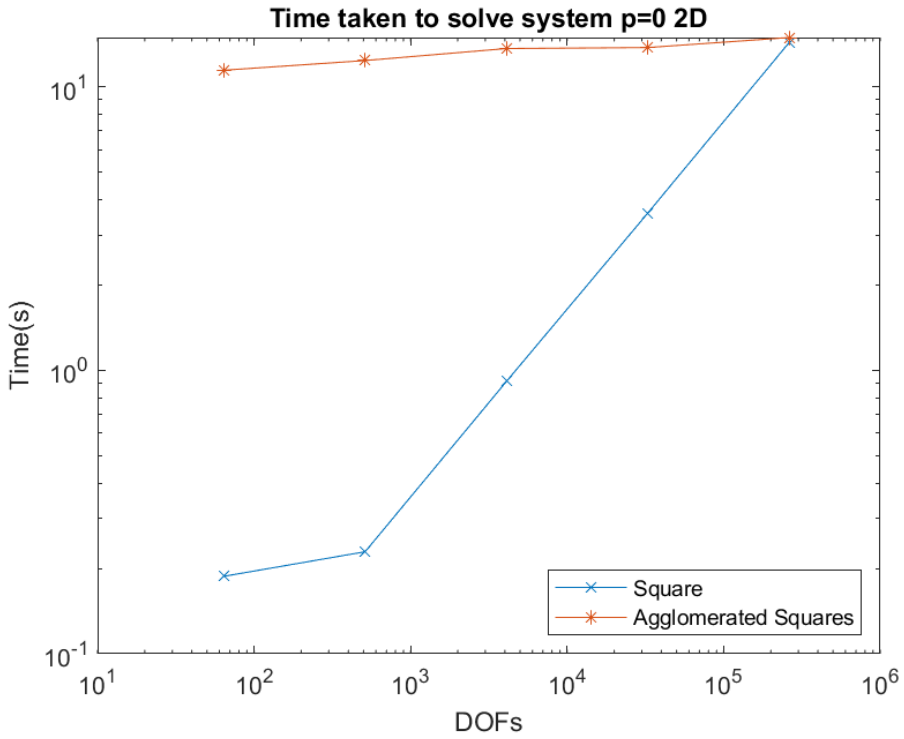

Figure 4.18: Time taken to solve the system with different element types with $\boldsymbol{p} = 0$

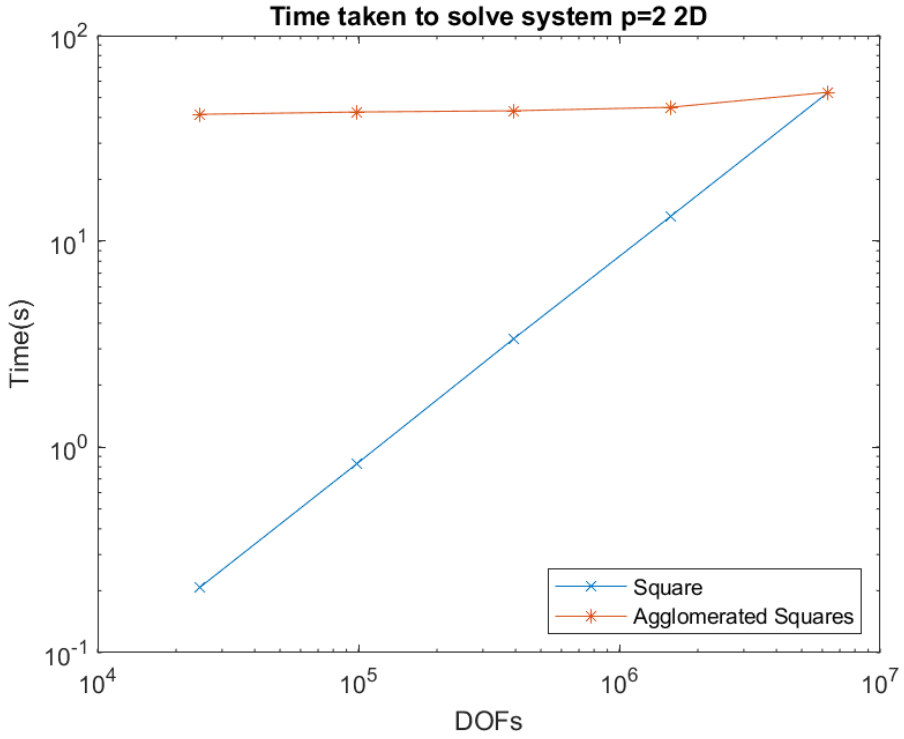

Figure 4.19: Time taken to solve the system with different element types with $\boldsymbol{p} = 2$

Our agglomerated squares have the same issue as the agglomerated polygons, in that they are still evaluated on the fine mesh as shown in Figures 4.18 and 4.19. Continuing our investigation in 3D gives similar results.

4.5.2 3D

For our numerical demonstrations in 3D, we will use this transport problem: With $\Omega = [0,1]^3$

$$\begin{aligned} \nabla \cdot (au) + u &= 2\pi \cos(2\pi x)\sin(2\pi y)\sin(2\pi z) + 2\pi \cos(2\pi y)\sin(2\pi x)\sin(2\pi z) \\ &\quad + 2\pi \cos(2\pi z)\sin(2\pi x)\sin(2\pi y) + \sin(2\pi x)\sin(2\pi y)\sin(2\pi z) \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial_- \Omega. \end{aligned} \tag{4.5.2}$$

Where $a = (1,1,1)^T$. This has an analytical solution $u = \sin(2\pi x)\sin(2\pi y)\sin(2\pi z)$. The tetrahedral and agglomerated polyhedral meshes have been made with the same number of DOFs, so we can see the effects of using the different element types on the time to solve this system.

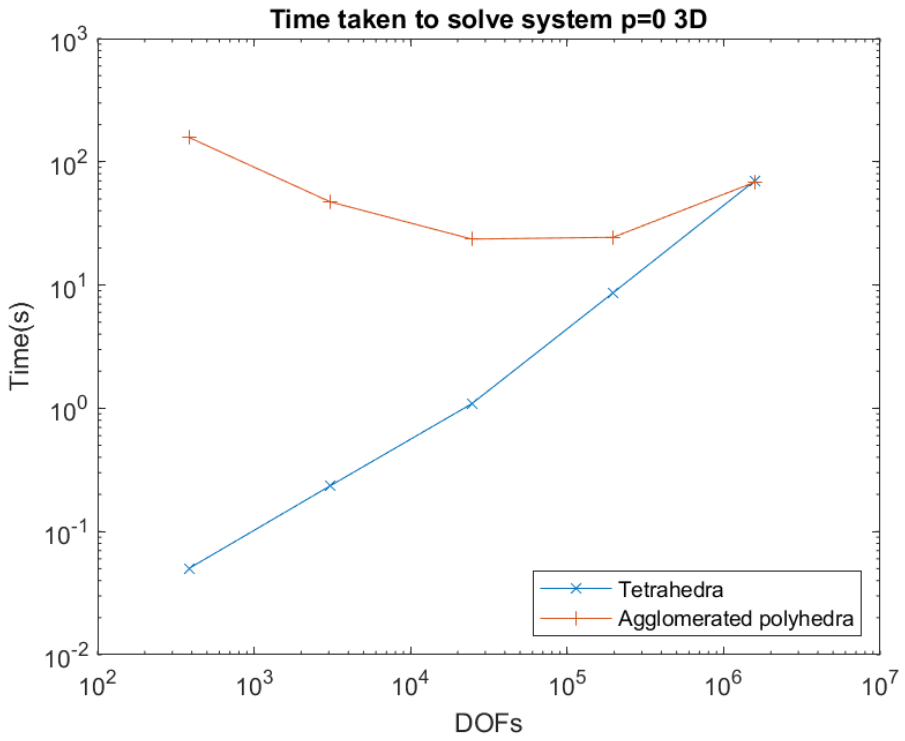

Figure 4.20: Time taken to solve the system with different element types with $\boldsymbol{p} = 0$

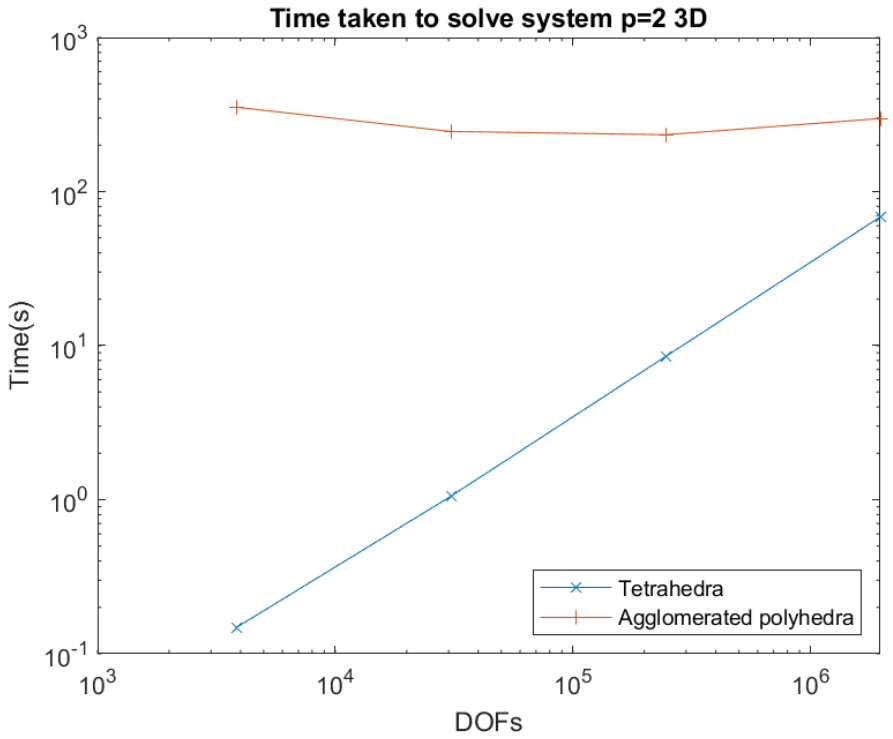

Figure 4.21: Time taken to solve the system with different element types with $\boldsymbol{p} = 2$

Figure 4.20 shows a very interesting phenomenon, where the time taken to solve the system appears to reduce as the matrix size increases.

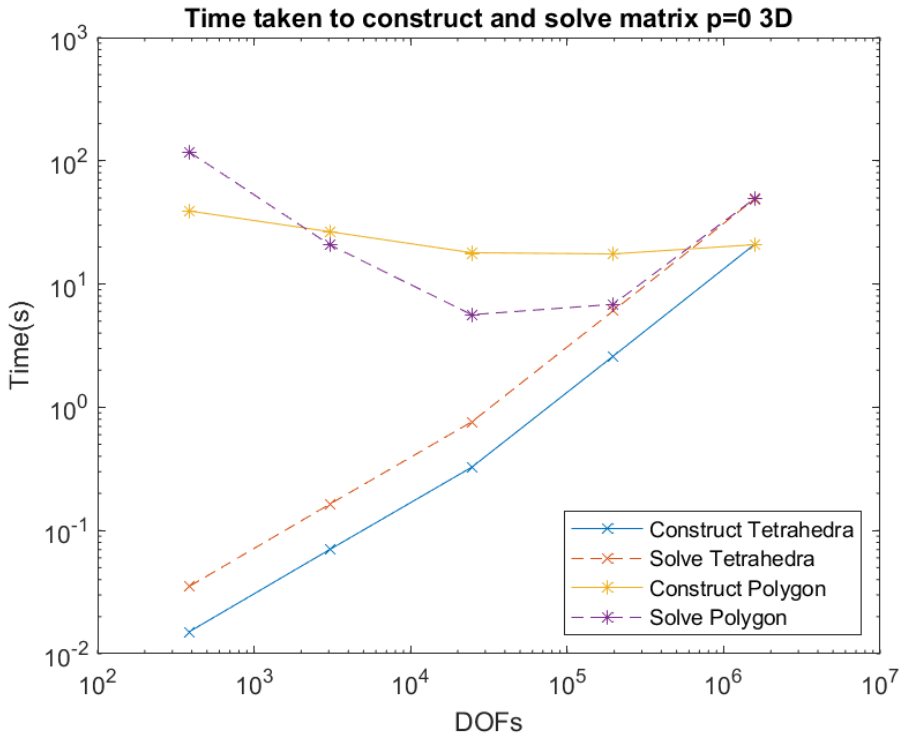

Figure 4.22: Time taken to construct and solve the matrix with $\boldsymbol{p} = 0$

Consulting Figure 4.22 shows that this is due to solving the matrix rather than

constructing it. This is likely to be due to a dense matrix formed by many coarse elements being connected by faces.

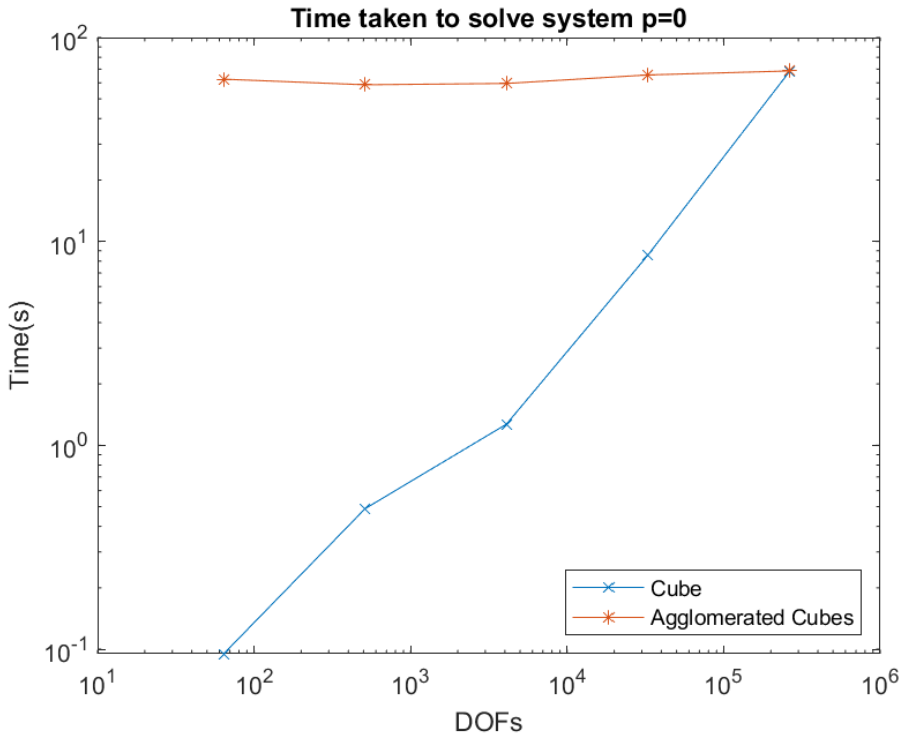

Figure 4.23: Time taken to solve the system with different element types with $\boldsymbol{p} = 0$

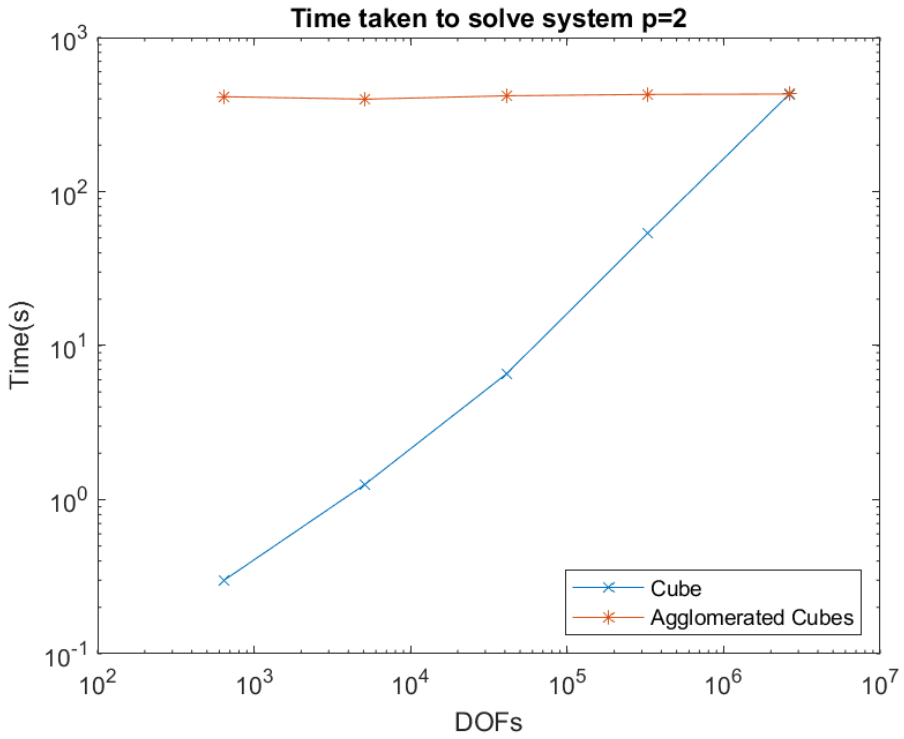

Figure 4.24: Time taken to solve the system with different element types with $\boldsymbol{p} = 2$

Agglomerated cubes, again, show the results we expect, evaluating the PDE on coarse elements being the dominating factor for the time to solve the system, as shown in Figures 4.23 and 4.24.

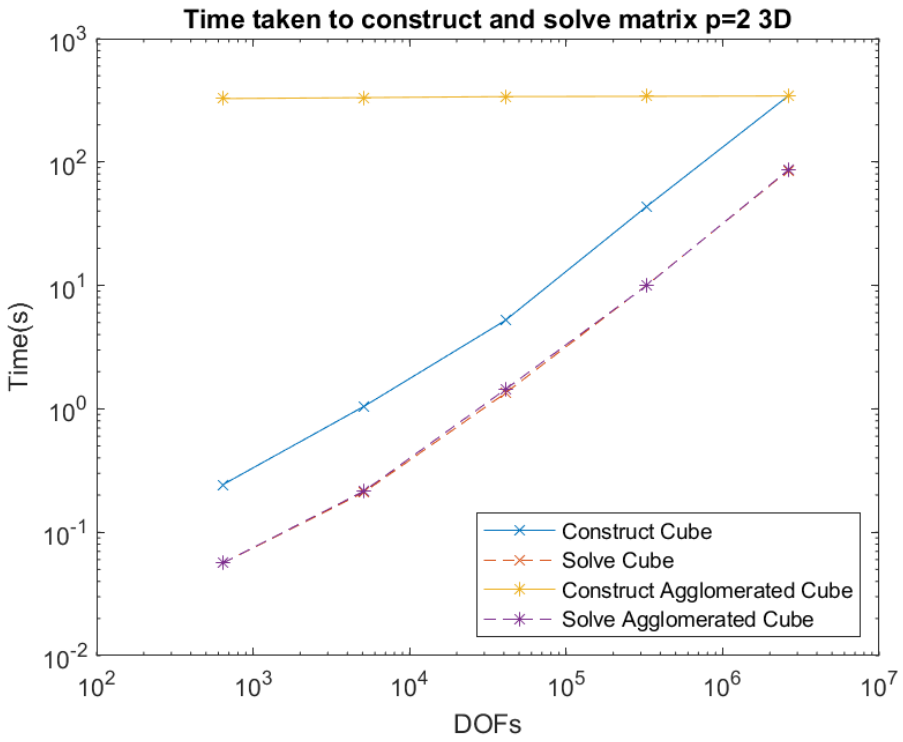

Figure 4.25: Time taken to construct and solve the matrix with $\boldsymbol{p} = 2$

We can demonstrate this in more detail with Figure 4.25, which shows that the time to solve the matrix formed by the agglomerated cube mesh is almost exactly the same as the cube mesh, and the only difference in the timing comes from the construction of this matrix. In fact, in the example, the resulting matrix should be identical due to how we have constructed our meshes and thus the solve times are identical.

4.6 Summary

Throughout this chapter, we have broken down and discussed the structure of the matrix that results from the DOG discretisation. We have focused on the matrices formed by the transport problem, as the DOG scheme results in a system of transport equations coupled in the right hand side. We discussed and demonstrated how the sparsity of these matrices is affected by the choice of elements in the spatial mesh, and how that, despite not increasing the DOFs in the system, the number of faces an element has directly correlates to the time taken to construct and solve the transport matrix.

We also showed how the elements can be grouped together via agglomeration, to reduce the number of DOFs in the system. Agglomerated elements, however, require the integrals over the fine elements to be evaluated to construct the coarse mesh matrix,

leading to construction times which are not dependent on the number of DOFs of the system but on the number of elements in the fine mesh.

In conclusion, although agglomerating elements through programs like METIS can reduce the size of the matrix, it does not follow that the time to solve the matrix system will be reduced. There are, however, still some advantages of having the smaller matrix size. So far we have been solving matrices that a computer can comfortably fit within its memory; however, for larger matrices such as those formed from MRI voxels, this limitation may become more important. Additionally, agglomerating into convex coarse elements, such as with our agglomerated cubes, leads to far more predictable results. In the next chapter, we show a new solver we developed, which allows our system to be solved without storing the entire matrix, and we will continue to experiment with different element types and show how they interact with our custom solver.

Chapter 5

Efficiently solving the linear Boltzmann transport equation

We showed in Section 3.8, that our DOG discretisation results in a system of transport equations coupled on the right hand side, solving the transport equation efficiently which will result in an efficient solver for the LBTE. As commented in the previous section, 4, the choice of element affects not only the construct time of the matrix, but also the time taken to solve the matrix system. We also commented on how expensive large matrices are to store. In this section, we present our customised transport equation solver, the sweep solver.

This solver evaluates the weak formulation on each element, or group of elements in the case of cyclic dependencies, individually, and then inverts the matrix equation for that element, to get the solution for that element. It then uses the solution of that element, to solve the upwind elements by moving the face blocks to the right hand side of the matrix equation. In addition, this solver uses quadrature free methods to construct the matrix blocks. This code was provided by our colleague, Dr. Thomas Radley, the details of the exact construction can be found in his thesis [55].

This results in a very fast solver that does not require the full matrix to ever be constructed or stored, for convex elements. For non-convex elements, cyclic dependencies can form. We suggest a solution to this, using Tarjan's algorithm to detect these dependencies and evaluate and invert the matrix for these groups of elements, instead of inverting for the individual elements. Finally, we test the effect that different element types have on the efficacy of our sweep solver.

To solve a matrix equation $Ax = b$ where A is a sparse matrix, there are many different approaches and algorithms. There are a variety of freely and commercially available software packages that offer highly optimised solvers for sparse matrixes. We will be using MUMPS [65] as our solver for the fully constructed sparse matrix systems; however, due to the size of the potential transport matrices formed from MRI or CAT

scan data, constructing the full transport matrix for each advection direction from our DOG discretisation, may not viable. If it is possible to construct the matrix, solving the matrix system may likewise be deemed to demand too much time and memory to be viable. We, therefore, have developed the algorithm, known as a sweep solver, to solve the matrix equation on an element by element basis. To do this, we exploit the known structure of the matrix of the transport problem, specifically the fixed advection direction that DOG discretisation gives us.

5.1 Sweep Solver

As the transport problem is strictly hyperbolic with respect to a constant wind direction, we know a priori which elements are dependent on the results of preceding elements. Exploiting this knowledge, we can pivot the rows of the matrix. By putting the downwind elements at the bottom of the matrix and making sure the element blocks are ordered to be below any element blocks it is upwind of, we obtain a matrix in block upper triangular form. This is the same as the pivots that many matrix equation solvers apply to get the matrix into row echelon form. For instance, using our example in Figure 4.1 we would normally get a matrix equation of the form:

$$\begin{bmatrix} [E1] & & [F13] & \\ [F21] & [E2] & & [F24] \\ & & [E3] & \\ & & [F43] & [E4] \end{bmatrix} \begin{bmatrix} [X1] \\ [X2] \\ [X3] \\ [X4] \end{bmatrix} = \begin{bmatrix} [B1] \\ [B2] \\ [B3] \\ [B4] \end{bmatrix}.$$

However, if we use our knowledge of the fixed wind direction $a = (1, 1)^T$, and make sure each element blocks is upwind of the element blocks under it, we get:

$$\begin{bmatrix} [E2] & [F24] & [F21] & \\ & [E4] & & [F43] \\ & & [E1] & [F13] \\ & & & [E3] \end{bmatrix} \begin{bmatrix} [X2] \\ [X4] \\ [X1] \\ [X3] \end{bmatrix} = \begin{bmatrix} [B2] \\ [B4] \\ [B1] \\ [B3] \end{bmatrix}.$$

Now that we have the matrix in upper triangular form, we know from the Gaussian elimination algorithm the matrix equation can be expressed as:

$$\begin{aligned} [E2][X2] &= [B2] - [F24][X4] - [F21][X1] \\ [E4][X4] &= [B4] - [F43][X3] \\ [E1][X1] &= [B1] - [F13][X3] \\ [E3][X3] &= [B3]. \end{aligned}$$

Notice that the solution $X3$ is now independent of any element block aside from $E3$.

Therefore, we can invert the element block to find the solution of the matrix equation on that element. Using this solution $X3$, we can then evaluate the right hand side of the matrix equations

$$[E1][X1] = [B1 - F13X3]$$
$$[E4][X4] = [B4 - F43X3].$$

So that the element blocks can be inverted to find the solutions $X1$ and $X4$. These solutions can then be used in the same way to find the solution for element two. Thus, the transport equation matrix can be inverted without ever constructing the full matrix.

5.1.1 Comparison between matrix and the sweep solver for convex elements

We will now compare the time taken to construct and solve a transport problem with a standard matrix solver and our sweep solver. For the matrix solver, we will be using a MUMPS based transport solver written by Prof. Paul Houston. In the sweep solver the element block matrix equations are inverted by a simple PLU matrix solver which I wrote, this is not an optimised solver but as the blocks it is inverting are rather small this should not make much of a difference. The sweep solver also incorporates quadrature free evaluation written by Dr Thomas Radley. The details of the exact construction of the quadrature free evaluation and the analysis of the procedure can be found in his thesis [55], but we provide a broad overview to aid the understanding of our results.

5.1.1.1 Quadrature free integration over polytopic domains

A function $f : \mathbb{R}^d \to \mathbb{R}$ is homogeneous of degree $k > 0$ if

$$f(\alpha \mathbf{x}) = \alpha^k f(\mathbf{x}) \quad \forall \alpha > 0 \text{ and } \forall \mathbf{x} \in \mathbb{R}^d$$

Euler's homogeneous function theorem [61] then states that if $f : \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable, then f is homogeneous of degree k if and only if

$$\mathbf{x} \cdot \nabla f(\mathbf{x}) = k f(\mathbf{x}) \quad \forall \mathbf{x} \in \mathbb{R}^d.$$

Then for some polytope $\mathcal{P}$ with faces $\{\mathcal{F}_i\}_{i=1}^{N_F}$ in $\mathbb{R}^d$ integrating over the polytope can be expressed as a series of surface integrals over its faces

$$\int_{\mathcal{P}} f(\mathbf{x}) \ d\mathbf{x} = \frac{1}{d+k} \sum_{i=1}^{m} a_i \int_{\mathcal{F}_i} f(\mathbf{x}) \ ds. \tag{5.1.1}$$

and then by the same arguments a surface integral over the face can be expressed as integrals over the $d-2$ boundary edges $\partial\mathcal{F}_i = \{\mathcal{E}_{ij}\}_{j=1}^{N_e}$

$$\int_{\mathcal{F}_i} f(\mathbf{x})\ ds = \frac{1}{d+k-1}\left(\sum_{j=1}^{N_e} d_{ij}\int_{\mathcal{E}_{ij}} f(\mathbf{x})\ d\nu + \int_{\mathcal{F}_i} \mathbf{x}_{i,0}\cdot\nabla f(\mathbf{x})\ ds\right). \tag{5.1.2}$$

With $\mathbf{x}_{i,0}$ being an arbitrary point lying in the same hyperplane $\mathcal{H}_i$ as $\mathcal{F}_i$. d_{ij} is the Euclidean distance between $\mathbf{x}_{i,0}$ and $\mathcal{E}_{ij}$. Thus, the integral over our polytope can be recursively dimensionally reduced to point evaluations [6].

We can then apply this to the DGFEM transport equation, thus we can reduce the integral on the polytope and its faces to be point evaluations. Another key property of quadrature free assembly is how it interacts with agglomerated meshes. Due to the reduction in dimension of the integral to the point evaluations, internal fine mesh elements, elements of the fine mesh that are in the coarse mesh but not contributing to its boundary, do not have to be evaluated. This means that for large coarse mesh elements, the number of fine mesh elements that have to be evaluated can be drastically reduced.

5.1.1.2 Square

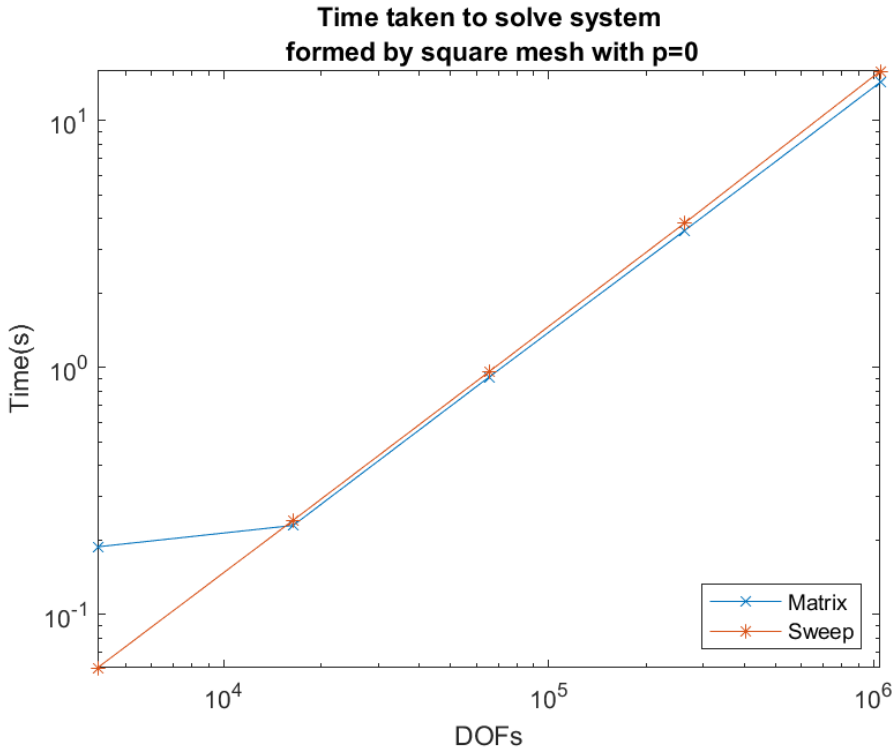

Figure 5.1: Time taken to solve system on a square mesh with $\boldsymbol{p} = 0$.

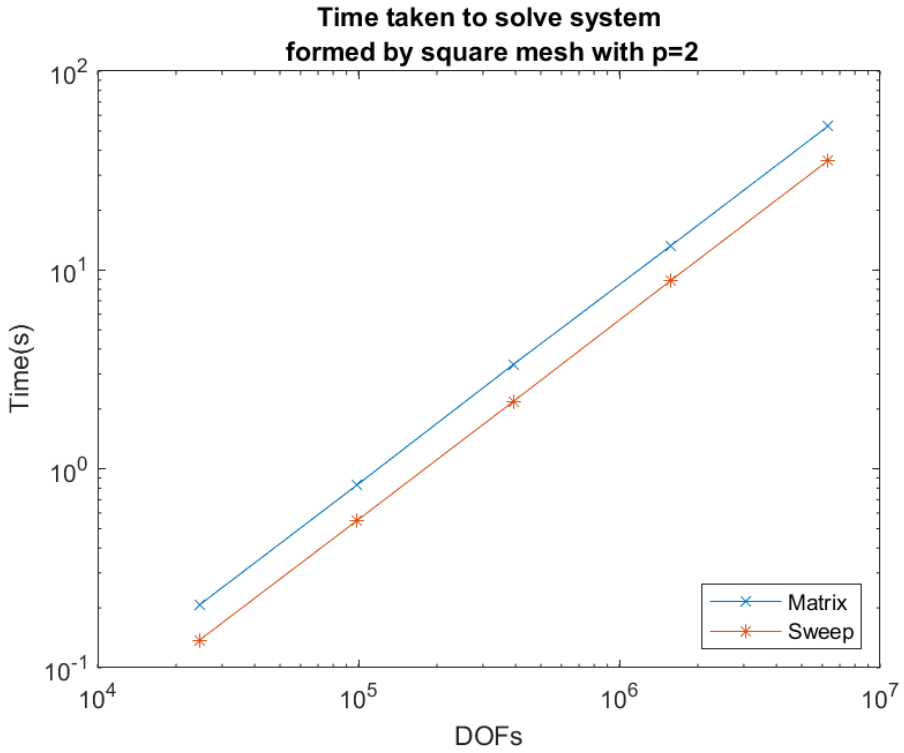

Figure 5.2: Time taken to solve system on a square mesh with $\boldsymbol{p} = 2$.

Comparing the time taken to solve the same transport problems, Figures 5.1 shows that, for the meshes with square elements, the sweep solver is slightly slower for $\boldsymbol{p} = 0$. This is not unexpected, the quadrature free evaluation is designed for large systems, and has some set up time associated, as Figure 5.3 shows. It is worth noting that due to the structure of the sweep solver there is no singular solver step, instead each time to invert each element's block is noted, and their total is taken away from the overall run time. As we increase $\boldsymbol{p}$ to two, Figure 5.2 shows a greater disparity in the time taken to solve the matrix system. Figure 5.4 shows that the time saved by the sweep solver is due to both the quadrature free evaluation and the sweep solver.

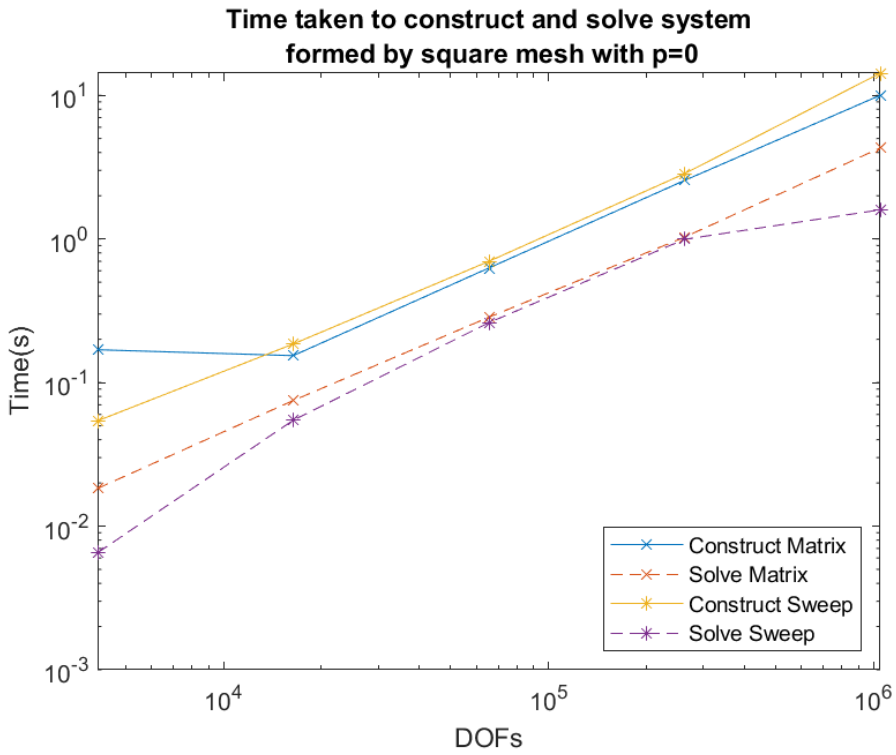

Figure 5.3: The breakdown of the construction and solve time for transport equation on square mesh $\boldsymbol{p} = 0$.

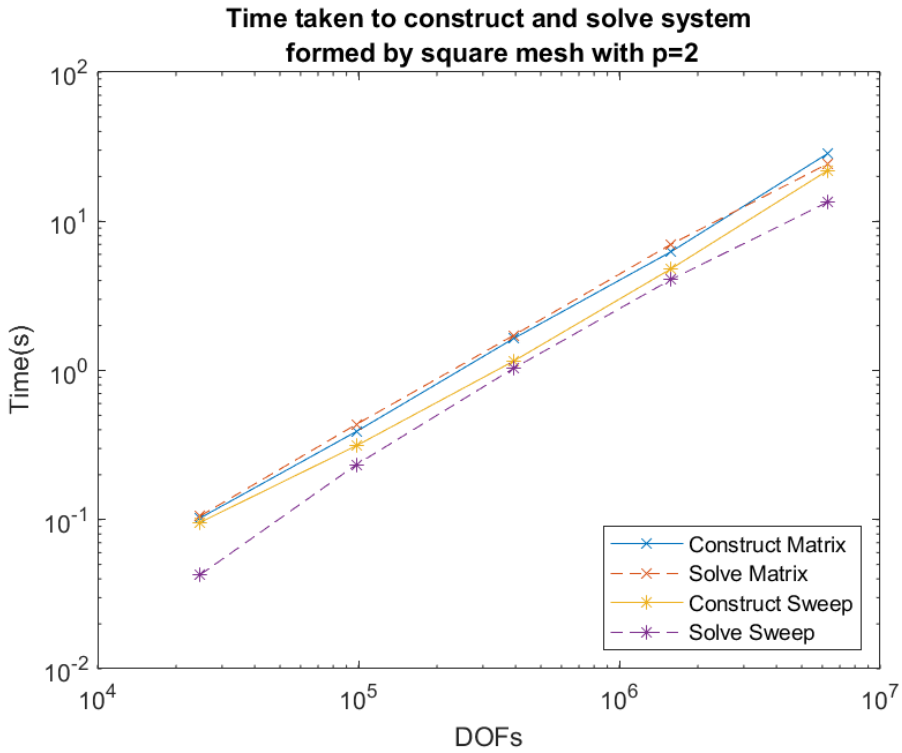

Figure 5.4: The breakdown of the construction and solve time for transport equation on square mesh $\boldsymbol{p} = 2$.

5.1.1.3 Triangular

Again, we see that the benefit of the sweep solver over an optimised matrix solver is minimal, though still existent, for a relatively low number of DOFs. Figure 5.6, however, shows at a higher polynomial degree the sweep solver is considerably faster than an optimised matrix solver.

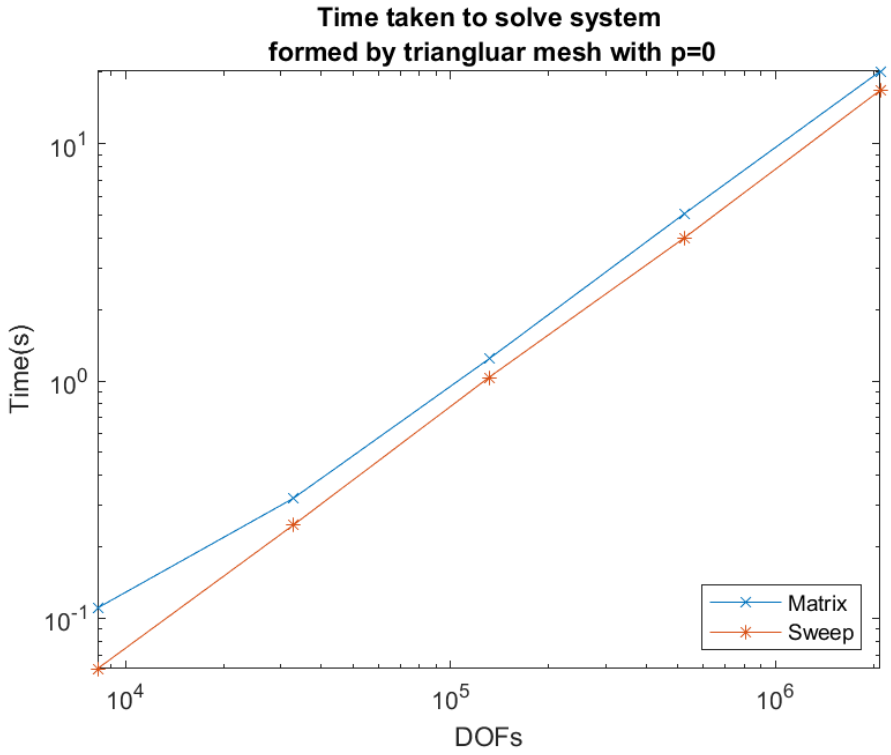

Figure 5.5: Time taken to solve system on a triangular mesh with $\boldsymbol{p} = 0$.

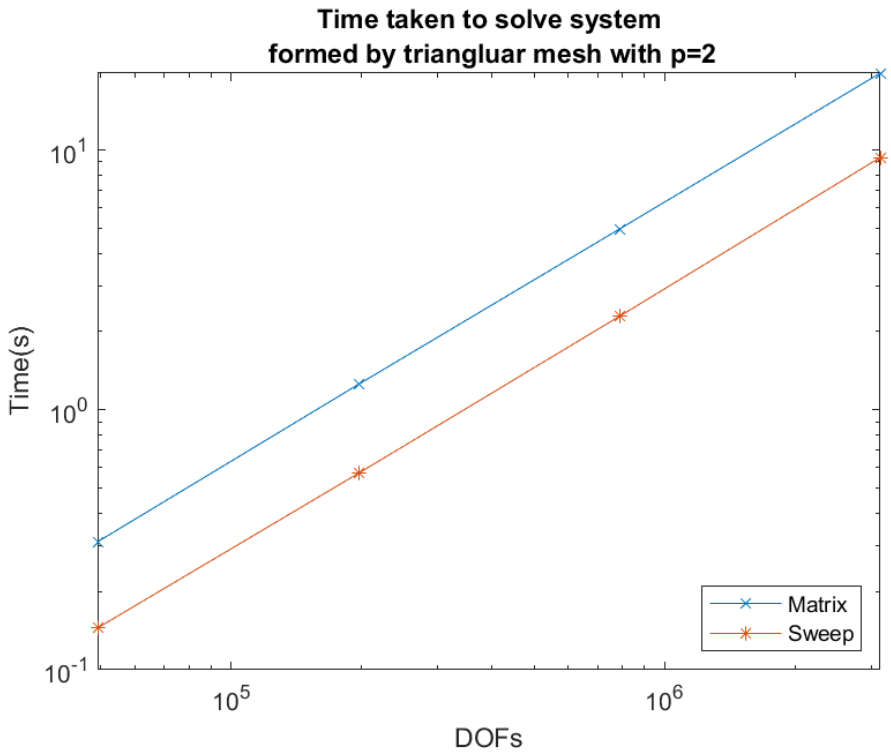

Figure 5.6: Time taken to solve system on a triangular mesh with $\boldsymbol{p} = 2$.

Figure 5.7 shows that we are making a saving on both the solve and the construction stages, at $\boldsymbol{p} = 2$.

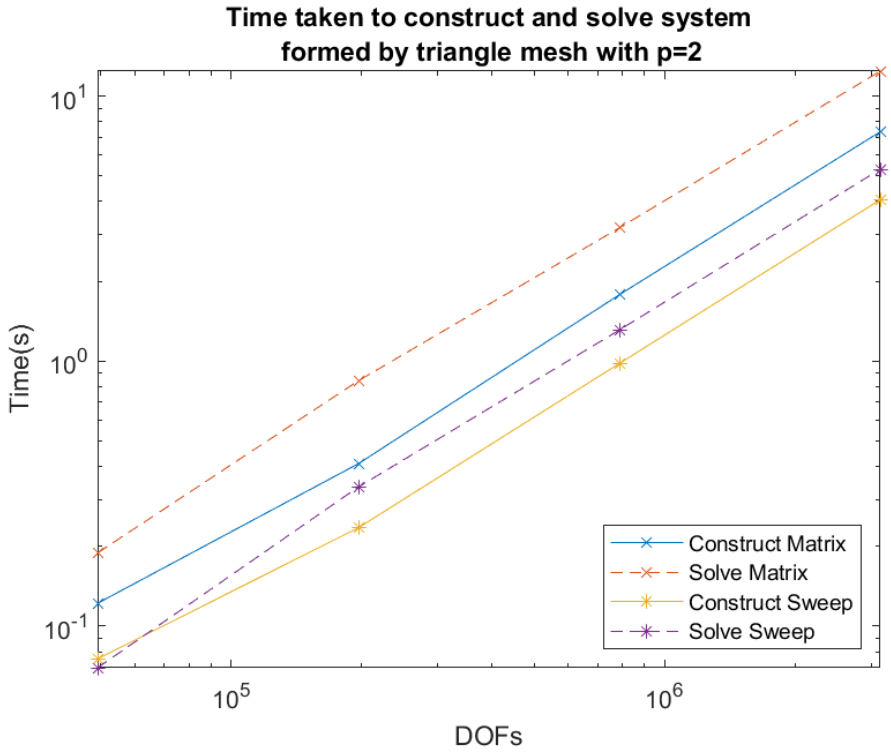

Figure 5.7: The breakdown of the construction and solve time for transport equation on triangluar mesh $\boldsymbol{p} = 2$.

5.1.1.4 Convex Polygons

Figure 5.8 shows that for our Voronoi tessellation generated meshes the sweep solver performs well for $\boldsymbol{p} = 2$, with constant saving at each step.

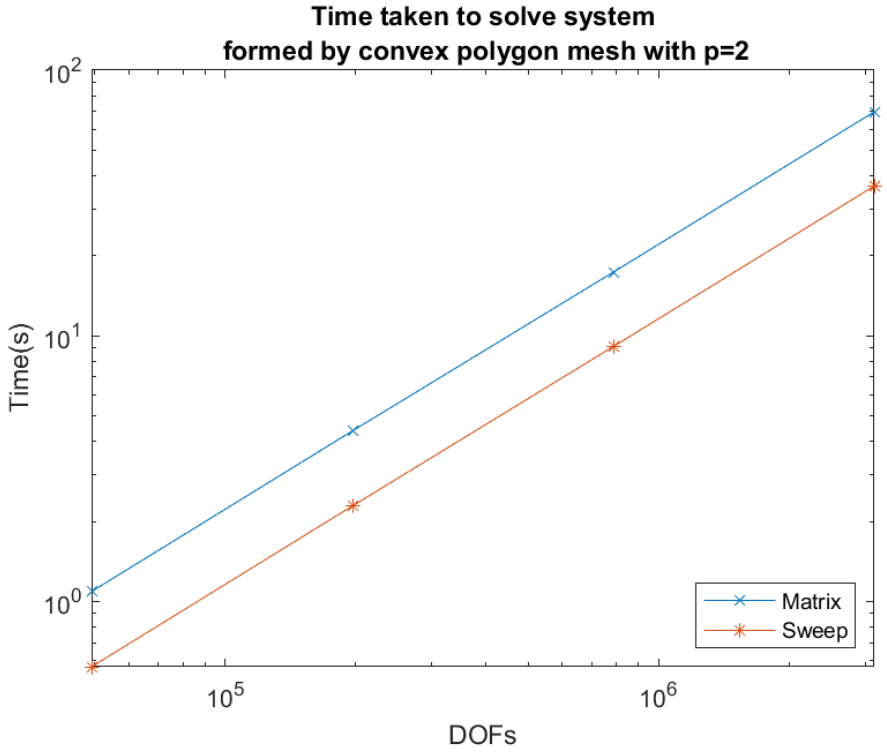

Figure 5.8: Time taken to solve on a convex polygonal mesh with $\boldsymbol{p} = 2$.

Convex polygons represent an additional challenge for the matrix solver, as it has to sub tessellate each element to define a quadrature on it. As with our agglomerated polygons, this increases the time taken for the evaluation of each element, as shown in Figure 5.9.

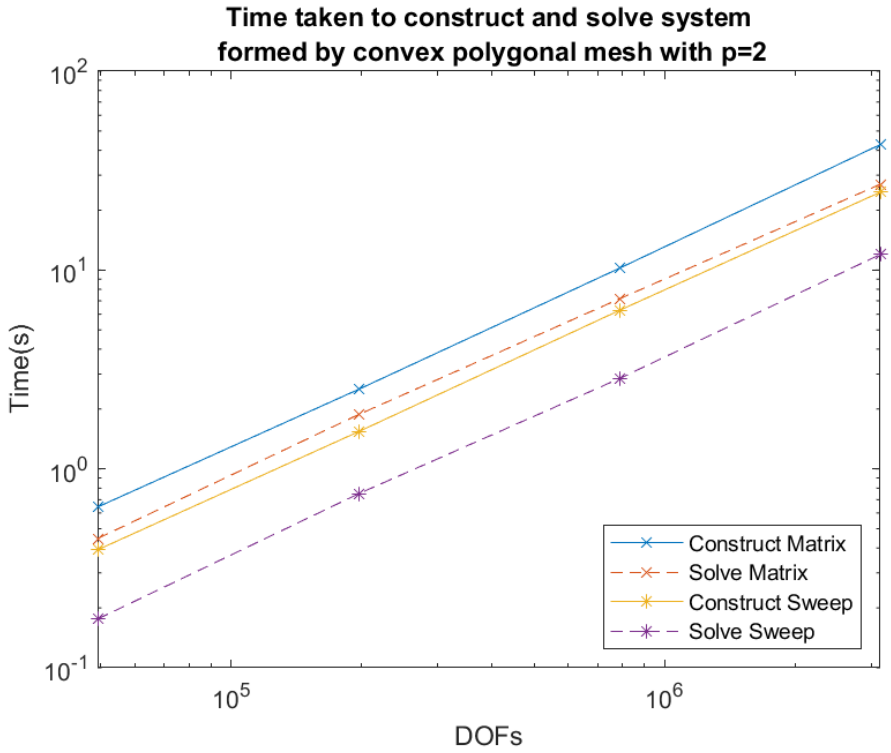

Figure 5.9: The breakdown of the construction and solve time for transport equation on convex polygonal mesh $\boldsymbol{p} = 2$.

5.1.1.5 Agglomerated Squares

As stated in the previous Section, agglomerating presents issues for the standard matrix solver, due to having to evaluate each element of the fine mesh. This explains the extreme difference shown in Figure 5.10. The quadrature free evaluation allows our sweep solver to ignore the fine elements that are not on the boundary of the coarse element.

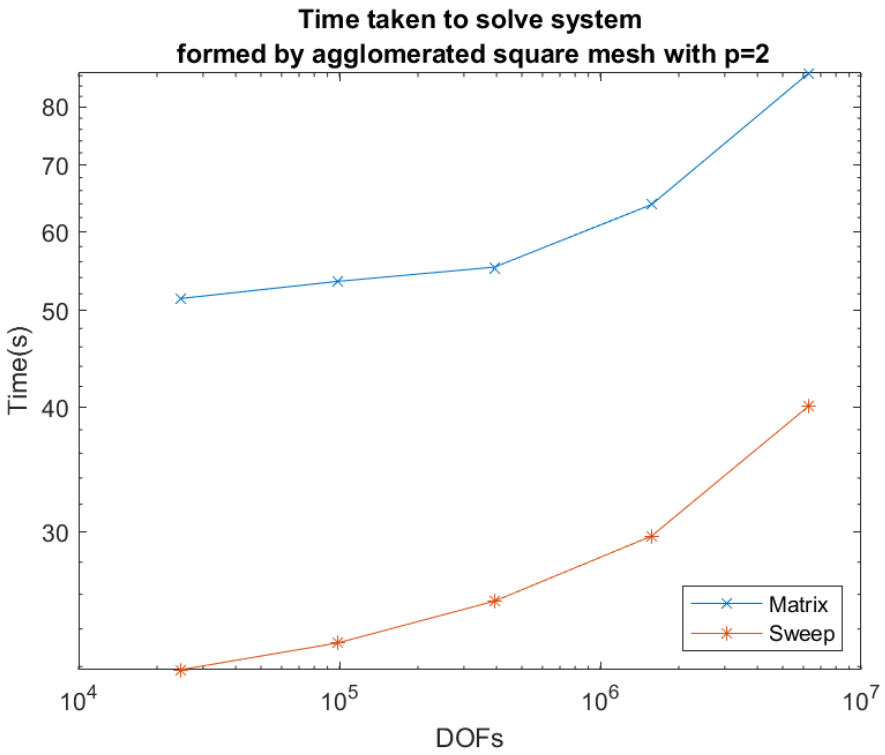

Figure 5.10: Time taken to solve system on an agglomerated square mesh with $\boldsymbol{p} = 2$.

Figure 5.11 shows just how dominated the matrix solver is by the construction of its matrix rather than solving the matrix. Happily, the sweep solver also offers a saving on the solve step, though nowhere near as significant as that offered by the quadrature free evaluation.

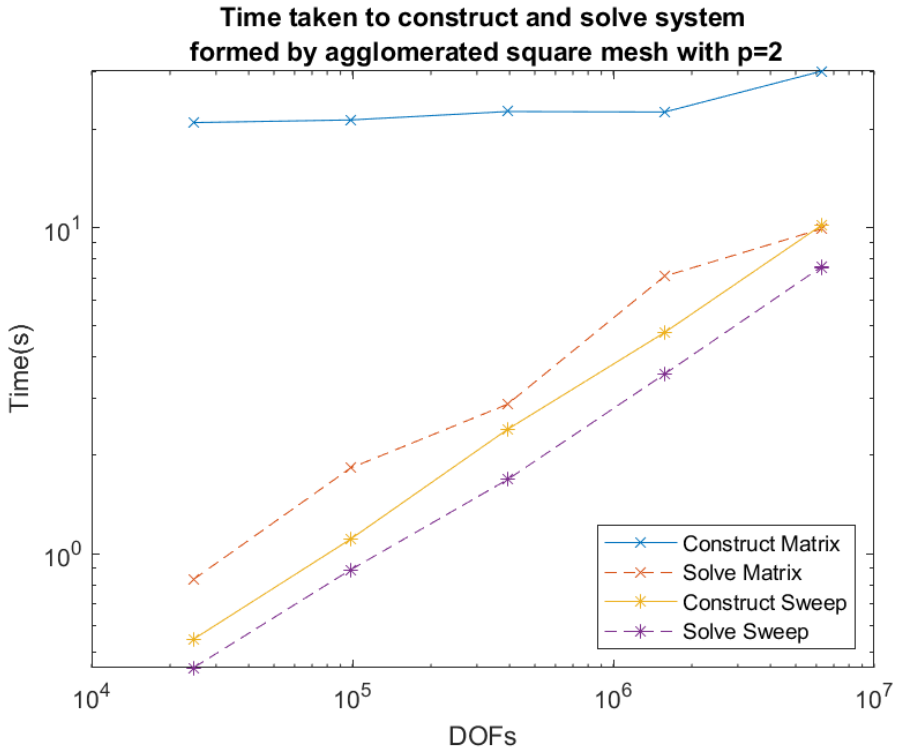

Figure 5.11: The breakdown of the construction and solve time for transport equation on agglomerated square mesh $\boldsymbol{p} = 2$.

5.1.1.6 Cube

Moving on to 3D we can see that the sweep solver offers savings for both $\boldsymbol{p} = 0$ and $\boldsymbol{p} = 2$. From Figure 5.14 we can see that for $\boldsymbol{p} = 0$ that the meshes with a low number of elements, the solve time is dominated by the construction of the matrix. We suspect that we have captured some dominating set up time rather than anything of particular significance.

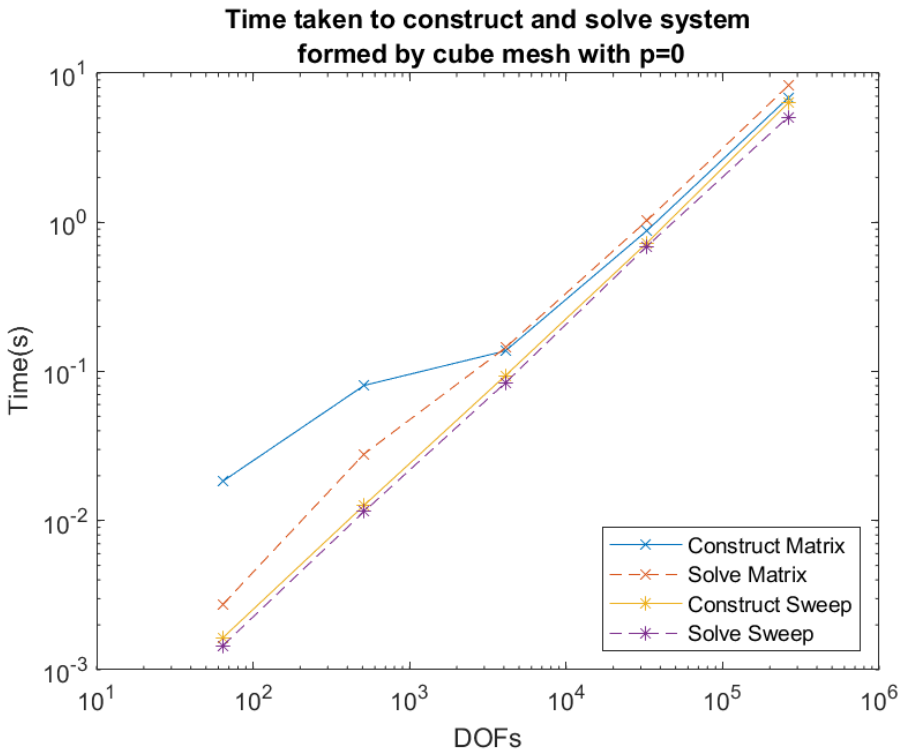

Figure 5.12: Time taken to solve system on a cube mesh with $\boldsymbol{p} = 0$.

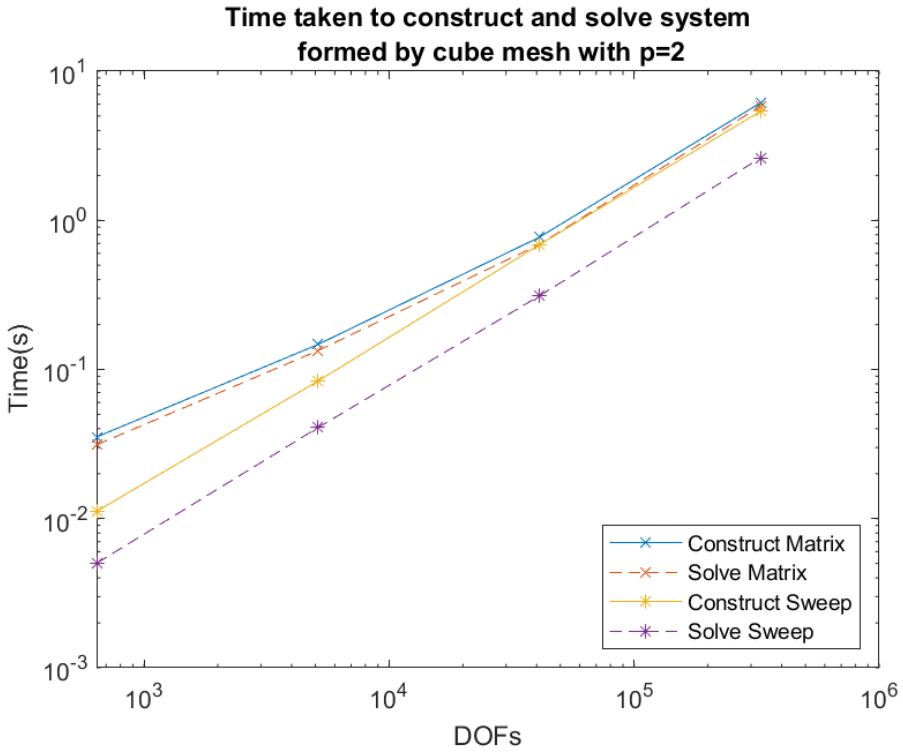

Figure 5.13: Time taken to solve system on a cube mesh with $\boldsymbol{p} = 2$.

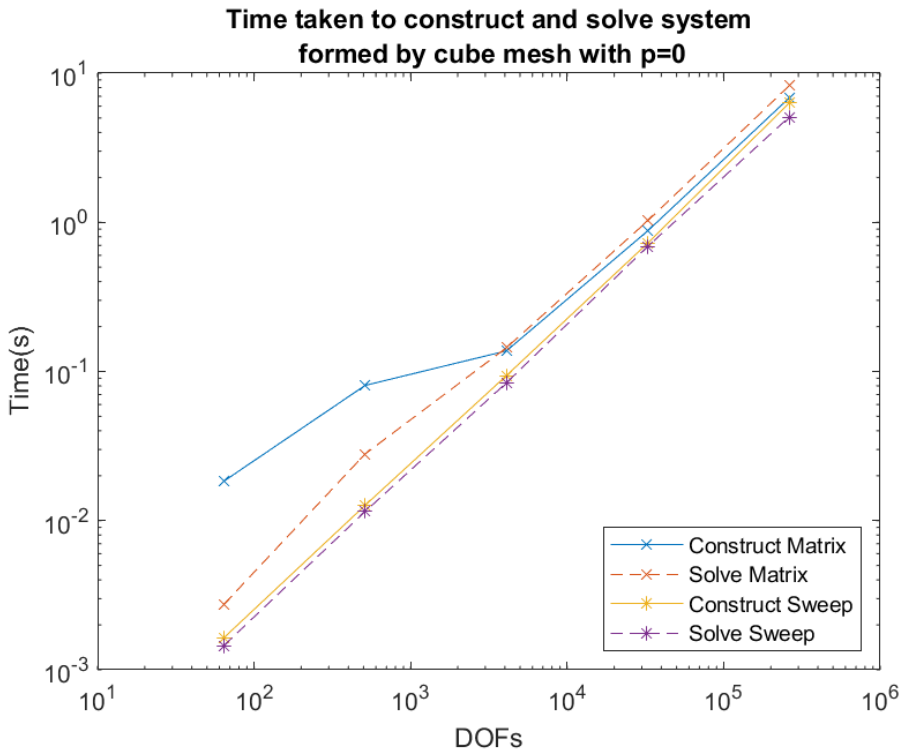

Figure 5.14: The breakdown of the construction and solve time for transport equation on cube mesh $\boldsymbol{p} = 0$.

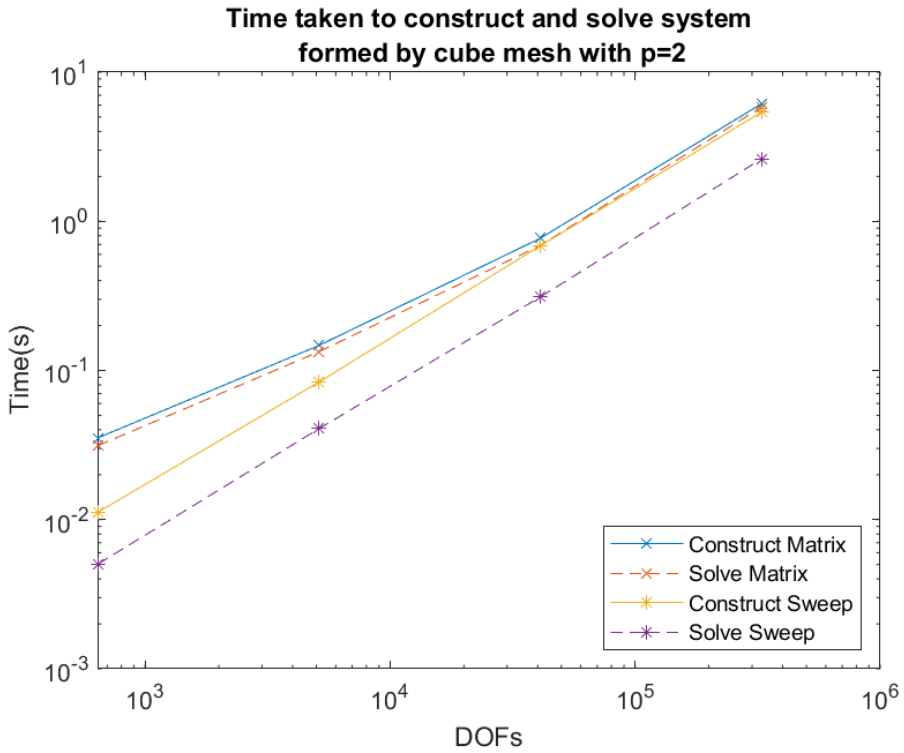

Figure 5.15: The breakdown of the construction and solve time for transport equation on cube mesh $\boldsymbol{p} = 2$.

5.1.1.7 Tetrahedral

The matrix formed by a tetrahedral mesh, is solved by our sweep solver more efficiently than the optimised matrix solver. Looking at Figure 5.17 shows that most of this saving can be attributed to the sweep solver rather than the quadrature free construction.

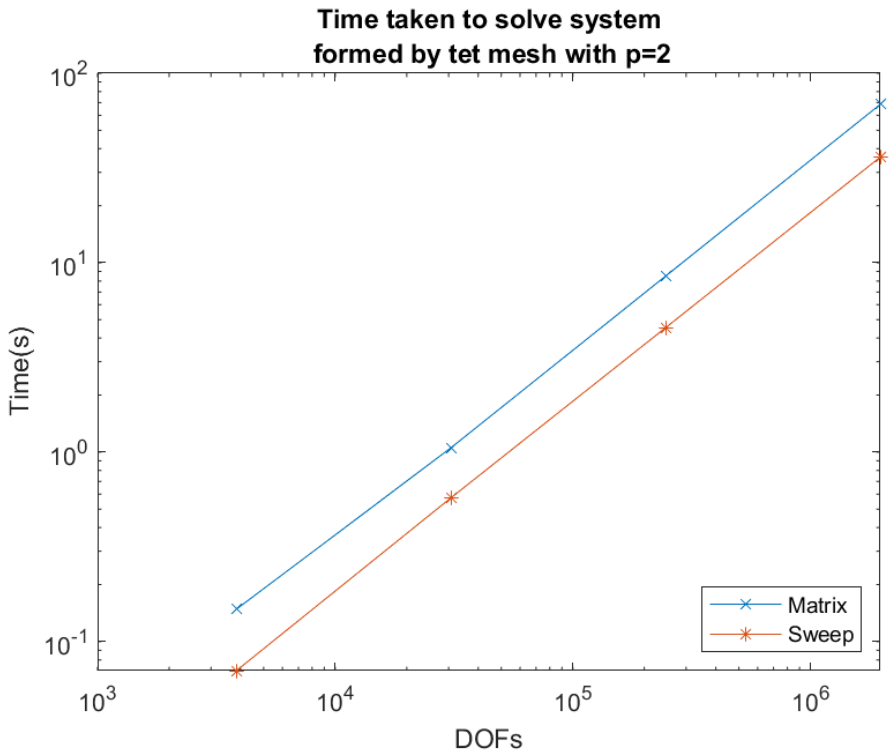

Figure 5.16: Time taken to solve system on a tet mesh with $\boldsymbol{p} = 2$.

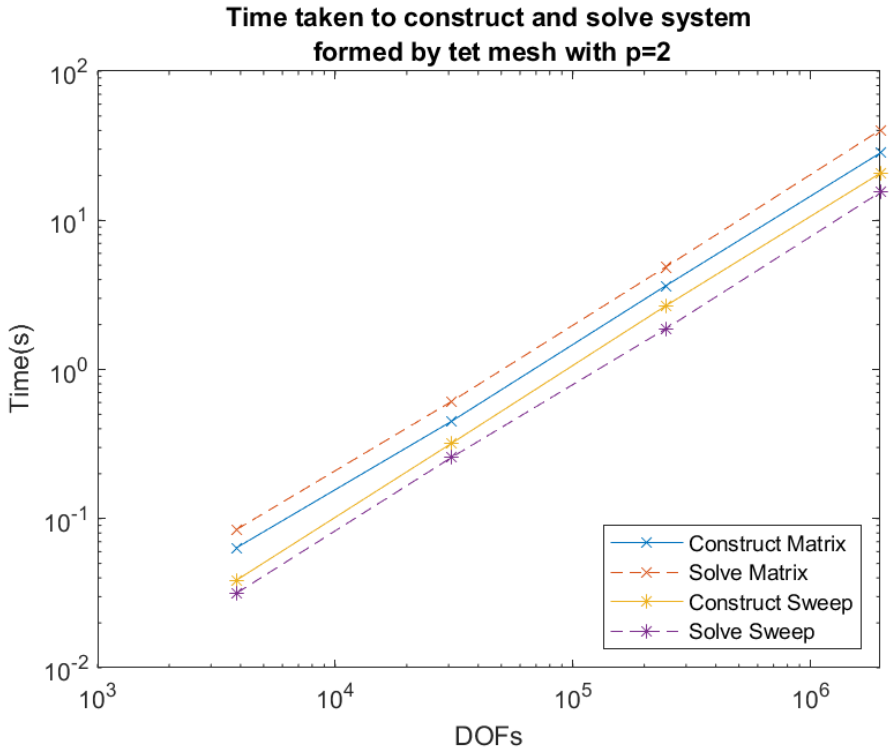

Figure 5.17: The breakdown of the construction and solve time for transport equation on tet mesh $\boldsymbol{p} = 2$.

5.1.1.8 Agglomerated Cubes

The sweep solver performs very favourably on the matrix formed by the agglomerated cube mesh. As with the agglomerated square example, this is in large part due to the quadrature free evaluation. Figure 5.19, however, shows that the sweep solver does provide some savings over the matrix solver.

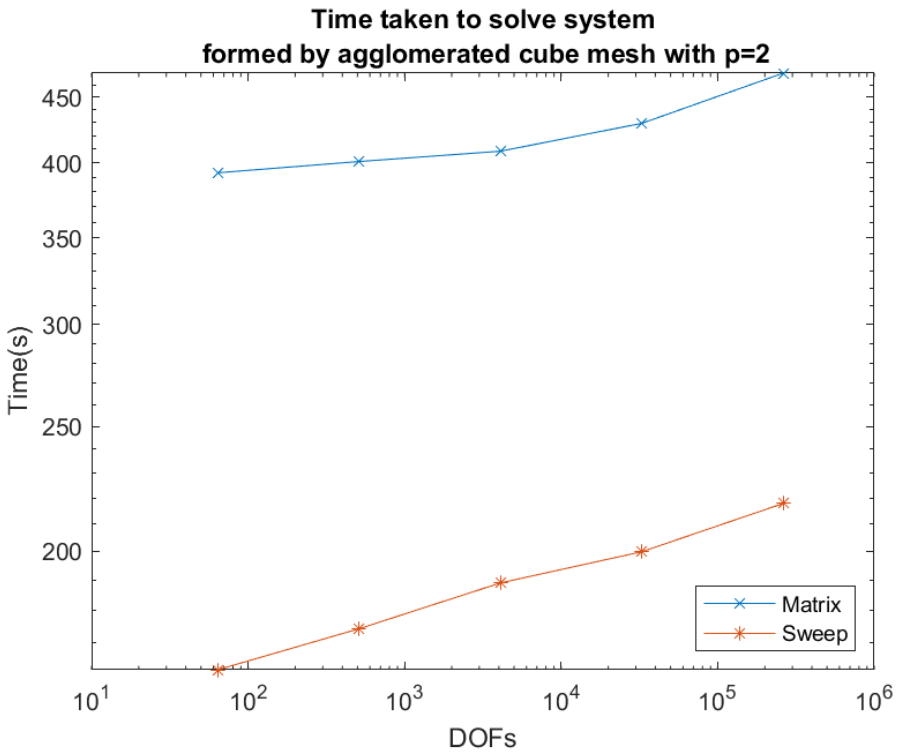

Figure 5.18: Time taken to solve system on an agglomerated cube mesh with $\boldsymbol{p} = 2$.

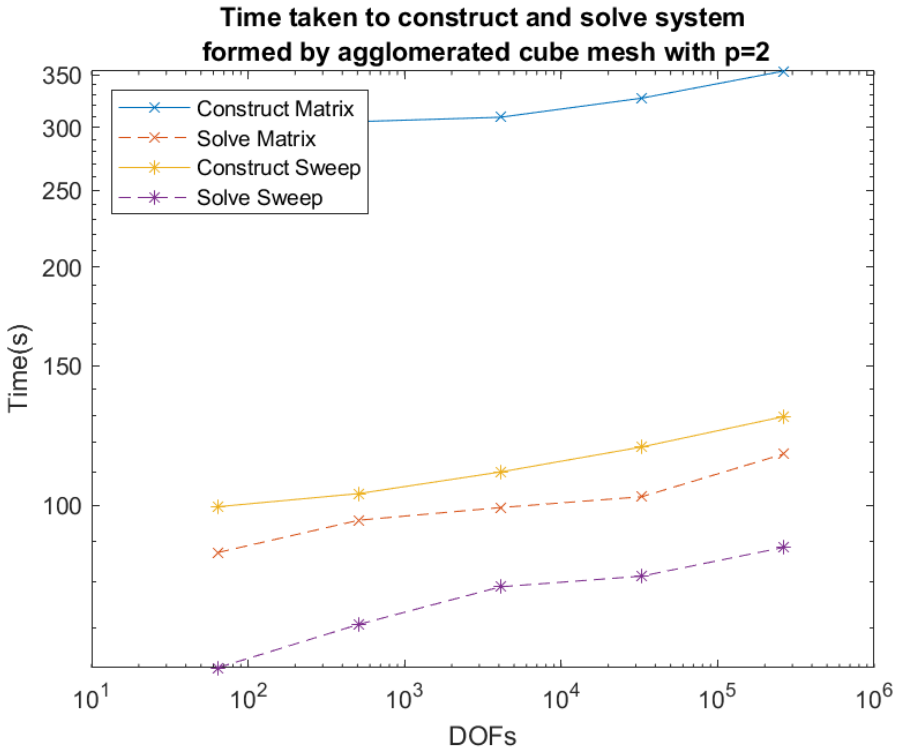

Figure 5.19: The breakdown of the construction and solve time for transport equation on agglomerated cube mesh $\boldsymbol{p} = 2$.

The sweep solver is shown to be a viable alternative to an optimised sparse matrix solver for convex elements. However, for non convex elements a problem may arise. If an

element is non convex it is possible for one element to be both the inflow and outflow of an element, directly or indirectly. This is called a cyclic dependence.

5.2 Cyclic Dependence

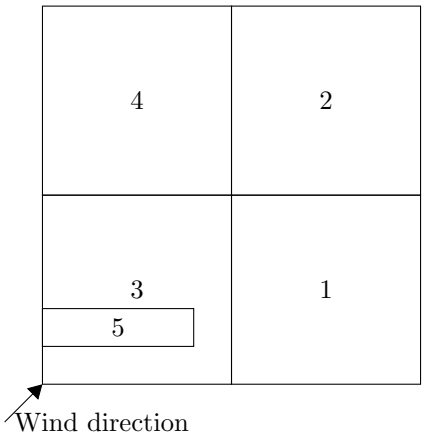

Figure 5.20: Mesh with a non-convex element with a north-east wind direction

Adding a non convex element to our example mesh, such as in Figure 5.20 gives an ordered matrix:

$$\begin{bmatrix} [E2] & [F24] & [F21] & & \\ & [E4] & & & [F43] \\ & & [E1] & & [F13] \\ & & & [E5] & [F53] \\ & & & [F35] & [E3] \end{bmatrix}.$$

This is obviously not in our desired upper triangular form, nor can any pivoting make it so. This effectively breaks our sweep solver, as we cannot evaluate a single element block and then invert it; however there is a larger block that could be inverted to solve the matrix equation. If we were to treat elements 3 and 5 as their own block and invert it to find the solutions to those elements, we could still invert the others using our sweep solver. So our matrix equation would become:

$$[E2][X2] = [B2] - [F42][X4] - [F12][X1]$$
$$[E4][X4] = [B4] - [F34][X3]$$
$$[E1][X1] = [B1] - [F31][X3]$$
$$\begin{bmatrix} [E5] & [F35] \\ [F53] & [E3] \end{bmatrix} \begin{bmatrix} [X5] \\ [X3] \end{bmatrix} = \begin{bmatrix} [B5] \\ [B3] \end{bmatrix}.$$

We then need to only invert one larger block, and then treat any elements that are not part of cyclic dependence as before. To do this, an algorithm to detect cyclic

dependencies is required. Tarjan's strongly connected components algorithm [64] is a relativity simple graph algorithm which will do this. We create a directed graph where each element is a node and each outflow face is an edge connecting the two elements vertex. Having created this directed graph to represent our mesh, this allows us to use Tarjan's algorithm to find the strongly connected components (SCC) in the problem. Any elements that form a cyclic dependence will be grouped into strongly connected components, and any elements that are not cyclically dependent will be considered as their own strongly connected component.

```
function TARJAN(node)
    node.visited ← true
    node.index ← indexCounter
    s.push(node)
    for all successor in node.successors do
        if node.visited then TARJAN(successor)
        end if
        node.lowlink ← MIN(node.lowlink, successor.lowlink)
    end for
    if node.lowlink == node.index then
        repeat
            successor ← stack.pop()
        until successor == node
    end if
end function
```

Tarjan's algorithm's run time is of the order $O(|V| + |E|)$, where $|V|$ is the number of vertices and $|E|$ is the number of edges of the directed graph [64]. Tarjan's algorithm also produces a reverse topological sort; therefore, if we perform Tarjan's algorithm on the mesh before we start we will produce the sweep ordering as well.

5.2.1 Comparison between matrix and the sweep solver for non-convex elements

5.2.1.1 Agglomerated polygons

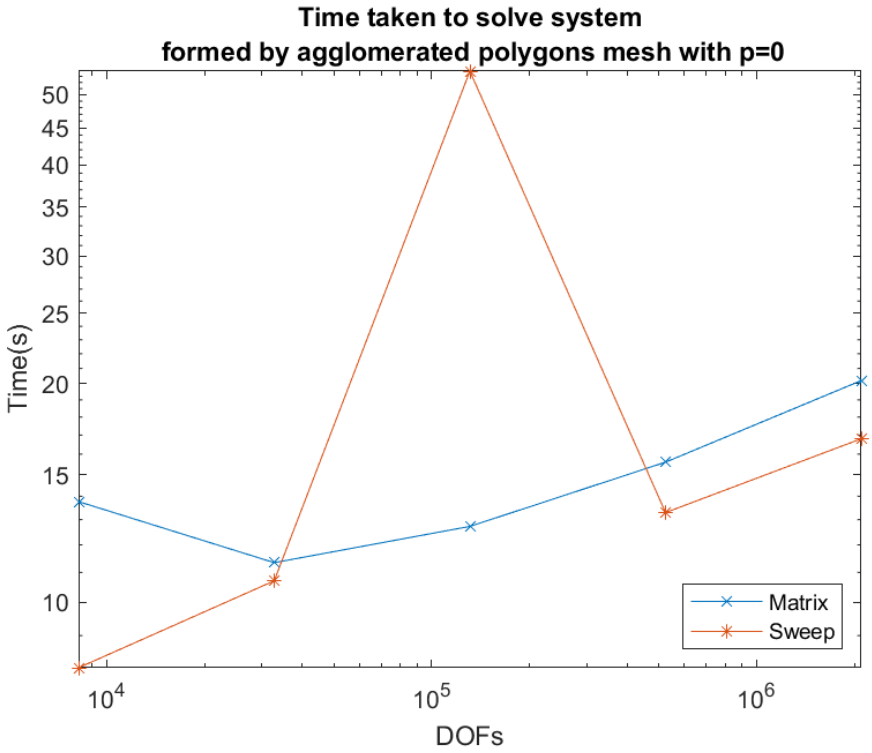

Figure 5.21: Time taken to solve system on an agglomerated polygon mesh with $\boldsymbol{p} = 0$.

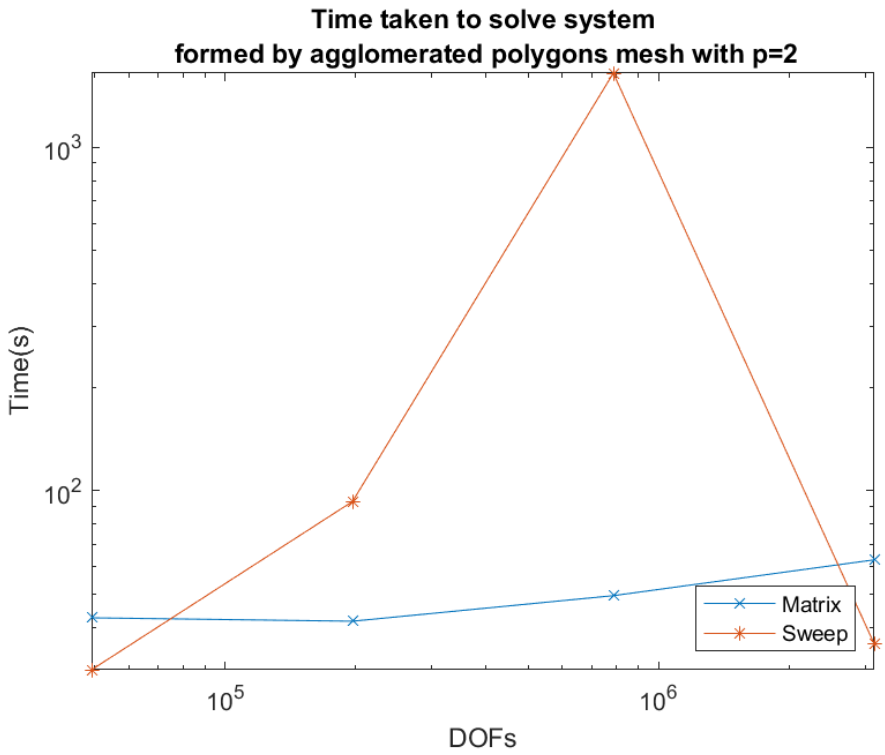

Figure 5.22: Time taken to solve system on an agglomerated polygonal mesh with $\boldsymbol{p} = 2$.

Figures 5.21 and 5.22 shows some very unexpected results, with the sweep solver taking considerably longer for some of the meshes compared to the matrix solver. Additionally, the matrix solver takes a longer time than one might expect to solve the first mesh.

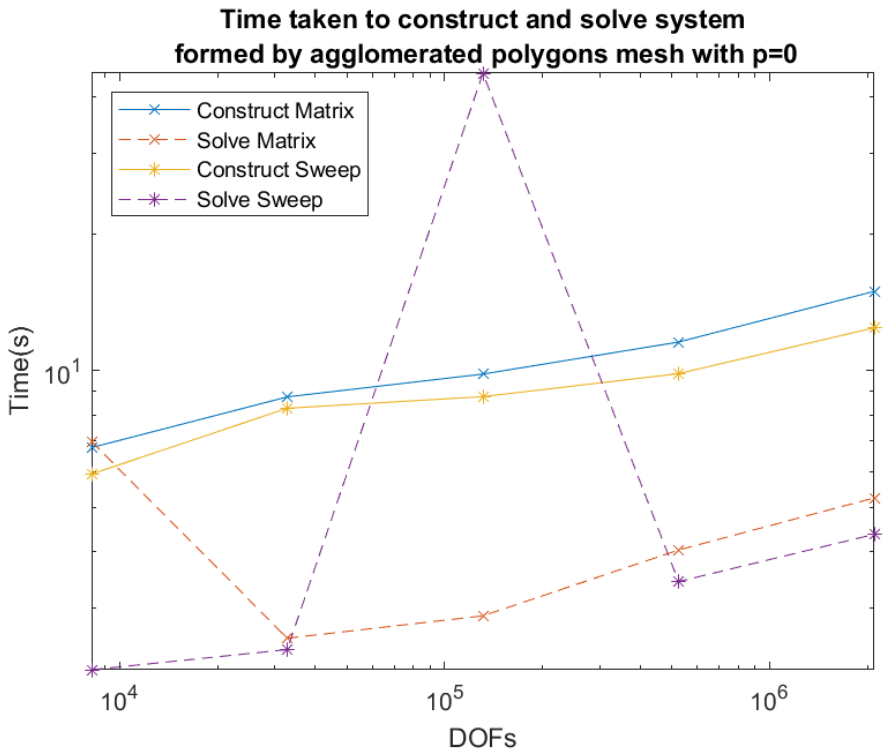

Figure 5.23: The breakdown of the construction and solve time for transport equation on agglomerated polygon mesh $\boldsymbol{p} = 0$.

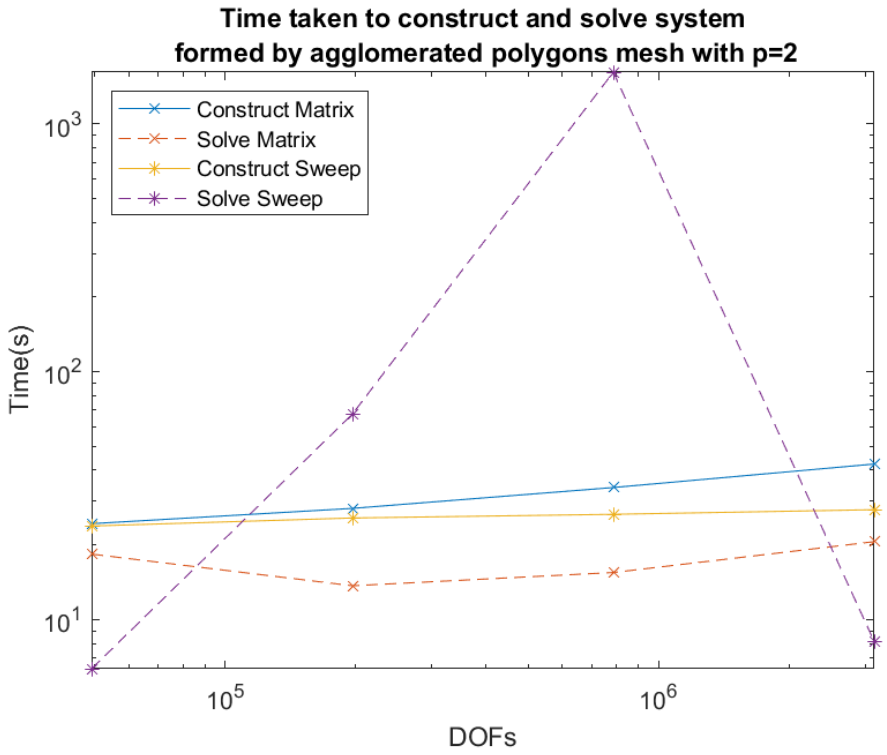

Figure 5.24: The breakdown of the construction and solve time for transport equation on agglomerated polygon mesh $\boldsymbol{p} = 2$.

Figures 5.23 and 5.24 provide some insight as to why these results are occurring. They

show that solving the matrix is the biggest change. The quadrature free construction is, as predicted, faster than the matrix solver. Additionally, the first solve for the matrix solver is shown to have an increased solve time, indicating that as expected a denser than usual matrix is present. Investigating the cause of these very slow solve times, we must look at the strongly connected components that Tarjan's algorithm is giving us, and thus the cyclic dependencies in the mesh.

Fine elements	Coarse elements	SCCs	Coarse elements in the largest SCC
2,097,152	8,192	6,640	13
2,097,152	32,768	28,680	30
2,097,152	131,072	69,540	86
2,097,152	524,279	523,895	2
2,097,152	2,097,152	2,097,152	1

Table 5.1: The number of strongly connected components and the number of coarse elements in the largest SCC.

From Table 5.1 we see that the cause of this slow down is due to the presence of large strongly connected components, with many elements that are in cyclic dependency, the largest dependency being formed of 86 elements. This results in our sweep solver trying to invert large sections of the matrix at once, causing a significant slow down.

To further investigate, we changed tack. Rather than changing the number of coarse elements with a fixed number of fine elements underneath, we fix the number of coarse elements and increased the number of fine elements, as we expected that the matrix solver's time will be dependent on the number of fine elements.

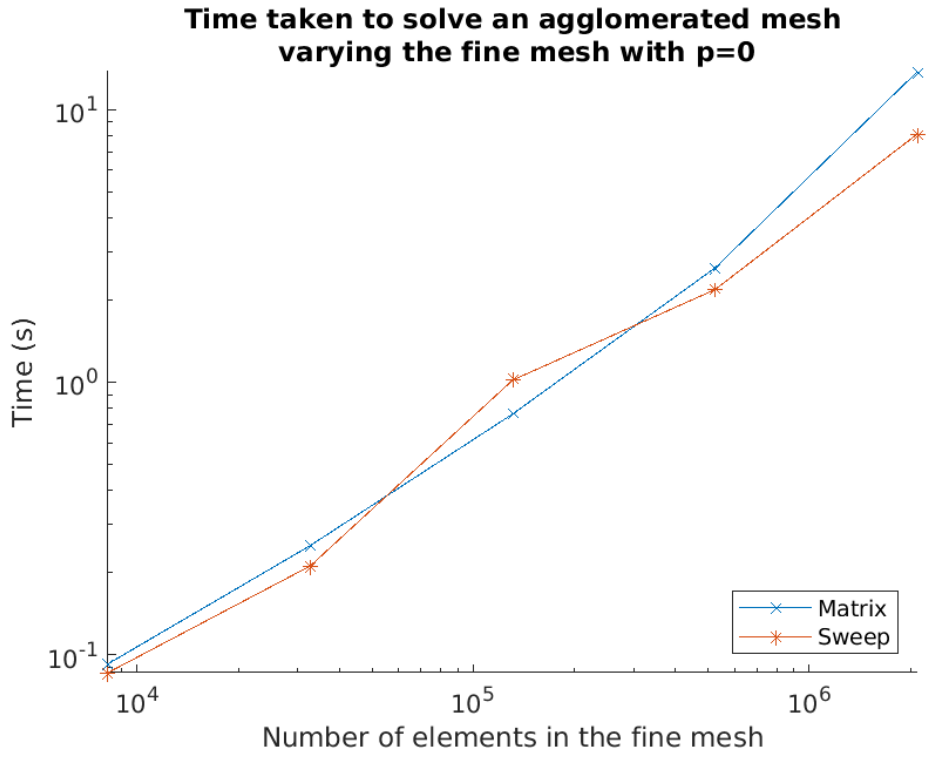

Figure 5.25: Time taken to solve on an agglomerated polygonal mesh, varying the fine mesh with $\boldsymbol{p} = 0$.

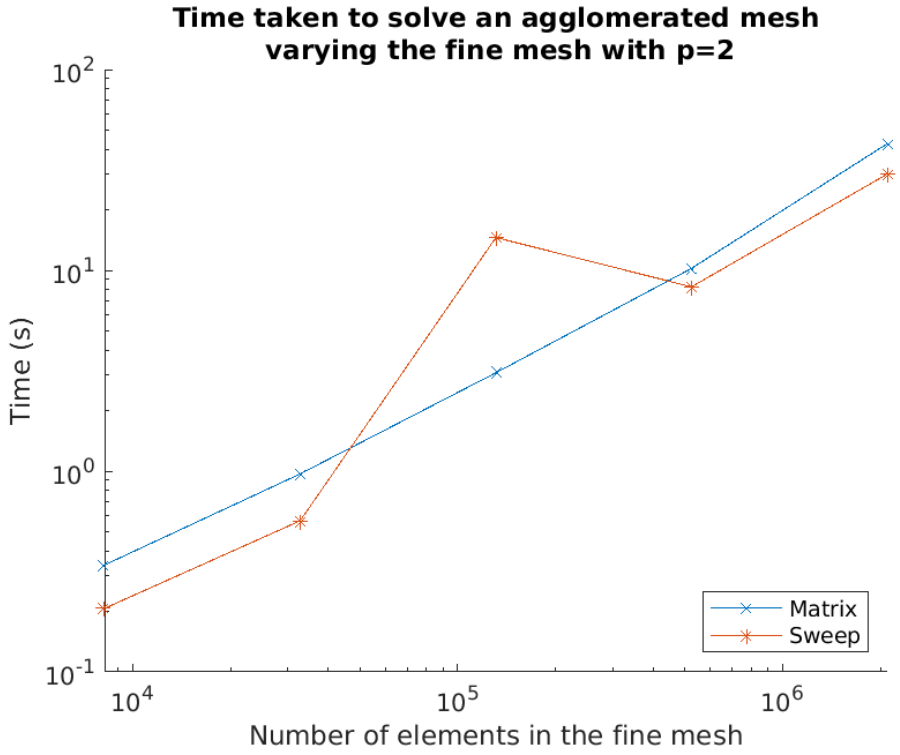

Figure 5.26: Time taken to solve on an agglomerated polygonal mesh, varying the fine mesh with $\boldsymbol{p} = 2$.

Figures 5.25 and 5.26 show the dependence of the matrix solver on the number of fine

elements as expected. We were not expecting the sweep solver to be dependent on the number of elements in our fine mesh, but it does appear to be. Again, the unexpected spike can be explained by the cyclic dependencies, shown in Table 5.2.

Fine elements	Coarse elements	SCCs	Coarse elements in the largest SCC
8,192	8,192	8,192	1
32,768	8,192	8,183	2
131,072	8,192	4,506	34
524,288	8,192	7,249	11
2,097,152	8,192	6,640	13

Table 5.2: The number of strongly connected components and the number of elements in the largest SCC.

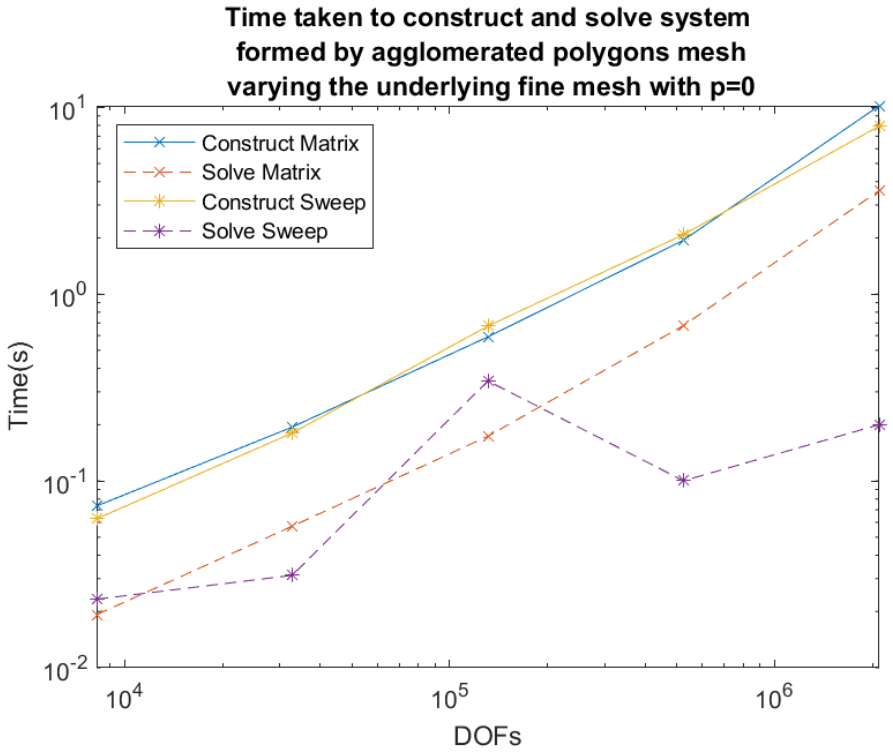

Figure 5.27: The breakdown of the construction and solve time for transport equation on agglomerated polygon mesh, varying the underlying fine mesh $\boldsymbol{p} = 0$.

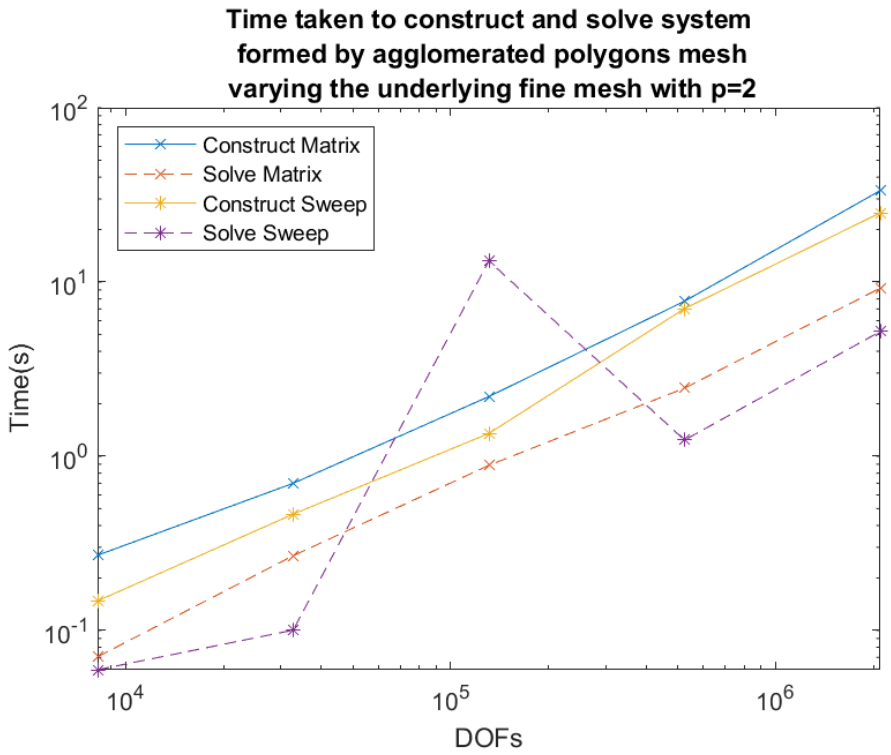

Figure 5.28: The breakdown of the construction and solve time for transport equation on agglomerated polygon mesh, varying the underlying fine mesh $\boldsymbol{p} = 2$.

Figures 5.27 and 5.28 show that the construction appears to be dependent on the number of elements in the fine mesh. This is very unexpected, as the quadrature free evaluation should have decoupled them.

5.2.1.2 Agglomerated squares

To avoid any confusion caused by cyclic dependencies, we will use our agglomerated squares meshes to investigate this further.

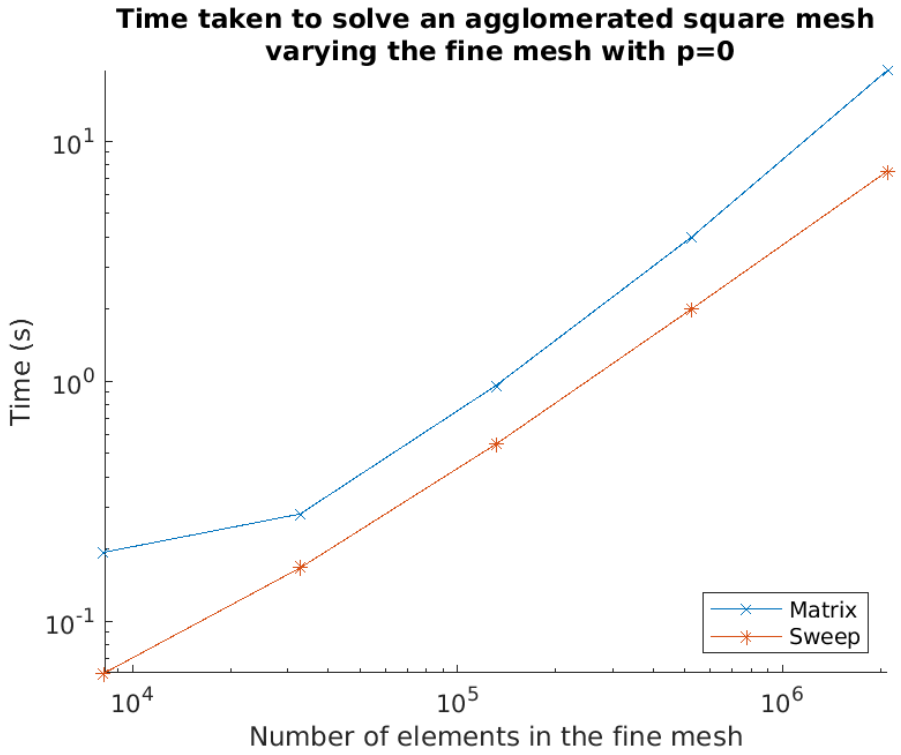

Figure 5.29: Time taken to solve system on an agglomerated square mesh, varying the fine mesh with $\boldsymbol{p} = 0$.

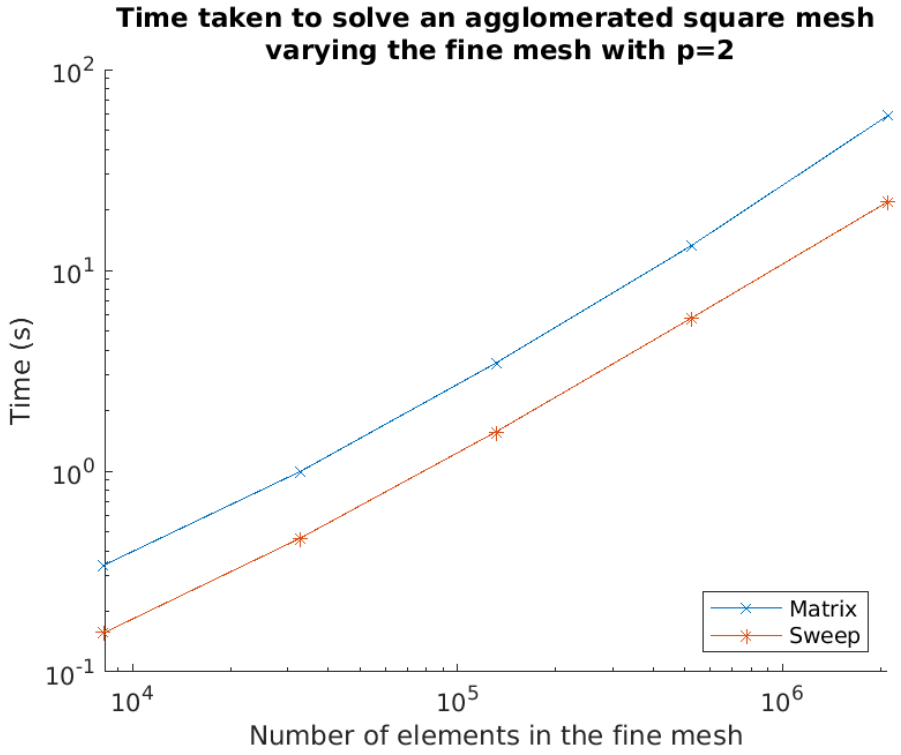

Figure 5.30: Time taken to solve system on an agglomerated square mesh, varying the fine mesh with $\boldsymbol{p} = 2$.

Again, both the sweep and matrix solver are dependent on the number of elements

in the fine mesh, as shown in Figures 5.29 and 5.30. Figures 5.31 and 5.32 show that the solve step is not as dependent on the number of elements in the fine mesh compared to the construction.

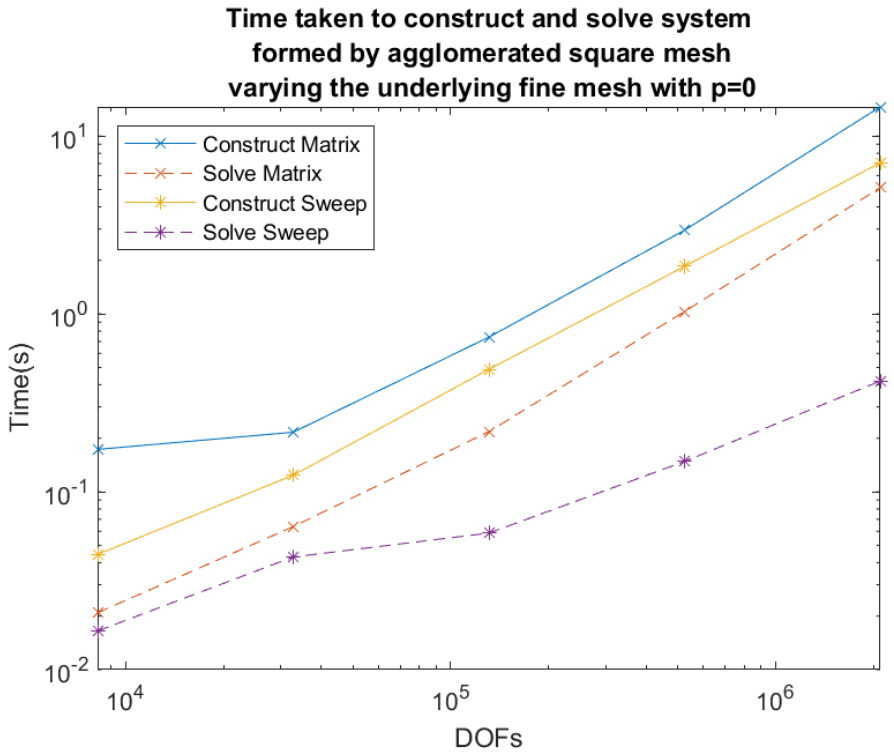

Figure 5.31: The breakdown of the construction and solve time for transport equation on agglomerated square mesh varying the underlying fine mesh $\boldsymbol{p} = 0$.

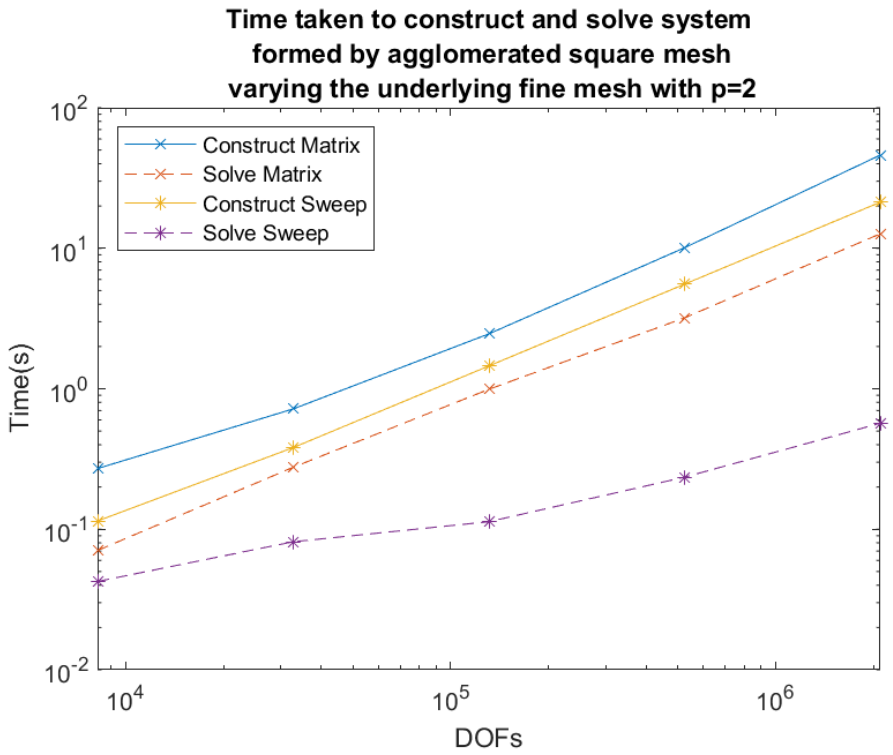

Figure 5.32: The breakdown of the construction and solve time for transport equation on agglomerated square mesh varying the underlying fine mesh $\boldsymbol{p} = 2$.

The Gaussian quadrature that is used to evaluate our forcing and boundary functions, to form the right hand side of the matrix equation, is dependent on the number of fine elements. This accounts for the dependency on the number of elements in the fine mesh during the construction step.

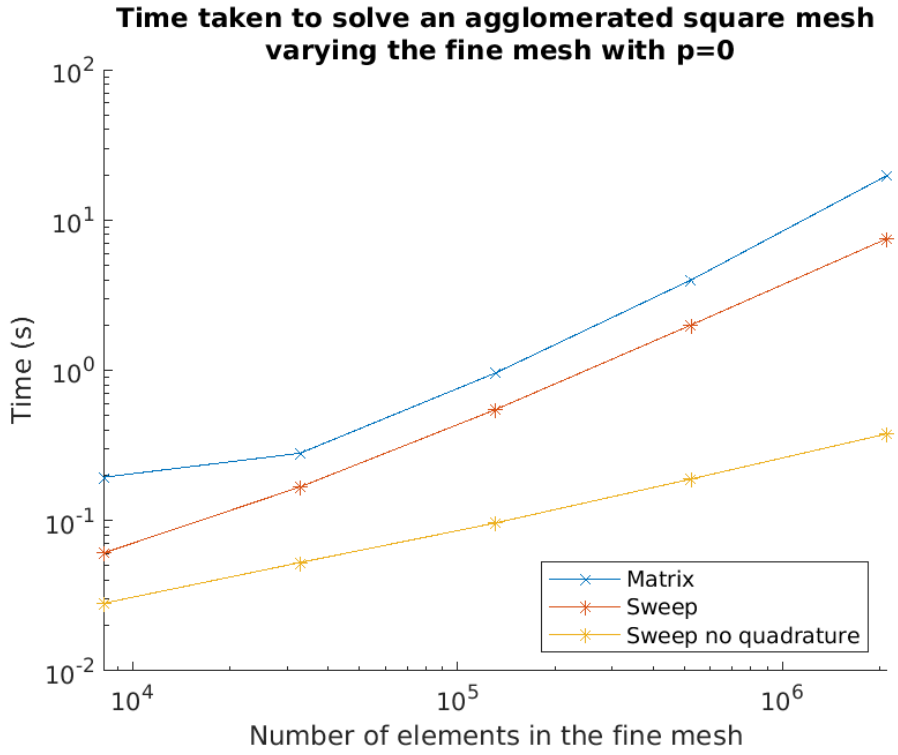

Figure 5.33: Time taken to solve on an agglomerated square mesh, varying the fine mesh with $\boldsymbol{p} = 0$.

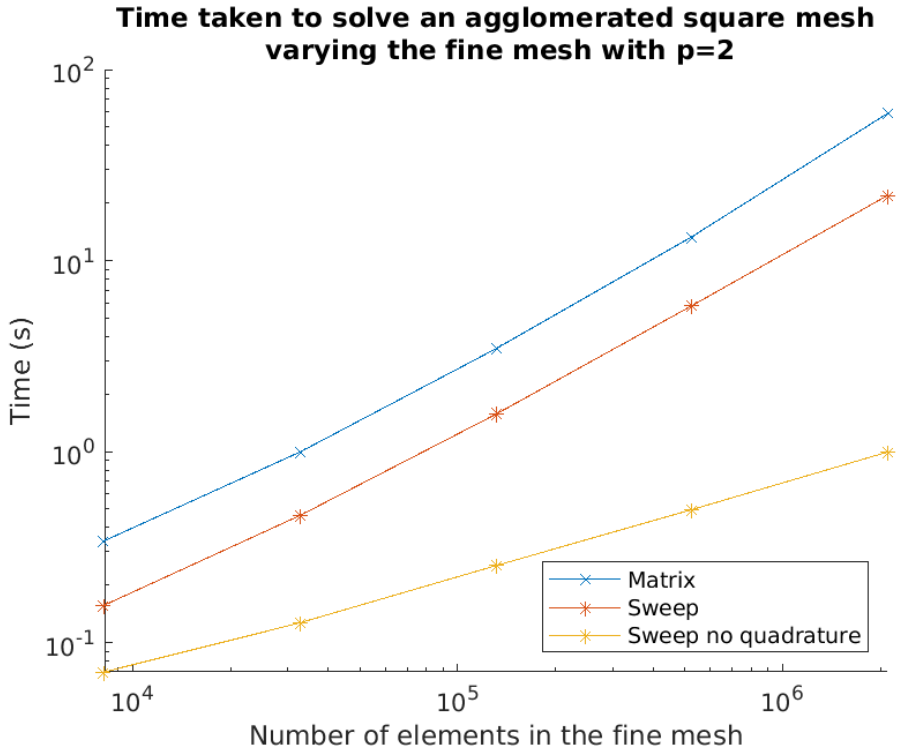

Figure 5.34: Time taken to solve on an agglomerated square mesh, varying the fine mesh with $\boldsymbol{p} = 2$.

To create Figures 5.35 and 5.36 we set our forcing and boundary functions to be

constant in our domain, and, thus, turned off the quadrature construction and evaluation for the right hand side. To that end, we can show that our sweep solver construction step time is actually dominated, at least at the number of DOFs we experimented with, by the quadrature required to evaluate the right hand side of the matrix equation.

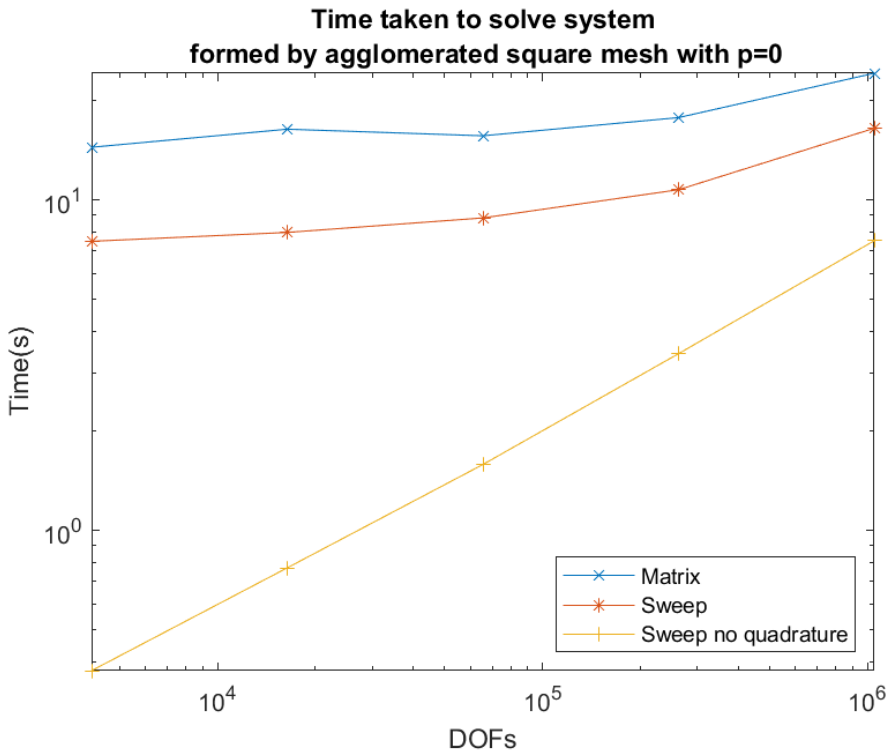

Figure 5.35: Time taken to solve on an agglomerated square mesh with no quadrature with $\boldsymbol{p} = 0$.

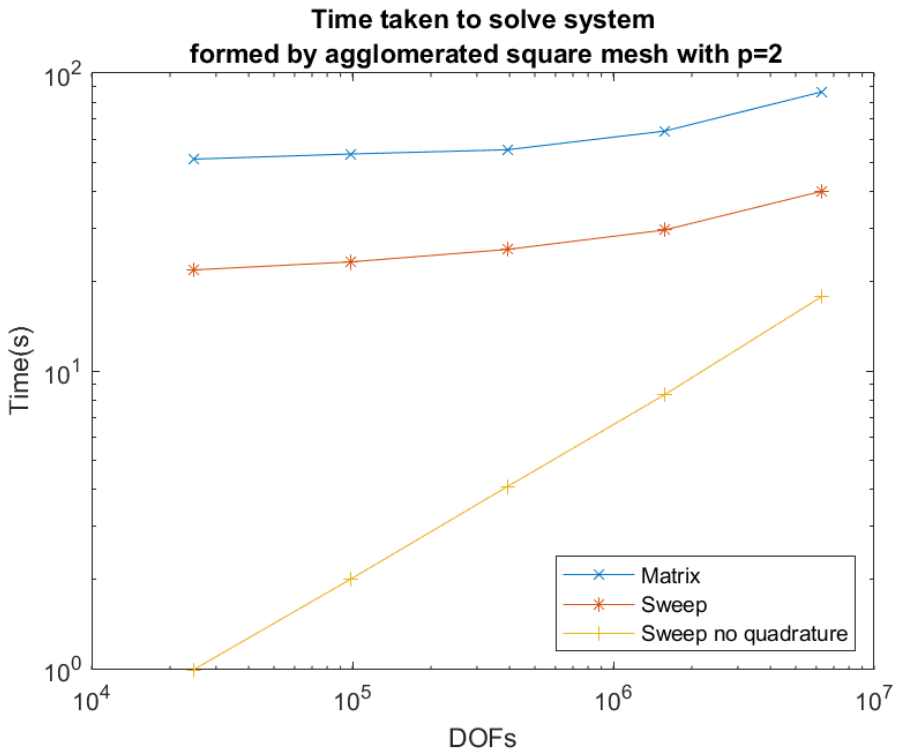

Figure 5.36: Time taken to solve on an agglomerated square mesh with no quadrature with $\boldsymbol{p} = 2$.

Figures 5.35 and 5.36 show that the quadrature dominates the vast majority of the sweep solve time. We presume that for a large enough mesh the construction of the matrix and solving the matrix system will dominate the time taken. This is obviously dependent on the elements that form the mesh. We revisited the agglomerated polygons meshes to see what effect the removal of quadrature by using a constant forcing and boundary functions has on the results.

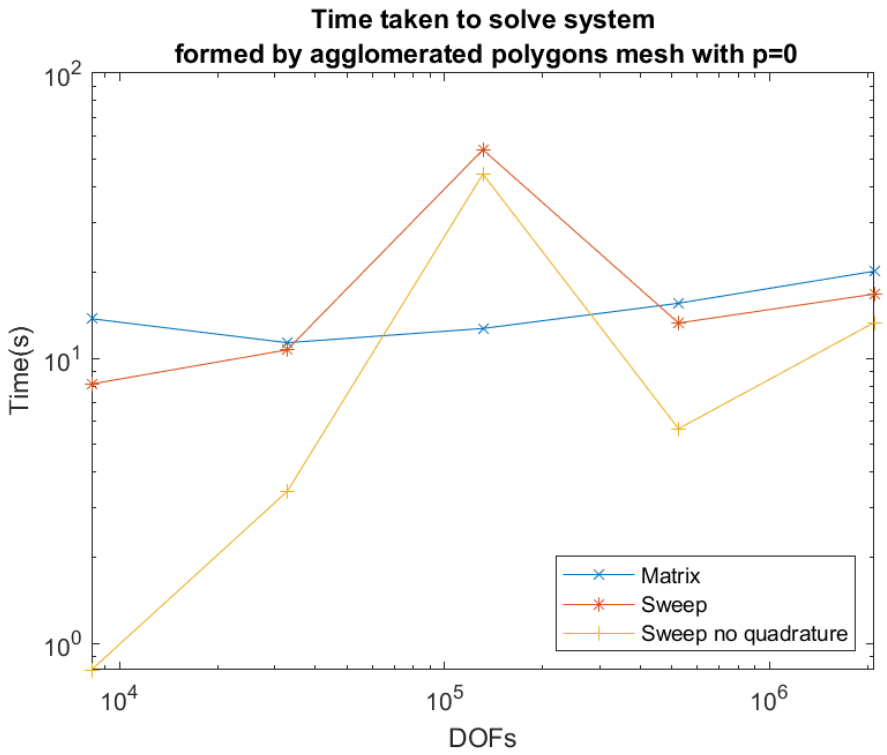

Figure 5.37: Time taken to solve on an agglomerated square mesh with no quadrature with $\boldsymbol{p} = 0$.

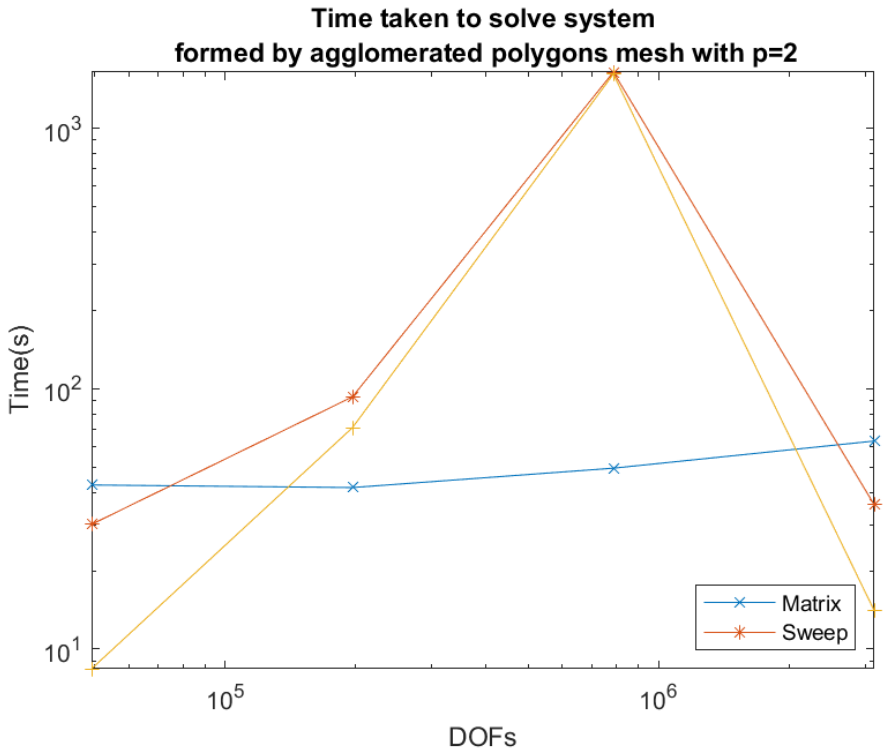

Figure 5.38: Time taken to solve on an agglomerated square mesh with no quadrature with $\boldsymbol{p} = 2$.

We can see that for meshes with less cyclic dependencies that quadrature was

Fine elements	Coarse elements	SCCs	Coarse elements in the largest SCC
384	384	384	1
3,072	384	9	348
24,576	384	3	382
196,608	384	2	383
1,572,864	384	1	384
1,572,864	384	1	384
1,572,864	3,072	2	3,071
1,572,864	24,576	5	24,570
1,572,864	196,608	2	196,572
1,572,864	1,572,864	1,572,864	1

Table 5.3: The number of strongly connected components (SCC) and the number of elements in the largest SCC.

dominating the time taken to solve the system, but for meshes with large SCC, the construction time, including the time to evaluate the quadrature, is dominated by the solve time.

5.2.1.3 Agglomerated polyhedra

Agglomerated polyhedra present many difficulties for the sweep solver. The meshes produced by METIS are so jagged, that they create cyclic dependence that span the majority of the mesh. METIS is not minimising the cyclic dependencies, rather it minimises the number of paths (faces) between vertices (elements).

Table 5.3 shows the number of elements in the largest SCC, which in the worst case is the entire mesh, meaning every element is cyclically dependent on every other element. This has a dramatic effect on the solve time of the sweep solver. Figures 5.39 and 5.40 show how the smallest solves are affected, with the sweep solver, at worst, taking around 19,000% longer than the matrix solver.

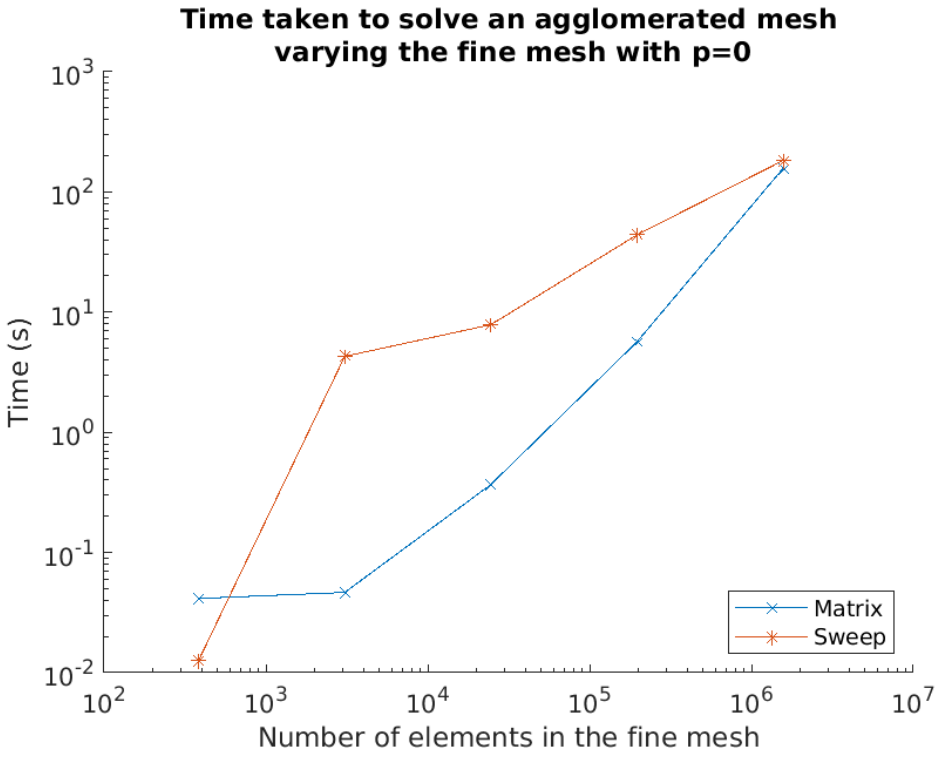

Figure 5.39: Time taken to solve on a polyhedral mesh, varying the number of fine mesh with $\boldsymbol{p} = 0$.

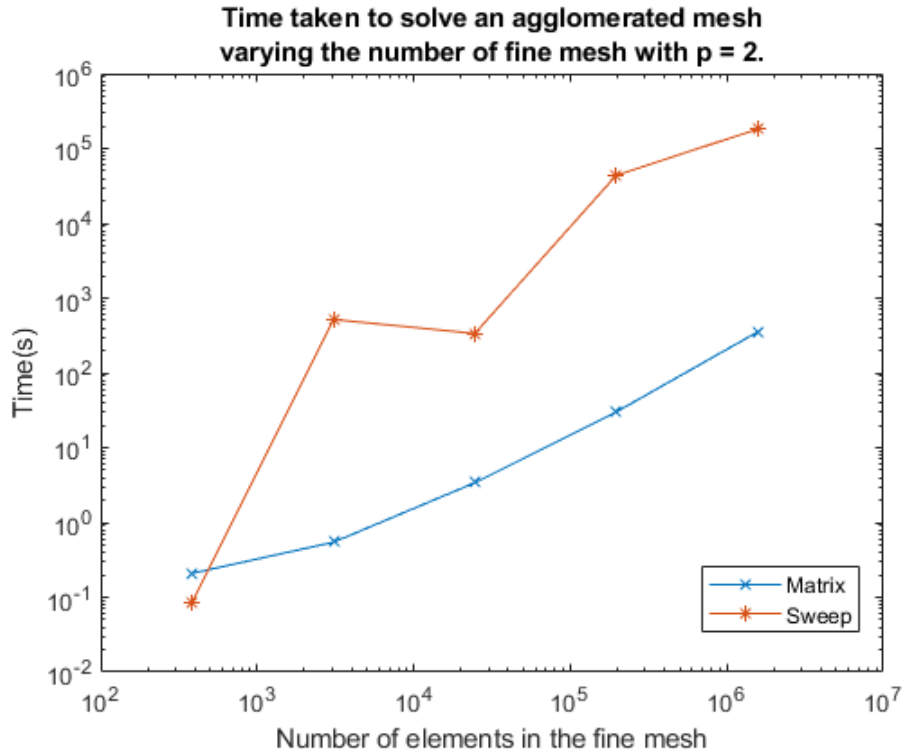

Figure 5.40: Time taken to solve on a polyhedral mesh, varying the number of fine mesh with $\boldsymbol{p} = 2$.

Again, we see that the sweep solver is completely dominated by the cyclic dependencies.

Unfortunately, there is little that can be done to mitigate this aside from using a different algorithm that results in convex coarse elements.

5.2.1.4 Agglomerated cubes

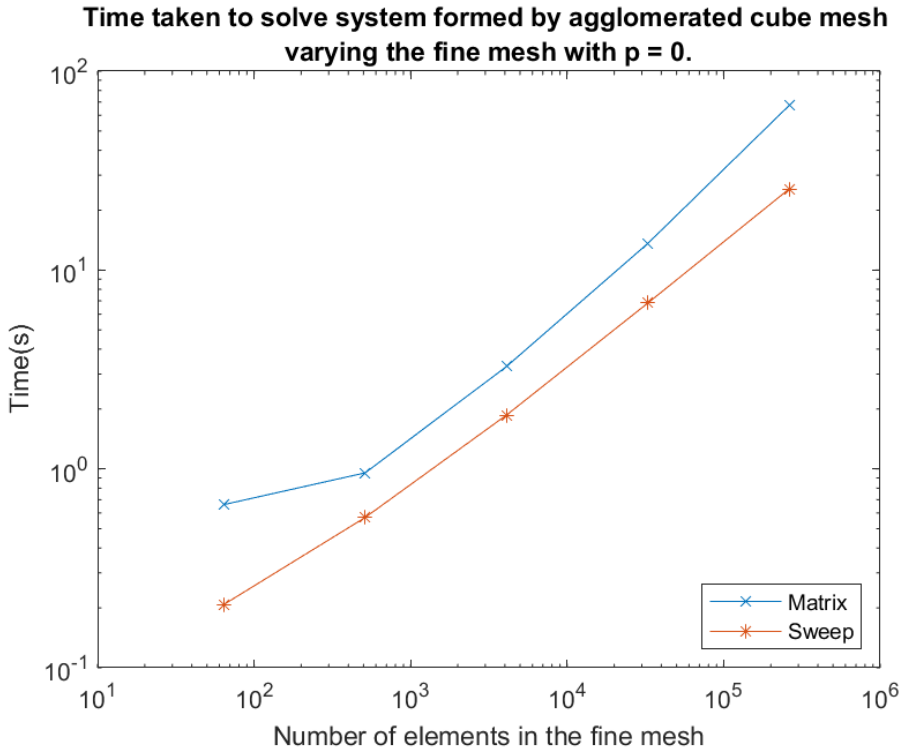

Figure 5.41: Time taken to solve on an agglomerated cube mesh, varying the number of fine mesh with $\boldsymbol{p} = 0$

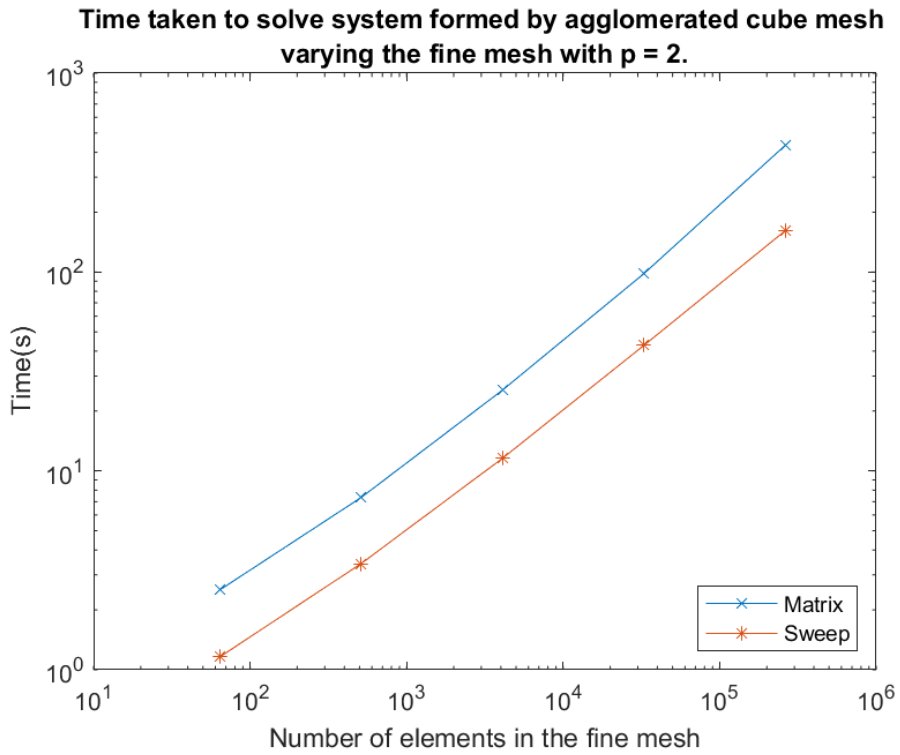

Figure 5.42: Time taken to solve on an agglomerated cube mesh, varying the number of fine mesh with $\boldsymbol{p} = 2$

Figures 5.41 and 5.42 show that for agglomerated cube mesh, the sweep solver time to solve the system is dependent on the number of elements in the fine mesh as well.

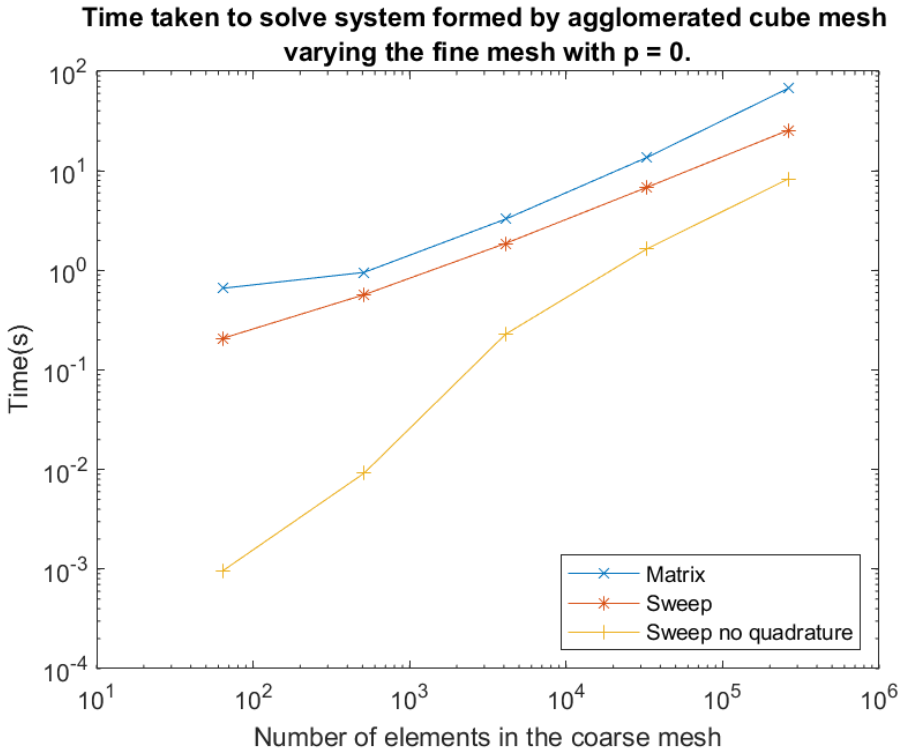

Figure 5.43: Time taken to solve on an agglomerated cube mesh, varying the number of fine mesh with o quadrature with $\boldsymbol{p} = 0$

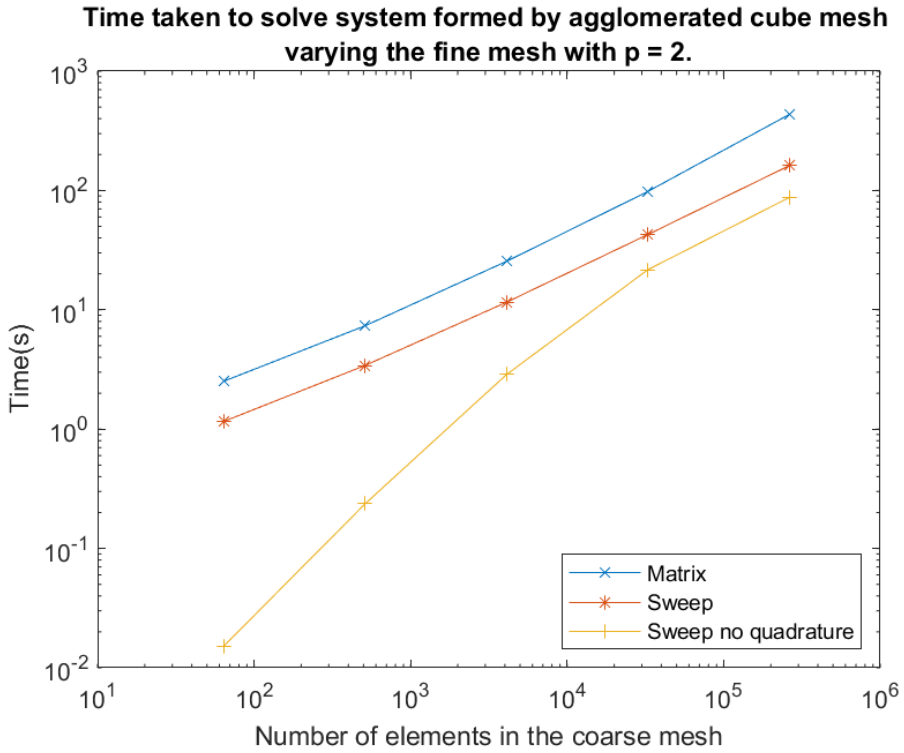

Figure 5.44: Time taken to solve on an agglomerated cube mesh, varying the number of fine mesh with no quadrature with $\boldsymbol{p} = 2$

Examining figures 5.43 and 5.44 we see that evaluating the right hand side with quadrature dominates the time taken to solve the system for a lower number of DOFs. For a larger number of DOFs, however, the time taken to solve the system actually starts to dominate the total time.

5.3 Summary

In conclusion, we have shown that the sweep solver is generally more efficient than the optimised matrix solver, MUMPS; however, the choice of elements in spatial domain can make a large difference to the efficiency of the solver. In particular, the sweep solver is a lot more sensitive to cyclic dependences than MUMPS is. This is especially true for 3D polytopes formed by METIS or similar programs that can result in highly irregular elements, causing large cyclic dependencies; this can make the sweep solver unusable. As the LBTE requires us to solve transport equations in every direction, it is best to chose convex elements to prevent any cyclic dependencies.

For clinical applications, our meshes would be based off of voxel data from MRI or CAT scans, so cube meshes would be very easy to define from that data, as we just use the voxel structure. This would, however, give us around seventeen million spatial elements, so the number of DOFs for LBTE would be extremely high. Therefore, some form of agglomeration would reduce the number of DOFs in the system and with the

quadrature free evaluation this would represent a saving in the matrix construction time as well.

As we have now shown that we have an efficient transport solver, let us link this back to the LBTE, and our solver for it. To do this, we will compare DOG discretisation with our sweep solver to the DG-DG discretisation with a MUMPS matrix solver.

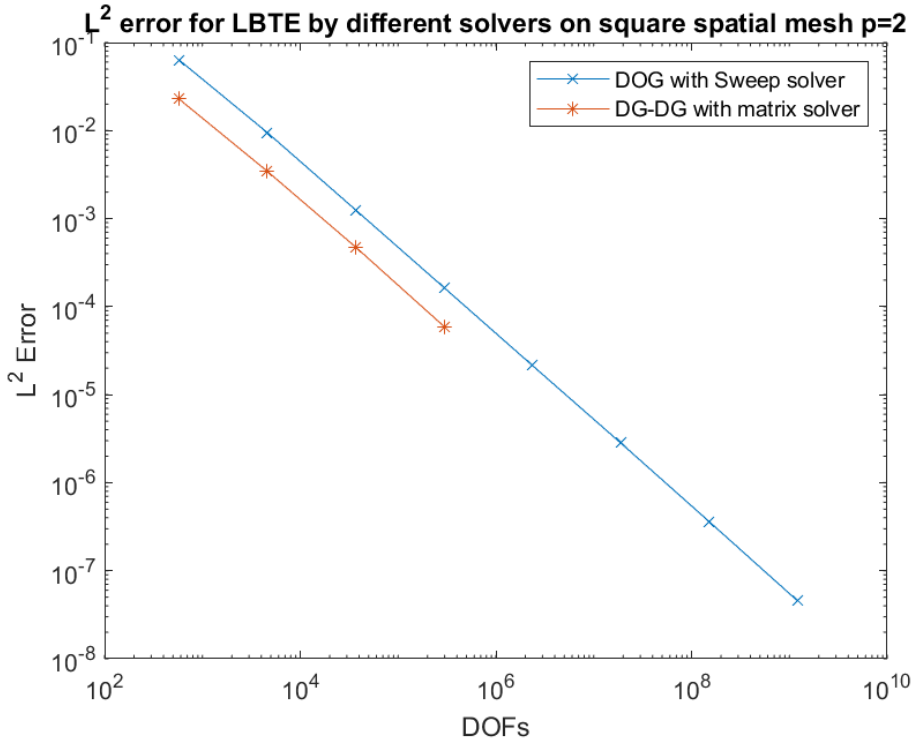

Figure 5.45: Error in the L^2 norm for different solvers on square spatial mesh $\boldsymbol{p} = 2$

Figure 5.45 shows that both the DG-DG and DOG solvers converge at the same rate in the L^2 norm. There is a constant difference, likely due to different quadrature being used. The DOG discretisation commits a variational crime by under resolving the quadrature, as we need the number of quadrature points to be exactly match our basis functions, see Section 3.8 for more details. The error incurred by this variational crime, is obviously not the dominating error as we still get the expected rate of convergence in the L^2 norm.

The reason the DG-DG solver stops prematurely, is the machine we used for these timings ran out of memory and the programme crashed while trying to solve the LBTE with 2,359,296 DOFs. This is also why the computations in this section have been restricted to 2D, the DG-DG solver simply can not handle number of DOFs required.

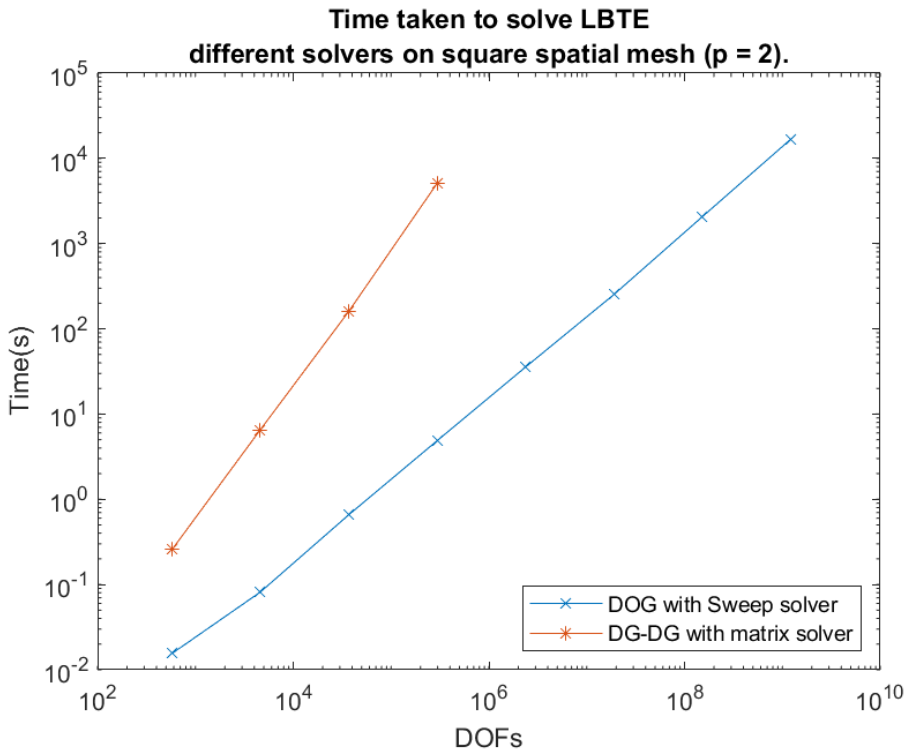

Figure 5.46: Time taken to solve LBTE with different solvers on square spatial mesh $\boldsymbol{p} = 2$

Comparing the run times, in Figure 5.46, shows how far we have come. The DOG discretisation with the sweep solver is significantly more efficient than the DG-DG discretisation with the matrix solver. The DOG solver is able to solve the LBTE with 150,994,944 DOFs in less than half the time it takes the DG-DG solver to solve the LBTE with 294,912 DOFs.

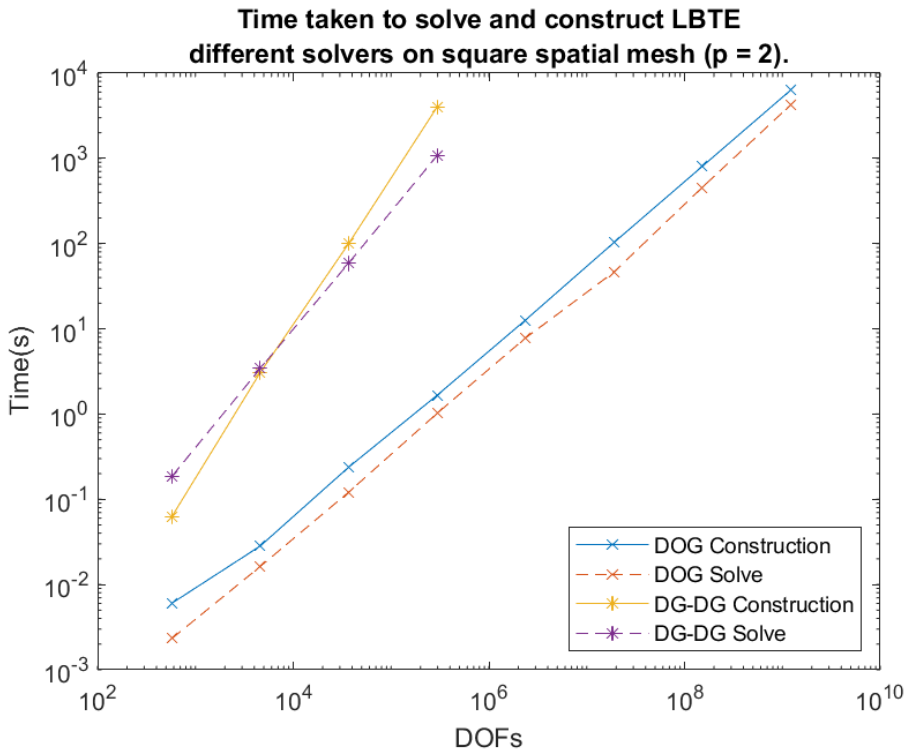

Figure 5.47: Time taken to solve and construct the LBTE with different solvers on square spatial mesh $\boldsymbol{p} = 2$

If we look at the breakdown of the time taken to solve and construct the matrixes for the two different solvers, in Figure 5.47, the construction and solve steps grow at a similar rate as the number of DOFs for DOG. For the DG-DG, however, we can see the construction step grows at a faster rate than the solver step. As noted in Chapter 3, the scattering steps are roughly the same, so have been excluded from this graph for clarity.

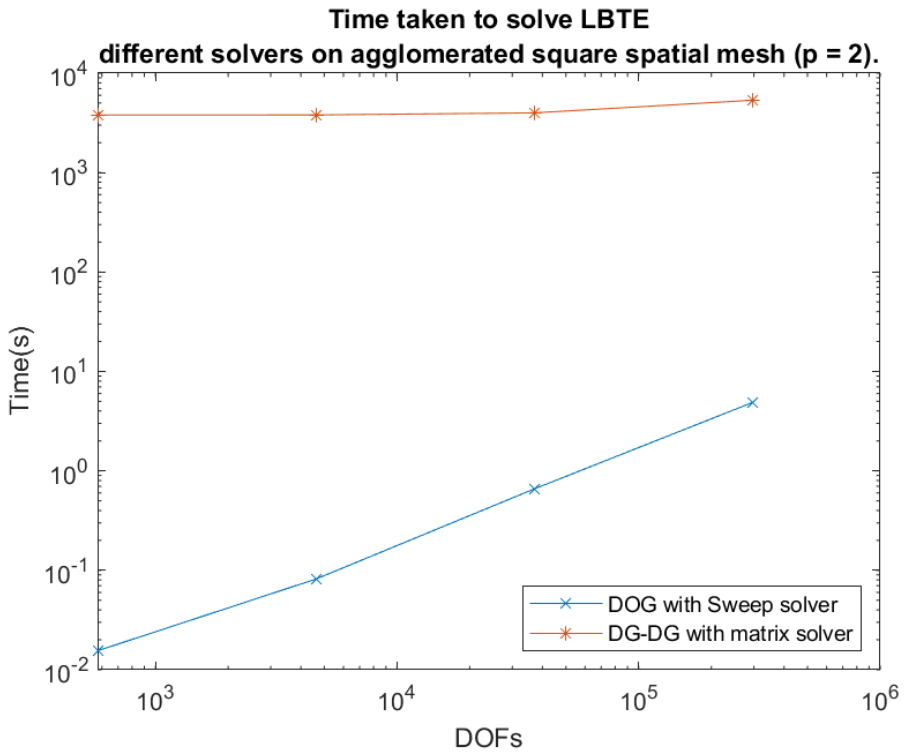

Figure 5.48: Time taken to solve LBTE with different solvers on agglomerated square spatial mesh $\boldsymbol{p} = 2$

Comparing the time taken for agglomerated squares for both solvers, with 294,912 fine square elements underneath. We see in Figure 5.48 that the time taken to solve the LBTE with the DG-DG solver is not entirely dependent on the number of DOFs in the matrix, while the DOG solver is. Figure 5.49 shows that the DG-DG solver time is dominated by the construction of the matrix.

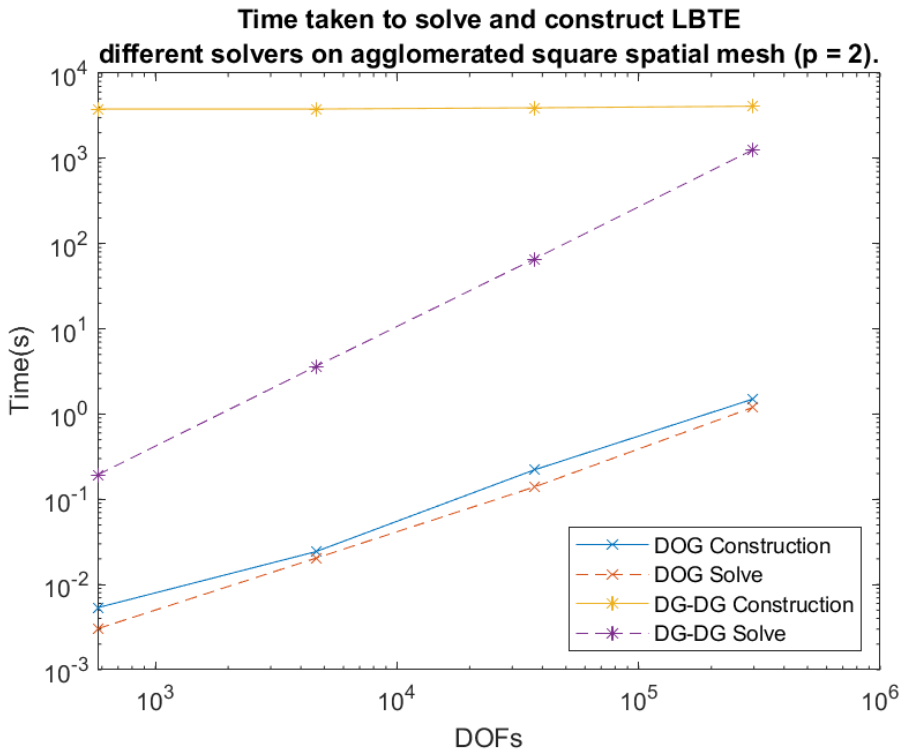

Figure 5.49: Time taken to solve and construct the LBTE with different solvers on agglomerated square spatial mesh $\boldsymbol{p} = 2$

Chapter 6

Adaptive Algorithms for the linear Boltzmann transport equation

Having developed an efficient solver for the transport equation, we now turn our sights back to the LBTE, and focus on accurately solving it. To this end we introduce the concept of goal oriented adaptivity, with the aim of reducing the error in our approximation of the LBTE, while maintaining an efficient solver.

Another reason adaptive refinements are popular is they can be used on complex geometries, while not requiring the whole mesh to be fine enough to capture the geometries. This is also equally true for solutions with a high degree of variation, which will be lost with too coarse a mesh.

Given the high dimensionality of the LBTE, uniform refinement is unfeasible for any practical problems due to the amount of memory that would be required to store the mesh and solutions. We, therefore, shall suggest using adaptive refinement algorithms. We have shown in Section 3 that increasing the number of elements in the meshes yields a smaller error; however, let us suppose that not every element contributes the same amount of error. Splitting an element with a larger error contribution would therefore be more efficient than splitting an element with a smaller error contribution. To this end, adaptive refinement algorithms aim to maximise the accuracy of the solution while minimising the number of DOFs used to compute that solution.

6.1 Motivation

Simulating physical phenomena always introduces some inaccuracies known as numerical errors. One type is a discretisation error. Discretisation errors occur from trying to

represent a continuous process within a discrete space; thus these errors can never be eliminated, only reduced. To reduce the discretisation error of a numerical technique, one must enrich the discrete space, increasing its dimensionality.
In relation to DGFEM, our discrete space is given in the form:

$$\mathbb{V}_{\Omega}^{\mathbf{p}} = \{v \in L_2(\Omega) : v|_{\kappa_\Omega} \in \mathbb{P}_{p_{\kappa_\Omega}}(\kappa_\Omega) \text{ for all } \kappa_\Omega \in \mathcal{T}_\Omega\}.$$

So we have two components of the discrete representation, the mesh $\mathcal{T}_\Omega$, and basis functions on each element of the mesh $\mathbb{P}_{p_{\kappa_\Omega}}(\kappa_\Omega)$. Thus, the discretisation error can be reduced by increasing the number of elements in our mesh, known as h-refinement, or by increasing the number of basis function in each element, p-refinement. In the case of polynomial basis functions, as described in Sections 2.3.1 and 3.8, this is done by increasing the degree of the family of polynomials defined on each element. Both of these refinement types have their own merits and uses.

h-refinement is best used to capture underlying geometries and non-smooth solutions on the element. p-refinement, on the other hand, is best used on elements with a smooth solution. These two techniques, however, can be combined while retaining the advantages of both methods into what is termed the hp-refinement algorithms, where each element is h or p refined depending on the smoothness of solution on the element [36].

In Section 2.3 and Chapter 3, we showed the effects of uniform h-refinement for a few global refinements of p on the transport and LBTE problems. While we did successfully reduce the discretisation error with respect to the L^2 norm, it can be assumed that some of the refinements on each element did little to change the overall error. This means that the time and resources dedicated to these additional degrees of freedom produced little for no reward. Ideally we would only refine the elements that are significantly contributing to error. This could be done by hand, i.e. individually choosing for each element whether it should be refined and if so what sort of refinement would be best. This would be tedious and time consuming; therefore, a computerised algorithm is advised.

6.2 General algorithm for adaptivity

For an adaptive algorithm we need some way of calculating the local error contribution of each element in the mesh and then using these local errors to decide which element to refine and in the hp case which refinement type to do. Figure 6.1 shows the general algorithm of adaptivity.

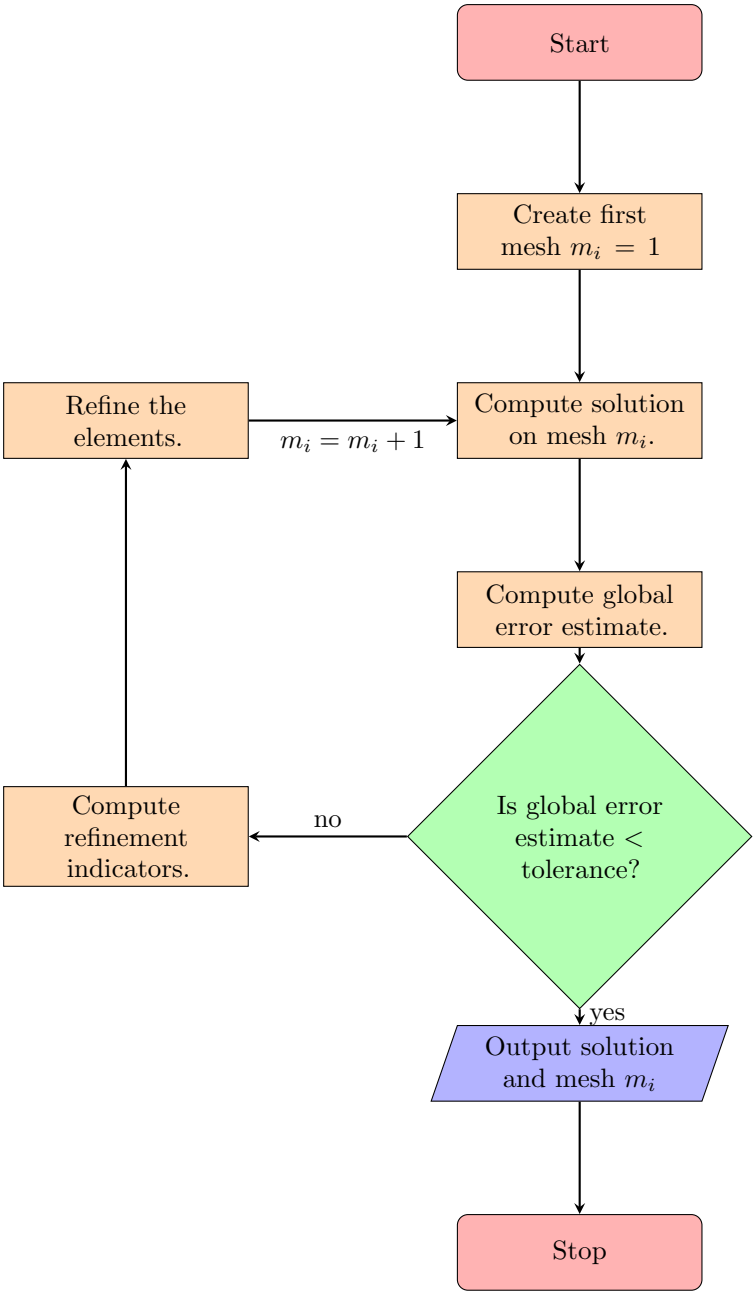

Figure 6.1: General form of mesh refinement algorithms [67].

Unfortunately, the L^2 norm we have used thus far can only be calculated using the analytical solution of the problem which heavily restricts the number of problems that it is useable for. We need to find a new way of computing the global error as well as the local errors.

6.3 *a posteriori* error estimator

Directly computing the error is impossible for many problems as they lack a known analytical solution; therefore, we must rely on an estimation of the error. To that end, we introduce an a posteriori error estimation. This estimation is based on an upper

bound of the error of a solution. Then we can guarantee analytically that the error, still unknown, is below this estimation. There are three main types of a posteriori error estimates: implicit, explicit, and recovery-based [2]. We will be focussing solely on the dual weighted residual (DWR) method [10] a type of goal oriented explicit a posteriori error estimation.

Consider a target functional, $J(\cdot)$, of physical interest, for example a mean value functional [30]:

$$J(v) = \int_{\Omega} \omega v d\mathbf{x} \tag{6.3.1}$$

where $\omega \in L^2(\Omega)$ is a given weight function [37].

We introduce the dual problem with respect to that functional, with the solution $z \in \mathbb{V}_{\Omega}^{\mathbf{p}}$ being defined by

$$J(v) = B(v, z) \quad \forall v \in \mathbb{V}_{\Omega}^{\mathbf{p}}.$$

where $B(\cdot, \cdot)$ is the bilinear from weak form of the primal problem. Therefore,

$$J(u) = B(u, z) = \ell(z).$$

So we can exploit the structure of our functional and bilinear form [31], to get

$$\begin{aligned} J(u) - J(u_h) &= J(u - u_h) \quad \text{by linearity} \\ &= B(u - u_h, z) \\ &= B(u - u_h, z - z_h) \quad \text{by Galerkin orthogonality (Section 2.3.2)} \\ &= B(u, z - z_h) - B(u_h, z - z_h) \\ &= \ell(z - z_h) - B(u_h, z - z_h) \\ &=: \mathcal{R}(u_h, z - z_h) \end{aligned}$$

Here $\mathcal{R}(u_h, z - z_h)$ is the *a posteriori* estimation of the error.

Due to the Galerkin orthogonality property, the dual solution z cannot be computed on the same finite dimensional subspace $\mathbb{V}_{\Omega}^{\mathbf{p}}$ as the primal solution, u, as this would lead to an error representation formula that is identically zero. Hence, the discrete dual solution $\hat{z}_h$ which we use to approximate $z - z_h$, again by Galerkin orthogonality, must come from an enriched space, we use $\mathbb{V}_{\Omega}^{\mathbf{p}+\mathbf{1}}$ [31]. We, therefore, approximate the a posteriori error using

$$|J(u) - J(u_h)| \approx \ell(\hat{z}_h) - B(u_h, \hat{z}_h) = |\mathcal{R}(u_h, \hat{z}_h)|.$$

Furthermore, this residual $\mathcal{R}(u_h, \hat{z}_h)$ can be decomposed into individual element contributions:

$$|J(u) - J(u_h)| \approx |\mathcal{R}(u_h, \hat{z}_h)| = |\sum_{\kappa_{\Omega} \in \mathcal{T}_{\Omega}} \eta_{\kappa_{\Omega}}| \leq \sum_{\kappa_{\Omega} \in \mathcal{T}_{\Omega}} |\eta_{\kappa_{\Omega}}|. \tag{6.3.2}$$

So it can be used to calculate the global and local error estimations.

To be able to compare error estimation techniques, it is customary to refer to the effectivity of the estimation given by:

$$I = \left| \frac{\sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \eta_{\kappa_\Omega}}{J(u) - J(u_h)} \right|$$

An effectivity of one indicates that the estimation is perfectly estimating the functional error.

The marking procedure is the method by which we chose which elements in our mesh to refine. This is of the utmost importance; if we mark too few elements, we will have to iterate the loop in Figure 6.1 many more times than we need to, making it very inefficient. On the other hand, if we choose to refine too many elements, we will end up having a less efficient programme. An 'optimal' choice would be defined as one that minimises this ratio [13]

$$\frac{\text{Work done to solve the new problem}}{\text{Gain in accuracy}}.$$

There are many marking procedures that are based around treating this ratio as an optimisation problem [21, 13]. We present these different options here:

- **Even distribution of error**

  Assume that the error is split evenly across all the elements, and then require all elements to have an error, such that when summed it will be lower than the tolerance [38].

  $$|\eta_{\kappa_\Omega}| \leq \frac{TOL}{N}$$

  where N is the number of elements in $\mathcal{T}_h$.

- **Fraction of Max**

  Use the maximum of the errors found [3]

  $$|\eta_{\kappa_\Omega}| < \omega \max_{\kappa_\Omega \in \mathcal{T}_\Omega} |\eta_{\kappa_\Omega}| \quad \omega \in (0, 1).$$

  The common choice is ω is $\frac{1}{3}$.

- **Dörfler marking**

  For a given $\omega \in (0, 1)$ find a subset $\mathcal{M}_\Omega$ of $\mathcal{T}_\Omega$, such that [23]

  $$\sum_{\kappa_\Omega \in \mathcal{M}_\Omega} |\eta_{\kappa_\Omega}|^2 \geq \omega \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} |\eta_{\kappa_\Omega}|^2.$$

  $\mathcal{M}_\Omega$ is constructed by the elements with the largest η_{κ_Ω} first.

- **Fixed fraction method**

  Order $|\eta_{\kappa_\Omega}|$ in terms of size and then select the top 20-30% of them for marking

[21]. The actual percentage is highly problem dependent, and tries to find a balance between over refining, where DOFs are created that do not have a meaningful impact on the overall error, and not performing too many runs.

The best choice is highly problem dependent, but for the rest of this report we will be using fixed fraction. Having decided which elements need refining and marking them, we then need to decide whether we should use h or p refinement. As hp-refinement is not our focus, we only mention two methods to do this.

- **Legendre Polynomial Coefficient**

  We estimate the decay rate of the coefficients by a least squares fit. Where $u^i_{\kappa_\Omega}, i = p_{\kappa_\Omega} - 4, \ldots, p_{\kappa_\Omega}$ is the ith Legendre coefficient in a one-dimensional expansion of the solution, we can assume that [51]

  $$u^i_h \approx Ce^{-i\sigma}.$$

  Then we p-refine if $\sigma > 1$ and h-refine otherwise (if $_{\kappa_\Omega} \leq 3$ then p-refine).

- **Sobolev Regularity Estimation**

  We use a root test derived in the paper [36].

  $$l_{\kappa_\Omega} = \frac{\log\left(\frac{2p_{\kappa_\Omega}+1}{2|u_h^{p_{\kappa_\Omega}}|^2}\right)}{2\log p_{\kappa_\Omega}} \quad p_{\kappa_\Omega} > 2,$$

  where $u^{p_{\kappa_\Omega}}$ is the p_{κ_Ω}th Legendre coefficient in a one-dimensional expansion of the solution. If $p_{\kappa_\Omega} \geq l_{\kappa_\Omega}$ use p-refinement, else use h-refinement.

Due to the high polynomial degree required for these tests, performing hp-refinement is impractical on the available hardware. We have, therefore, focused solely on h-refinement for our implementation section.

6.3.1 Transport problem

Applying the DWR (6.3.2) to a transport problem as presented in Section 2.3.1 we can define our element residual:

$$\begin{aligned}\eta_{\kappa_\Omega} = &\int_{\kappa_\Omega} f\hat{z}_h \, d\mathbf{x} - \int_{\partial_- \kappa_\Omega \cap \partial\Omega} (\boldsymbol{\mu}\cdot\boldsymbol{n}_{\kappa_\Omega}) g_{\mathrm{D}} \hat{z}_h \, ds \\ &- \int_{\kappa_\Omega} (\boldsymbol{\mu}\cdot\nabla u_h \hat{z}_h + (\alpha) u_h \hat{z}_h) \, d\mathbf{x} \\ &+ \int_{\partial_- \kappa_\Omega \setminus \partial\Omega} (\boldsymbol{\mu}\cdot\boldsymbol{n}_{\kappa_\Omega}) \lfloor u_h \rfloor \hat{z}_h^+ \, ds \\ &+ \int_{\partial_- \kappa_\Omega \cap \partial\Omega} (\boldsymbol{\mu}\cdot\boldsymbol{n}_{\kappa_\Omega}) u_h^+ \hat{z}_h^+ \, ds.\end{aligned}$$

We define ω of our mean value functional $J(\cdot)$ (6.3.1) as a Gaussian function centred on $\boldsymbol{b} = (0.3, 0.3)^T$, of height $a = 1$, and width of $\boldsymbol{c} = (1, 1)^T$:

$$\omega(\mathbf{x}) = a \exp\left(-\left(\frac{(x-b_1)^2}{2c_1^2} + \frac{(y-b_2)^2}{2c_2^2}\right)\right).$$

Using the example problem from Section 4.5.1, we can compute the effectivities of our error estimator. Table 6.1 show that our effectivities are close to one and converge towards it with additional refinement. This shows that DWR is a good error estimator for the transport problem.

$\boldsymbol{p}$	Total DOFs	Effectivities
0	4,096	0.9894823564
0	16,384	0.9946546551
0	65,536	1.0001351506
0	262,144	1.0000654540
0	1,048,576	0.9999945653
1	12,288	0.9916364352
1	49,152	0.9964397215
1	196,608	1.0000054546
1	786,432	1.0000009854
1	3,145,728	0.9999998564
2	24,576	0.9986464116
2	98,304	0.9999983476
2	393,216	1.0000008653
2	1,572,864	1.0000006351
2	6,291,456	1.0000005854

Table 6.1: The effectivities for transport problem (10 s.f.)

Algorithm 1 h-refinement algorithm 1 with fixed fraction of 30%.

```
1: function h-REFINEMENT(1)(η)
2:     Sort η high to low, populate elementsnum[1:N] ▷ Record the order of the elements
3:     elements[1:N]=0
4:     for m ← 1 to ⌈N × 0.3⌉ do
5:         i= elementsnum[m]
6:         elements[i]=1
7:     end for
8:     Return elements                    ▷ elements[i]=1 marks element i for refinement
9: end function
```

The h-refinement algorithm employed, h-refinement(1), here is very simple: if an element's error indicator, η_{κ_Ω}, is in the top 30% of the errors in the mesh, then it is refined. We arrived at the fixed fraction of 30% by experimenting with different fixed fractions.

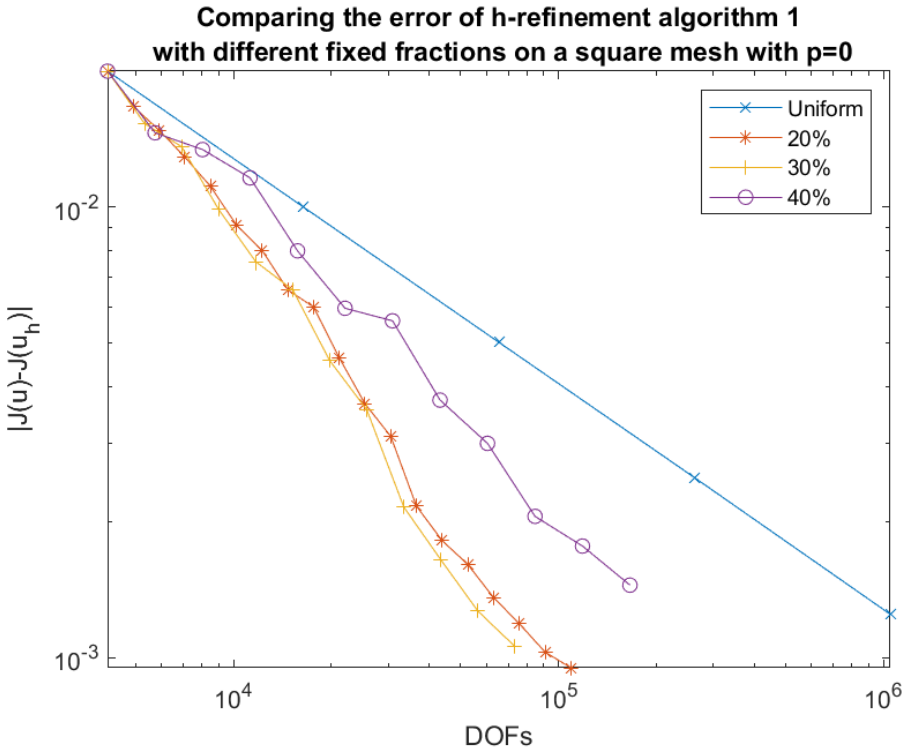

Figure 6.2: Comparing the functional errors of h-refinement algorithm 1 with different fixed fractions on a square mesh with $\boldsymbol{p} = 0$.

Figure 6.2 shows the functional error of different fixed fractions, we can see that refining the top 40% of elements gives suboptimal convergence, due to elements being refined that are not contributing enough to the overall error. Fixed fractions of 20% and 30% give similar convergence, but a very different number of iterations of the loop in Figure 6.1. With each iteration representing a solve of the primal and dual problem, in order to save time we want to minimise the number of iterations required to reach an appropriate level of accuracy. While a fixed fraction of 40% has a lower number of iterations in the graph shown, it will be likely to require more iterations than the fixed fraction of 30% to reach a functional error of around 10^{-3}.

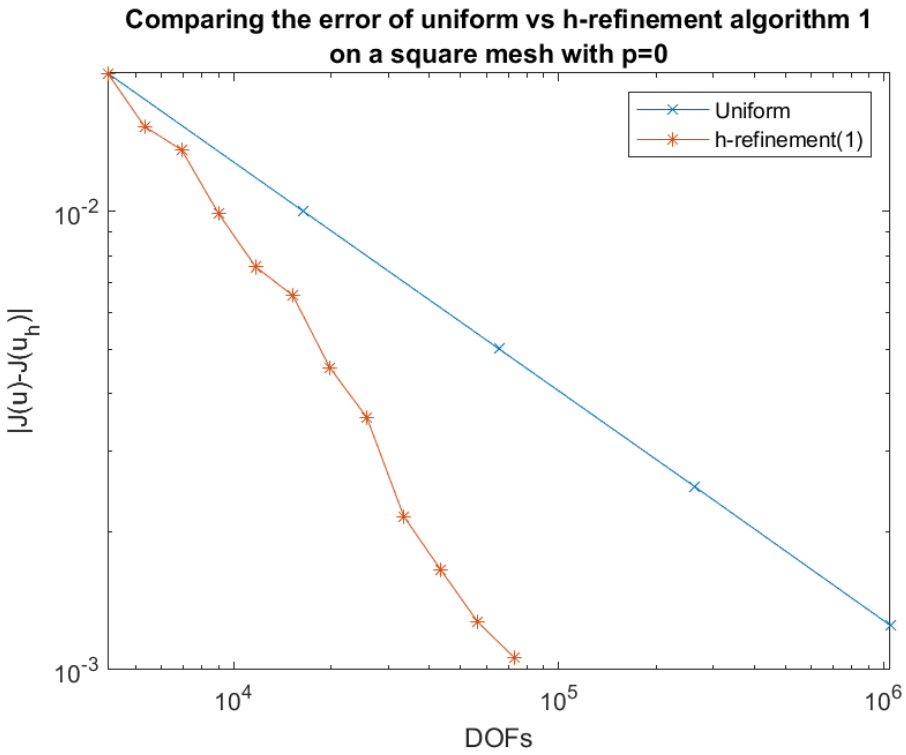

Figure 6.3: The error estimates of uniform vs h-refinement algorithm 1 for a transport problem on a square mesh with p=0.

Figure 6.3 shows the h-adaptive refinement algorithm compared to uniform refinement, with the h-refinement terminated once the error indicator is below the error indicator of the finest mesh in the uniform refinement we considered. It is clear that the h-refinement uses many fewer DOFs to get the same accuracy as uniform refinement.

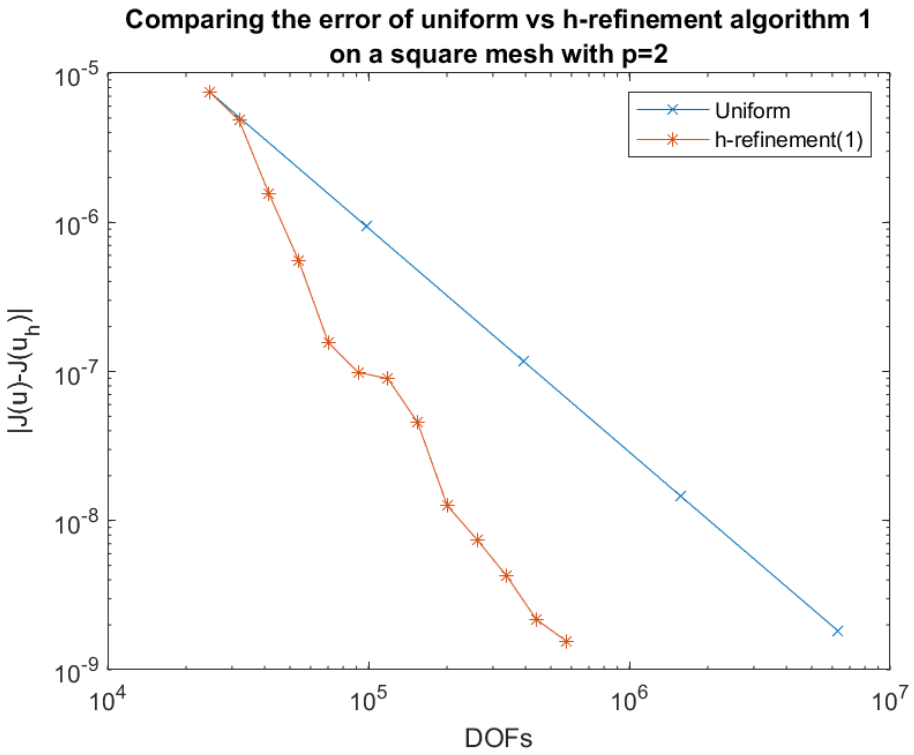

Figure 6.4: The error estimates of uniform vs h-refinement algorithm 1 for a transport problem on a square mesh with p=2.

Figure 6.4 shows the same thing but for $\boldsymbol{p} = 2$. Again, h-refinement requires many fewer DOFs for the same level of accuracy. It also shows a relatively common phenomenon, where refining some elements leads to little or no reduction in the error. Both Figures 6.3 and 6.4 indicate that for uniform refinement, the functional error $|J(u) - J(u_h)|$ is of order $O(h^{2(\boldsymbol{p}+\frac{1}{2})})$ which is the expected optimal rate of convergence [38].

6.3.2 LBTE

Applying DWR to the LBTE, we can define the residual for the tensor element as

$$\begin{aligned}
\eta_\kappa = &\int_{\kappa_g}\int_{\kappa_{\mathbb{S}}}\int_{\Omega} f\hat{z}_h \, d\mathbf{x}\, d\boldsymbol{\mu}\, dE \\
&- \int_{\kappa_g}\int_{\kappa_{\mathbb{S}}}\int_{\partial_-\kappa_\Omega\cap\partial\Omega} (\boldsymbol{\mu}\cdot\boldsymbol{n}_{\kappa_\Omega}) g_{\mathrm{D}}\hat{z}_h \, ds\, d\boldsymbol{\mu}\, dE \\
&- \int_{\kappa_g}\int_{\kappa_{\mathbb{S}}}\int_{\kappa_\Omega} (\boldsymbol{\mu}\cdot\nabla_{\mathbf{x}} u_h\hat{z}_h + (\alpha+\beta)u_h\hat{z}_h)\, d\mathbf{x}\, d\boldsymbol{\mu}\, dE \\
&+ \int_{\kappa_g}\int_{\kappa_{\mathbb{S}}}\int_{\partial_-\kappa_\Omega\setminus\partial\Omega} (\boldsymbol{\mu}\cdot\boldsymbol{n}_{\kappa_\Omega}) \lfloor u_h\rfloor \hat{z}_h^+ \, ds\, d\boldsymbol{\mu}\, dE \\
&+ \int_{\kappa_g}\int_{\kappa_{\mathbb{S}}}\int_{\partial_-\kappa_\Omega\cap\partial\Omega} (\boldsymbol{\mu}\cdot\boldsymbol{n}_{\kappa_\Omega}) u_h^+\hat{z}_h^+ \, ds\, d\boldsymbol{\mu}\, dE \\
&- \int_{\kappa_g}\int_{\kappa_{\mathbb{S}}}\int_{\kappa_\Omega} \mathcal{S}[u_h](\mathbf{x},\boldsymbol{\mu},E)\hat{z}_h \, d\mathbf{x}\, d\boldsymbol{\mu}\, dE.
\end{aligned}$$

We define ω of our mean value functional $J(\cdot)$ as a von Mises type distribution, centred on a point $\boldsymbol{b}_\Omega = (0.3, 0.3)^T$ and $\boldsymbol{b}_\mathbb{S} = (1)$ for the space and angular domains, with concentrations of $\kappa_\Omega = 50$ and $\kappa_\mathbb{S} = 0$, and noting that for $\kappa_\Omega = 0$ the distribution is normal:

$$\omega(\mathbf{x}, \boldsymbol{\mu}) = \frac{1}{2\pi I_0(\kappa_\mathbb{S})} \exp \kappa_\Omega(\mathbf{x} - \boldsymbol{b}_\Omega)^2,$$

here $I_0(\kappa_\mathbb{S})$ is the modified Bessel function of the first kind of order 0. We use this weight for all the dual problems of the LBTE we compute in this chapter.

6.4 2D mono-energetic

Using the model problem from Section 3.7, we can see the effectivities of the error estimation in Tables 6.2 are close to one. As the error estimation is a good approximation

$\boldsymbol{p} = \boldsymbol{q}$	Total DOFs	Effectivities
0	4,096	0.9894823564
0	16,384	0.9946546551
0	65,536	0.9998648494
0	262,144	1.0000654546
0	1,0485,76	0.9999945653
1	122,88	0.9916364352
1	49,152	0.9964397215
1	196608	1.0000054546
1	786,432	1.0000009854
1	3,145,728	0.9999998564
2	24,576	0.9986464116
2	98,304	1.0000016524
2	393,216	0.9999991347
2	1,572,864	1.0000006351
2	6,291,456	0.9999994146

Table 6.2: The effectivities for 2D mono-energetic problem (10 s.f.)

of the functional error. However, when we commence with h-refinement algorithm 1 with a fixed fraction of 20% we see some unexpected results.

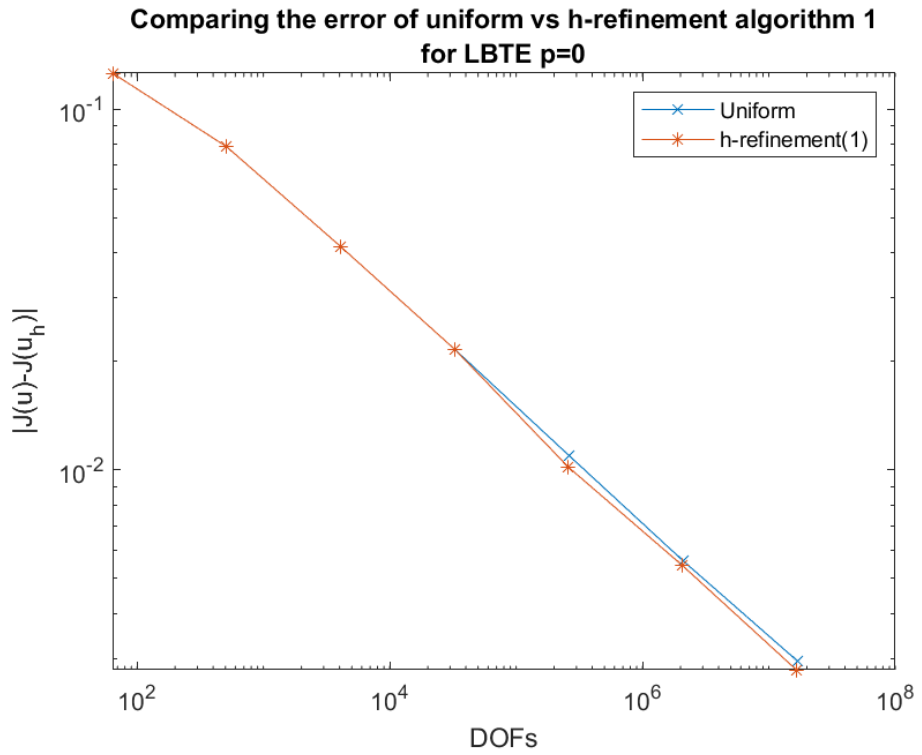

Figure 6.5: Comparing the functional error of uniform vs h-refinement algorithm 1 for LBTE with $\boldsymbol{q} = \boldsymbol{p} = 0$.

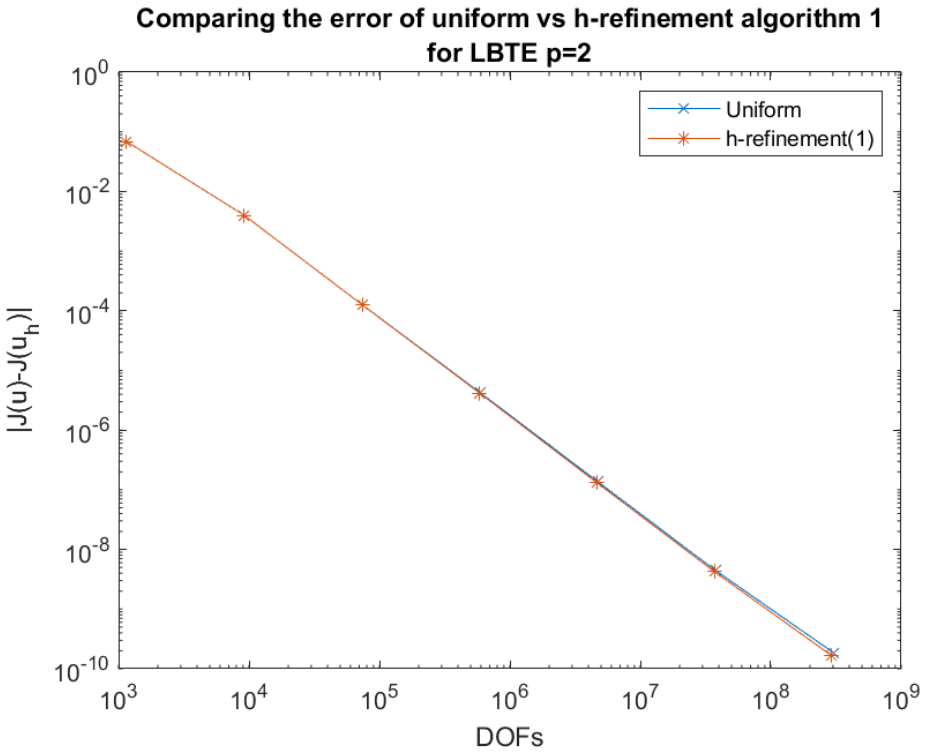

Figure 6.6: Comparing the functional error of uniform vs h-refinement algorithm 1 for LBTE with $\boldsymbol{q} = \boldsymbol{p} = 2$.

Figures 6.5 and 6.6 show that the h-refinement algorithm 1 on the LBTE using

the tensor element error indicators results in almost the same as uniform refinement despite only marking 20% of the elements for refinement. This is a result of the tensor structure of the elements. Marking a tensor element for refinement, in fact, marks an element in each of the space and angle meshes. Due to this, marking 20% of the tensor elements results in nearly every space and angle element being refined thus giving uniform refinement. Figures 6.5 and 6.6 also show that uniform refinement with fixed $\boldsymbol{q}, \boldsymbol{p}$ gave a convergence of the functional error of order $O(h^{2(\boldsymbol{p}+\frac{1}{2})})$.

To combat this, it would be helpful to be able to compute error indicators for both the space and angle meshes separately.
Let $\Pi_\Omega, \Pi_\mathbb{S}$ denote the L^2-projection operators onto the spatial and angular finite element spaces, so that $z_h = \Pi_\Omega \Pi_\mathbb{S} z$. Therefore,

$$\begin{aligned} J(u) - J(u_h) &= \mathcal{R}(u_h, z - z_h) \\ &= \mathcal{R}(u_h, z - \Pi_\Omega \Pi_\mathbb{S} z) \\ &= \mathcal{R}(u_h, z - \Pi_\Omega z + \Pi_\Omega z - \Pi_\Omega \Pi_\mathbb{S} z) \\ &= \mathcal{R}(u_h, z - \Pi_\Omega z) + \mathcal{R}(u_h, \Pi_\Omega(z - \Pi_\mathbb{S} z)) \end{aligned}$$

Approximating the dual solution z with $\hat{z}_h \in \mathbb{V}_\Omega^{\mathbf{p+1}}$ via DWR, gives

$$J(u) - J(u_h) \approx \mathcal{R}(u_h, \hat{z}_h - \Pi_\Omega \hat{z}_h) + \mathcal{R}(u_h, \Pi_\Omega(\hat{z}_h - \Pi_\mathbb{S} \hat{z}_h)).$$

$$\begin{aligned} |J(u) - J(u_h)| &\lessapprox |\mathcal{R}(u_h, \hat{z}_h - \Pi_\Omega \hat{z}_h)| + |\mathcal{R}(u_h, \Pi_\Omega(\hat{z}_h - \Pi_\mathbb{S} \hat{z}_h))| \\ &=: |\sum_{\kappa_\Omega \in \mathcal{T}_\Omega} \eta_{\kappa_\Omega}| + |\sum_{\kappa_\mathbb{S} \in \mathcal{T}_\mathbb{S}} \eta_{\kappa_\mathbb{S}}| \\ &\leq \sum_{\kappa_\Omega \in \mathcal{T}_\Omega} |\eta_{\kappa_\Omega}| + \sum_{\kappa_\mathbb{S} \in \mathcal{T}_\mathbb{S}} |\eta_{\kappa_\mathbb{S}}| \end{aligned}$$

Where,

$$\begin{aligned} \eta_{\kappa_\Omega} = &\int_{\kappa_\Omega} f(\hat{z}_h - \Pi_\Omega \hat{z}_h)\, d\mathbf{x} - \int_{\partial_- \kappa_\Omega \cap \partial\Omega} (\boldsymbol{\mu} \cdot \boldsymbol{n}_{\kappa_\Omega}) g_\mathrm{D} (\hat{z}_h - \Pi_\Omega \hat{z}_h)\, ds \\ &- \int_{\kappa_\Omega} (\boldsymbol{\mu} \cdot \nabla u_h (\hat{z}_h - \Pi_\Omega \hat{z}_h) + (\alpha + \beta) u_h (\hat{z}_h - \Pi_\Omega \hat{z}_h))\, d\mathbf{x} \\ &+ \int_{\partial_- \kappa_\Omega \setminus \partial\Omega} (\boldsymbol{\mu} \cdot \boldsymbol{n}_{\kappa_\Omega}) \lfloor u_h \rfloor (\hat{z}_h^+ - \Pi_\Omega \hat{z}_h^+)\, ds \\ &+ \int_{\partial_- \kappa_\Omega \cap \partial\Omega} (\boldsymbol{\mu} \cdot \boldsymbol{n}_{\kappa_\Omega}) u_h^+ (\hat{z}_h^+ - \Pi_\Omega \hat{z}_h^+)\, ds \\ &+ \int_{\kappa_\Omega} \mathcal{S}[u_h](\mathbf{x}, \boldsymbol{\mu}, E)(\hat{z}_h - \Pi_\Omega \hat{z}_h)\, d\mathbf{x}, \end{aligned}$$

and

$$\begin{aligned}\eta_{\kappa_{\mathbb{S}}} = &\int_{\kappa_{\mathbb{S}}} f\Pi_{\Omega}(\hat{z}_h - \Pi_{\mathbb{S}}\hat{z}_h)\, d\boldsymbol{\mu} \\ &- \int_{\kappa_{\mathbb{S}}} (\boldsymbol{\mu}\cdot\nabla_{\mathbf{x}} u_h \Pi_{\Omega}(\hat{z}_h - \Pi_{\mathbb{S}}\hat{z}_h) + (\alpha+\beta)u_h\Pi_{\Omega}(\hat{z}_h - \Pi_{\mathbb{S}}\hat{z}_h))\, d\boldsymbol{\mu} \\ &- \int_{\kappa_{\mathbb{S}}} \mathcal{S}[u_h](\mathbf{x},\boldsymbol{\mu},E)\Pi_{\Omega}(\hat{z}_h - \Pi_{\mathbb{S}}\hat{z}_h)\, d\boldsymbol{\mu}.\end{aligned}$$

6.4.1 h-refinement algorithm 2

As we wish to minimise the error of the LBTE on the tensor element, it does not make sense to treat these two error indicators separately. While we could refine the elements on each mesh that have the highest error indicators, that does not guarantee that the tensor element with the most error is refined. We must, therefore, develop a more customised algorithm to decide which elements of the space and angle meshes to refine.

Using the split error indicators, we developed a new refinement algorithm, h-refinement(2). The h-refinement algorithm 2 marks the tensor elements with the largest local error estimates, the space, and angle elements that form that tensor element have one added to their tally. The space and angle elements with the highest tallies are then refined.

Algorithm 2 h-refinement algorithm 2 with fixed fraction of 20%.

function h-REFINEMENT(2)($\eta_\Omega[1:N_\Omega], \eta_\mathbb{S}[1:N_\mathbb{S}]$,tensorelements[1:N])
  for $i \leftarrow 1$ to N **do**
    j,k=tensorelements[i] ▷ Fetch space, angle element numbers for tensor element i
    η[i]=$\eta_\Omega[j] + \eta_\mathbb{S}[k]$
  end for
  Sort η high to low, populate tensorelementsnum[1:N] ▷ Record the order of the elements
  spacetally $[1:N_\Omega]$=0
  angletally $[1:N_\mathbb{S}]$=0
  for $m \leftarrow 1$ to $\lceil N/5 \rceil$ **do**
    i=tensorelementsnum$[m]$
    j,k=tensorelements $[i]$
    spacetally$[j] + +$
    angletally$[k] + +$
  end for
  Sort spacetally high to low, populate spaceelementsnum[1:N_Ω]
  Sort angletally high to low, populate angleeelementsnum[1:$N_\mathbb{S}$]
  spaceelements[1:N_Ω]=0
  angleelements[1:$N_\mathbb{S}$]=0
  for $m \leftarrow 1$ to $\lceil N_\Omega/5 \rceil$ **do**
    j=spaceelementsnum$[m]$
    spaceelements$[j] = 1$
  end for
  for $m \leftarrow 1$ to $\lceil N_\mathbb{S}/5 \rceil$ **do**
    k=angleelementsnum$[m]$
    angleelements$[k] = 1$
  end for
  Return spaceelements and angleelements ▷ spaceelements[j]=1 marks space element j for refinement, analogous for angle
end function

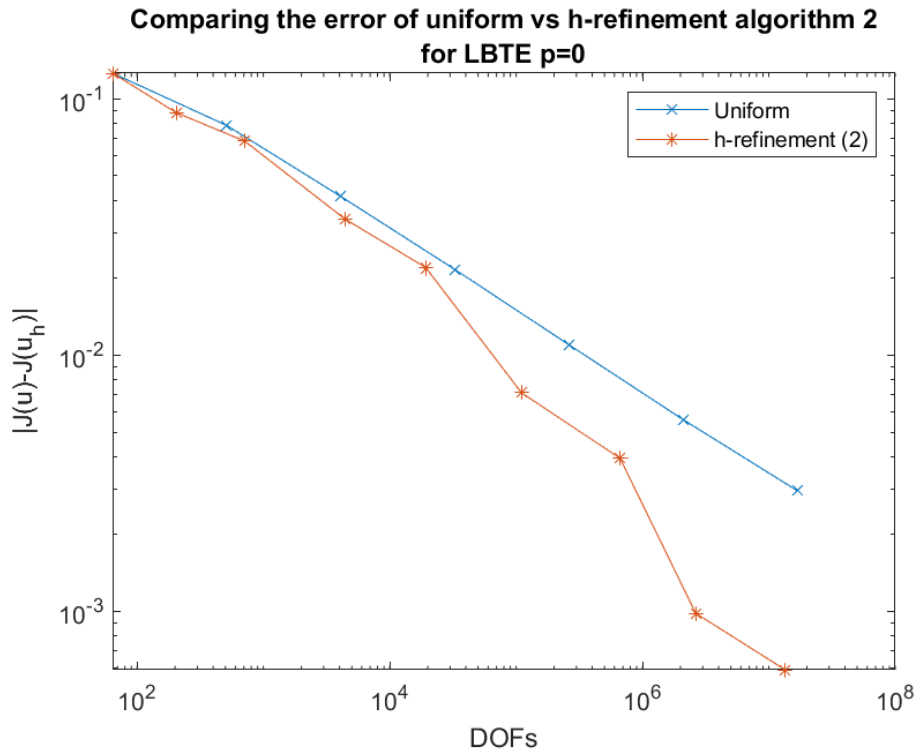

Figure 6.7: Comparing the functional error of uniform vs h-refinement algorithm 2 for LBTE with $\boldsymbol{q} = \boldsymbol{p} = 0$.

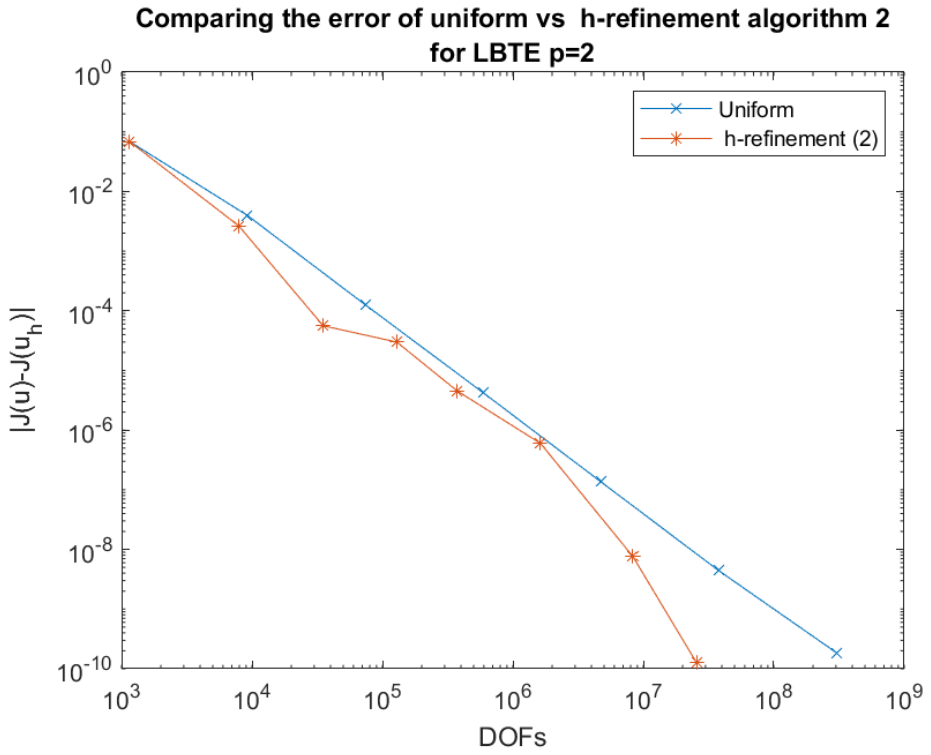

Figure 6.8: Comparing the functional error of uniform vs h-refinement algorithm 2 for LBTE with $\boldsymbol{q} = \boldsymbol{p} = 2$.

As Figures 6.7 and 6.8 show, this is a marked improvement on uniform refinement

and thus the tensor residual; however, it is not as efficient as we might have hoped. This is likely to be due to it being possible for the space or angle element that is marked most to not correspond to the tensor element with the highest error.

6.4.2 h-refinement algorithm 3

To address this issue, we introduce h-refinement algorithm 3, which multiplies each tally by the error contribution of the tensor element. Thus, the space and angle elements selected to be refined are the ones that contribute the most to the global tensor error.

Algorithm 3 h-refinement algorithm 3 with fixed fraction of 20%.

function h-REFINEMENT(3)($\eta_\Omega[1:N_\Omega], \eta_\mathbb{S}[1:N_\mathbb{S}]$,tensorelements[1:N])
  for $i \leftarrow 1$ to N **do**
    j,k=tensorelements[i] ▷ Fetch space, angle element numbers for tensor element i
    η[i]=$\eta_\Omega[j] + \eta_\mathbb{S}[k]$
  end for
  Sort η high to low, populate tensorelementsnum[1:N] ▷ Record the order of the elements
  spacetally $[1:N_\Omega]$=0
  angletally $[1:N_\mathbb{S}]$=0
  for $m \leftarrow 1$ to $\lceil N/5 \rceil$ **do**
    i=tensorelementsnum[m]
    j,k=tensorelements [i]
    spacetally[j]= spacetally[j]+$\eta[i]$
    angletally[k]= angletally[k] +$\eta[i]$
  end for
  Sort spacetally high to low, populate spaceelementsnum[1:N_Ω]
  Sort angletally high to low, populate angleeelementsnum[1:$N_\mathbb{S}$]
  spaceelements[1:N_Ω]=0
  angleelements[1:$N_\mathbb{S}$]=0
  for $m \leftarrow 1$ to $\lceil N_\Omega/5 \rceil$ **do**
    j=spaceelementsnum[m]
    spaceelements[j] = 1
  end for
  for $m \leftarrow 1$ to $\lceil N_\mathbb{S}/5 \rceil$ **do**
    k=angleelementsnum[m]
    angleelements[k] = 1
  end for
  Return spaceelements and angleelements ▷ spaceelements[j]=1 marks space element j for refinement, analogous for angle
end function

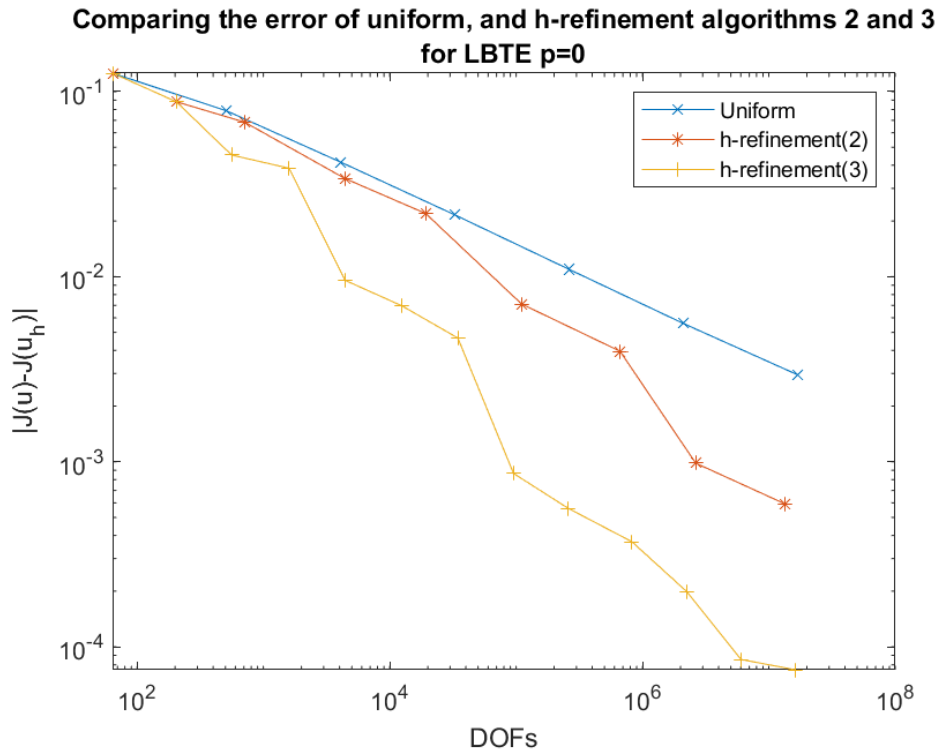

Figure 6.9: Comparing the functional error of uniform, h-refinement algorithms 2 and 3 for LBTE with $\boldsymbol{q} = \boldsymbol{p} = 0$.

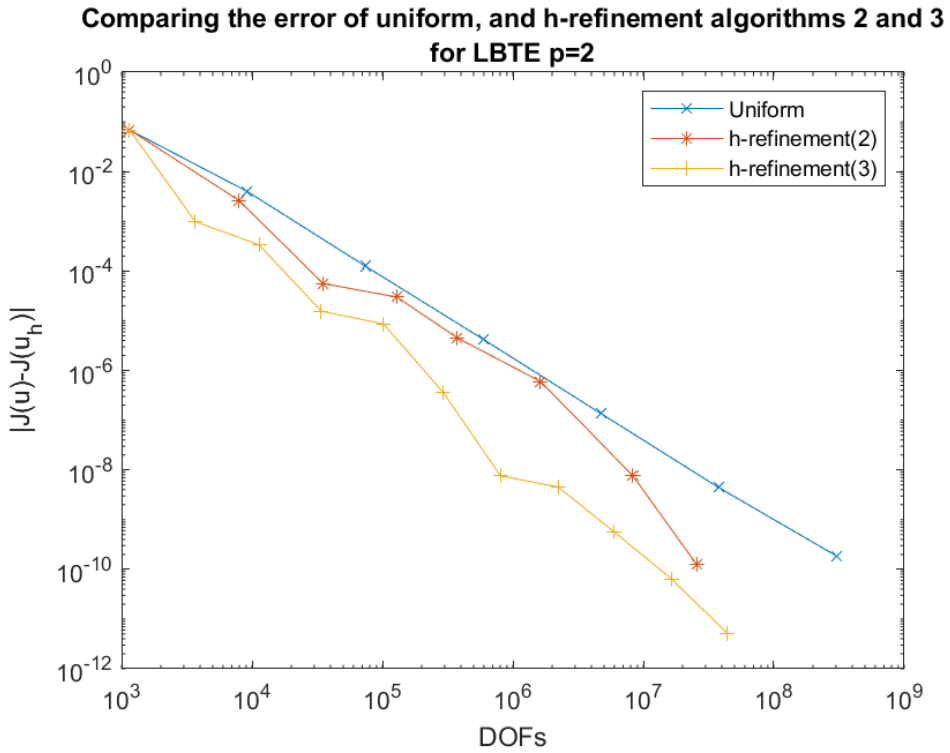

Figure 6.10: Comparing the functional error of uniform, h-refinement algorithms 2 and 3 for LBTE with $\boldsymbol{q} = \boldsymbol{p} = 2$.

Figures 6.9 and 6.10 show that h-refinement algorithm 3 is a great improvement on

h-refinement algorithm 2, especially for $\boldsymbol{q} = \boldsymbol{p} = 0$. This probably means that many space and/ or angle elements formed tensor elements marked for refinement multiple times, thus were marked for refinement by algorithm 2 but didn't necessarily contribute the most to the overall error. h-refinement algorithm 3 has proved to be very efficient compared to uniform refinement, requiring around one order of magnitude fewer DOFs for the same error for $\boldsymbol{q} = \boldsymbol{p} = 0$.

6.5 2D mono-energetic beam

So far we have only used a fairly smooth non-physical problem for the LBTE, but to demonstrate the effectiveness of our refinement techniques we introduce the beam problem. This is a closer match to the type of problem that would be used in radiography. It represents an emitter with a focused beam of photons being fired from it, into a unit square of water, $\Omega = [0, 1]^2$cm. The beam enters on the $x = 0$ boundary and is shone horizontally, with an initial radius of 5cm. The boundary condition is given by

$$g(\mathbf{x}, \boldsymbol{\mu}) = \begin{cases} \frac{1}{2}(1 - \tanh(100(\|\mathbf{x} - \boldsymbol{s}\| - 5))) \text{ if } \boldsymbol{d} \cdot \boldsymbol{\mu} > 0.99995 \\ 0 \text{ otherwise} \end{cases}$$

where $\boldsymbol{s}$ is the source of the beam on the boundary in this case $\boldsymbol{s} = (0, 0.5)$, and $\boldsymbol{d}$ is the direction of travel of the beam in this case $\boldsymbol{d} = (1, 0)$. There is no forcing in this problem as the source of photons is outside of our domain of interest, so $f = 0$.

The data terms α and $\mathcal{S}$ are chosen to mimic the Compton scattering of photons travelling through water. To that end the macroscopic absorption cross-section is set at $\alpha = 0$, and the differential scattering cross-section is:

$$\theta(\mathbf{x}, \boldsymbol{\mu} \cdot \boldsymbol{\mu}') = \rho(x)\frac{1}{2}r(2 - (1 - \cos^2(\vartheta))^2)$$

where $\rho(x)$ is the electron density of the medium, in this case water $\rho(x) \approx 3.34281 \times 10^{29}\text{m}^{-3}$ [27], and $r \approx 2.81794 \times 10^{-15}$m is a physical constant, the classical electron radius [32]. $\cos(\vartheta) = \boldsymbol{\mu} \cdot \boldsymbol{\mu}'$ is the scattering angle.

The macroscopic scattering cross-section of the medium can be calculated from the differential scattering cross-section

$$\beta(\mathbf{x}, \boldsymbol{\mu}) = \int_{\mathbb{S}} \theta(\mathbf{x}, \boldsymbol{\mu} \to \boldsymbol{\mu}')\, d\boldsymbol{\mu}';$$

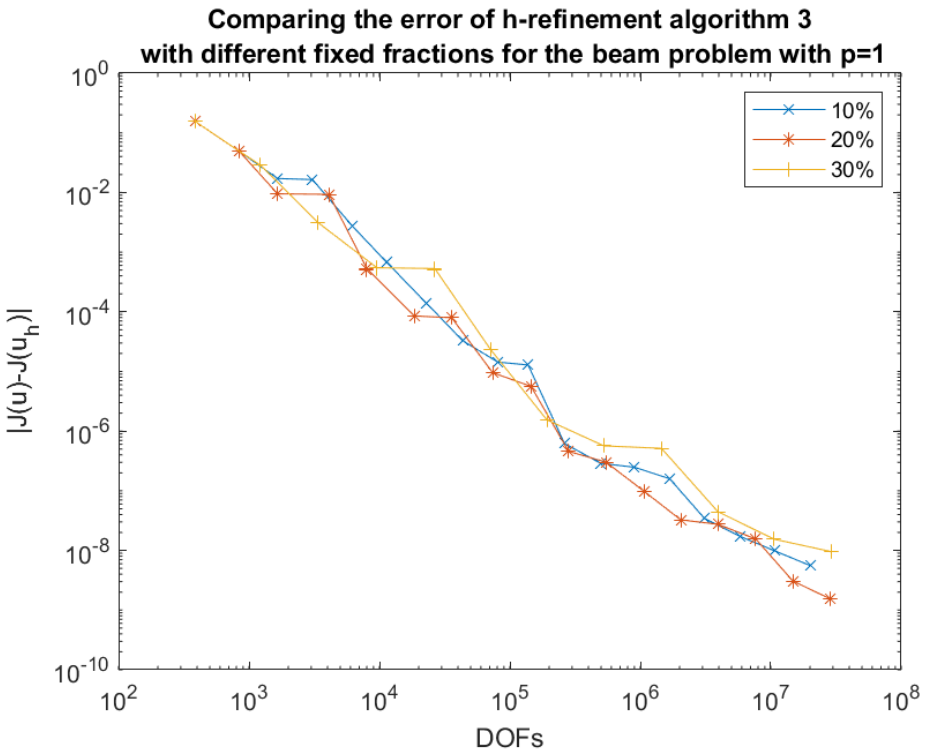

Figure 6.11: Comparing the functional errors of h-refinement algorithm 3 with different fixed fractions for the beam problem with $\boldsymbol{q} = \boldsymbol{p} = 1$.

Figure 6.11 shows the functional error for the beam problem with different fixed fractions, we choose a fixed fraction of 20% as it requires fewer iterations of the loop in Figure 6.1 than the 10% fixed fraction to reach an error below 10^{-7}. A fixed fraction of 30%, requires fewer iterations but over refines. To reach an error below 10^{-7}. a fixed fraction of 30% required 3,933,828 DOFs whereas our choice of the fixed fraction of 20% only required 896,112 DOFs.

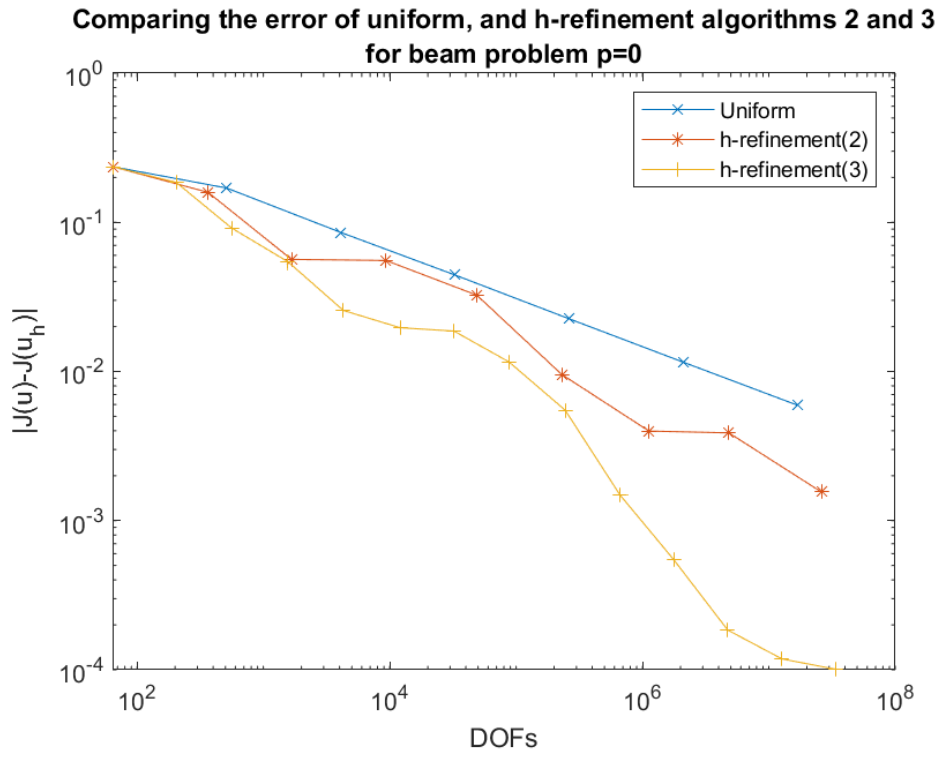

Figure 6.12: Comparing the functional error of uniform, h-refinement algorithms 2 and 3 for beam problem with $\boldsymbol{q} = \boldsymbol{p} = 0$.

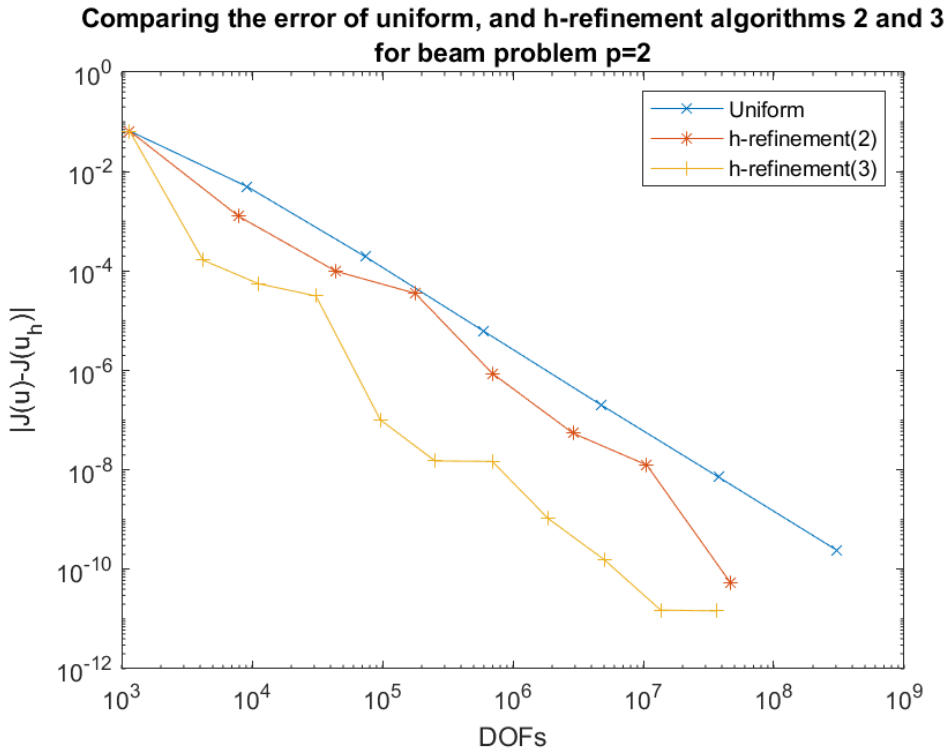

Figure 6.13: Comparing the functional error of uniform, h-refinement algorithms 2 and 3 for beam problem with $\boldsymbol{q} = \boldsymbol{p} = 2$.

Figures 6.12 and 6.13 show that the functional error converges in order $O(h^{2(\boldsymbol{p}+\frac{1}{2})})$

for uniform refinement. Figures 6.12 and 6.13, also show the h-refinement algorithm 3 is far more efficient in terms of DOFs for functional error than the h-refinement algorithm 2, with an order of magnitude difference for higher DOFs. Both algorithms have instances where the h-refinement does not result in any meaningful reduction in the functional error. This is likely to be due to the h-refinements not providing sufficient resolution to capture the details of the solution. Both algorithms, however, perform better than the uniform refinement and thus the h-refinement algorithm 1.

6.6 3D mono-energetic beam

We take the beam problem from Section 6.5, and applied it to a 3D cube of water, so the mono-energetic LBTE becomes 5D. To do this, we take the same problem definition, as in Section 6.5, but use $\Omega = [0,1]^3\text{cm}$, $\boldsymbol{s} = (0, 0.5, 0.5)$ and $\boldsymbol{d} = (1, 0, 0)$. The rest of the definition of the problem is dimensionally invariant.

Unfortunately, with this many dimensions, we cannot do uniform refinement due to the number of degrees of freedom it would require, but we have included a predicted rate of convergence for uniform refinement, $O(h^{2(p+\frac{1}{2})})$.

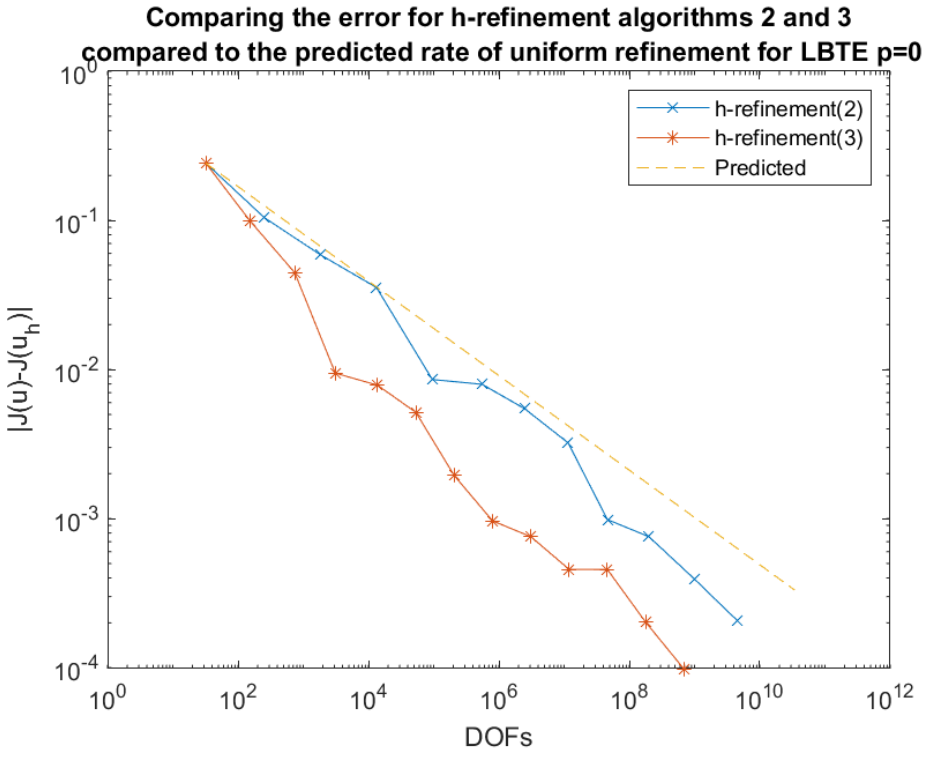

Figure 6.14: Comparing the functional error from h-refinement algorithms 2 and 3 to the predicted rate of uniform refinement for LBTE $\boldsymbol{q} = \boldsymbol{p} = 0$.

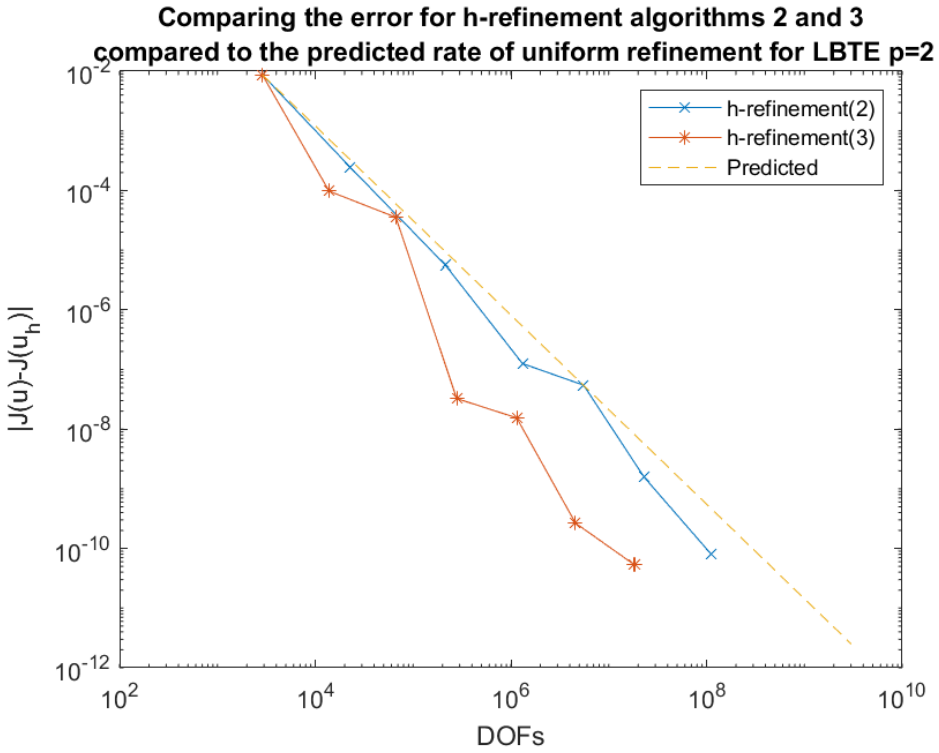

Figure 6.15: Comparing the functional error from h-refinement algorithms 2 and 3 to the predicted rate of uniform refinement for LBTE $\boldsymbol{q} = \boldsymbol{p} = 2$.

From Figures 6.14 and 6.15 we once again see that h-refinement algorithm 3 outperforms h-refinement algorithm 2. Also, we do see refinements that have little to no effect on the overall functional error. This can be explained by the resolution, failing to capture details of the solutions. Additionally, we see refinements that provide a steep drop in the functional error. This is likely to be a consequence of the resolution of the mesh capturing the underlying behaviour of the solution.

6.7 Summary

In conclusion, we have shown how an a posteriori error estimates can be computed using the dual weighted residual method for both the transport and LBTE problems. We have also shown how this residual error can be then be divided into residuals for each element in the mesh. This allows us to identify the elements that are contributing the most to the global error.

Using these error indicators, we showed how a basic h-refinement algorithm, h-refinement(1), can be used to solve the transport problem more efficiently than uniform refinement. Additionally we saw that these residuals were a good approximation of the functional error for the transport problem as we got effectivities of around one. We also saw that uniform h-refinement resulted in the functional error reducing at order $O(h^{2(\boldsymbol{p}+\frac{1}{2})})$.

When we tried h-refinement algorithm 1 on the LBTE we obtained nearly uniform refinement due to the tensor nature of the elements. We, therefore, created h-refinement algorithm 2 to tally the number of times a space or angle element is marked for refinement, and then refine the elements with the most tallies. h-refinement algorithm 2 proved to be an upgrade to h-refinement algorithm 1, being more efficient than uniform refinement. h-refinement algorithm 2 did not always refine the elements that were contributing the most to the global error. To combat this, we created h-refinement algorithm 3 which weighted the tallies with the tensor error indicators. h-refinement algorithm 3 performed well, providing an order of magnitude reduction to the functional error for the same number of DOFs as uniform refinement.

Finally, we tested h-refinement algorithm 2 and 3 on a LBTE problem that incorporated more realistic physics. We simulated a beam of photons being fired in to a block of water, and undergoing Compton scattering. Once again, h-refinement algorithm 3 proved to be more efficient.

Chapter 7

Conclusion

In this thesis, we have developed efficient high-order discontinuous Galerkin finite element methods (DGFEMs) for the numerical approximation of the mono- and poly-energetic forms of the linear Boltzmann transport equation (LBTE). We have employed DGFEMs in each of the spatial, angular, and energetic domains to create a three-way tensor discretisation of the LBTE. We have then shown how this tensor discretisation can be implemented as a sequence of transport equations coupled in the right hand side. The remaining work has focussed on optimising the construction and solving of this discretisation.

In Chapter 3, we introduced discretisations for each of the spatial, angular, and energy domains, and specified spaces of discontinuous piecewise-polynomial basis functions used in each case. These function spaces were necessary to develop a full space-angle-energy discretisation of the poly-energetic linear Boltzmann transport equation using the discontinuous Galerkin finite element method. We then provided numerics to show that the space-angle DG-DG discretisation of the mono-energetic LBTE is high order, with the error in the L^2 norm converging at $O(h^{\boldsymbol{p}+1})$ for a space-angle mesh-size parameter h and the global polynomial degree of approximation p. We then showed how the angle and energy domains' DG discretisations could become a sequence of coupled transport equations, like a multigroup discrete-ordinates-like solver.

Chapter 4 provided a deeper insight into the form of the matrix resulting from the discretisation in the previous chapter. We discussed and presented the construction of the transport equation blocks. We then examined how meshes of different element types could affect the matrix sparsity, and thus the construction and inverting of the matrix of the transport equation. The creation of polytopic meshes was covered, with emphasis being placed on agglomeration. We also examined the agglomeration of elements into polytopes and how the underlying fine elements are still employed by a "standard" matrix solver.

In Chapter 5 we presented a novel transport equation solver. This solver showed itself

to be considerably more efficient at constructing and inverting the system's matrix than a direct matrix solver for convex elements, while not requiring the system's matrix to be fully constructed. We then showed that the same solver could be efficient for non-convex elements if there were not many cyclic dependencies present. If, however, large cyclic dependencies were present, this sweep solver would then be less efficient than the direct matrix solver. Finally, in this chapter, we showed the speed-up we had achieved for solving the LBTE. We compared the tensor discretisation with a direct matrix solver to the coupled transport equations solved by our sweep solver for both square elements and elements agglomerated into squares.

Finally, in Chapter 6 we presented an a posteriori error bound for the transport equation and the linear Boltzmann transport equation. We used these error bounds to create an adaptive h-refinement solve for each of them. Noting the h-refinement algorithm was underperforming for the LBTE, we presented an error estimate that was split across the domains of the LBTE and then used this to develop two h-refinement algorithms. We then tested these h-refinement algorithms on a problem that modelled real world physics to great success. The h-refinement algorithm that marks the element with the error of each tensor element it is part of, showing a clear advantage over the other methods.

7.1 Further work

There are several topics of research that can be suggested as extensions of the work contained in the thesis. We will briefly discuss them here.

7.1.1 Poly-energetic LBTE

We provided a novel way of discretising the energy domain of the LBTE in Section 3.8.1; however, we have not provided any numerics to support it. Some numerics do appear in [33] and [55] but no comparisons to the "standard" multigroup approximations exist. This could be informative as the DGFEM of the energy domain is a high order approximation, but multigroup is low order. Additionally, a comparison to the full three-way tensor system of DG-DG-DG, outlined in Section 3.4, could be done.

7.1.1.1 a posteriori error bound and adaptivity

There are a number of open questions about solving the dual problem of the poly-energetic LBTE. It is unclear if one solves the dual problem from high energy to low or from low energy to high. The split a posteriori error bound, discussed in Section 6.4, can be expanded to include poly-energetic LBTE problems. We believe that an energy error

indicator can be created of the form

$$\begin{aligned} J(u) - J(u_h) &= \mathcal{R}(u_h, z - \Pi_\Omega \Pi_\mathbb{S} \Pi_\mathbb{E} z) \\ &= \mathcal{R}(u_h, z - \Pi_\Omega \Pi_\mathbb{S} z + \Pi_\Omega \Pi_\mathbb{S} z - \Pi_\Omega \Pi_\mathbb{S} \Pi_\mathbb{E} z) \\ &= \mathcal{R}(u_h, z - \Pi_\Omega z + \Pi_\Omega (z - \Pi_\mathbb{S} z) - \Pi_\Omega \Pi_\mathbb{S} (z - \Pi_\mathbb{E} z)) \\ &= \mathcal{R}(u_h, z - \Pi_\Omega z) + \mathcal{R}(u_h, \Pi_\Omega (z - \Pi_\mathbb{S} z)) + \mathcal{R}(u_h, \Pi_\Omega \Pi_\mathbb{S} (z - \Pi_\mathbb{E} z)). \end{aligned}$$

We can then incorporate this into any of our h-refinement algorithms, and the results can be compared. Further to this, a full hp-refinement run could be done for the mono-energetic, as well as the poly-energetic LBTE. This will require immense computer resources, as most of the smoothness checks, such as the Sobolev Regularity Estimation, shown in Section 6.3, require high polynomial degrees to be reliable [36].

7.1.2 Mesh creation and agglomeration

We have shown that convex elements are required by our sweep solver to be efficient, but there are many ways to create convex polygonal meshes. The Voronoi tessellation provides convex polygonal shapes, but has no respect for underlying problem geometry. An adapted tessellation may be able to be created which allows this.

Alternatively, agglomerating the MRI or CAT scan data into convex polygons is possible. While agglomerating square/cube elements into a larger square/cube elements is possible, there are still problems associated with it. There can be many faces evaluated by the quadrature free algorithm, as we have to evaluate each face in the fine mesh that forms the boundary of an element in the coarse mesh. This could be fixed by agglomerating the faces of the coarse mesh element. This is relatively simple for a square coarse mesh element, but is more complex for a cube or arbitrary shaped polytope. Also, refining our coarse mesh would require de-agglomerating the coarse mesh and then re-agglomerating the fine mesh into more elements. If we manage to complete these two tasks, we could remove the fine mesh entirely. So once we have agglomerated the fine mesh, we could store our coarse mesh as a completely new mesh and remove the underlying mesh. The question then becomes how to h-refine a convex polygon into N convex polygons. This is an interesting open problem, that would have many other applications in other fields.

7.1.3 Medical physics applications

In order to demonstrate that our deterministic approach can be used on real world problems, it will be necessary to benchmark our method vs gold standard Monte Carlo methods [44]. These benchmarks will, by necessity, have to be much more complex, with inhomogeneities in the material coefficients [68].

7.1.3.1 Electron Transport

To actually be able to calculate dosage for radiotherapy, we will need to move on to the study of electrons. We know from physics that the interactions between photons and a medium can produce an electron and electrons can scatter from colliding with each other [11]. At energy levels used for radiology, electron interactions do not produce a photon, but they may at lower energies. We, therefore, form a coupled system of equations, one models photons and their interactions, feeding into the second equation which models electrons and their interactions [32].

$$\begin{aligned}
&\boldsymbol{\mu}\cdot\nabla_{\mathbf{x}}u_\gamma(\mathbf{x},\boldsymbol{\mu},E)+(\alpha_\gamma(\mathbf{x},E)+\beta_\gamma(\mathbf{x},E))u_\gamma(\mathbf{x},\boldsymbol{\mu},E)\\
&\qquad=\int_{\mathbb{E}}\int_{\mathbb{S}}\theta_{\gamma\gamma}(\mathbf{x},\boldsymbol{\mu}'\cdot\boldsymbol{\mu},E'\to E)u_\gamma(\mathbf{x},\boldsymbol{\mu}',E')\;\,d\boldsymbol{\mu}'\,dE+f_\gamma(\mathbf{x},\boldsymbol{\mu},E),\\
&\boldsymbol{\mu}\cdot\nabla_{\mathbf{x}}u_e(\mathbf{x},\boldsymbol{\mu},E)+(\alpha_e(\mathbf{x},E)+\beta_e(\mathbf{x},E))u_e(\mathbf{x},\boldsymbol{\mu},E)\\
&\qquad=\int_{\mathbb{E}}\int_{\mathbb{S}}\theta_{\gamma e}(\mathbf{x},\boldsymbol{\mu}'\cdot\boldsymbol{\mu},E'\to E)u_\gamma(\mathbf{x},\boldsymbol{\mu}',E')\;\,d\boldsymbol{\mu}'\,dE\\
&\qquad+\int_{\mathbb{E}}\int_{\mathbb{S}}\theta_{ee}(\mathbf{x},\boldsymbol{\mu}'\cdot\boldsymbol{\mu},E'\to E)u_e(\mathbf{x},\boldsymbol{\mu}',E')\;\,d\boldsymbol{\mu}'\,dE\\
&\qquad+f_e(\mathbf{x},\boldsymbol{\mu},E),
\end{aligned}$$

where, the terms α_γ, β_γ, and $\theta_{\gamma\gamma}$ denote the typical macroscopic absorption, macroscopic scattering, and differential scattering cross-sections associated with the physics of photons. Likewise α_e, β_e and θ_{ee} for electrons and the differential cross-section $\theta_{\gamma e}$ represents electron production from photon interactions.

From this, the electron scalar flux

$$\phi_e(\mathbf{x},E)=\int_{\mathbb{S}}u_e(\mathbf{x},\boldsymbol{\mu},E)\,d\boldsymbol{\mu}$$

could be used to calculate the dosage of radiation. Once we can calculate the dosage, we can use a dosage functional to inform our adaptivity algorithms and do goal oriented adaptive refinements.

Bibliography

[1] Adams, M. L., and Larsen, E. W. Fast iterative methods for discrete-ordinates particle transport calculations. *Progress in Nuclear Energy 40*, 1 (2002), 3 – 159.

[2] Ainsworth, M., and Oden, J. A posteriori error estimation in finite element analysis. *Computer Methods in Applied Mechanics and Engineering 142*, 1 (1997), 1–88.

[3] Ainsworth, M., and Senior, B. Aspects of an adaptive *hp*-finite element method: Adaptive strategy, conforming approximation and efficient solvers. *Computer Methods in Applied Mechanics and Engineering 150*, 1 (1997), 65 – 87. Symposium on Advances in Computational Mechanics.

[4] Antonietti, P. F., Dedner, A., Madhavan, P., Stangalino, S., Stinner, B., and Verani, M. High order discontinuous Galerkin methods for elliptic problems on surfaces. *SIAM J. Numer. Anal. 53* (2015), 1145–1171.

[5] Antonietti, P. F., Giani, S., and Houston, P. *hp*-version composite discontinuous galerkin methods for elliptic problems on complicated domains. *SIAM Journal on Scientific Computing 35*, 3 (2013), A1417–A1439.

[6] Antonietti, P. F., Houston, P., and Pennesi, G. Fast numerical integration on polytopic meshes with applications to discontinuous Galerkin finite element methods. *Journal of Scientific Computing* (2018), 1–32.

[7] Arnold, D. N., Brezzi, F., Cockburn, B., and Marini, L. D. Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM journal on numerical analysis 39*, 5 (2002), 1749–1779.

[8] Asadzadeh, M. Analysis of a fully discrete scheme for neutron transport in two-dimensional geometry. *SIAM journal on numerical analysis 23*, 3 (1986), 543–561.

[9] Aurenhammer, F. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv. 23*, 3 (Sept. 1991), 345–405.

[10] Becker, R., and Rannacher, R. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica 10* (2001), 1–102.

[11] Bedford, J. L. Calculation of absorbed dose in radiotherapy by solution of the linear Boltzmann transport equations. *Physics in Medicine and Biology* (2018).

[12] Bell, G. I., and Glasstone, S. Nuclear reactor theory. Tech. rep., US Atomic Energy Commission, Washington, DC (United States), 1970.

[13] Berndt, M., Manteuffel, T. A., and McCormick, S. F. Local error estimates and adaptive refinement for first-order system least squares (fosls). *Electron. Trans. Numer. Anal 6* (1997), 35–43.

[14] Bey, K. S., and Tinsley Oden, J. *hp*-version discontinuous galerkin methods for hyperbolic conservation laws. *Computer Methods in Applied Mechanics and Engineering 133*, 3 (1996), 259–286.

[15] Brooks, A. N., and Hughes, T. J. R. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering 32* (Sept. 1982), 199–259.

[16] Cangiani, A., Dong, Z., Georgoulis, E. H., and Houston, P. *hp-Version Discontinuous Galerkin Methods on Polygonal and Polyhedral Meshes*. Springer International Publishing, 2017.

[17] Carson, M. The spherical harmonic method. Tech. rep., National Research Council of Canada, Atomic Energy Project, Division of Research, 1947.

[18] Cockburn, B. *Discontinuous Galerkin Methods for Computational Fluid Dynamics*. American Cancer Society, 2017, pp. 1–63.

[19] Cockburn, B., Karniadakis, G. E., and Shu, C.-W. The development of discontinuous galerkin methods. B. Cockburn, G. E. Karniadakis, and C.-W. Shu, Eds., Springer Berlin Heidelberg.

[20] Cockburn, B., and Shu, C.-W. Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws ii: General framework. *Mathematics of Computation 52*, 186 (1989), 411–435.

[21] De Sterck, H., Manteuffel, T., McCormick, S., Nolting, J., Ruge, J., and Tang, L. Efficiency-based h-and *hp*-refinement strategies for finite element methods. *Numerical Linear Algebra with Applications 15*, 2-3 (2008), 89–114.

[22] Demlow, A. Higher-order finite element methods and pointwise error estimates for elliptic problems on surfaces. *SIAM J. Numer. Anal. 47* (2009), 805–827.

[23] Dörfler, W. A convergent adaptive algorithm for poisson's equation. *SIAM Journal on Numerical Analysis 33*, 3 (1996), 1106–1124.

[24] Dziuk, G., and Elliott, C. M. Finite element methods for surface PDEs. *Acta Numer. 22* (2013), 289–396.

[25] Eppstein, D. How many tetrahedra?, https://www.ics.uci.edu/~eppstein/projects/tetra/, 2001.

[26] Fletcher, J. K. A solution of the neutron transport equation using spherical harmonics. *Journal of Physics A: Mathematical and General 16*, 12 (aug 1983), 2827–2835.

[27] Geant, et al. Physics reference manual. *Version: Geant4 9*, 0 (2016).

[28] Gifford, K. A., Horton Jr, J. L., Wareing, T. A., Failla, G., and Mourtada, F. Comparison of a finite-element multigroup discrete-ordinates code with monte carlo for radiotherapy calculations. *Physics in Medicine & Biology 51*, 9 (2006), 2253.

[29] Green, P. J., and Sibson, R. Computing Dirichlet Tessellations in the Plane. *The Computer Journal 21*, 2 (05 1978), 168–173.

[30] Harriman, K., Houston, P., Senior, B., and Suli, E. *hp*-version discontinuous galerkin methods with interior penalty for partial differential equations with nonnegative characteristic form. Tech. rep., Unspecified, 2002.

[31] Hartmann, R., and Houston, P. Adaptive discontinuous galerkin finite element methods for the compressible euler equations. *Journal of Computational Physics 183*, 2 (2002), 508–532.

[32] Hensel, H., Iza-Teran, R., and Siedow, N. Deterministic model for dose calculation in photon radiotherapy. *Physics in Medicine & Biology 51*, 3 (2006), 675.

[33] Houston, P., Hubbard, M. E., Radley, T. J., Sutton, O. J., and Widdowson, R. S. J. Efficient high-order space-angle-energy polytopic discontinuous galerkin finite element methods for linear boltzmann transport, https://doi.org/10.48550/arXiv.2304.09592, 2023.

[34] Houston, P., Schwab, C., and Suli, E. Stabilized *hp*-finite element methods for first-order hyperbolic problems. *SIAM Journal on Numerical Analysis 37*, 5 (2000), 1618–1643.

[35] Houston, P., Schwab, C., and Süli, E. Discontinuous *hp*-finite element methods for advection-diffusion-reaction problems. *SIAM Journal on Numerical Analysis 39*, 6 (2002), 2133–2163.

[36] Houston, P., Senior, B., and Süli, E. Sobolev regularity estimation for *hp*-adaptive finite element methods.

[37] Houston, P., Senior, B., and Süli, E. *hp*-discontinuous galerkin finite element methods for hyperbolic problems: error analysis and adaptivity. *International Journal for Numerical Methods in Fluids 40*, 1-2, 153–169.

[38] Houston, P., and Süli, E. *hp*-adaptive discontinuous galerkin finite element methods for first-order hyperbolic problems. *SIAM Journal on Scientific Computing 23*, 4 (2001), 1226–1252.

[39] Inform, N. Radiotherapy, https://www.nhsinform.scot/tests-and-treatments/non-surgical-procedures/radiotherapy, 2022.

[40] Johnson, C., and Pitkäranta, J. An analysis of the discontinuous galerkin method for a scalar hyperbolic equation. *Mathematics of computation 46*, 173 (1986), 1–26.

[41] Karypis, G. Metis, https://github.com/KarypisLab/METIS, 2023.

[42] Karypis, G., and Kumar, V. METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. *Computer Science Engineering (CSE) Technical Reports 749* (1997).

[43] Karypis, G., and Kumar, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing 20*, 1 (1998), 359–392.

[44] Kawrakow, I., and Fippel, M. Investigation of variance reduction techniques for Monte Carlo photon dose calculation using XVMC. *Physics in Medicine & Biology 45*, 8 (2000), 2163.

[45] Koch, R., Krebs, W., Wittig, S., and Viskanta, R. Discrete ordinates quadrature schemes for multidimensional radiative transfer. *Journal of Quantitative Spectroscopy and Radiative Transfer 53*, 4 (1995), 353 – 372.

[46] Larsen, E. W., Miften, M., Fraass, B. A., and Bruinvis, I. A. D. Electron dose calculations using the method of moments. *Medical physics 24 1* (1997), 111–25.

[47] Lasaint, P., and Raviart, P.-A. On a finite element method for solving the neutron transport equation. In *Mathematical aspects of finite elements in partial differential equations*. Elsevier, 1974, pp. 89–123.

[48] Lathrop, K. D., and Carlson, B. G. Discrete ordinates angular quadrature of the neutron transport equation. Tech. Rep. LA-3186, Los Alamos Scientific Laboratory, 1965.

[49] Lesaint, P., and Raviart, P.-A. On a finite element method for solving the neutron transport equation. *Publications mathématiques et informatique de Rennes*, S4 (1974), 1–40.

[50] Lewis, E. E. E. E., and Miller, W. F. *Computational methods of neutron transport*. New York : Wiley, 1984. "A Wiley-Interscience publication.".

[51] Mavriplis, C. Adaptive mesh strategies for the spectral element method. *Computer methods in applied mechanics and engineering 116*, 1-4 (1994), 77–86.

[52] Persson, P.-O. *Mesh generation for implicit geometries*. PhD thesis, Massachusetts Institute of Technology, 2005.

[53] Peterson, T. E. A note on the convergence of the discontinuous galerkin method for a scalar hyperbolic equation. *SIAM Journal on Numerical Analysis 28*, 1 (1991), 133–140.

[54] ProgrammerSought. Cg-polygon triangulation, https://www.programmersought.com/article/6011775469/, 2001.

[55] Radley, T. *Discontinuous Galerkin Methods for the Linear Boltzmann Transport Problem*. PhD thesis, University of Nottingham, 2022.

[56] Reed, W. H., and Hill, T. Triangular mesh methods for the neutron transport equation. Tech. rep., Los Alamos Scientific Lab., N. Mex.(USA), 1973.

[57] Schötzau, D., and Schwab, C. An *hp* a priori error analysis of the dg time-stepping method for initial value problems. *CALCOLO 37*, 4 (Dec 2000), 207–232.

[58] Schötzau, D., Schwab, C., and Wihler, T. P. *hp*-dgfem for second-order elliptic problems in polyhedra i: Stability on geometric meshes. *Siam journal on numerical analysis* (2013), 1610–1633.

[59] Schötzau, D., and Schwab, C. Time discretization of parabolic problems by the *hp*-version of the discontinuous galerkin finite element method. *SIAM Journal on Numerical Analysis 38*, 3 (2001), 837–875.

[60] Seaïd, M., Klar, A., and Pinnau, R. Numerical solvers for radiation and conduction in high temperature gas flows. *Flow, Turbulence and Combustion 75* (2005), 173–190.

[61] Simon, C. P., Blume, L., et al. *Mathematics for economists*, vol. 7. Norton New York, 1994.

[62] Stacey, W. M. *Nuclear reactor physics*. John Wiley & Sons, 2018.

[63] Süli, E., Houston, P., and Schwab, C. *hp*-finite element methods for hyperbolic problems. *The Mathematics of Finite Elements and Applications X* (2000), 143–162.

[64] Tarjan, R. Depth-first search and linear graph algorithms. *SIAM Journal on Computing 1*, 2 (1972), 146–160.

[65] Technologies, M. Multifrontal massively parallel sparse direct solver, https://mumps-solver.org/, 2022.

[66] UK, C. R. Cancer statistics for the uk, https://www.cancerresearchuk.org/health-professional/cancer-statistics-for-the-uk, 2023.

[67] Verfürth, R. A posteriori error estimation and adaptive mesh-refinement techniques. *Journal of Computational and Applied Mathematics 50*, 1-3 (1994), 67–83.

[68] Wang, L., Lovelock, M., and Chui, C.-S. Experimental verification of a CT-based Monte Carlo dose-calculation method in heterogeneous phantoms. *Medical Physics 26*, 12 (1999), 2626–2634.