PhD First Year Report

Finite Element Methods for Fluid-Structure Interaction Problems

Adam M Blakey, *Mathematical Sciences*

14th July 2021

This report contains my progress so far, ultimately aiming to construct a reliable and realistic in-silico model of the haemodynamics in the human placenta. We plan to take advantage of hp-adaptive FEMs, as well as specialised techniques in FSI and multiscale modelling to achieve this.

1 PhD Work

1.1 Introduction and Literature Review

Differential equations are widely used in science and engineering to explain and predict facts about real-world phenomena – from predicting the planetary orbits and the weather, to the modelling of epidemics and traffic flow, to the design of buildings and aeroplanes. However, even simple models often do not yield analytical solutions, so one of two approaches is usually taken instead: solve a simplified model to find an approximate solution to the original equations, or approximate the equations directly using some form of numerical method; the latter is the focus of this report. In particular, this project aims to develop a validated biophysical model of blood flow distribution across the intervillous space (IVS) of the human placenta, incorporating the recently observed placental pump [15].

Jensen and Chernyavsky explain the role and modelling techniques of the placenta eloquently in [28]. In summary, the placenta has two main purposes: to exchange blood and nutrients between mother and fetus. In human placentas, fetal blood flows through many small villi that are submerged in maternal blood. Current models describe the IVS containing the villi as a deformable porous medium. We specifically plan to model the flow of maternal blood surrounding the villi in the placenta, as shown in Figure 1; the figure outlines the villi containing the fetal blood, surrounded by maternal blood in the IVS.

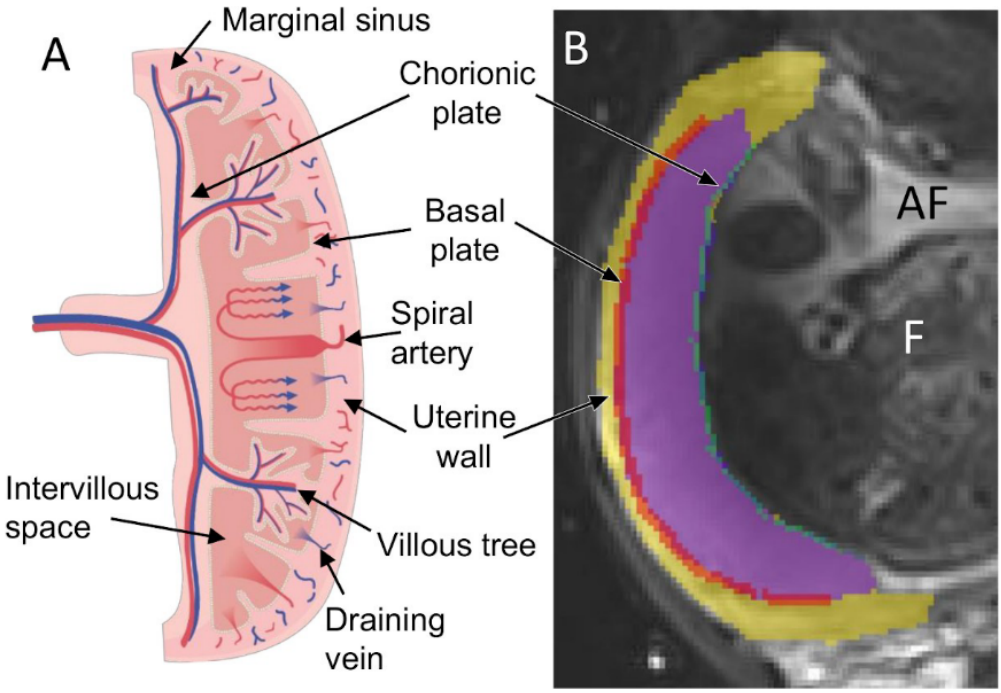


Figure 1: *(A) Schematic showing flow of both maternal and fetal blood through the placenta. Fetal blood flows through fetal vessels across the chorionic plate to the villous tree and returns along the same route. Maternal blood flows through spiral arteries into the IVS and exits via the draining veins. (B) Typical placental MRI image, with the purple colouring corresponding to the placental region of interest. Image taken from [15].*

The 'utero-placental pump' is a new phenomenon that is documented by Dellschaft et al. in [15] where it has been found experimentally that placental volume can reduce by up to 40% in contractions over a 10-minute period. We plan to model this phenomenon computationally.

It is quite early into this project to specify exactly what tools will be used to build up the described biophysical model, but suitable approaches include:

- Darcy flow in the IVS [13][28];
- Fluid-structure interaction (FSI) and moving boundary techniques to capture placental pump phenomenon [3][4][14][15];
- Multiscale modelling techniques for villous structures [27][30].

We choose to use finite element methods for modelling these effects computationally — and more specifically, we choose to use discontinuous Galerkin finite element methods (DGFEMs). Continuous Galerkin finite element methods (CGFEMs) also exist and enforce continuity between 'elements' in each computational mesh. Loosely speaking, DGFEMs are somewhat of a hybrid between classical continuous FEMs and finite volume methods (FVMs) [10], the latter being used extensively in computational fluid dynamics (CFD) simulations. Cangiani et al. [10] explain that DGFEMs are a great choice for modelling in many practical settings due to their simple treatment of complicated geometries, ease of adaptivity in both the local mesh subdivision and local polynomial enrichment, and availability of rigorous error analysis tools. A posteriori tools are of particular practical interest for both error quantification as well as automatic adaptivity strategies, and form a great choice of method for our application.

The IVS of the placenta can be, and has been, modelled as a deformable porous medium. There are two main approaches to modelling flow in this region, either using tree-scale models that are on the scale of an entire villous tree, or using pore-scale models that resolve features at the level of individual villous

branches (approximately 50 µm) [28]; models at each scale model different behaviour, one of which may be more important for certain applications. Chernyavsky, Jensen, and Leach use a tree-scale model, similar to what we are hoping to achieve, which used a 3D Darcy model in a hemispherical domain to model the maternal blood [12]. In somewhat of a contrast, Lecarpentier et al. made comparison between two models of the flow that they had developed: one using the Brinkman equations, and one given by the full Navier-Stokes equations [30]. Although there are many mathematical biological applications that take into account elastic effects on geometries (e.g. [33]), we are currently unaware of any models that take into account the elasticity of a realistic geometry on the placenta-scale, although some do include elastic effects on smaller placental components (e.g. [18]). This is something we plan to incorporate in our biophysical computational model.

As described in [14], we may utilise some form of FSI to accurately model the interaction between the fluid in the IVS, and as well as the surrounding elastic tissue to capture the placental pump phenomenon described in [15]. FSI problems are, perhaps obviously by their name, associated with the interaction between fluids and solids (or structures). These problems arise in many areas including the aeroelasticity of aircraft wings [37], the playing of reeded woodwind instruments [34], the construction of bridges [38], and the haemodynamics of animals [19]. In terms of numerical simulations of fluid-structure interaction problems, there are generally two approaches: the monolithic approach, and the partitioned approach. The former treats the fluid and structure dynamics in the same mathematical framework and is solved simultaneously by a unified algorithm, and the latter treats the fluid and structure as two separate fields — which can be solved separately with their own meshes and numerical algorithms — and can be useful when using 'legacy' codes that have already been validated for solving many complicated problems [23].

In addition to FSI, we may use techniques from moving meshes to change how the nodes in our discretised domain move [4]. An Arbitrary Lagrangian-Eulerian (ALE) description may also be utilised to easily map between the material, spatial, and reference frames of reference when quantities are evolving in moving mesh methods [16].

Processes which occur on the microscopic scale in a physical system can have macroscopic effects on the larger scale behaviour we want to understand; multiscale modelling can be vital in solving such problems computationally primarily due to a scale gap [9] where, without the use of multiscale techniques, computational power may be vastly wasted. Multiscale problems are usually formulated by introducing a macroscale variable that is separated from the microscale variable by some small constant as demonstrated in [32]; O'Dea et al. continue to show how multiscale modelling is used in a biological setting to inform the dynamics of macroscale tissue growth from the microscale structure of biological tissue, and also detail how they used a Darcy-type equation governing flow through a porous medium.

The structure for the remainder of the report is as follows. In Section 1.2 we will introduce the idea of Galerkin methods, and in Section 1.3 some notes on what may be useful modelling-wise. In Section 1.4 we will fully derive both a continuous and discontinuous Galerkin finite element method discretisation for Poisson's equation, and we will further note the discretisations for Stokes flow; we will end this section by presenting some specific a priori and a posteriori error bounds and discuss how these may be utilised with smoothness indicators in *hp*-adaptivity. In Section 1.6 we briefly outline how a typical OOP FEM code may be structured along with a discussion of some key features, and then conclude the report with some numerical experiments — including simulations in 1D and 2D, simulations of Poisson's equation and Stokes flow, and using continuous and discontinuous Galerkin FEMs. Section 2 contains not-so-academic details about future targets for this project.

1.2 Continuous and Discontinuous Galerkin Finite Element Methods

Finite element methods first appeared with Courant in 1942 where they used the idea of minimising a functional using linear approximations over subregions, which is fundamental in the form required for deriving a finite element method; it wasn't until Clough in 1960 when the method was given its name, describing how the methods were used for work on flexibility in wing structures at the Boeing Company [22]. Gupta and Meek in [22] also credit the initial development of FEMs to papers by Argyris, Turner et al., and Zienkiewicz.

Since FEMs' initial work, many different flavours of FEM have been developed including CGFEMs and DGFEMs as described in Section 1.1, but also methods such as extended finite element methods (XFEMs), cut finite element methods (CutFEMs), and *hpk*-FEMs — some of which can be combined to make very powerful methods. All methods have in common that they roughly try to discretise the domain and solve many sub-problems using some discretised variational form of the original problem. In this report we consider the crossover between CGFEMs and DGFEMs with *hp*-FEMs.

Although we may ultimately end up using the DGFEM formulation for simulations, it is still useful to study the CGFEM formulation: firstly because the derivation is somewhat easier in that we do not have to derive extra terms for inter-element face contributions; secondly because boundary conditions can be applied strongly, which can simplify both the mathematical derivation and the programming implementation; and thirdly because the literature for CGFEM is more expansive. However we note that CGFEM can exhibit numerical instabilities in the form of spurious oscillations in the vicinity of sharp features of the analytical solution [10]. On the other hand DGFEMs can be very useful in solving on complicated meshes with for instance hanging nodes, as well as their close relationship with FVMs in CFD.

1.3 Some PDE Problems

The incompressible Navier-Stokes equations are currently the best-known model of time-dependent, viscous, incompressible, Newtonian fluid flow. We want to find the velocity, u, and pressure, p, subject to:

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho\nabla(\mathbf{u}\otimes\mathbf{u}) - \nabla\cdot\underline{\underline{\sigma}} = \rho\mathbf{f}, \qquad \text{in } \Omega, \tag{1a}$$

$$\nabla\cdot\mathbf{u} = 0, \qquad \text{in } \Omega, \tag{1b}$$

$$\mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^\top) - p\underline{\underline{I}} = \underline{\underline{\sigma}}, \qquad \text{in } \Omega, \tag{1c}$$

$$\mathbf{u} = \mathbf{g}_D, \qquad \text{on } \Gamma_D \neq \{\Omega, \emptyset\}, \tag{1d}$$

$$\mu\nabla\mathbf{u} : \mathbf{n} - p\mathbf{n} = \mathbf{0}, \qquad \text{on } \Gamma_N = \Gamma_D \setminus \Gamma_D, \tag{1e}$$

where ρ is the constant density of the fluid, μ is the dynamic viscosity, $\mathbf{g}_D$ is some Dirichlet boundary condition, and $\mathbf{f}$ is some body force acting on the fluid.

The details are omitted here, but in the limit of $\rho u L \ll \mu$, where L is some characteristic lengthscale, one obtains Stokes flow, which can be used to describe so-called creeping flow. Stokes flow has successfully been used to model, among other things, the flow of blood in the IVS at the pore scale [28]. For this project we intend to use Stokes flow as a stepping stone in which to ultimately study other models of flow in the IVS.

Stokes flow implicitly contains a $\nabla^2\mathbf{u}$ diffusion term (arising from substituting the stress tensor, $\underline{\underline{\sigma}}$, in Equation (1c)). It was therefore useful to study the Poisson equation as an initial problem as a stepping stone to study Stokes flow. On a separate topic, there are also occasions where the Poisson equation itself my be directly used later in the project: for example in the mesh-smoothing techniques used in ALE [16].

1.4 PDE Discretisation

For this discretisation section, we first make some definitions. Firstly, the domain in which we will seek solutions will be denoted $\Omega \subset \mathbb{R}^d$ for $d = 1, 2, 3$. Secondly, we will denote $\Gamma := \partial\Omega$ as the boundary of Ω, and we further split this boundary into Γ_D and Γ_N to represent the parts of the boundary with Dirichlet and Neumann boundary data, respectively; we note that $\Gamma = \Gamma_D \cup \Gamma_N$ and $\emptyset = \Gamma_D \cap \Gamma_N$. Finally, we define $\mathbf{n}$ to be the standard outward normal to a given surface, S.

1.4.1 Discretising the Poisson Problem

In this section we consider the Poisson problem in the open and bounded domain $\Omega \subset \mathbb{R}^d$, for $d = 1, 2, 3$. For this problem, we have: find $u(\mathbf{x})$ such that

$$-\nabla^2 u = f \text{ in } \Omega, \tag{2a}$$

$$u = g_D \text{ on } \Gamma_D \neq \emptyset, \tag{2b}$$

$$\nabla u \cdot \mathbf{n} = g_N \text{ on } \Gamma_N := \partial\Omega \setminus \Gamma_D, \tag{2c}$$

where f, g_D, g_N are functions of the spatial coordinates, and n is the usual outward normal on Γ_N.

From Equations (2) we perform the standard technique in finite element methods of multiplying Equation (2a) by a test function, v, that does not necessarily live in the same space as the solution, u, and integrate over the domain. This gives us

$$-\int_\Omega \nabla^2 uv \,\mathrm{d}x = \int_\Omega fv \,\mathrm{d}x.$$

Recall integration by parts in multiple dimensions (or rather Green's first identity) as

$$\int_\Omega \nabla^2 uv \,\mathrm{d}x = \int_\Gamma v\nabla u \cdot n \,\mathrm{d}s - \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x \tag{3}$$

where we have assumed that u and v are absolutely continuous. The integration by parts formula in Equation (3) allows us to write the second-order term as

$$\int_{\Gamma_D} v\nabla u \cdot n \,\mathrm{d}s + \int_{\Gamma_N} g_N v \,\mathrm{d}s - \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x, \tag{4}$$

where we have also split up the boundary integrals over Γ into Γ_D and Γ_N for the Dirichlet and Neumann segments respectively, and substituted the Neumann boundary data if $\Gamma_N \neq \emptyset$.

We now introduce a function space in which we seek our solution. Here we strongly imposed the boundary conditions, and therefore define the space we seek solutions within as

$$V := \{v \in H^1(\Omega) : v|_{\Gamma_D} = g_D\}.$$

We also further introduce our test space to be a similar space, but instead with zero Dirichlet boundary conditions:

$$V^0 := \{v \in H^1(\Omega) : v|_{\Gamma_D} = 0\},$$

which ultimately makes the first term in Equation (4) vanish when $v \in V^0$.

Our original Poisson problem in Equations (2) can therefore be written equivalently in a weak form: find $u \in V$ such that

$$\int_{\Gamma_N} g_N \,\mathrm{d}s + \int_\Omega \nabla u \cdot \nabla v \,\mathrm{d}x = \int_\Omega fv \,\mathrm{d}x \tag{5}$$

for all $v \in V^0$. Equivalence of both the original strong formulation and the weak formulation can be shown using an argument similar to that presented in [29].

We note that the problem presented in Equation 5 is still infinite-dimensional, which is not usable for solving numerically on a computer. Therefore we first discretise our domain into a mesh, which consists of non-overlapping elements (smaller shapes within the domain) for which the union of all elements is the entire domain[1]. Depending upon how the domain is created, these shapes may be restricted to triangles or squares in 2D, or tetrahedra or cubes in 3D, or may permit general polygons or polyhedra. We denote our mesh as $\mathcal{T}_h$, which is made up of many elements $\kappa \in \mathcal{T}_h$. With correct

[1] This assumes that the boundary of the domain can be constructed by piecewise straight lines, although this can be somewhat relaxed with more effort.

handling, these elements can be general polygons; however, for our simulations in this report, all simulations are performed either with intervals, or regular squares.

We define the following finite-dimensional space, in which we will ultimately seek solutions:

$$V_h := \{v \in V : v|_\kappa \in \mathcal{P}_p(\kappa), \kappa \in \mathcal{T}_h \setminus \Gamma_D\}, \tag{6}$$

where $\mathcal{P}_p(\kappa)$ is the set of polynomials of at most degree p on element κ. We similarly define our test space to be

$$V_h^0 := \{v \in V^0 : v|_\kappa \in \mathcal{P}_p(\kappa), \kappa \in \mathcal{T}_h \setminus \Gamma_D\}. \tag{7}$$

Our finite-dimensional problem, which we will refer to as the CGFEM formulation, is therefore the following: find $u_h \in V_h$ such that

$$\int_{\Gamma_N} g_N \,\mathrm{d}s + \int_\Omega \nabla u_h \cdot \nabla v_h \,\mathrm{d}x = \int_\Omega f v_h \,\mathrm{d}x, \tag{8}$$

for all $v \in V_h^0$. We stress here that the formulation is now finite-dimensional and can be approximated using a computer.

As discussed in Section 1.1, DGFEMs exist and do not have the rigid assumption that interfaces between element faces are continuous: they may instead be discontinuous. To derive a DGFEM formulation for Equations 2, the first thing we do is to write the problem as a system of first-order equations:

$$-\nabla \cdot \mathbf{s} = f \text{ in } \Omega, \tag{9a}$$

$$\mathbf{s} = \nabla u \text{ in } \Omega, \tag{9b}$$

$$u = g_D \text{ on } \Gamma_D \neq \emptyset, \tag{9c}$$

$$\nabla u \cdot n = g_N \text{ on } \Gamma_N := \partial\Omega \setminus \Gamma_D, \tag{9d}$$

where we have introduced a new variable, $\mathbf{s}$. We multiply Equations (9a) and (9b) by two test functions, v and $\boldsymbol{\tau}$, and integrate over each element in the mesh, $\kappa \in \mathcal{T}_h$, to obtain our weak formulation:

$$\int_\kappa \mathbf{s} \cdot \boldsymbol{\tau} \,\mathrm{d}x = -\int_\kappa u \nabla \cdot \boldsymbol{\tau} \,\mathrm{d}x + \int_{\partial\kappa} u \mathbf{n}_\kappa \cdot \boldsymbol{\tau} \,\mathrm{d}s, \tag{10a}$$

$$\int_\kappa \mathbf{s} \cdot \nabla v \,\mathrm{d}x = \int_\kappa f v \,\mathrm{d}x + \int_{\partial\kappa} \mathbf{s} \cdot \mathbf{n}_\kappa v \,\mathrm{d}s. \tag{10b}$$

For each element in our mesh, $\kappa \in \mathcal{T}_h$, we introduce some new discontinuous Galerkin vector spaces for some maximum polynomial degree, p, in which we seek our solutions:

$$V_h := \{v \in L^2(\Omega) : v|_\kappa \in \mathcal{P}_p(\kappa), \kappa \in \mathcal{T}_h \setminus \Gamma_D\},$$
$$\Sigma_h := \{\boldsymbol{\tau} \in [L^2(\Omega)]^2 : \boldsymbol{\tau}|_\kappa \in \mathcal{P}_p(\kappa), \kappa \in \mathcal{T}_h \setminus \Gamma_D\}.$$

The essence of creating a DGFEM is in choosing an approximation of the solution in the face terms of the weak formulation, for which there are many choices (see [2]). The most general DGFEM formulation is then: find $u_h \in V_h$ and $\mathbf{s}_h \in \Sigma_h$ such that for all $\kappa \in \mathcal{T}_h$ we have

$$\int_\kappa \mathbf{s}_h \cdot \boldsymbol{\tau}_h \,\mathrm{d}x = -\int_\kappa u_h \nabla \cdot \boldsymbol{\tau}_h \,\mathrm{d}x + \int_{\partial\kappa} \hat{u}_\kappa \mathbf{n}_\kappa \cdot \boldsymbol{\tau}_h \,\mathrm{d}s, \tag{11a}$$

$$\int_\kappa \mathbf{s}_h \cdot \nabla v_h \,\mathrm{d}x = \int_\kappa f v_h \,\mathrm{d}x + \int_{\partial\kappa} \hat{\mathbf{s}}_\kappa \cdot \mathbf{n}_\kappa v_h \,\mathrm{d}s, \tag{11b}$$

for all $v_h \in V_h$ and $\boldsymbol{\tau}_h \in \Sigma_h$ where the written numerical fluxes $\hat{u}_\kappa$ and $\hat{\mathbf{s}}_\kappa$ are approximations to u and s, respectively, on the boundary of each element, κ.

We now introduce two operators that are defined on element faces: the average operator and the jump operator, as given in [10]. Let u be a vector-valued function and p be a scalar-valued function.

The average operator, which is defined on the face between two elements κ_i and κ_j, denoted as $F \in \mathcal{F}^{\mathcal{I}}$, where $\mathcal{F}^{\mathcal{I}}$ denotes the set of interior faces between elements in $\mathcal{T}_h$, is given by

$$\begin{aligned}\{\!\{\mathbf{u}\}\!\} &:= \frac{1}{2}(\mathbf{u}^+_{\kappa_i} + \mathbf{u}^+_{\kappa_j}),\\ \{\!\{p\}\!\} &:= \frac{1}{2}(p^+_{\kappa_i} + p^+_{\kappa_j}),\end{aligned}$$

where $\mathbf{u}^+_{\kappa_i}$ and $p^+_{\kappa_i}$ denote the trace values of $\mathbf{u}$ and p on element κ_i. We may similarly introduce the jump operator as

$$\begin{aligned}[\![\mathbf{u}]\!] &:= \mathbf{u}^+_{\kappa_i} \cdot \mathbf{n}_{\kappa_i} + \mathbf{u}^+_{\kappa_j} \cdot \mathbf{n}_{\kappa_j},\\ [\![p]\!] &:= p^+_{\kappa_i}\mathbf{n}_{\kappa_i} + p^+_{\kappa_j}\mathbf{n}_{\kappa_j},\end{aligned}$$

where $\mathbf{n}_{\kappa_i}$ and $\mathbf{n}_{\kappa_j}$ denote the outward normals to κ_i and κ_j, respectively, on F. In a similar way, we may define $\mathcal{F}^{\mathcal{B}}$ to be the faces of elements in $\mathcal{T}_h$ which lie on the boundary of the domain, such that the faces of $\mathcal{T}_h$, $\mathcal{F}$, satisfy $\mathcal{F} = \mathcal{F}^{\mathcal{I}} \cup \mathcal{F}^{\mathcal{B}}$ and $\mathcal{F}^{\mathcal{I}} \cap \mathcal{F}^{\mathcal{B}} = \emptyset$. The average and jump operators on these boundary faces are defined as

$$\begin{aligned}\{\!\{\mathbf{u}\}\!\} &:= \mathbf{u}^+_{\kappa_i},\\ \{\!\{p\}\!\} &:= p^+_{\kappa_i},\\ [\![\mathbf{u}]\!] &:= \mathbf{u}^+_{\kappa_i} \cdot \mathbf{n}_{\kappa_i},\\ [\![p]\!] &:= p^+_{\kappa_i}\mathbf{n}_{\kappa_i}.\end{aligned}$$

For the remainder of this report, we will only consider the interior penalty method originally suggested by Douglas and Dupont in [17], and specifically we will consider only the symmetric case. This amounts to choosing

$$\begin{aligned}\hat{u}_\kappa &:= \begin{cases}\{\!\{u_h\}\!\} & \text{on } F \in \mathcal{F}^{\mathcal{I}},\\ g_D & \text{on } F \in \mathcal{F}^{\mathcal{B}},\end{cases}\\ \hat{\mathbf{s}}_\kappa &:= \begin{cases}\{\!\{\nabla u_h\}\!\} - \sigma([\![u_h]\!]) & \text{on } F \in \mathcal{F}^{\mathcal{I}},\\ \nabla u_h - \sigma(u_h - g_D)\mathbf{n}_{\kappa_i} & \text{on } F \in \mathcal{F}^{\mathcal{B}},\end{cases}\end{aligned}$$

where σ is a penalisation parameter specific to the Poisson problem that scales like $\mathcal{O}(p^2/h)$.

Once we substitute $\mathbf{s}_h = \nabla u_h$ from above and add Equations (11), we recover the DGFEM formulation: Find $u_h \in V_h$ such that:

$$\begin{aligned}\int_\Omega \nabla u_h \cdot \nabla v_h \,\mathrm{d}x - \int_{\mathcal{F}} (\{\!\{\nabla u_h\}\!\} \cdot [\![v_h]\!] + \{\!\{\nabla v_h\}\!\} \cdot [\![u_h]\!] - \sigma [\![u_h]\!] \cdot [\![v_h]\!])\,\mathrm{d}s\\ = \int_\Omega f v_h \,\mathrm{d}x + \int_{\mathcal{F}^{\mathcal{B}}} g_D(\sigma v_h - \nabla v_h \cdot \mathbf{n})\,\mathrm{d}s,\end{aligned}$$

for all $v_h \in V_h$, where we also note that here that derivatives denoted by ∇ denote the broken gradient (N.B. [10]).

1.4.2 Discretising Stokes Flow

We now consider Stokes Flow in the open and bounded domain $\Omega \subset \mathbb{R}^d$, for $d = 2, 3$. For this problem, we have: find $\mathbf{u}$ and p such that

$$-\mu\nabla^2\mathbf{u} + \nabla p = \mathbf{f} \text{ in } \Omega, \tag{12a}$$

$$\nabla \cdot \mathbf{u} = 0 \text{ in } \Omega, \tag{12b}$$

$$\mathbf{u} = \mathbf{g_D} \text{ on } \Gamma_D \neq \emptyset, \tag{12c}$$

$$\nabla\mathbf{u} \cdot \mathbf{n} = \mathbf{g_N} \text{ on } \Gamma_N := \partial\Omega \setminus \Gamma_D, \tag{12d}$$

where $\mathbf{f}, \mathbf{g_D}, \mathbf{g_N}$ are functions of the spatial coordinates, μ is the dynamic viscosity, and $\mathbf{n}$ is the usual outward normal on Γ_N.

We omit the full details of the derivation of the discretisation, but note the full details are outlined in [29]; we just note the discretisation here for use with the final simulations in Section 1.6.2. We first define

$$A(\mathbf{u}, \mathbf{v}) := \mu \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} \, \mathrm{d}\mathbf{x}, \tag{13}$$

$$B(\mathbf{v}, p) := \int_\Omega p \nabla \cdot \mathbf{v} \, \mathrm{d}\mathbf{x}, \tag{14}$$

$$\mathcal{L}(\mathbf{v}) := \int_\Omega f \mathbf{v} \, \mathrm{d}\mathbf{x}, \tag{15}$$

and subsequently define

$$\mathcal{A}((\mathbf{u}, p), (\mathbf{v}, q)) := A(\mathbf{u}, \mathbf{v}) - B(\mathbf{v}, p) + B(\mathbf{u}, q). \tag{16}$$

We also note here the finite element spaces in which we seek our solutions, given for the velocity and pressure, respectively given by:

$$\mathbf{V}_h := \{\mathbf{v} \in L^2(\Omega)^d : \mathbf{v}|_\kappa \in \mathcal{P}_p(\kappa)^d, \kappa \in \mathcal{T}_h\}, \tag{17}$$

$$Q_h := \{q \in L^2(\Omega) : q|_\kappa \in \mathcal{P}_{p-1}(\kappa), \kappa \in \mathcal{T}_h\}, \tag{18}$$

where we note that the polynomial degree of the pressure space is chosen at one lower than the velocity space in order to ensure that the Babuška and Brezzi inf-sup constant is not zero [20].

Using similar arguments to those in Section 1.4.1, we may now easily give the continuous Galerkin discretisation of Stokes in Equation 12 to be: find $(\mathbf{u_h}, p_h) \in \mathbf{V}_h \times Q_h$ such that

$$\mathcal{A}((\mathbf{u}_h, p_h), (\mathbf{v}_h, q_h)) = \mathcal{L}(\mathbf{v}_h), \tag{19}$$

for all $(\mathbf{v}_h, q_h) \in \mathbf{V}_h \times Q_h$.

For the DGFEM formulation of the Stokes problem, as stated in [24], we redefine $A(\mathbf{u}, \mathbf{v})$ and $B(\mathbf{v}, p)$ to

$$\tilde{A}(\mathbf{u}, \mathbf{v}) := \mu \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} \, \mathrm{d}\mathbf{x} - \int_\mathcal{F} (\{\!\{\mu \nabla \mathbf{v}\}\!\} : [\![\mathbf{u}]\!] + \{\!\{\mu \nabla \mathbf{u}\}\!\} : [\![\mathbf{v}]\!]) \, \mathrm{d}s + \mu \int_\mathcal{F} \gamma h^{-1} [\![\mathbf{u}]\!] : [\![\mathbf{v}]\!] \, \mathrm{d}\mathbf{x},$$

$$\tilde{B}(\mathbf{v}, q) := -\int_\Omega q \nabla \cdot \mathbf{v} \, \mathrm{d}\mathbf{x} + \int_F \{\!\{q\}\!\} [\![\mathbf{v}]\!] \, \mathrm{d}s,$$

where we note that the jump operators on vectors, such as $[\![\mathbf{u}]\!]$, are implicitly two-dimensional.

Just like in the continuous version, we can add these equations together to give a single equation, so we can therefore define $\tilde{\mathcal{A}}((\mathbf{u}, p), (\mathbf{v}, q))$ again as

$$\tilde{\mathcal{A}}((\mathbf{u}, p), (\mathbf{v}, q)) := \tilde{A}(\mathbf{u}, \mathbf{v}) - \tilde{B}(\mathbf{v}, p) + \tilde{B}(\mathbf{u}, q), \tag{20}$$

and we therefore have our full DGFEM formulation: find $(\mathbf{u}_h, p_h) \in \mathbf{V}_h \times Q_h$ such that

$$\tilde{\mathcal{A}}((\mathbf{u}_h, p_h), (\mathbf{v}_h, q_h)) = \mathcal{L}(\mathbf{v}_h), \tag{21}$$

for all $(\mathbf{v}_h, q_h) \in \mathbf{V}_h \times Q_h$.

1.5 Error Control and hp-adaptivity

A priori error bounds for specific model problems when solved with FEMs are known (e.g. [1][11][21]) and take the form

$$\|u - u_h\| \leq \mathcal{E}_1(h, p, u),$$

where u is the exact solution, u_h is the FEM approximation of the solution, and $\mathcal{E}_1$ is some function of h, p, and u. Such a bound is given for 1D CGFEM in [35]:

$$\mathcal{E}_1(h,p,u) = C\frac{h^{\min(p,k)}}{p^k}\|u\|_{H^k(0,1)}, \tag{22}$$

where $C > 0$ is some constant, k is an integer that roughly relates to the smoothness of u [2], and we note that although the bound is on the interval $(0,1)$ this can be easily mapped to any other interval in 1D.

Throughout this report we will only consider DEs which we know the solution to, which therefore allows us to compute exact a priori bounds if wanted. However in practice this is of course not the case and, although it gives us an insight into how our errors may converge when refining, it is of little practical use. We therefore need some other robust bounds to control the errors induced due to our FEM approximation.

Opposed to a prior error bounds, a posteriori error bounds exist (e.g. [24][31][39]) and take the form

$$\|u - u_h\| \leq \mathcal{E}_2(h,p,u_h),$$

where, crucially, $\mathcal{E}_2$ depends upon the approximate solution, u_h, *not* the unknown exact solution, u. Not only does this give us access to bounds on our error from the true solution without ever knowing the true solution, it also paves the way for automatic local hp-adaptivity. The forms of these a posteriori error bounds can ultimately be quite complicated; we therefore present in this section the idea of how they are used in 1D and present references for use in higher dimensions.

hp-adaptivity aims to combine the benefits of exponential convergence rates of p-adaptivity and the benefits of increasing local resolution around non-smooth areas of the domain with h-adaptivity; p-adaptivity involves enriching the polynomial degree locally in areas of high estimated error, and h-adaptivity involves subdividing elements in areas of high estimated error. Two key ingredients are generally required for such an approach:

1. Element-wise error indicators: to tell us where the approximation is worst;
2. A smoothness indicator: to tell us where the approximation is most "smooth".

For the first key ingredient, in 1D CGFEMs for the Poisson equation, we may easily derive an a posteriori error bound as shown in [6]. The Poisson problem we will consider until the end of this short section takes the form

$$-\epsilon u''(x) + c(x)u(x) = f(x), \text{ on } x \in \Omega, \tag{23}$$

with $\epsilon = 1$, $c \equiv 0$, $f(x) = \pi\cos(\pi x)$, and vanishing Dirichlet boundary conditions. We can derive a residual-based error bound and is written as

$$\|u - u_h\|_E \leq \sqrt{\sum_{i=1}^{N} \frac{1}{\epsilon p_i(p_i+1)}\left\|w_i^{1/2}R(u_h)\right\|^2_{L^2(x_{i-1},x_i)}}, \tag{24}$$

where the residual is defined as $R(u_h)|_{x_{i-1},x_i} := f + \epsilon u'' - cu, \forall i$, p_i is the polynomial degree on the ith element, $w_i := (x_i - x)(x - x_{i-1})$, and the energy norm is defined as the usual bilinear form $\|u\|_E := \epsilon\|u'\|_{L^2(\Omega)} + \|\sqrt{c}u\|_{L^2(\Omega)}$. We note the bound is very similar to [40] except Wihler does not assume that the projection of f to be equal to the function itself.

By taking the term within the sum of Equation (24),

$$\eta_{\kappa_i}(u_h) := \frac{1}{\sqrt{\epsilon p_i(p_i+1)}}\left\|w_i^{1/2}R(u_h)\right\|_{L^2(x_{i-1},x_i)}, \tag{25}$$

[2] k corresponds to the $H^k(0,1)$ space that u lives in, so higher k is a smoother function and lower k is a less-smooth function. Note that we need sufficient smoothness for higher p to give us any 'benefit' in the numerator.

we have what is known as a local error indicator; whilst we cannot say that η_{κ_i} is an upper bound of the error on an individual element, κ_i, we can use the indicator to approximately calculate where the error in the solution is largest. We may then use adaptive strategies to refine locally depending upon the size of the indicator in certain locations on the mesh.

For 1D DGFEMs, there exists a similar residual-based error bound as noted in [8]:

$$\eta_{\kappa_i}(u_h) := h_{\kappa_i}^2 \|R(u_h)\|_{L^2(\kappa_i)} + h_\kappa \|J(u_h)\|_{L^2(\partial\kappa_i)}, \tag{26}$$

where $J(u_h) := [\![\epsilon u_h']\!]$ denotes the so-called jump residual between κ_i and its surrounding elements. This bound can be used again as an error indicator as suggested above. In fact, you can give this a try yourself with an interactive example using Blakey FEM: `https://fem.blakey.family/`. The code at this URL performs 5 steps of *hp*-adaptivity with a DGFEM in 1D for given ϵ, $c(x)$, $f(x)$, and a starting number of elements (N).

Similar a posteriori error bounds and resulting local error indicators can be generalised to higher dimensions for various model problems (for example [8][26][36]).

The second key ingredient for automatic *hp*-adaptivity is a smoothness indicator, which tells us whether to refine h or p in a given adaptive step. In 1D we can refer once again to [40] for the smoothness indicator given in Equation (27), which is close to 1 when u_h is smooth and is close to 0 when u is non-smooth. We note that smoothness indicators may take many forms and can be derived for 1D and higher [25].

$$\mathcal{F}_K[u_h] := \begin{cases} \|u_h\|_{\infty(\kappa)}^2 \left[\coth(1)\left(h_\kappa^{-1}\|u_h\|_{L^2(\kappa)}^2 + h_\kappa\,|u_h|_{H^1(\kappa)}^2\right)\right]^{-1} & \text{if } u_h \not\equiv 0, \\ 1 & \text{if } u_h \equiv 0. \end{cases} \tag{27}$$

hp-adaptivity will indeed help in this project's work by sparing computational resources in areas where it is not needed. As shown in the following section, the degrees of freedom of the computations (directly related to how long a computation will take) can be greatly reduced by using *hp*-adaptivity.

1.6 Numerics

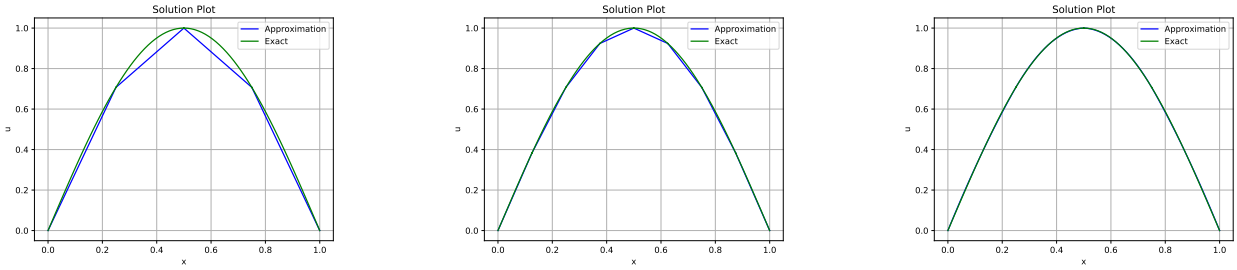
In this section we present some simple numerics with various boundary conditions. We then use a CGFEM and DGFEM method for a Poisson problem, as well as CGFEM for a Stokes problem, in a mixture of 1D and 2D domains.

1.6.1 Design of an OOP Code

Many readily-available FEM solvers already exist such as deal.II, FEniCS, and DUNE. There is no single FEM solver package that can perform everything that the user may want: for example deal.II can perform h, p, and *hp*-adaptivity whilst FEniCs can only perform h-adaptivity. However deal.II cannot use elements that are not intervals, squares, or cubes; in contrast, FEniCS can have triangular and tetrahedral elements. This is just due to the priorities of the problems the authors of these packages wanted to solve. We therefore have opted to use an in-house developed software package called AptoFEM for this project, which allows us to have an in-depth understanding of the structure of the software and easily add features ourselves if the needed arises.

As part of Blakey's Master's dissertation, there was a simple solver created — titled Blakey FEM — that can perform *hp*-adaptive simulations in 1D on a specific form of one linear and one nonlinear problem with CGFEM [6]. This code has since been further extended to include DGFEM [5], and has been ported from C++ to Fortran [7]. For the efficiency of an FEM solver it is important that at least core of an FEM code is written in a low level language such as Fortran or C++, since these methods can be very computationally demanding and ideally no resource would go wasted on higher-level languages such as Java or Delphi.

With FEMs being finite-dimensional, the solving of them amounts to solving a linear system. Therefore it is vital that accurate and efficient methods for solving linear systems is incorporated.



(a) *4 elements equal-sized with linear basis functions.*

(b) *8 elements equal-sized with linear basis functions.*

(c) *4 elements equal-sized with quadratic basis functions.*

Figure 2: *Comparison of three different simulations of a Poisson problem in 1D using Blakey FEM. The plots have been generated by sampling 10 linearly-spaced points per element and plotting using Matplotlib.*

Depending upon the sparsity of the resulting matrix different algorithms can be used such as the Thomas algorithm for solving tridiagonal matrices exactly, or GMRES for general matrices iteratively.

Another key component of FEM codes comes in the approximation of the integrals that arise from the variational form of the problem; in many FEM codes, including Blakey FEM and AptoFEM, the quadrature formula for approximating these integrals is Gauss-Legendre. Gauss-Legendre quadrature is the de facto quadrature formula because the quadrature is flexible enough to allow us to specify the order to which we want to solve (and therefore not wasting computational resources on simple problems), and also due to its optimality in the order of the polynomials that it can integrate exactly.

In terms of the design of an object-oriented programming (OOP) code, Blakey FEM makes extensive use of inheritance and polymorphism to make the code easily extensible and with minimal redundancy. Please see [6] for full details on how Blakey FEM is implementation.

1.6.2 Some Numerical Experiments

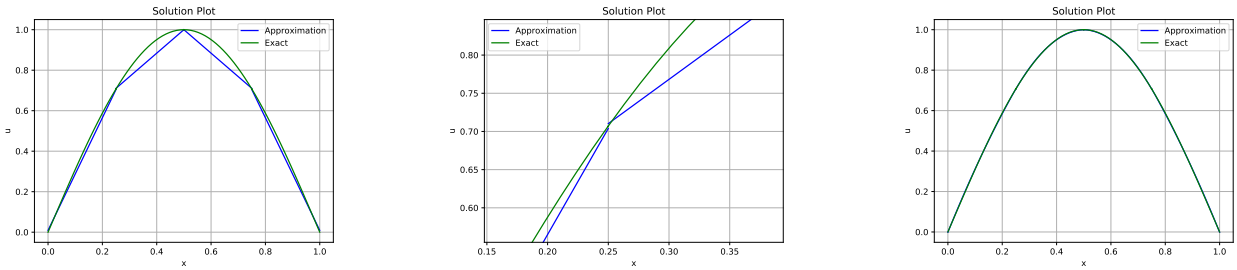
Here, we will present a short comparison of simulations with variable p in 1D with Blakey FEM and then move onto only comparing convergence rates in AptoFEM.

Figure 2 shows a comparison between three 1D Poisson problem simulations performed in Blakey FEM, where we have our forcing function $f = \pi \cos(\pi x)$, domain $[0, 1]$, and vanishing Dirichlet boundary conditions. In Figure 2a we have 4 equal-sized elements using linear ($p = 1$) basis functions which give us a crude approximation to the exact solution. We see that the approximation improves for Figures 2b and 2c but these are achieved in two very different ways: Figure 2b achieves this by doubling the number of elements (h-refinement), whereas Figure 2c achieves this by everywhere increasing the polynomial degree by 1 (p-refinement). It's important to note here that the degrees of freedom (DoF) for solving are roughly similar for these latter two figures, yet by the 'eyeball norm' the solution with quadratic basis functions approximates the solution better. However it is also important to note that a more accurate quadrature formula will need to be employed for basis functions with a higher degree, which itself will be more costly computationally.

We can also run similar simulations using DGFEM, as shown in Figure 3. We notice in particular from Figure 3b that the solution is not continuous between elements. Figure 4 shows further simulations in 2D using AptoFEM where the forcing function is now $f = 2\pi^2 \sin(\pi x) \sin(\pi y)$ and domain is $[0, 1]^2$ and we keep zero Dirichlet boundary conditions.

We may apply h-refinement globally on our mesh to get a better approximation of the solution. By the a priori bound given in Equation (22) we have a rough idea of how this may converge: for a 'smooth enough' function (k large) with linear approximating polynomials we would have convergence of the errors against DoFs that looks like $\mathcal{O}(h)$ in the energy norm. For higher-order polynomials we would essentially see order $\mathcal{O}(h^p)$. As shown in Figure 5 we see this convergence behaviour for $p = 1, 2, 3$ for the 1D Poisson problem using Blakey FEM; note that the estimator and error are indistinguishable from one another.

As discussed in [40][6] we can recover exponential convergence rates with p-refinement, provided that

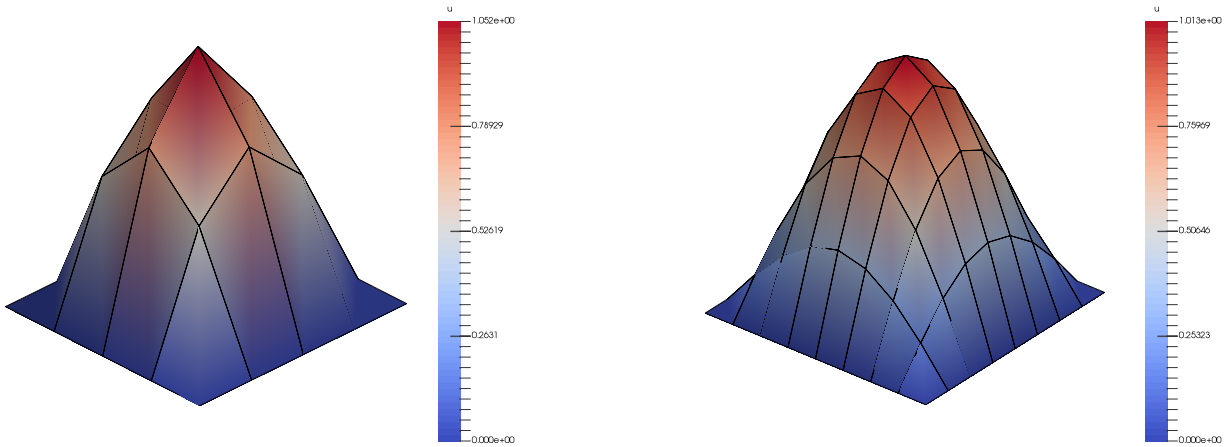


(a) *4 elements equal-sized with linear basis functions.*

(b) *Zoomed-in view of Figure 3a.*

(c) *4 elements equal-sized with quadratic basis functions.*

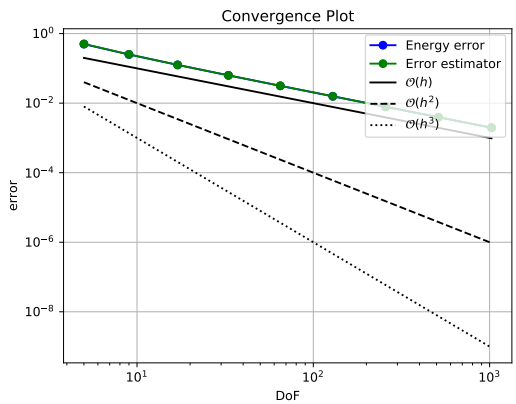
Figure 3: *Comparison of three different simulations of a Poisson problem in 1D using Blakey FEM. The plots have been generated by sampling 10 linearly-spaced points per element and plotting using Matplotlib.*



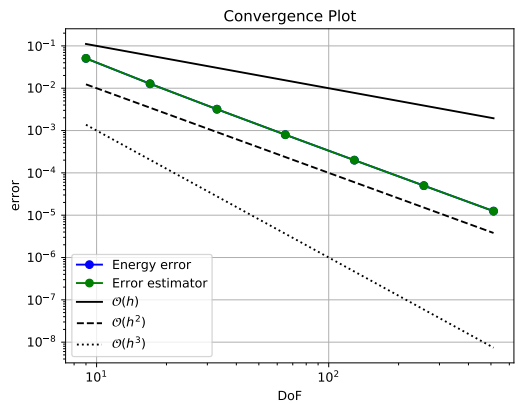
(a) *Poisson problem from Equation* (2) *with* $\times 4 \times 4 = 16$ *square elements, solved with the AptoFEM CGFEM solver with linear basis functions.*

(b) *Poisson problem from Equation* (2) *with* $\times 8 \times 8 = 64$ *square elements, solved with the AptoFEM CGFEM solver with linear basis functions.*

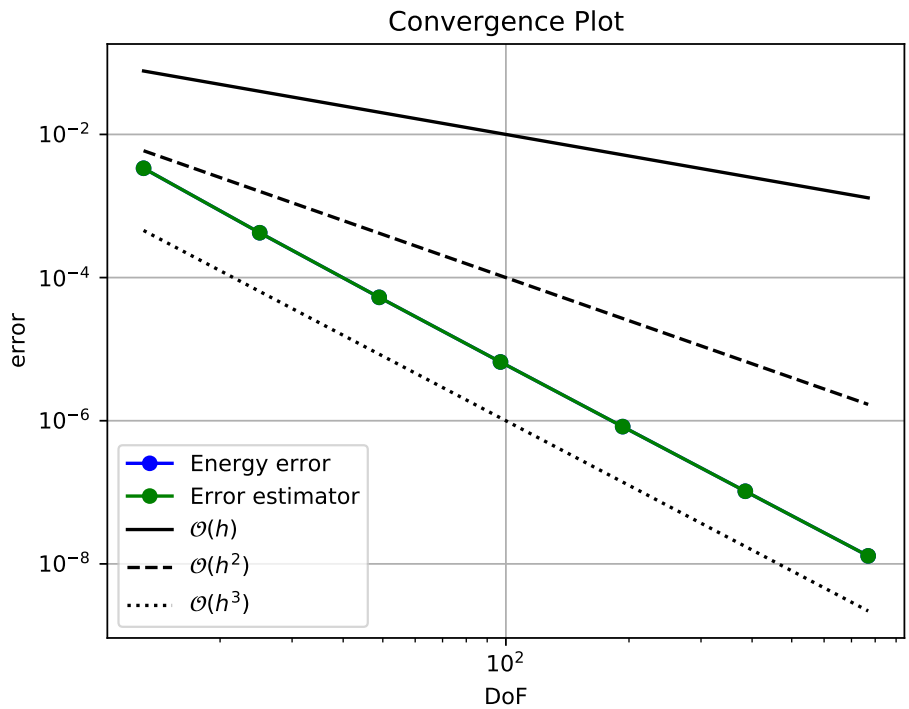
Figure 4: *Comparison of two different simulations of a Poisson problem in 2D using AptoFEM. The plots have been generated using ParaView.*



(a) *Global h-refinement with $p = 1$.*



(b) *Global h-refinement with $p = 2$.*



(c) *Global h-refinement with $p = 3$.*

Figure 5: *Convergence of Poisson problem from Equation (2) in 1D with CGFEM with varying p.*

the true solution is 'smooth enough' — this is defined more formally through the use of a smoothness indicator, as given in 1D earlier in Equation (27). There are problems that don't fit the criteria for being 'smooth enough' but can still recover these exponential convergence rates by using hp-refinement, where we refine in h when we aren't 'smooth enough', and otherwise refine in p. We note that these refinement steps do not have to be applied everywhere and can be performed element-wise. Loosely speaking, Blakey FEM's hp-adaptive code is created with the following steps, similarly to the steps outlined in [8]:

1. Solve
   - Solve the original FEM problem 'as usual' with a given mesh and polynomial degree;
2. Estimate
   - Estimate the error on every element using local error indicators, for example the indicators given in Equation (26);
3. Mark
   - Use some strategy to make those with a large enough error for refinement, for example by marking all those that are above 66.7% of the maximum estimated error;
4. Decide
   - Use some type of smoothness indicator to decide whether to h- or p-refine on each marked element, for example by using Equation (27) in 1D and p refining those elements for which the smoothness indicator is greater than 0.5;
5. Refine
   - Apply either h- or p-refinement to each marked element individually, depending upon what has been decided above.

Using this process, we have five refinement/adaptivity strategies: Global h-refinement, h-adaptivity, global p-refinement, p-adaptivity, and hp-adaptivity. We layout convergence plots for two contrasting problems: one for the Poisson problem in Equation (2) in 1D shown in Figure 7, and a boundary layer problem in 1D which yields a solution with a sharp gradient at the boundaries (see [40, Example 2]) in Figure 8. This problem solves the following differential equation:

$$-\epsilon u'' + u = 1, \text{ in } (0,1), \tag{28a}$$

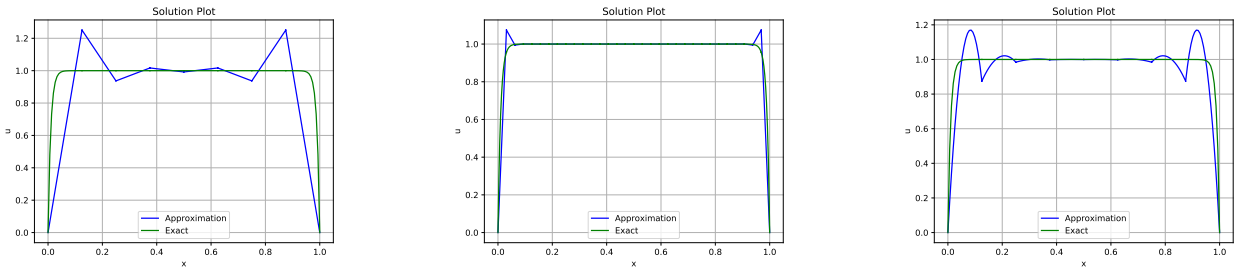
$$u = 0, \text{ on } \{0,1\}, \tag{28b}$$

where $\epsilon = 10^{-4}$ and permits exact solution

$$u(x) = -\frac{\exp(x/\sqrt{\epsilon})}{\exp(1/\sqrt{\epsilon})+1} - \frac{\exp(-x/\sqrt{\epsilon})\exp(1/\sqrt{\epsilon})}{\exp(1/\sqrt{\epsilon})+1} + 1. \tag{29}$$

Some example solutions of this boundary layer problem is shown in Figure 6. Both the Poisson and boundary layer problems are plotted with the exact error in the energy norm as well as the a posteriori bound found in Equation (24).

Figure 7 shows five plots for each of the refinement/adaptivity strategies outlined above. Although the plots for global h-refinement and h-adaptivity look very similar, we note that for h-adaptivity that steps are taken more cautiously: in that the convergence rates follow roughly the same trajectory, but the h-adaptivity plot increases by a small number of DoFs with each iteration. We actually see something very similar between the p-refinement and p-adaptivity in that with the same number degrees of freedom the error is roughly similar, although the p-adaptivity simulation takes more cautious steps. We further note that there is not much difference between the plot for p-adaptivity and for hp-adaptivity: in fact this is because the solution to this Poisson problem is very smooth, and the hp-adaptive algorithm — although not recorded here — actually opts to refine p at every step rather than h.



(a) *8 elements equal-sized with linear basis functions.*

(b) *32 elements equal-sized with linear basis functions.*

(c) *8 elements equal-sized with quadratic basis functions.*

Figure 6: *Comparison of three different simulations of the boundary layer problem in Equation* (28) *in 1D using Blakey FEM. The plots have been generated by sampling 10 linearly-spaced points per element and plotting using Matplotlib.*

Contrastingly, Figure 8 shows convergence plots for the 1D boundary layer problem given in Equation (28) with zero Dirichlet boundary conditions; this example is contrasting due to the sharp gradient of the solution near the boundaries, unlike the very smooth solution of the Poisson problem. Both the global h-refinement and h-adaptivity begin with the error indicator not being tight on the actual error for earlier simulations, but they soon become tight. The actual convergence of the two is very different: for 20 DoF the error is about 10 times smaller for h-adaptivity — this is likely because the h-refinement will be taking place near the boundaries where the refinement is needed, and not wasting resources refining in the centre of the domain. The global p-refinement and p-adaptivity give similar results, but does appear to give exponential convergence. The hp-adaptivity strategy is the best strategy here: although plotted for a higher number of DoFs, we note that around 20 DoFs the error is about equal with that of the h-adaptivity plot, and for around 40 DoFs performs about 10 times better than p-refinement and p-adaptivity. The convergence here is also very clearly exponential and converging faster than the already exponentially converging p-refinement could have achieved alone.

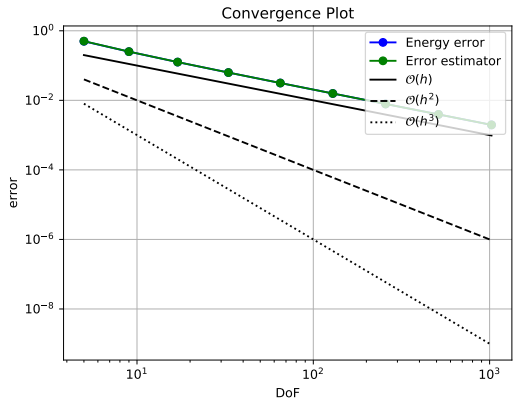
We now take a two-fold step from the CGFEM 1D Poisson problem: We can take a 2D Poisson problem from Equation (2) and can use AptoFEM with DGFEM to run simulations for each of the five refinement/adaptivity strategies to produce Figure 9. We note that there are some shared similarities with Figure 2, which is the 1D Poisson problem using CGFEM; in particular, the plots for the energy error and DG error for global h-refinement and h-adaptivity on both figures convergence with the same $\mathcal{O}(h)$, the difference being that the error estimator is not as tight in the DG case. Global p-refinement and p-adaptivity also give exponential convergence rates in both cases. The p-adaptive plot here seems to increase in error after around 40 DoFs — some investigation would need to be needed to determine the cause of this, as this is not yet at machine precision (about 10^{-16}) but it may be associated with that. Interestingly, the hp-adaptivity does not perform as well as may be expected in the 2D case; however this may not be quite as surprisingly as it first seems, as the error estimators and smoothness indicators are different between the 1D and 2D simulations. Perhaps for this problem there is a better set of indicators to do this process more efficiently.

We present some final simulations here for a Stokes problem from (12) in 2D. We choose to solve over an L-shaped domain in the region $[-1,1]^2 \setminus [0,1] \times [-1,0]$. We solve with $\mu = 1$ set up such that we have exact solution:

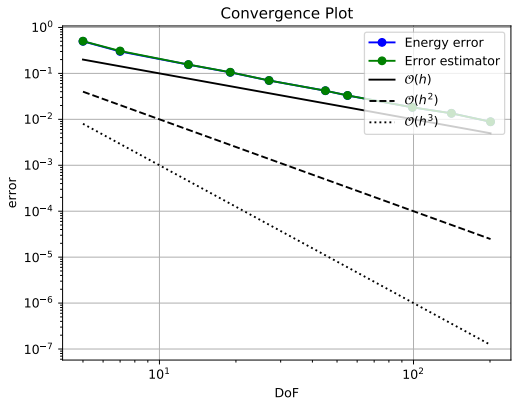
$$\mathbf{u} = \begin{bmatrix} -(y\cos(y) + \sin(y))e^x \\ y\sin(y)e^x \end{bmatrix}, \tag{30a}$$

$$p = 2e^x \sin(y). \tag{30b}$$

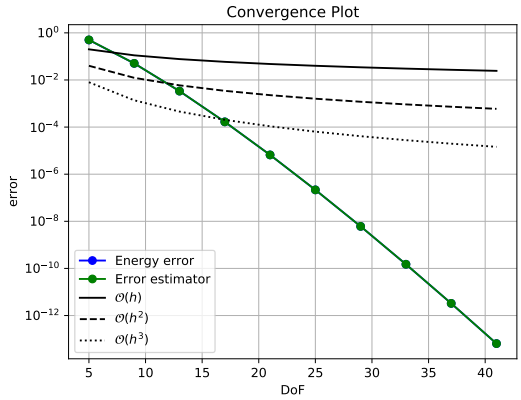
To keep the pressure uniquely defined we prescribe a Neumann boundary condition on the side with $x = 1$ and Dirichlet boundary conditions everywhere else. Figure 10 shows two different simulations for this Stokes problem with a different number of elements, where the contours on the mesh represent the values of the solution pressure, and the magnitude and direction of the glyphs represent the solution flow velocity.



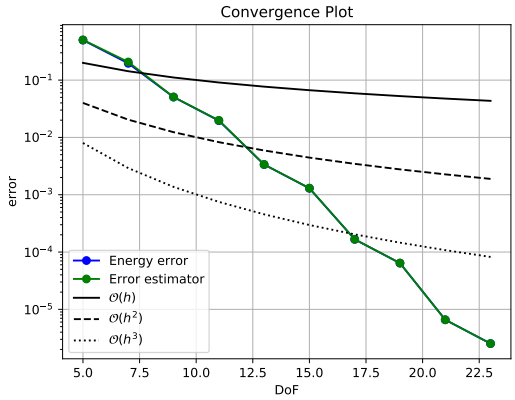
(a) *Global h-refinement.*



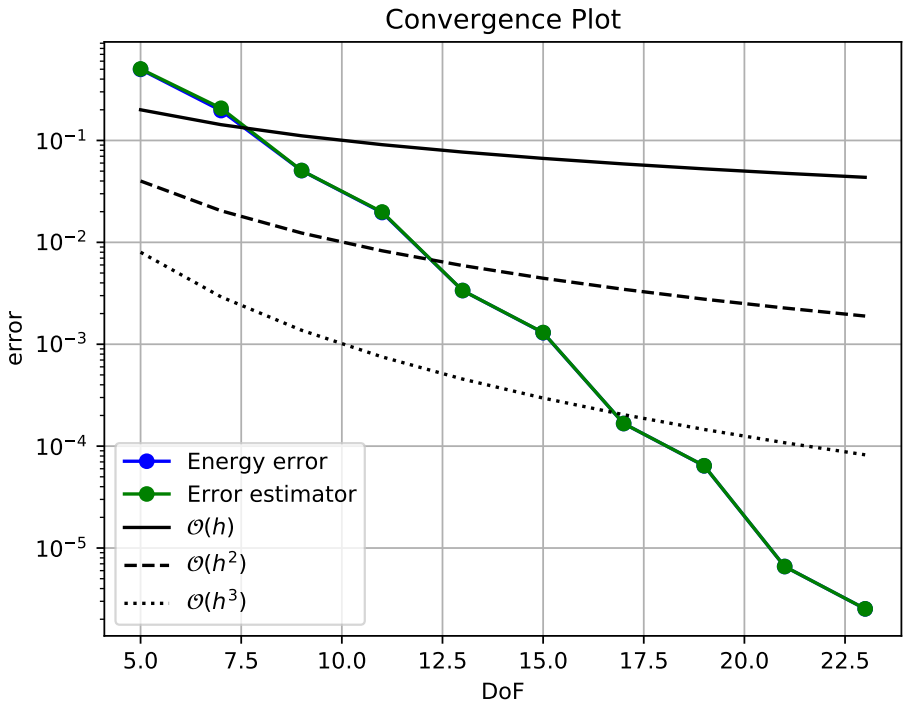
(b) *h-adaptivity.*



(c) *Global p-refinement.*

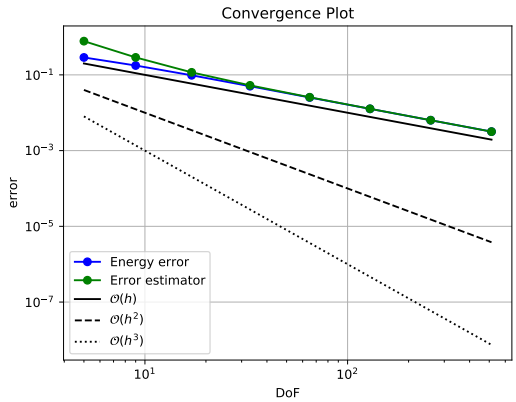


(d) *p-adaptivity.*

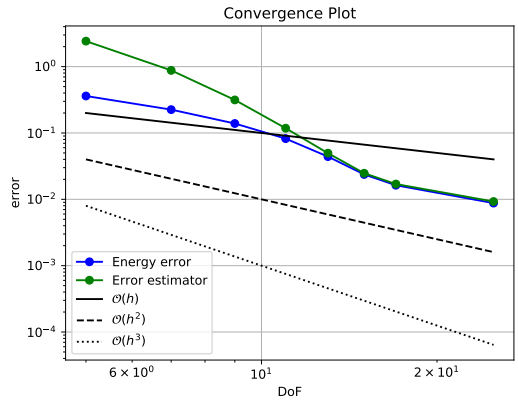


(e) *hp-adaptivity.*

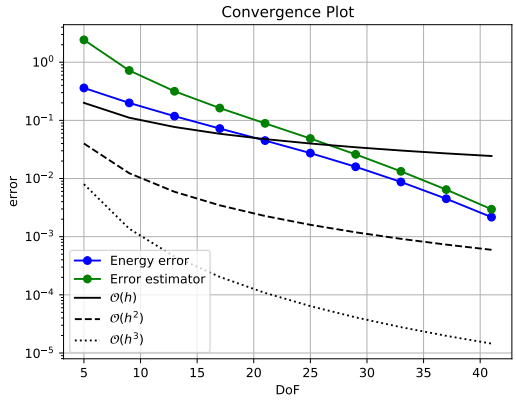
Figure 7: *Convergence of Poisson problem from Equation (2) in 1D using CGFEM with various refinement/adaptivity strategies.*



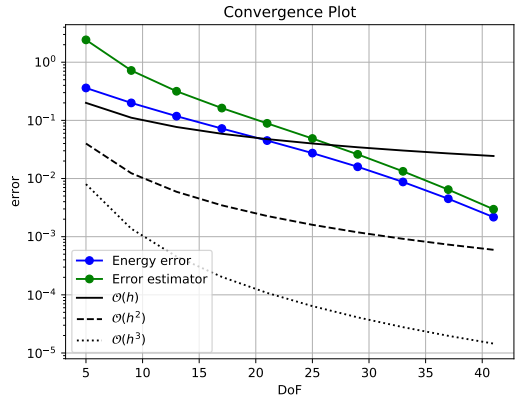
(a) *Global h-refinement.*



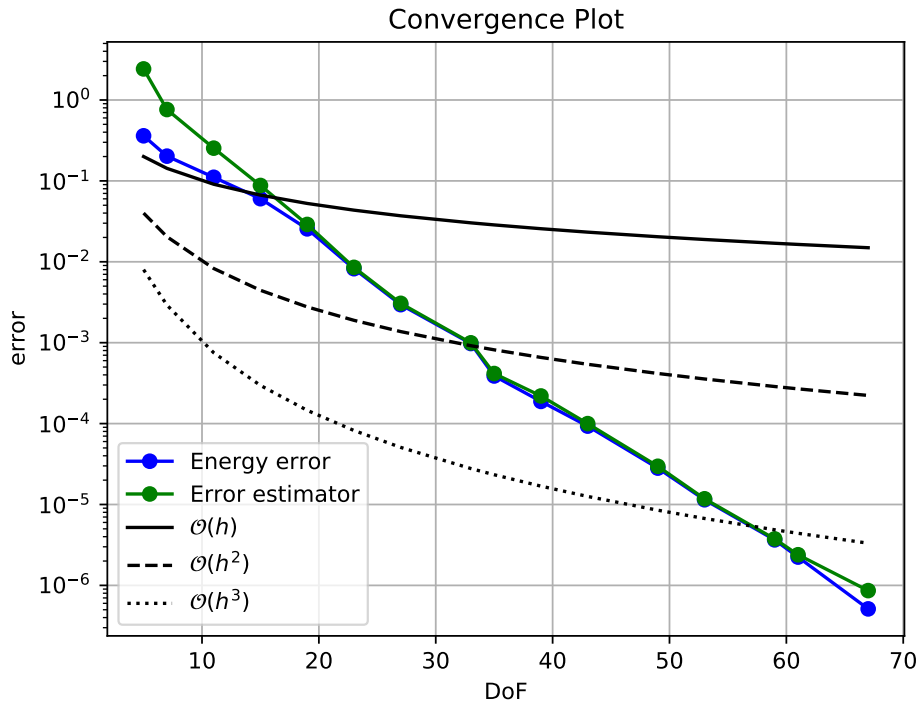
(b) *h-adaptivity.*



(c) *Global p-refinement.*

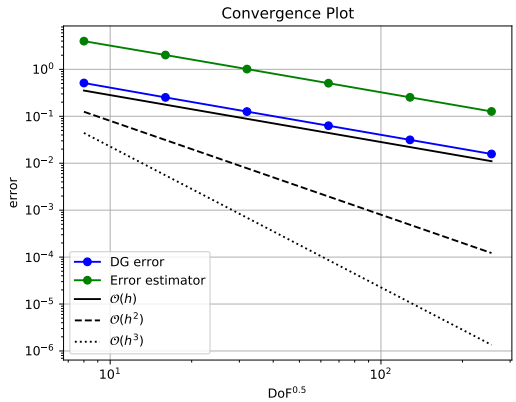


(d) *p-adaptivity.*

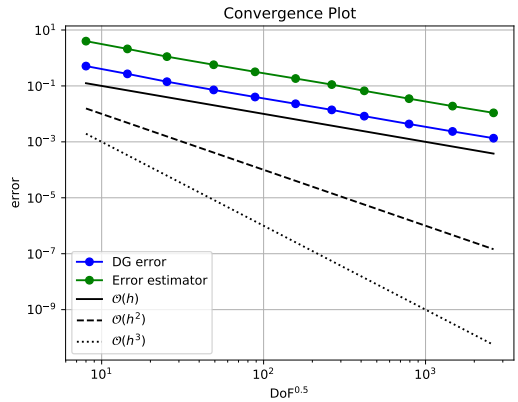


(e) *hp-adaptivity.*

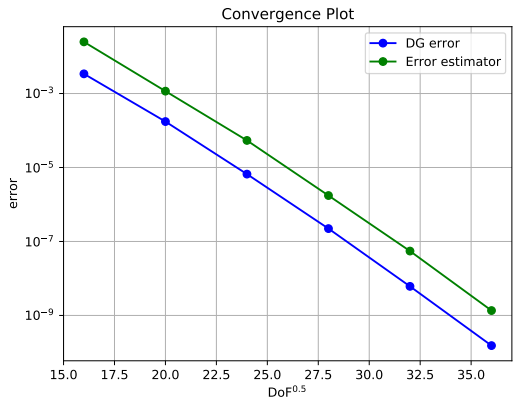
Figure 8: *Convergence of boundary layer problem from Equation (28) in 1D using CGFEM with various refinement/adaptivity strategies.*



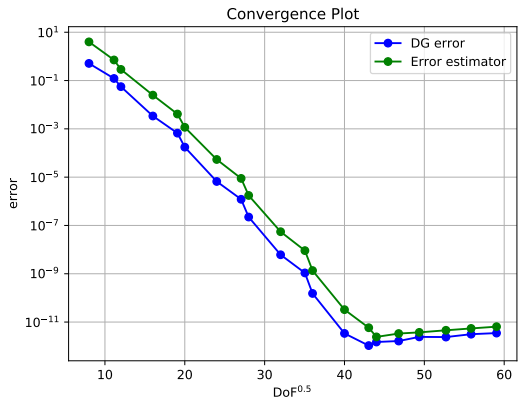
(a) *Global h-refinement.*



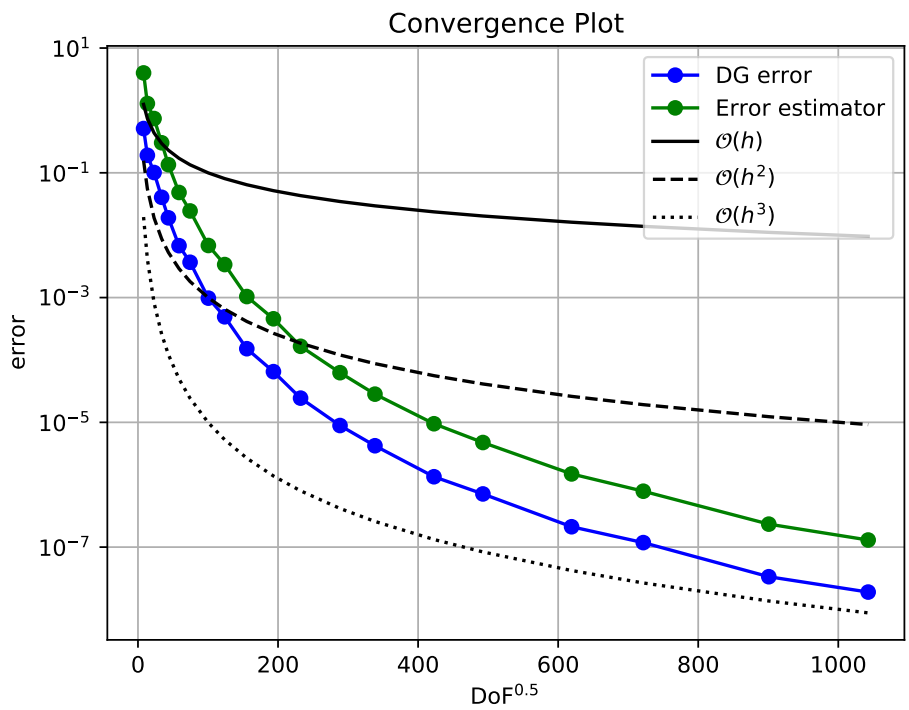
(b) *h-adaptivity.*



(c) *Global p-refinement.*

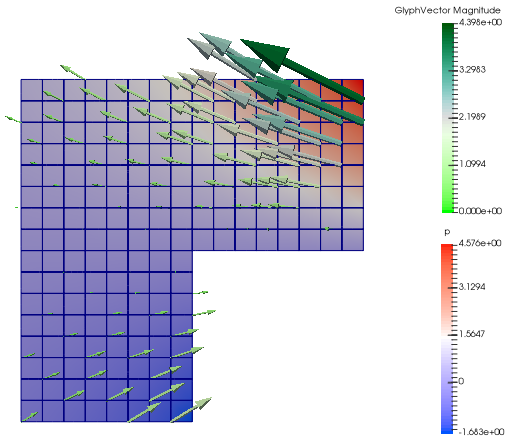


(d) *p-adaptivity.*

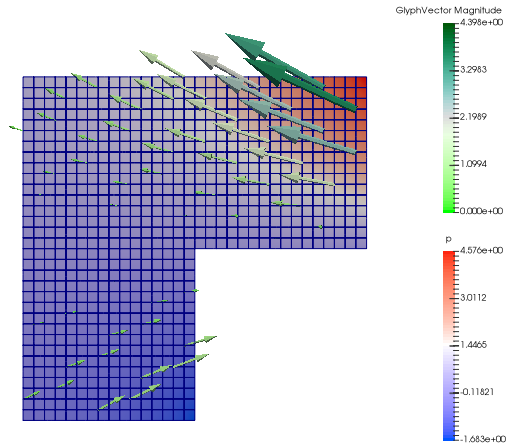


(e) *hp-adaptivity.*

Figure 9: *Convergence of Poisson problem from Equation (2) in 2D using DGFEM in AptoFEM with various refinement/adaptivity strategies.*



(a) *Stokes problem from Equation* (12) *with* $8 \times 8 - 4 \times 4 = 48$ *square elements, solved with the AptoFEM CGFEM solver with quadratic basis functions in space and linear basis functions in pressure.*



(b) *Stokes problem from Equation* (12) *with* $16 \times 16 - 8 \times 8 = 192$ *square elements, solved with the AptoFEM CGFEM solver with quadratic basis functions in space and linear basis functions in pressure.*

Figure 10: *Comparison of two different simulations of a Poisson problem in 2D using AptoFEM. The plots have been generated using ParaView. Note that the velocity plots here have been linearly interpolated between nodal values and therefore may not exact represent the velocity calculated.*

1.7 Overview of Work (so far)

In this report we have outlined what we are aiming towards with this project — namely creating a biophysical computational model of the maternal blood flow in the IVS of the human placenta — and provided some relevant background literature on the biology, modelling, and numerics required. We then introduced the continuous and discontinuous Galerkin methods, followed by a full derivation of the discretisation for both methods for Poisson's equation; we noted the discretisations for Stokes flow, and discussed the use of error indicators and smoothness indicators and how they are useful in *hp*-adaptivity. We ended by describing a typical layout for an FEM code and presented some numerical experiments that mixed together CGFEM and DGFEM, Poisson's equations and Stokes flow, and 1D and 2D simulations; we also investigated how the convergence rates of these simulations are affected by five different types of refinement/adaptivity.

At the time of writing this report, I am hoping to imminently finish writing a DGFEM solver for Stokes flow in AptoFEM, and then look to apply either Stokes or Darcy flow to a realistic geometry of the placenta.

1.8 Future Work

By the end of my project, as discussed in Section 1.1, I hope to couple together techniques from FSI and moving boundary techniques to capture the placental pump phenomenon, as well as potentially incorporating multiscale modelling techniques for modelling the villous structures in the placenta. A suitable model (e.g. Darcy flow) will be chosen for modelling flow within the IVS itself.

The placenta is very much a three-dimensional, time-dependent object, and I therefore expect that simulations will have to take place in 3D with some form of time-stepping; as one might imagine, the computational complexity of such numerics can be huge and as such we would most likely rely on techniques such as *hp*-adaptivity to reduce the number of DoFs to a computable level. Additionally, I would also like to obtain some realistic geometries in which to create meshes.

2 Plan for Coming 12 Months

Key targets for my project include:

October 2021 Use a realistic model of time-independent flow within a realistic geometry;

January 2021 Use some form of time numerical scheme for time-dependent problems;

April 2021 Use some prescribed motion to move the domain/mesh of the problem in a specific way;

July 2022 Use a FSI in to inform how the mesh motion should be done according to the placenta contractions.

In terms of training, I believe that I have completed all compulsory training. At this moment, I do no plan to undertake any more; however that may change if there is a specific need.

I'm quite lucky that COVID-19 has not directly affected my project too much. However, indirectly, I've missed out on a lot of social life of being a PhD student as well as the natural academic discussions that come from seeing people each day. In the past few months I've been fortunate enough to meet a select few of the PhD students in the school, but I still feel as though my project may progress more slowly than it would have until 'normality' returns. I think the technology for discussing complicated mathematics has not yet been perfected and have found Microsoft Whiteboard a little difficult to derive and discuss mathematics.

References

[1] Harbir Antil, Ricardo H. Nochetoo, and Patrick Sodré. "Optimal control of a free boundary problem with surface tension effects: a priori error analysis". In: *SIAM Journal on Numerical Analysis* 53.5 (2015), pp. 2279–2306. DOI: https://doi.org/10.1137/140958360.

[2] Douglas N. Arnold et al. "Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems". In: *SIAM Journal on Numerical Analysis* 39.5 (2006), pp. 1749–1779. DOI: https://doi.org/10.1137/S0036142901384162.

[3] M.J. Baines, M.E. Hubbard, and P.K. Jimack. "A moving mesh finite element algorithm for the adaptive solution of time-dependent partial differential equations with moving boundaries". In: *Applied Numerical Mathematics* 3.3 (2005), pp. 450–469. DOI: https://doi.org/10.1016/j.apnum.2004.09.013.

[4] MJ Baines, ME Hubbard, and PK Jimack. "Velocity-based moving mesh methods for nonlinear partial differential equations". In: *Communications in Computational Physics* 10.3 (2011), pp. 509–576. DOI: https://doi.org/10.4208/cicp.201010.040511a.

[5] Adam M Blakey. *Blakey FEM Software Package*. https://github.com/JustAdamHere/PhD_BlakeyFEM. 2020.

[6] Adam M Blakey. *FEM Algorithms in C++*. Master's Dissertation. 2020. URL: https://r.blakey.family/BlakeyFEM.

[7] Adam M Blakey. *Forkey FEM Software Package*. https://github.com/JustAdamHere/Forkey-FEM. 2020.

[8] Andrea Bonito. "Quasi-Optimal Convergence Rate of an Adaptive Discontinuous Galerkin Method". In: *SIAM Journal on Numerical Analysis* 48.2 (2010), pp. 734–771. DOI: https://doi.org/10.1137/08072838X.

[9] Achi Brandt. "Multiscale Scientific Computation: Review 2001". In: *Multiscale and Multiresolution Methods*. Springer Berlin Heidelberg, 2002, pp. 3–95. DOI: https://doi.org/10.1007/978-3-642-56205-1_1.

[10] Andrea Cangiani et al. *hp-version discontinuous Galerkin methods on polygonal and polyhedral meshes*. Springer Briefs in Mathematics. https://ebookcentral.proquest.com/lib/nottingham/detail.action?docID=5163236. Springer, 2017. ISBN: 9783319676739.

[11] Paul Castillo et al. "Optimal a priori error estimates for the hp-version of the local discontinuous Galerkin method for convection–diffusion problems". In: *Mathematics of Computation* 71.238 (2001), pp. 455–478. DOI: https://doi.org/10.1090/S0025-5718-01-01317-5.

[12] I. L. Chernyavsky, O. E. Jensen, and L. Leach. "A mathematical model of intervillous blood flow in the human placentone". In: *Placenta* 31.1 (2010), pp. 44–52. DOI: https://doi.org/10.1016/j.placenta.2009.11.003.

[13] Igor L. Chernyavsky et al. "Transport in the placenta: homogenizing haemodynamics in a disordered medium". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 369.1954 (2011), pp. 4162–4182. DOI: https://doi.org/10.1098/rsta.2011.0170.

[14] J. Collis et al. "Effective equations governing an active poroelastic medium". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2198 (2017), p. 20160755. DOI: https://doi.org/10.1098/rspa.2016.0755.

[15] Neele S Dellschaft et al. "The haemodynamics of the human placenta in utero". In: *PLOS Biology* 18.5 (2020), e3000676. DOI: http://dx.doi.org/10.1371/journal.pbio.3000676.

[16] Jean Donea et al. "Arbitrary Lagrangian–Eulerian Methods". In: *Encyclopedia of Computational Mechanics*. Cambridge University Press, 2004. Chap. 14. ISBN: 9780470091357. DOI: https://doi.org/10.1002/0470091355.ecm009.

[17] Jim Douglas and Todd Dupont. "Interior Penalty Procedures for Elliptic and Parabolic Galerkin Methods". In: *Computing Methods in Applied Sciences*. Springer Berlin Heidelberg, 1976, pp. 207–216. ISBN: 978-3-540-37550-0. DOI: https://doi.org/10.1007/BFb0120591.

[18] F.F. Erian, S. Corrsin, and S.H. Davis. "Maternal, placental blood flow: A model with velocity-dependent permeability". In: *Journal of Biomechanics* 10.11 (1977), pp. 807–814. DOI: https://doi.org/10.1016/0021-9290(77)90095-1.

[19] Luca Formaggia, Alfio Quarteroni, and Allesandro Veneziani. *Cardiovascular Mathematics: Modeling and simulation of the circulatory system*. MS&A. Springer Science & Business Media, 2010. DOI: https://doi.org/10.1007/978-88-470-1152-6.

[20] Klaus Gerdes and Dominik Schötzau. "hp-finite element simulations for Stokes flow — stable and stabilized". In: *Finite elements in analysis and design* 33.3 (1999), pp. 143–165. DOI: https://doi.org/10.1016/S0168-874X(99)00018-9.

[21] Wei Gong, Michael Hinze, and Zhaojie Zhou. "Optimal control of a free boundary problem with surface tension effects: a priori error analysis". In: *Journal of Scientific Computing* 66.3 (2016), pp. 941–967. DOI: https://doi.org/10.1007/s10915-015-0051-2.

[22] K. K. Gupta and J. L. Meek. "A brief history of the beginning of the finite element method". In: *International Journal for Numerical Methods in Engineering* 39.22 (1996), pp. 3761–3774. DOI: https://doi.org/10.1002/(SICI)1097-0207(19961130)39:22<3761::AID-NME22>3.0.CO;2-5.

[23] Gene Hou, Jin Wang, and Anita Layton. "Numerical Methods for Fluid-Structure Interaction — A Review". In: *Communications in Computational Physics* 12.2 (2012), pp. 337–377. DOI: https://doi.org/10.4208/cicp.291210.290411s.

[24] Paul Houston, Dominik Schötzau, and Thomas P. Wihler. "Energy norm shape a posteriori error estimation for mixed discontinuous Galerkin approximations of the stokes problem". In: *Journal of Scientific Computing* 22–23.June (2005), pp. 347–370. DOI: https://doi.org/10.1007/s10915-004-4143-7.

[25] Paul Houston and Endre Süli. "A note on the design of hp-adaptive finite element methods for elliptic partial differential equations". In: *Computer Methods in Applied Mechanics and Engineering* 194.2 (2005), pp. 229–243. DOI: https://doi.org/10.1016/j.cma.2004.04.009.

[26] Paul Houston and Thomas P. Wihler. "An hp-adaptive Newton-discontinuous-Galerkin finite element approach for semilinear elliptic boundary value problems". In: *Mathematics of Computation* 87.1 (2018), pp. 2641–2674. DOI: https://doi.org/10.1090/mcom/3308.

[27] L. Irons, J. Collis, and R.D. O'Dea. *Modeling of Microscale Transport in Biological Processes: Chapter 12 - Microstructural Influences on Growth and Transport in Biological Tissue — A Multiscale Description*. Academic Press, 2017, pp. 311–334. DOI: https://doi.org/10.1016/B978-0-12-804595-4.00012-2.

[28] Oliver E Jensen and Igor L Chernyavsky. "Blood Flow and Transport in the Human Placenta". In: *Annual review of fluid mechanics* 51.1 (2019), pp. 25–47. DOI: https://doi.org/10.1146/annurev-fluid-010518-040219.

[29] Claes Johnson. *Numerical solution of partial differential equations by the finite element method*. Cambridge University Press, 1987. ISBN: 0 521 345 146.

[30] E. Lecarpentier et al. "Computational Fluid Dynamic Simulations of Maternal Circulation: Wall Shear Stress in the Human Placenta and Its Biological Implications". In: *PLOS ONE* 11.1 (2016), pp. 1–18. DOI: https://doi.org/10.1371/journal.pone.0147262.

[31] J.M. Melenk and B.I. Wohlmuth. "On residual-based a posteriori error estimation in hp-FEM". In: *Advances in Computation Mathematics* 15.1 (2001), pp. 311–331. DOI: https://doi.org/10.1023/A:1014268310921.

[32] Reuben D O'Dea et al. "A multiscale analysis of nutrient transport and biological tissue growth in vitro". In: *Mathematical medicine and biology: a journal of the IMA* 32.3 (2015), pp. 345–366. DOI: https://doi.org/10.1093/imammb/dqu015.

[33] Hannah J. Pybus et al. "Reduced biomechanical models for precision-cut lung-slice stretching experiments". In: *Journal of Mathematical Biology* 82.35 (2021). DOI: https://doi.org/10.1007/s00285-021-01578-2.

[34] Andrey Ricardo Da Silva, Gary P Scavone, and Maarten Van Walstijn. "Numerical simulations of fluid-structure interactions in single-reed mouthpieces". In: *The Journal of the Acoustical Society of America* 122.3 (2007), pp. 1798–1809. DOI: https://doi.org/10.1121/1.2759166.

[35] Christoph Schwab. *p- and hp-finite element methods: Theory and applications in solid and fluid mechanics*. Oxford University Press, 1998. ISBN: 978-0198503903.

[36] Pavel Šolín, Jakub Červený, and Ivo Doležel. "Arbitrary-level hanging nodes and automatic adaptivity in the hp-FEM". In: *Mathematics and Computers in Simulation* 77.1 (2008), pp. 117–132. DOI: https://doi.org/10.1016/j.matcom.2007.02.011.

[37] Wei-Kang Sun, Lu-Wen Zhang, and K.M. Liew. "A smoothed particle hydrodynamics-peridynamics coupling strategy for modeling fluid-structure interaction problems". In: *Computer methods in applied mathematics and engineering* 371 (2020). DOI: https://doi.org/10.1016/j.cma.2020.113298.

[38] Gergely Szabó, József Györgyi, and Gergely Kristóf. "Three-dimensional FSI Simulation by Using a Novel Hybrid Scaling – Application to the Tacoma Narrows Bridge". In: *Periodica Polytechnica Civil Engineering* 64.4 (2020), pp. 975–988. DOI: https://doi.org/10.3311/PPci.15586.

[39] R. Verfürth. "A posteriori error estimation and adaptive mesh-refinement techniques". In: *Journal of Computational and Applied Mathematics* 50.1-3 (1994), pp. 67–83. DOI: https://doi.org/10.1016/0377-0427(94)90290-9.

[40] Thomas P. Wihler. "An hp-adaptive strategy based on continuous Sobolev embeddings". In: *Journal of Computational and Applied Mathematics* 235.8 (2011), pp. 2731–2739. DOI: https://doi.org/10.1016/j.cam.2010.11.023.