Automating goal-based adaptive refinement for partial differential equations and eigenvalue problems

Joseph Flood

St Hugh's College

University of Oxford

A thesis submitted for the degree of

M.Sc. in Mathematical Modelling and Scientific Computing

Trinity Term 2025

Acknowledgements

First and foremost, I wish to extend my sincere gratitude to my supervisor, Patrick Farrell, for his invaluable support, patience and guidance throughout the project.

I am also grateful to my cohort and friends for their moral support, shared experiences and inspiration and extend this to all in the university who have crossed my path with their wisdom during this journey.

I would like to also acknowledge and thank my family and close friends. Their unwavering belief in me has kept my spirits and motivation strong throughout the process. The MMSC basketball sessions in particular have been a highlight of the year.

I dedicate this report to my future wife and kids.

Abstract

We extend the Firedrake framework with automated goal-based adaptive meshing solvers for partial differential equations (PDEs) and eigenvalue problems (eigenproblems). For PDEs, the user provides the UFL form, a target functional $J(u)$, and a tolerance; the solver estimates the goal error $|J(u) - J(u_h)|$ and refines cells in the mesh that contribute most to this error, iterating until the tolerance is met. For eigenproblems, the user provides a target eigenvalue, and the solver minimises the error in the closest eigenvalue, $|\lambda - \lambda_h|$. Both solvers utilise the theory of the dual-weighted residual method to accomplish these tasks.

A major barrier to adoption of goal-based adaptivity has been the requirement for error estimators to be manually derived for each variational problem. Our solver removes the need for manual derivations by automatically computing local error estimators, making goal-based adaptivity accessible to those without specialist expertise in the dual-weighted residual method.

For PDEs, we demonstrate efficiency on standard benchmark problems, including mixed formulations such as weakly symmetric linear elasticity and Navier-Stokes with a lift boundary integral. For eigenproblems, we demonstrate strong results for Maxwell and Stokes problems.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

In finite element methods, engineers and scientists are frequently interested in derived quantities of interest (QoI), as opposed to the solution field across the entire domain. Canonical examples include the stress on a particular surface or the lift and drag generated by an object. Estimating the error in these quantities is much more useful in these scenarios than estimating the error in the global energy norm, which is the traditional goal of *a posteriori* error estimation [3]. Estimating and controlling the error in QoIs is the goal of the Dual-Weighted Residual (DWR) method, established by Rolf Rannacher and Roland Becker, amongst others, in the 1990s and 2000s. A comprehensive review can be found in [8].

Let $J(u)$ be a quantity of interest derived from the solution u by the 'goal functional' $J(\cdot)$. Let u_h be a computed approximation to u using the finite element method. The goal of the DWR method is multi-faceted: We seek to both estimate and control the error $\eta \coloneqq |J(u) - J(u_h)|$. To estimate the error, we first define and solve an adjoint (also known as dual) problem for a function, denoted as z, defined across the domain. Roughly speaking, the adjoint solution z measures the influence of the goal functional $J(\cdot)$ on the PDE. The error estimate is then directly obtained by weighting the weak global residual against the adjoint solution. Controlling the error is done in two stages: Firstly, the local contributions to the global error are obtained by decomposing the weighted global residual into the sum of local weighted residuals. To obtain sharp estimates, the local residual is (generally) integrated by parts to decompose it into cell and facet terms, which are then individually weighted against z [8]. Secondly, cells that contribute most to the global error in the goal

functional are refined, with the aim of reducing the error, although by an unknown amount. These steps are repeated until a desired error estimate is reached.

Unfortunately, decomposing the local residual into cell and facet terms can be challenging for complex PDEs and/or for those without specific expertise [39]. To remedy this, Rognes and Logg developed an algorithm which approximates the local residual on cells and facets with a polynomial representation, implementing it in the open-source finite element software FEniCS [6]. Initial results indicate the algorithm is effective, but the implementation had some difficulties. In particular, hand-written code was required to assemble and solve the polynomial problems over the cells and facets.

In our project, we implement an updated version of the algorithm developed by Rognes and Logg for PDEs, and additionally develop a similar algorithm that controls the error of targeted eigenvalues of eigenproblems. Both algorithms are implemented in Firedrake, a finite element software that abstracts the coding and solving of the finite element method away from the user, enabling the user to solve complex PDEs and eigenproblems with less extensive software knowledge [19]. The user expresses the problem in Unified Form Language (UFL), stating the variational formulation on the desired finite element function space [1]. The function space is defined using elements contained within the FIAT library [26]. Firedrake then assembles the system and solves the underlying matrix problem using PETSc [11, 4].

Using tools available in Firedrake, we reinterpret the residual calculations of Rognes and Logg as variational problems over suitable DG0 (cell) and DG0 trace (facet) spaces, and implemented the latter in the finite element library FIAT [26]. This allowed for the residual calculations to be coded in exactly the same way as the PDE itself, in a handful of lines of Python. We then solved the problem with one matrix-free Jacobi iteration, which solves the problem in a local manner without any laborious implementation of custom code.

1.2 Main contributions

In addition to the reinterpretation of the algorithm in Firedrake, we also extend the error estimates to optionally include the adjoint residual, which theoretically improves error estimates for nonlinear problems. Additionally, we extend the automated method to eigenproblems. An algorithm is developed in which the goal functional is the error in a selected eigenvalue, which is obtained by the user passing in a target.

In terms of outputs, we develop a two new Firedrake solver classes, one for goal-

oriented error control of PDEs, and one for eigenvalue-targeted error control of eigenvalue problems.

The PDE solver, `GoalAdaptiveNonlinearVariationalSolver`, takes in a variational finite element problem defined upon an initial mesh, a goal functional (QoI), and a tolerance. The solve method computes the finite element solution, estimates the error in the goal functional applied to the computed solution field, and adaptively refines the mesh until the error estimate is below the tolerance, or the maximum iteration is reached, whichever is first.

The eigenproblem solver, `GoalAdaptiveEigenSolver`, takes in a variational finite element eigenvalue problem defined upon an initial mesh, a target eigenvalue, and a tolerance. The solve method computes the finite element solution, estimates the error in the closest computed eigenvalue to the target, and adaptively refines the mesh until the error estimate is below the tolerance, or the maximum iteration is reached, whichever is first.

We apply the algorithms to a wide range of problems found in the literature, including mixed formulations of fluid flow and elasticity, and a highly nonlinear problem. Good results are obtained for the problems considered.

1.3 Code

The new solver classes are currently undergoing the process of integration with the extended Firedrake sotware library. The development repository can be found at https://github.com/joeflood/adaptiveFEM, which contains the PDE and eigenproblem solver classes, and the numerical examples included within this report, amongst other examples which were not included.

1.4 AI Input

AI was used occasionally to aid in the coding of standard functions. The author notes however, that the AI programs consulted were generally not capable of producing accurate Firedrake code, which is likely due to the fast-paced nature of current Firedrake development.

1.5 Structure of the Dissertation

The remainder of the dissertation is structured as follows. In Chapter 2 we discuss the mathematical framework for the PDE solver, including the underpinning DWR method and automated residual calculations. In Chapter 3 we present numerical experiments on PDE problems. In Chapter 4 we extend the method to eigenvalue problems and results of numerical experiments are presented. In Chapter 5 we summarise the work and suggest future developments to the new Firedrake solvers.

Chapter 2

Mathematical framework

2.1 Global error estimate

We consider the problem of computing the value of a functional of the solution of a variational equation. To make these ideas precise, let V be a function space. On V, let $A : V \times V \to \mathbb{R}$ be a differentiable semilinear form, $F : V \to \mathbb{R}$ be a linear functional, and $J : V \to \mathbb{R}$ be a (possibly nonlinear) goal functional. We seek to solve the variational problem:
Find $u \in V$ such that

$$A(u; v) = F(v), \quad \forall v \in V, \tag{2.1}$$

then compute $J(u)$.
The semicolon means that the function is linear in any arguments to the right. A prototypical example to keep in mind is that A, F represent the stationary Navier–Stokes equations describing flow around a wing, and J evaluates the drag force exerted on that wing. The philosophy here is that J is the quantity of interest; we would accept a poor approximation of u if we had an excellent approximation of J.

To approximate (2.1), we must cast it in a finite-dimensional form. To this end, we consider the Galerkin projection of (2.1) to the finite-dimensional subspace $V_h \subset V$, parametrised by $h \in \mathbb{R}_+$. This gives: *Find $u_h \in V_h$ such that*

$$A(u_h; v_h) = F(v_h), \quad \forall v_h \in V_h, \tag{2.2}$$

then compute $J(u_h)$.
We assume that (2.1) and (2.2) possess locally unique solutions.

We seek to estimate the error in the computed functional $J(\cdot)$, defined as

$$\eta \coloneqq |J(u) - J(u_h)|. \tag{2.3}$$

A general framework for accomplishing this is known as the Dual-Weighted Residual (DWR) method, developed by Rannacher et al. [8]. In this section, we state and prove the most important results in this method, applicable to the general case of semilinear forms and nonlinear goal functionals. We begin by reformulating the problem as a constrained optimization problem for $u \in V$:

$$J(u) \in \min!, \quad A(u; v) = F(v), \quad \forall v \in V. \tag{2.4}$$

This optimisation problem just has as stationary points the (locally unique) solutions of the PDE. Solutions u correspond to stationary points $\{u, z\} \in V \times V$ of the Lagrangian

$$L(u; z) \coloneqq J(u) + F(z) - A(u; z), \tag{2.5}$$

with the adjoint variable $z \in V$. Solutions $\{u, z\}$ are then given by the corresponding Euler–Lagrange system

$$A(u; v) = F(v), \qquad \forall v \in V, \tag{2.6a}$$
$$A'(u; v, z) = J'(u; v), \quad \forall v \in V, \tag{2.6b}$$

where A' is the Fréchet derivative of A. (2.6b) is known as the adjoint equation. The Galerkin approximation of the Euler–Lagrange system is:
Find $\{u_h, z_h\} \in V_h \times V_h$ *such that*

$$A(u_h; v_h) = F(v_h), \qquad \forall v_h \in V_h \tag{2.7a}$$
$$A'(u_h; v_h, z_h) = J'(u_h; v_h), \quad \forall v_h \in V_h. \tag{2.7b}$$

We assume that solutions to this system exist and are locally unique.
At this point, it is useful to introduce the notion of primal and adjoint residuals. The primal residual is associated with (2.2) and is defined by

$$\rho(u_h, v) \coloneqq F(v) - A(u_h; v), \quad v \in V, \tag{2.8}$$

which vanishes for $v \in V_h$. The adjoint residual is associated with (2.7b) and is defined by

$$\rho^*(z_h; v) \coloneqq J'(u_h, v) - A'(u_h; v, z_h), \quad v \in V. \tag{2.9}$$

We also define the forward error $e \coloneqq u - u_h$ and the adjoint error $e^* \coloneqq z - z_h$ Our goal is to associate these residuals with the error in the goal functional $J(\cdot)$. The following proposition, obtained by Becker and Rannacher [8], is one of the key results which we will use for this.

Proposition 2.1. *For the Galerkin approximation of the Euler–Lagrange system given by* (2.2) *and* (2.7b)*, we have the a posteriori error representation*

$$J(u) - J(u_h) = \frac{1}{2} \min_{v_h \in V_h} \rho(u_h; z - v_h) + \frac{1}{2} \min_{v_h \in V_h} \rho^*(z_h; v - v_h) + R, \tag{2.10}$$

with the remainder term given by

$$\begin{aligned} R \coloneqq \frac{1}{2} \int_0^1 \big\{ J'''(u_h + se; e, e, e) - A'''(u_h + se; e, e, e, z_h + se^*) \\ -3A''(u_h + se; e, e, e^*) \big\} s(s-1)\, ds. \end{aligned} \tag{2.11}$$

We remark that the remainder term is cubic in the error $e \coloneqq \{e, e^*\}$ and can therefore usually be neglected in practice [8]. Additionally, for the minimiser over $v_h \in V_h$, we note that due to linearity (in arguments after ;) and Galerkin orthogonality

$$\min_{v_h \in V_h} \rho(u_h; z - v_h) = \rho(u_h; z) - \min_{v_h \in V_h} \underbrace{\rho(u_h; v_h)}_{=0} = \rho(u_h; z) = \rho(u_h; z - z_h)$$

and

$$\min_{v_h \in V_h} \rho^*(z_h; u - u_h) = \rho^*(z_h; u) - \min_{v_h \in V_h} \underbrace{\rho^*(z_h; v_h)}_{=0} = \rho^*(z_h; u) = \rho^*(z_h; u - u_h).$$

In practice, we use $\rho(u_h; z - z_h)$ and $\rho^*(z_h; u - u_h)$ because they give the sharpest estimates in the presence of errors introduced during numerical computation, such as quadrature [8].

One then obtains the error estimate

$$\eta_h \coloneqq |J(u) - J(u_h)| \approx \frac{1}{2} |\rho(u_h; z - z_h) + \rho^*(z_h; u - u_h)|. \tag{2.12}$$

A simpler error estimator can be obtained by solely evaluating the primal residual, at the cost of a quadratic rather than cubic remainder term. This is done by relating the primal and adjoint residuals [8]:

Proposition 2.2. *For the Galerkin approximation of the Euler–Lagrange system given by* (2.2) *and* (2.7b)*, we have the a posteriori error representation*

$$J(u) - J(u_h) = \min_{v_h \in V_h} \rho(u_h; z - v_h) + R, \tag{2.13}$$

where R is given by

$$R = \int_0^1 \big\{ A''(u_h + se; e, e, z) - J''(u_h + se; e, e) \big\} s \; ds. \tag{2.14}$$

As a consequence of Proposition 2.2, the error in the goal functional can be estimated by evaluating the primal residual alone, removing the requirement of obtaining a high-order approximation to the exact solution u of the primal problem, at the penalty of raising the order of the remainder term from cubic to quadratic in the error. In practice, the primal error estimator is

$$\eta_h \coloneqq |J(u) - J(u_h)| \approx |\rho(u_h; z - z_h)|. \tag{2.15}$$

Becker and Rannacher emphasise that, for non-linear problems, the combined error estimate eq. (2.12) may produce more accurate estimates. We also remark that in the special case of a linear variational equation and linear goal functional, the remainder vanishes when solely considering the primal residual Proposition 2.2.

To evaluate these error estimates, we need to approximate $u - u_h$ and $z - z_h$. We discuss this further in Section 2.4 below, but briefly comment here. One option for doing so is to compute better approximations of u and z (e.g. with higher-degree finite elements or on a finer mesh) and compute the difference between these and the solutions in the finite element space we are working with. This is expensive but gives the best error estimates. Various cheaper alternatives have also been proposed [8].

In summary, these error estimates may be evaluated on suitable finite element solutions to predict the accuracy of evaluation of a particular quantity of interest (QoI), defined by a functional of u. In our automatic adaptive refinement algorithm, these estimates are used as a stopping criterion to determine when the estimated error in the goal functional is below a user-specified tolerance. However, to drive the adaptive procedure, we must *localise* the contributions to the error estimator: we must decide *where* in our domain matters for the accurate evaluation of the functional. This is the subject of the next section.

2.2 Local error estimates

As mentioned above, for an adaptive algorithm we also require local error "estimates" that can be used to iteratively drive refinement from one mesh to another. With a local error estimator in hand, one can order the estimates by magnitude and mark cells above a set threshold for refinement. In this section, we discuss how to obtain the local error estimates from the global error estimate (2.15), with the averaged primal and adjoint error estimate (2.12) following similarly. The general approach is to integrate the weak residual by parts, obtaining a local residual on each cell with additional flux terms (which are usually redistributed equally among shared facets).

The local cell-wise and facet-wise residuals are then weighted by the dual solution, in the same manner as Proposition 2.2, but on a local level. Automating the localisation of cell-wise and facet-wise residuals will be one of the key challenges in automating the DWR method below.

To illustrate these ideas, we apply this procedure to three model problems: The Poisson equation, general diffusion equations, and the Navier–Stokes equations.

2.2.1 Poisson equation

We consider the Poisson equation:

$$-\Delta u = f \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \partial\Omega, \tag{2.16}$$

posed on a polygonal domain $\Omega \subset \mathbb{R}^2$. We obtain the standard weak form by multiplying by a test function v and integrating by parts. *Find* $u \in H_0^1(\Omega)$ *such that*

$$A(u, v) = (f, v), \quad \forall v \in H_0^1(\Omega), \tag{2.17}$$

where $A(u, v) \coloneqq (\nabla u, \nabla v)$. Throughout this report, we use $(\cdot, \cdot)$ to denote the L^2 inner product, with corresponding norm $\|\cdot\| \coloneqq \sqrt{(\cdot, \cdot)}$, unless otherwise specified. To solve this equation with the finite element method, we divide the domain Ω into a union of non-overlapping simplicial elements, denoting the decomposition as $\mathbb{T}_h$. We discretize (2.17) using finite-dimensional subspaces

$$V_h \coloneqq \{v \in V : v|_K \in P_n(K),\ K \in \mathbb{T}_h\},$$

where K indexes each cell in $\mathbb{T}_h$, and $P_n(K)$ denotes the space of polynomial functions defined on each cell K of degree at most n. Our approximations $u_h \in V_h$ to the true solution u are thus obtained by solving the discrete problem

$$A(u_h, v_h) = (f, v_h), \quad \forall v_h \in V_h. \tag{2.18}$$

We seek to estimate the element-wise contributions to the error in a linear goal functional $J(\cdot)$ to drive an adaptive refinement algorithm. To do this, we use Proposition 2.2. The Poisson equation (2.16) and the goal functional are linear, so the remainder term vanishes. We have the error estimate

$$\eta_h \coloneqq |J(u) - J(u_h)| = |\rho(u_h; z - z_h)| = |\rho(u_h; e^*)| \tag{2.19}$$

due to Galerkin orthogonality. For the Poisson equation, the weak residual is

$$\rho(u_h; e^*) \coloneqq (f, e^*) - A(u_h, e^*) = (f, e^*) - (\nabla u_h, \nabla e^*) \tag{2.20}$$

We split the residual into element-wise contributions and integrate by parts on each cell:

$$\begin{aligned}\rho(u_h, e^*) &= \sum_{K\in\mathbb{T}_h} \big\{(f, e^*)_K - (\nabla u_h, \nabla e^*)_K\big\} \\ &= \sum_{K\in\mathbb{T}_h} \big\{(f + \Delta u_h, e^*)_K + (-\partial_n u_h, e^*)_{\partial K}\big\},\end{aligned} \tag{2.21}$$

where $\partial_n u_h \coloneqq \nabla u_h \cdot n$, with n the unit normal on cell facet S. Now consider an interior facet S. Since it is shared by two cells K and K', the two contributions from ∂K and $\partial K'$ combine to yield the jump operator

$$[\![\partial_n u_h]\!]|_S \coloneqq \nabla u_h|_K \cdot n + \nabla u_h|_{K'} \cdot n'. \tag{2.22}$$

In other words, the boundary term measures by how much our approximation fails to have continuous normal fluxes across cell boundaries. Both the cell and flux residuals approach zero as the element size $h \to 0$.

Since our goal is to determine the cellwise contribution to the error, we must assign the facet contribution to each cell. To avoid double-counting the jump, one must decide a distribution between the neighbouring cells, the simplest solution being to distribute the flux equally [39]. Our residual decomposition (2.21) then becomes

$$\rho(u_h, e^*) = \sum_{K\in\mathbb{T}_h} \big\{(R(u_h), e^*)_K + (r(u_h), e^*)_{\partial K}\big\}, \tag{2.23}$$

with the cell residual $R(u_h)$ and the edge residual $r(u_h)$ defined by

$$R(u_h) \coloneqq f + \Delta u_h, \qquad r(u_h)_{|S} \coloneqq \begin{cases} \frac{1}{2}[\![\partial_n u_h]\!], & \text{if } S \subset \partial K \backslash \partial\Omega \\ 0, & \text{if } S \subset \partial\Omega. \end{cases}$$

The case $S \subset \partial\Omega$ would have nonzero facet residual if we were imposing boundary data that cannot be represented exactly in V_h. If we define

$$\eta_K \coloneqq |(R(u_h), e^*)_K + (r(u_h), e^*)_{\partial K}| \tag{2.24}$$

then

$$\eta_h = |\rho(u_h, e^*)| \leq \sum_{K\in\mathbb{T}_h} \eta_K \tag{2.25}$$

by the triangle inequality. Adaptive refinement can then be driven by the magnitude of the error indicators η_K. Different strategies may be used, which will be discussed in Section 2.5.1. The general approach is to obtain a positive scalar M for which cells with $\eta_K > M$ are marked for refinement.

We note that since this is a linear problem, by Proposition 2.2 the primal and adjoint residuals are equal. The error in the goal functional could therefore equally well be evaluated as

$$\eta_h = |\rho^*(z_h; e)| = |J'(u_h; v) - A'(u_h; v, z_h)| \tag{2.26}$$

By definition, $J'(u_h; v) = A'(u_h; v, z)$, and for the Poisson equation $A'(u_h; v, z) = a(v, z)$ and $A'(u_h; v, z_h) = a(v, z_h)$, so

$$\eta_h = |a(v, z) - a(v, z_h)| = |a(v, e^*)| = |(f, e^*)| \tag{2.27}$$

which we could split across the mesh in the same manner as (2.21).

2.2.2 General diffusion equation:

To give an example of how these estimators extend to more complicated problems, we apply the same procedure to the following reaction-diffusion operator

$$Lu \coloneqq -\nabla \cdot \{a\nabla u\} + bu = f \quad \text{in } \Omega, \tag{2.28}$$

with mixed boundary conditions. We divide the boundary into Dirichlet and Neumann parts, $\partial\Omega = \Gamma_D \cup \Gamma_N$, where $\Gamma_N \cap \Gamma_D = \emptyset$, then

$$u = g_D \quad \text{on } \Gamma_D, \qquad n \cdot \{a\nabla u_h\} = g_N \quad \text{on } \Gamma_N. \tag{2.29}$$

The error indicators are

$$\eta_K = |(R(u_h), e^*)_K + (r(u_h), e^*)_{\partial K}|, \tag{2.30}$$

with

$$R(u_h) \coloneqq f + \nabla \cdot \{a\nabla u_h\} - bu_h, \qquad r(u_h)|_S \coloneqq \begin{cases} \frac{1}{2} n \cdot [\![\nabla u_h]\!], & \text{if } S \subset \partial K \backslash \partial\Omega \\ 0, & \text{if } S \subset \Gamma_D, \\ n \cdot \{a\nabla u_h\} - g_N, & \text{if } S \subset \Gamma_N. \end{cases}$$

Across a large portion of PDEs, the form of (2.30) is consistent, the challenge comes with obtaining the forms of the cell and edge (flux) residuals [39]. The general approach, beginning with the weak form, is to

1. Split the weak residual into a summation over each cell in the mesh.
2. Integrate by parts the semilinear form $A(\cdot;\cdot)$. This recovers the strong form of the residual on each cell for $R(u_h)$ (this may be split in mixed formulations with multiple test and trial functions present) and a flux term on each cell.
3. Apply the jump operator to each facet and redistribute the fluxes for facets shared between cells.

2.2.3 Stationary incompressible Navier–Stokes equation

To convince the reader this approach can be applied to more complex PDE systems, we apply it to the incompressible Navier–Stokes equations. Given a positive viscosity $\nu > 0$, we solve

$$\begin{aligned} -\nu\Delta v + v \cdot \nabla v + \nabla p &= f, \\ \nabla \cdot v &= 0, \end{aligned} \tag{2.31}$$

for the velocity v and pressure p in a bounded domain $\Omega \in \mathbb{R}^2$. For boundary conditions, we decompose the boundary into rigid walls, inlets, and outlets: $\Gamma = \Gamma_{\text{rigid}} \cup \Gamma_{\text{in}} \cup \Gamma_{\text{out}}$. Along rigid walls, we impose the the no-slip condition, at the inlet we specify the velocity, and at the outlet we impose 'free-stream' conditions,

$$v|_{\Gamma_{\text{rigid}}} = 0, \quad v|_{\Gamma_{\text{in}}} = \hat{v}, \quad \nu\partial_n v - pn|_{\Gamma_{\text{out}}} = 0. \tag{2.32}$$

The variational formulation of (2.31) uses the function spaces

$$\hat{V} \coloneqq L \times \hat{H}, \quad V \coloneqq L \times H \subset V, \tag{2.33}$$

where $L \coloneqq L^2(\Omega), \quad \hat{H} \coloneqq H^1(\Omega)^2, \quad H \coloneqq \{v \in H^1(\Omega)^2 : v|_{\Gamma_{\text{in}} \cup \Gamma_{\text{rigid}}} = 0\}$.

For the solution pair $\{p, v\}$ and test pair $\{q, w\}$, we define the semilinear form

$$A(u; \varphi) \coloneqq \nu(\nabla v, \nabla w) + (v \cdot \nabla v, w) - (p, \nabla \cdot w) + (q, \nabla \cdot v). \tag{2.34}$$

We seek solution $u = \{p, v\} \in V + \{0, \hat{v}\}$, such that

$$A(u; \varphi) = (f, w), \quad \forall \varphi = \{q, w\} \in \hat{V}. \tag{2.35}$$

We assume that this problem possesses a locally unique, stable, solution.

To discretize (2.35), we formulate the problem in the standard way, seeking the solution pair $u_h = \{p_h, v_h\} \in V_h \subset V$ tested against the test pair $\varphi_h = (q_h, w_h) \in \hat{V}_h \subset \hat{V}$. The Galerkin approximation of source term f is denoted by f_h.

Now, we turn our attention to obtain local error estimators to the linear goal functional $J(u)$. The primal residual is given by

$$\rho(u_h; \varphi) = \nu(\nabla v, \nabla w) + (v \cdot \nabla v, w) - (p, \nabla \cdot w) + (q, \nabla \cdot v) - (f, w), \quad \varphi \in \hat{V}. \tag{2.36}$$

We divide the integrals into contributions from each cell $K \in \mathbb{T}_h$ and integrate by parts cell-wise to obtain

$$\rho(u_h; \varphi) = \sum_{K \in \mathbb{T}_h} \left\{ (R^v(u_h), w)_K + (r^v(u_h), w)_{\partial K} + (q, R^p(u_h))_K \right\}, \tag{2.37}$$

with

$$R^v(u_h) \coloneqq -\nu\Delta v_h + v_h \cdot \nabla v_h + \nabla p_h - f_h$$
$$R^p(u_h) \coloneqq \nabla \cdot v_h$$
$$r^v(u_h) \coloneqq r(u_h)|_S \coloneqq \begin{cases} -\frac{1}{2}[\![\nu\partial_n v_h - p_h n]\!], & \text{if } S \subset \partial K \backslash \partial\Omega \\ -(\nu\partial_n v_h - p_h n), & \text{if } S \subset \Gamma_{\text{out}}, \\ 0, & \text{if } S \subset \Gamma_{\text{rigid}} \cup \Gamma_{\text{in}}. \end{cases}$$

The adjoint problem is given by

$$A'(u;\varphi,z) = J(\varphi), \quad \forall \varphi \in \hat{V}, \tag{2.38}$$

from which we obtain the dual solution z. Splitting our dual error e^* into its corresponding velocity and pressure components, $e^* = (e_p^*, e_v^*) = (z^p - z_h^p, z^v - z_h^v)$. Then, as in (2.25), we use the local refinement indicators

$$\eta_K \coloneqq |(R^v(u_h), e_v^*)_K + (r^v(u_h), e_v^*)_{\partial K} + (e_p^*, R^p(u_h))_K|. \tag{2.39}$$

We remark that these outflow conditions are not consistent with the axioms of continuum mechanics [31]. That said, the analysis carries over to the symmetric gradient formulation once appropriate adjustments have been made.

Note that in general the semilinear form $A(\cdot;\cdot)$ may include additional stabilization terms, an example being the streamline-upwind/Petrov–Galerkin formulation developed by Brooks and Hughes for the convection-diffusion equation [10].

We note that it is certainly possible to develop an error indicator without integrating by parts and redistributing the normal derivative, instead simply directly using the weak form. For the Poisson equation (2.16), this would be

$$\eta_K = |(\nabla u_h, \nabla e^*) - (f, e^*)|. \tag{2.40}$$

Unfortunately, doing so loses the sharpness of the indicators and turns out to be much less efficient at reducing the error with mesh refinement [39].

Local refinement indicators similar to (2.25) have been derived by hand for many PDEs. Focusing on duality-based estimators (to distinguish from general norm-based estimators), estimators have been developed for ordinary differential equations [15], plasticity [36], hyperbolic systems [27], reactive compressible flow [42], systems of nonlinear reaction-diffusion equations [16, 41], eigenvalue problems [22, 12], wave propogation [5], radiative transfer [37], nonlinear elasticity [30], the incompressible Navier–Stokes equations [7, 24], variational multiscale equations [29], multiphysics problems, [28], and fluid-interaction problems [38, 44].

Unfortunately, a barrier for widespread development of these estimators exists because of the expertise required to derive them. In the next section, we present an automatic algorithm, developed initially by Rognes and Logg [39] and refined in our work, which removes the need for the user to derive the form of the error estimators by hand.

2.3 Automation of local error estimates

The form of the primal residual $\rho(u_h; v)$, is consistent for a large class of variational problems:

$$\rho(u_h; v) = \sum_{K \in \mathbb{T}_h} \left\{ (R(u_h), v)_K + (r(u_h), v)_{\partial K} \right\} \tag{2.41}$$

Rognes and Logg show that it is possible to automatically calculate approximations R_K and $R_{\partial K}$ to $R(u_h)$ and $r(u_h)$ respectively by solving local problems across the mesh, removing the need to derive these expressions by hand [39]. The approximations of the residual contributions (or scalar components of vector/tensor fields, if this is the case), use piecewise polynomials:

$$R_K \in \mathcal{P}^p(K), \quad R_{\partial K}|_S \in \mathcal{P}^q(K), \quad \forall S \in \partial K, \;\; \forall K \in \mathbb{T}_h, \;\; p, q \in \mathbb{N}. \tag{2.42}$$

To obtain R_K and $R_{\partial K}$, two assumptions are made:

A1 (global decomposition). The residual is a sum of local contributions:

$$\rho(u_h; v) = \sum_{K \in \mathbb{T}_h} \rho_K(v) \tag{2.43}$$

A2 (local decomposition). Each local residual ρ_K offers a local decomposition:

$$\rho_K(v) = (R_K, v)_K + (R_{\partial K}, v)_{\partial K} \tag{2.44}$$

Assumption A1 is satisfied if the semilinear form A is expressed as integrals over the cells and facets of the mesh $\mathbb{T}_h$. Assumption A2 is satisfied if the variational problem has been derived by testing a partial differential equation against a test function and (if required) integrating by parts to move derivatives from the solution to the test function.

Before moving on to the automatic computation of local residuals, we define the barycentric coordinate on a simplex, which are useful for simplifying the subsequent definitions of bubble and facet bubble functions.

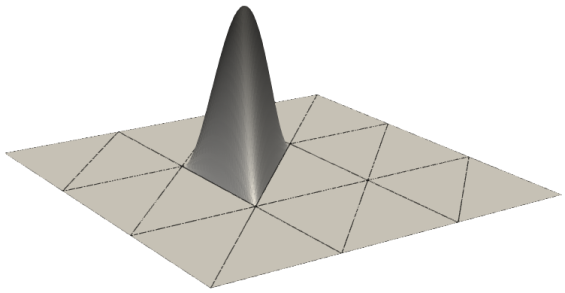


Figure 2.1: Bubble function.

Let K be a d-dimensional simplex. Any point $p \in K$ can be expressed uniquely as a convex combination of the vertices. To be specific, there are $d+1$ numbers $\{\lambda_1, ..., \lambda_{d+1}\}$ such that $\sum_i^{d+1} \lambda_i = 1$ and

$$p = \sum_i^{d+1} \lambda_i r_i,$$

where r_i represent the vertices of the simplex.

2.3.1 Computation of cell residuals R_K

To compute the cell residual R_K, we first define the bubble function b_K on cell K, a polynomial function that vanishes on the boundary of K:

$$b_K \coloneqq \prod_{i=1}^{d+1} \lambda_i^K, \tag{2.45}$$

where λ_i^K is the barycentric coordinate function on K associated with vertex i. The bubble function is plotted in Figure 2.1.

Next, we define the basis $\{\phi_i\}_{i=1}^m$ for $\mathcal{P}^p(K)$. To obtain a local problem for R_K, we test the local residual $\rho_K(v)$ against $b_K\phi_i$, where $b_K\phi_i$ is the bubble function multiplied by the i^{th} basis function. We obtain the local problem: *Find* $R_K \in \mathcal{P}^p(K)$ *such that*

$$(R_K, b_K\phi_i)_K = \rho_K(b_K\phi_i), \quad i = 1, ..., m. \tag{2.46}$$

This may easily be computed cell-by-cell.

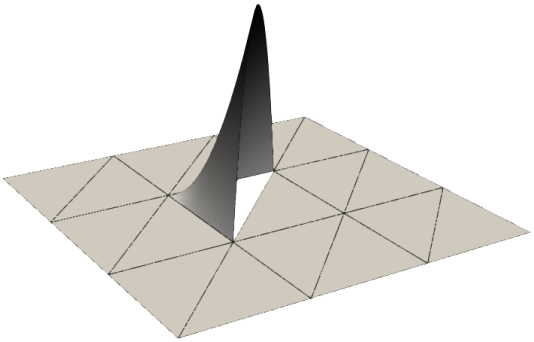


Figure 2.2: Cone function.

2.3.2 Computation of facet residuals $R_{\partial K}$

To obtain the facet residuals $R_{\partial K}$, we begin by defining the cone function on facet S, the analogue of the bubble function on the facet:

$$\beta_S^K = \prod_{i \in I_S^T} \lambda_i^K, \tag{2.47}$$

where I_S^T is an index set, defined such that

$$\beta_S^K|_K = \begin{cases} b_S, & T = S, \\ 0, & T \neq S. \end{cases}$$

The cone function is plotted in Figure 2.2. We test the local residual ρ_K against $\beta_S^K \phi_i$, obtaining the following local problem for each facet residual: *Find* $R_{\partial K}|_S \in \mathcal{P}^q(S)$ *such that*

$$(R_{\partial K}|_S, \beta_S^K \phi_i)_S = \rho_K(\beta_S^K \phi_i) - (R_K, \beta_S^K \phi_i)_K, \quad \forall i \in I_S^K. \tag{2.48}$$

Thus, one may compute the residual representation (2.41) by solving the local problems (2.50) and (2.51) across the mesh. For vector or tensor valued spaces, the local problems are solved for each scalar component.

2.3.3 Solvability of the local problems

Rognes and Logg prove that the local problems (2.50) and (2.51) uniquely determine the polynomial approximations R_K and $R_{\partial K}$ The local problem (2.50) uniquely determines the orthogonal projection of the exact local cell residual $R(u_h)|_K$ with respect to the bubble-weighted inner product $(u, v)_{B_K} \coloneqq (b_K u, v)_K$. Likewise, the local facet problem (2.51) uniquely determines the orthogonal projection of the exact

local facet residual $r(u_h)|_S$ with respect to the facet bubble-weighted inner product $(u,v)_{\beta_S^K} \coloneqq (\beta_S^K u, v)_S$. The accuracy of the approximation is then governed by the degrees p and q of the polynomial spaces (2.42).

The use of bubble and cone functions to localise the weak residual is a standard technique for proving reliability and effectivity for norm-based error estimates [39, 47].

2.3.4 Extension to local adjoint errors

In Rognes and Logg's paper, only the primal residual was considered [39]. Given that Becker and Rannacher highlight the increased effectiveness of using the averaged primal and adjoint residuals for nonlinear problems [8], we extend the automatic computation method to compute approximations to the local dual residuals $\rho^*(z_h; v)$ on each cell and facet.

The adjoint residual affords the same form of decomposition as the primal residual, that is

$$\rho(u_h; v) = \sum_{K \in \mathbb{T}_h} \big\{ (R^*(u_h), v)_K + (r^*(u_h), v)_{\partial K} \big\}. \tag{2.49}$$

It is then straightforward to approximate $R^*(u_h)$ and $r^*(u_h)$ using R_K^* and $R_{\partial K}^*$ in the same fashion as for the primal residual. For our adjoint local cell problem, we seek $R_{\partial K}|_S \in \mathcal{P}^q(S)$ such that

$$(R_K^*, b_K \phi_i)_K = \rho_K^*(b_K \phi_i), \quad i = 1, ..., m. \tag{2.50}$$

For our adjoint the adjoint local facet problem we seek $R_{\partial K}|_S \in \mathcal{P}^q(S)$ such that

$$(R_{\partial K}|_S, \beta_S^K \phi_i)_S = \rho_K(\beta_S^K \phi_i) - (R_K, \beta_S^K \phi_i)_K, \quad \forall i \in I_S^K. \tag{2.51}$$

Uniqueness of the representations R_K^* and $R_{\partial K}^*$ is guaranteed in the same manner as for the primal representations.

2.3.5 Connection to local refinement indicators

Now we have considered the decomposition of the primal and adjoint residuals, we return briefly to connect these decompositions to the form of the dual-weighted local refinement indicators. Considering the primal residual alone, the local refinement indicator is

$$\eta_K^{\text{primal}} \coloneqq \big| (R_K, e^*)_K + (R_{\partial K}, e^*)_{\partial K} \big| \tag{2.52}$$

Considering the combined primal and adjoint residuals, our refinement indicator is

$$\eta_K^{\text{combined}} \coloneqq \frac{1}{2} \big| (R_K^*, e)_K + (R_{\partial K}^*, e)_{\partial K} + (R_K, e^*)_K + (R_{\partial K}, e^*)_{\partial K} \big| \tag{2.53}$$

2.3.6 Implementation

Rognes and Logg's approach was limited by the capabilities of FEniCs when the algorithm was developed. The local cell and facet residual problems hand to be coded by hand, and were solved in serial, which is computationally inefficient [39]. One of Firedrake's key advantages over other codes is the ability to automatically generate parallelized code [19].

To take advantage of this automated parallelism, we formulate the bubble and cone problems as variational problems in UFL and solve these using Firedrake's `solve` function. For our bubble and cone finite element spaces, we use the readily available bubble and facet bubble function spaces in FIAT:

```
B = FunctionSpace(mesh, "bubble", dim+1, variant="integral")
bubbles = Function(B).assign(1)

FB = FunctionSpace(mesh, "facetbubble", dim, variant="integral")
cones = Function(FB).assign(1)
```

The variational problems could then be formulated with ease, the cell problem is shown below[1]:

```
LHS = inner(Rcell_trial, bubbles*Rcell_test)*dx
RHS = assemble(residual(F, bubble*Rcell_test))
solve(LHS == RHS, Rcell, solver_parameters=sp_cell)
```

Where `Rcell_trial` and `Rcell_test` are the trial and test polynomials of degree p on a DG0 function space. `Rcell` is the DG0 function that stores the output.

The corresponding variational formulation (2.51) was implemented for the facet residual `Rfacet`, using the computed value of `Rcell` on the right-hand side. Both variational formulations are solved using matrix-free Jacobi iteration, which is dictated to PETSc by the `solver_parameters` argument.

2.4 Approximating the exact errors e and e^*

To estimate the error in the goal functional with quadratic error, we weight the primal residual $\rho(u_h;\cdot)$ against the dual error $e^* = z - z_h$, where z is the true solution of the dual equation. For cubic error, we weight the adjoint residual $\rho^*(z_h;\cdot)$ against the

[1] Variable names have been renamed for clarity.

primal error $e = u - u_h$. In all practical examples, the exact u and z is impossible to obtain. Instead, we solve for an approximation of z in an 'enriched' finite element space $\hat{V}$, that is, a space of higher polynomial order than V_h. In this report, the degree we use for this space is $p+1$, where p is the polynomial degree of the problem formulation. We have not noticed any significant difference in numerical results by further increasing the polynomial order, that said, in our implementation the user has the option of selecting any appropriate order via the `adaptive_parameters` argument.

$$\hat{V} \coloneqq \{v \in V : v_{|K} \in \mathcal{P}^{p+1}(K),\ K \in \mathbb{T}_h\}. \tag{2.54}$$

For vector, tensor, or mixed spaces, we simply raise the degree of each component of each function space. We denote the 'enriched' approximations in the enriched space $\hat{V}$ of the dual solution z by $\hat{z}$ and the primal solution u by $\hat{u}$. We note that using a refined mesh could also work.

There are two main methods for obtaining the enriched approximations. We explain the methods for the dual solution, but note that approach for the primal solution is exactly the same.

1. **Global high-order approximation.** The straightforward approach is to solve for $\hat{z}$ by assembling and solving the system in the $\hat{V}$ space, with an iterative or direct solver. Then, by interpolating $\hat{z}$ to the lower-degree space via the interpolation operator π_h^-, the dual error is then approximated as

$$e^* \approx \hat{z} - \pi_h^- \hat{z}. \tag{2.55}$$

   An advantage to this approach is that it is simple to implement. The machinery is already in place for linear and nonlinear solves, and a global interpolation operator already exists in Firedrake. A significant drawback of this approach is that it incurs a computational cost similar to that of solving the primal equation in this higher space. One could argue it may be more effective to simply solve the primal equation one degree higher in the first place.

   A disadvantage is that in practice the cost to solve the dual or primal problem in the higher space may be too much to tolerate.

2. **Local high-order approximation.** An alternative to the first approach is to solve for the dual solution on the current mesh and with the primal polynomial degree p, then use patch-wise higher-order interpolation $\pi_h^+ z_h$ to approximate the dual error as

$$e^* \approx \pi_h^+ z_h - z_h. \tag{2.56}$$

Rognes and Logg in particular implement a method of this type by computing least-squares approximations on cell patches, followed by a global smoothing operation [39].

For the duration of this report, we take the approach of (1.). The reason for this is that the focus of the report is on the adaptive algorithm itself, and patchwise extrapolation is not a feature currently readily available in Firedrake. We also note the DWR method can be unreliable if the error incurred by the numerical approximation of the dual solution is not negligible [34]. Therefore, to more reliably assess the implementation of the adaptive algorithm, it is justified to use the direct method for obtaining the enriched solutions.

A new method for approximating the dual error was explored, using vertex-star relaxation, however, the results were not convincing enough and the idea is too underdeveloped to include within this report. That said, results for the Poisson equation in 2D were good enough for this to be promising avenue to consider for future work.

2.5 Mesh refinement

Mesh refinement occurs in two stages. Firstly, the array of local error estimators is used to determine if cells are marked for refinement or left alone. The array of markers is then passed to a refinement function that refines the marked cells. Numerous strategies are available for both stages. In the implementation of our solvers, we chose strategies that were reliable and readily available or straightforward to implement.

2.5.1 Cell marking strategy

In our implementation we use Dörfler marking [13]. Let $\{\eta_K\}_{K\in\mathcal{T}}$ be nonnegative local error indicators on a mesh $\mathbb{T}_h$. Given a bulk parameter $0 < \theta \leq 1$, *Dörfler marking* selects a minimal subset $\mathcal{M} \subseteq \mathbb{T}_h$ such that

$$\sum_{K\in\mathcal{M}} \eta_K \geq \theta \sum_{K\in\mathbb{T}_h} \eta_K. \tag{2.57}$$

Dörfler marking is effective because it concentrates refinement where the estimator is largest, guaranteeing a quantitative reduction of the global estimator after refinement while keeping the number of refined elements controlled. This mechanism underlies convergence (and, for many linear problems, optimal algebraic rates) [32, 45].

The bulk parameter θ can be adjusted to vary the amount of cell refinement between iterations. Adaptive θ adjustment strategies are common [18]. In this report, we fix $\theta = 0.5$, unless stated otherwise.

2.5.2 Mesh refinement strategy

To refine the mesh, we use Firedrake's `Mesh.refine_marked_elements` function [48], which calls Netgen's own refinement method [43]. Netgen refinement is uses a bisection method, based on the paper [2].

2.6 Adaptive algorithm

We are now in a position to state the automatic goal-adaptive algorithm. Although the `GoalAdaptiveNonlinearVariationalSolver` class implements both the primal-only and combined methods, we state the algorithms separately here for clarity. We begin with the primal-residual-only algorithm, Algorithm 1.

If $\rho(u; v)$ is nonlinear, and a more accurate error estimate and refinement procedure is desired, we develop the combined residual algorithm by modifying steps 1, 3, 5, and 6. See Algorithm 2.

Between mesh iterations, the primal solution was interpolated from the coarse mesh to the fine mesh, known as prolongation. This is not only computationally efficient, for nonlinear problems it is often essential for convergence that a solution close to the true solution is used as an initial guess.

The `GoalAdaptiveNonlinearVariationalSolver` class is used in the following manner:

```
# Define UFL variational form F, Function u, boundary conditions
J = u * dx # Example goal functional
tolerance = 0.0001
adaptive_parameters = {} # Dictionary
problem = NonlinearVariationalProblem(F, u, bcs)
adaptive_problem = GoalAdaptiveNonlinearVariationalSolver(problem,
J, tolerance, adaptive_parameters)
adaptive_problem.solve()
```

The user also has the options of passing in separate solver parameters for the PETsc solves of the primal and dual problems, the exact solution field or exact goal

Algorithm 1 Adaptive algorithm, primal residual only

Let $\rho : V \times \hat{V} \to \mathbb{R}$ be a given semilinear form, $J : V \to \mathbb{R}$ a given goal functional, and $\epsilon > 0$ a given tolerance. The user provides an initial tessellation $\mathbb{T}_h$ of the domain Ω and the finite-element formulation of F on spaces $V_h \subset V$ and $\hat{V}_h \subset \hat{V}$, including the element family and degree.

1. Compute the finite element solution $u_h \in V_h$ of the primal problem $\rho(u_h; v_h) = 0$ for the primal approximation u_h.

2. Approximate the dual error $e^* \coloneqq z - z_h$ by $e^* \approx \tilde{z} - z_h$, with $\tilde{z} \in V_h^{p+1}$. Do this by either:

   - Assemble the linearized adjoint semilinear form ρ^* in the space $V_h^{p+1} \times \hat{V}_h^{p+1}$: $\rho^*(\tilde{z}; \tilde{v}) \coloneqq A'(u_h; \tilde{v}, \tilde{z}) - J'(u_h; \tilde{z})$. Compute its finite element solution $\tilde{z}$. Extrapolate $\tilde{z}$ using the interpolation operator $\pi_h^- : V_h^{p+1} \to V_h$ to obtain z_h.
   - Assemble the linearized adjoint semilinear form ρ^* in the space $V_h \times \hat{V}_h$: $\rho^*(z_h; v_h) \coloneqq A'(u_h; v_h, z_h) - J'(u_h; z_h)$. Compute its finite element solution z_h. Extrapolate z_h using the extrapolation operator $\pi_h^+ : V_h \to V_h^{p+1}$ to obtain $\tilde{z}$.
   - Vertex star relaxation to approximate high-frequency modes.

3. Evaluate the error estimate $\eta_h = |\rho(u_h; \tilde{z} - z_h)|$.

4. If $|\eta_h| \leq \epsilon$, accept the solution u_h and break.

5. Compute the cell and facet residuals R_K and $R_{\partial K}$ of the residual representation (2.41) by solving the local problems (2.50) and (2.51).

6. Compute the error indicators $\eta_K = \left|(R_K, e^*)_K + (R_{\partial K}, e^*)_{\partial K}\right|$.

7. Sort the error indicators in order of magnitude and apply the Dörfler marking strategy discussed in Section 2.5.1.

8. Refine all cells marked for refinement.

9. Go back to step 1.

Algorithm 2 Adaptive algorithm, combined primal and adjoint residuals

1. a) Compute the finite element solution $u_h \in V_h$ of the primal problem $\rho(u_h; v_h) = 0$.

   b) Approximate the primal error using $e \approx \tilde{u} - u_h$, with $\tilde{u} \in V_h^{p+1}$. Do this by either:

   - Solve $\rho(\tilde{u}; \tilde{v}) = 0, \forall \tilde{v} \in \hat{V}_h^{p+1}$, interpolate to obtain u_h by $\pi_h^- \tilde{u}$.
   - Solve $\rho(u_h; v_h) = 0, \forall v_h \in \hat{V}_h$, extrapolate to obtain $\tilde{u}$ by $\pi_h^+ u_h$.

   Then obtain the primal error approximation by evaluating $\tilde{u} - u_h$.

⋮

3. Evaluate the error estimate $\eta_h = \frac{1}{2}|\rho(u_h; \tilde{z} - z_h) + \rho^*(z_h; \tilde{u} - u_h)|$.

⋮

5. Compute the primal cell and facet residual representations R_K and $R_{\partial K}$ of (2.41) and the adjoint cell and facet residual representations R_K^* and $R_{\partial K}^*$ by solving (2.50) and (2.51) and the corresponding adjoint local problems.

6. Compute the error indicators $\eta_K = \frac{1}{2}|(R_K^*, e)_K + (R_{\partial K}^*, e)_{\partial K} + (R_K, e^*)_K + (R_{\partial K}, e^*)_{\partial K}\big|$.

⋮

9. Go back to step 1.

functional for calculation of the exact error. A user guide will be published once the code is pushed to Firedrake that details every options available to the user.

Chapter 3

Numerical PDE examples

In this section, we present numerical examples to demonstrate the effectiveness of the goal adaptive algorithm. We validate our algorithm using problems (8.1) and (8.2) in Rognes and Logg [39], then test the effectiveness of the algorithm on further problems. Before continuing, we note that all examples were cross-checked against manual indicators to ensure correct implementation of the automated algorithm.

3.1 Poisson equation

We begin with the Poisson equation,

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega_D, \\ \partial_n u &= g \quad \text{on } \partial\Omega_N. \end{aligned} \tag{3.1}$$

We use the standard variational formulation of (3.1), seeking $u \in H^1_{0,\partial\Omega_D}(\Omega)$ such that

$$\rho(u; v) \coloneqq A(u, v) - F(v) = 0, \quad \forall v \in H^1_{0,\partial\Omega_D}(\Omega), \tag{3.2}$$

with the bilinear form A and linear functional L defined as

$$\begin{aligned} A(u, v) &\coloneqq (\nabla u, \nabla v), \\ L(v) &\coloneqq (f, v) + (g, v)_{\partial\Omega_N}. \end{aligned}$$

We discretize (3.2) in V_h, the space of continuous piecewise linear polynomials (continuous Lagrange elements).

The test case we use is taken from [39], for validation of our implementation of the algorithm. Consider the three-dimensional L-shaped domain

$$\Omega = \big((-1,1)\times(-1,1)\backslash(-1,0)\times(-1,0)\big)\times(-1,0). \tag{3.3}$$

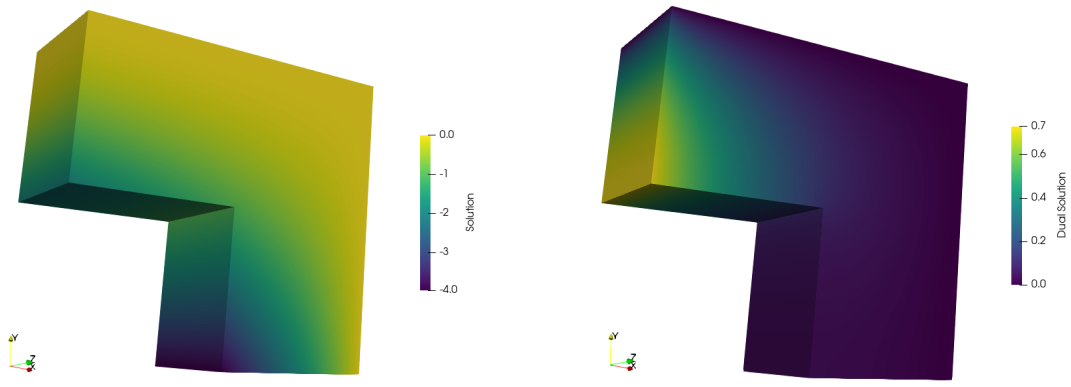
We take the Dirichlet boundary to be $\partial\Omega_D \coloneqq \{(x,y,z) : x = 1 \text{ or } y = 1\}$ and the Neumann boundary to be $\partial\Omega_N \coloneqq \partial\Omega\backslash\partial\Omega_D$. For the source term, we take $f(x,y,z) = -2(x-1)$, and for the Neumann boundary we let $g = G\cdot n$, where $G(x,y,z) = \big((y-1)^2, 2(x-1)(y-1), 0\big)$. The exact solution is then given by

$$u(x,y,z) = (x-1)(y-1)^2.$$

For the goal functional, we take the average value of the solution on the left boundary $\Gamma = \{(x,y,z) : x = -1\}$; that is,

$$J(u) = \int_\Gamma u \, \mathrm{d}s. \tag{3.4}$$

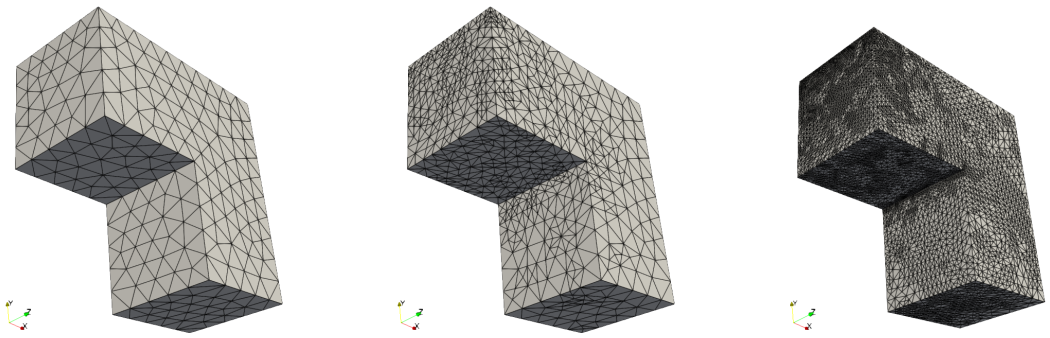
We solve this problem using Algorithm 1. For our approximation of the dual solution, we take $\tilde{z} \in V_h^2$ (the space of piece continuous quadratic elements), in which we directly solve for $\tilde{z}$ then interpolate to obtain $e^* \approx \tilde{z} - \pi_h^- \tilde{z}$. Figure 3.1 shows the computed primal solution u_h and the computed enriched dual solution $\tilde{z}_h$.



(a) Primal solution u_h. (b) Enriched dual solution $\hat{z}_h$.

Figure 3.1: Computed primal and dual solution of the Poisson problem at mesh iteration 4.

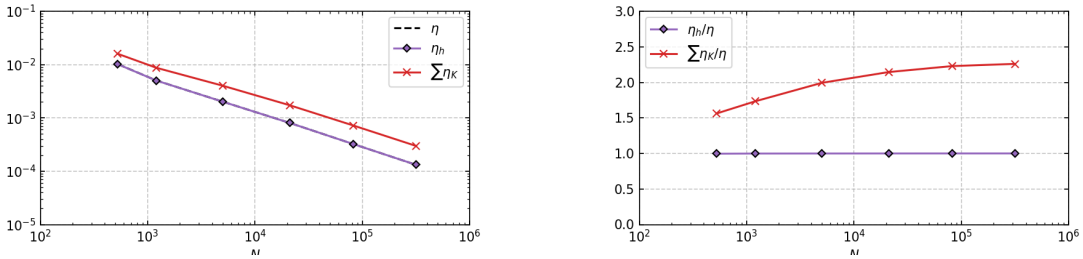
Figure 3.2 shows selected meshes generated in the application of Algorithm 1 to the Poisson problem.



(a) Initial (b) Level 2 (c) Level 4

Figure 3.2: Selected meshes generated by Algorithm 1 solving the Poisson problem.

Figure 3.3 shows the true error η, the error estimate η_h, the sum of the error indicators $\sum_K \eta_K$, and effectivity indices η_h/η and $\sum_K \eta_K/\eta$, plotted against N, the number of degrees of freedom in the function space V_h. The estimated error η_h gives a very good approximation of the true error η, with the effectivity index $\eta_h/\eta \approx 1$ even on the coarsest mesh. The second effectivity index $\sum_K \eta_K/\eta$ gives an indication of the sharpness of the local error estimators. Since the sum of local error indicators is not used to predict the error, its overall magnitude is not strictly important, but it provides a measure of the sharpness of the error indicators. If summed without signs, the summed error indicator is equivalent to the predicted error, up to floating point errors.



(a) Errors. (b) Effectivity indices.

Figure 3.3: Errors and effectivity indices for the Poisson problem.

3.2 Weakly symmetric linear elasticity

Next, we consider a three-field formulation of linear isotropic elasticity which weakly enforces the symmetry of the stress tensor. This problem was tested by Rognes and Logg, 8.2 [39]. For a domain $\Omega \subset \mathbb{R}^2$, the unknowns are the stress tensor

$\sigma \in H(\mathrm{div}, \Omega; \mathbb{R}^{2\times 2})$, the displacement $u \in L^2(\Omega; \mathbb{R}^2)$, and the rotation $\gamma \in L^2(\Omega)$. The bilinear and linear forms are

$$A\big((\sigma, u, \gamma), (\tau, v, \eta)\big) \coloneqq (B\sigma, \tau) + (\mathrm{div}\,\sigma, v) + (u, \mathrm{div}\,\tau) + (\sigma, \eta) + (\gamma, \tau), \tag{3.5}$$

$$F\big((\tau, v, \eta)\big) \coloneqq (g, v) + (u_0, \tau \cdot n)_{\partial\Omega}, \tag{3.6}$$

where g is the given body force, u_0 is a prescribed boundary displacement field, and B is the compliance tensor. We consider an isotropic homogeneous elastic material, reducing the form of $B\sigma$ to

$$B\sigma = \frac{1}{2\mu}\Big(\sigma = \frac{\lambda}{2(\mu+\lambda)}(\mathrm{tr}\,\sigma)I\Big), \tag{3.7}$$

with shear modulus μ and elastic modulus λ. $I \in \mathbb{R}^{2\times 2}$ is the identity tensor.

We use the same discretization scheme as Rognes and Logg, with a mixed finite element space $V_h = \hat{V}_h \coloneqq [\mathrm{BDM}_1]^2 \times [\mathrm{DG}_0]^2 \times P_1$ [39]. The stress tensor is discretized using two first-order Brezzi-Douglas-Marini elements $[\mathrm{BDM}_2]^2$, the displacement field with piecewise constant vector elements, $[\mathrm{DG}_0]^2$, and the rotation field with continuous piecewise linear elements, CG_1.

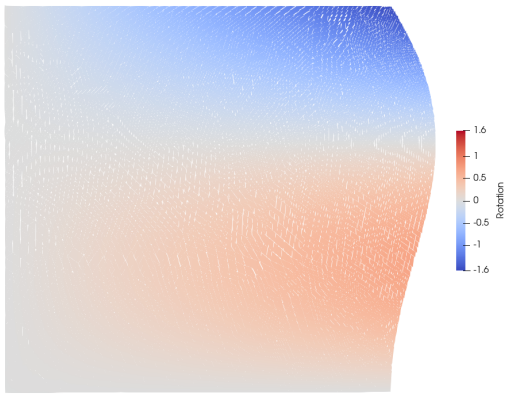
For the particular example taken from [39], we consider the domain $\Omega \coloneqq (0,1) \times (0,1)$. We take the exact displacement solution $u(x,y) = (xy\sin\pi y, 0)$ for $\mu = 1$ and $\lambda = 100$. The exact stress is then $\sigma(x) = A^{-1}\varepsilon(u = A^{-1}\big(\frac{1}{2}(\nabla u + (\nabla u)^T\big)$, and $g = \mathrm{div}\,\sigma(x)$, $u_0(x) = u(x)$ on $\partial\Omega$.

For our goal functional, we take a weighted measure of the average shear stress on the right boundary,

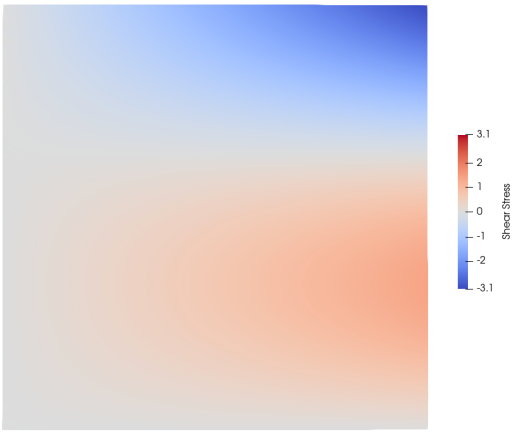
$$J((\sigma, u, \gamma)) = \int_\Gamma \sigma \cdot n \cdot (\psi, 0)\,\mathrm{d}s \approx -0.06029761071, \tag{3.8}$$

where $\Gamma = \{(x,y) : x = 1\}$ and $\psi = y(y-1)$.

We apply Algorithm 1 to the elasticity problem. Figure 3.4 shows the computed solution on the final mesh. Figure 3.4a shows the shape of the deformed object with exaggerated warping. The rotation field is shown with the colourmap, and is seen to correspond to the shear stresses shown in Figure 3.4b.



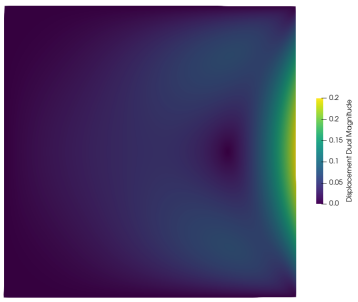
(a) Displacement (warped by vector) and rotation (colourmap).



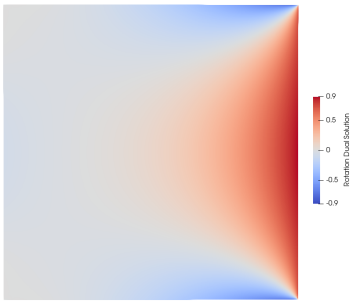
(b) Shear stress.

Figure 3.4: Solution fields of the elasticity problem.

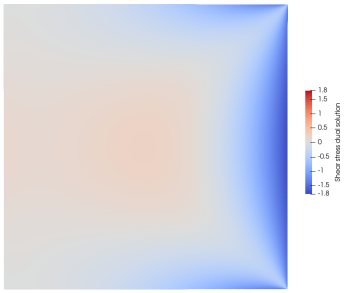
Figure 3.5 shows the enriched dual solution fields on the final mesh. As expected, we see the dual solution dominates on the right face of Ω.



(a) Dual displacement magnitude



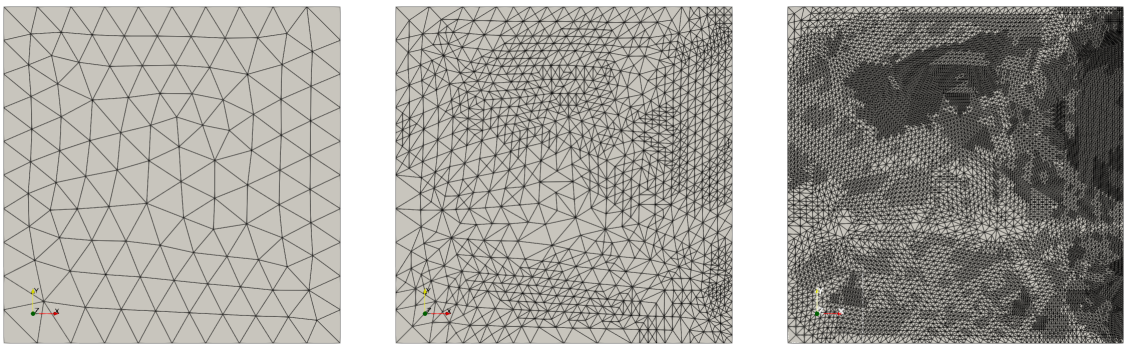
(b) Dual rotation field



(c) Dual shear stress field

Figure 3.5: Enriched dual solution fields of the elasticity problem.

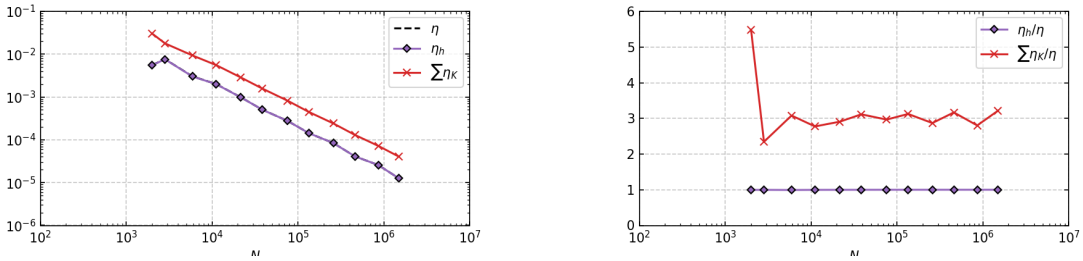
Figure 3.6 shows selected meshes generated in the application of Algorithm 1 to the elasticity problem. The mesh is visibly refined in areas corresponding to high shear stress gradients.



(a) Initial (b) Iteration 4 (c) Iteration 8

Figure 3.6: Mesh progression for the elasticity problem.

Figure 3.7 shows the errors and effectivity indices for the elasticity problem. Like the Poisson problem, the effectivity index $\eta_h/\eta \approx 1$ even on the coarsest mesh. The effectivity index $\sum_K \eta_K/\eta$ is also low, remaining below 4 for the duration of the algorithm, indicating sharpness of the error indicators. These results are in line with what is expected from Rognes and Logg's implementation.



(a) Errors. (b) Effectivity indices.

Figure 3.7: Errors and effectivity indices for the elasticity problem.

3.3 Navier-Stokes problem

Next, we consider an example Navier-Stokes problem for the computation of lift around a cylinder. We take the computational domain to be $\Omega := (0, H) \times (0, 2.2) \backslash \mathcal{B}$, where $H = 0.41$ and $\mathcal{B}$ is a circle of radius $1/20$ and centre $(0.2, 0.2)$. The strong

form of the problem is

$$\begin{aligned} -\nu\Delta u + (u\cdot\nabla)u - \nabla p = 0 \quad &\text{in } \Omega, \\ \nabla\cdot u = 0 \quad &\text{in } \Omega, \\ u = 0 \quad &\text{on } \Gamma_{\text{no-slip}}, \\ u = \hat{u}(x,y) \quad &\text{on } \Gamma_{\text{inflow}}, \\ \nu\frac{\partial u}{\partial n} - p\cdot n = 0 \quad &\text{on } \Gamma_{\text{outflow}}, \end{aligned} \tag{3.9}$$

where $\nu = 10^{-3}$. The boundary parts are

$$\begin{aligned} \Gamma_{\text{outflow}} &\coloneqq (\{x = 2.2\}\cap\partial\Omega)\backslash\partial(\{x=2.2\}\cap\partial\Omega), \\ \Gamma_{\text{inflow}} &\coloneqq \{x=0\}\cap\partial\Omega, \\ \Gamma_{\text{no-slip}} &\coloneqq \overline{\partial\Omega\backslash(\Gamma_{\text{inflow}}\cup\Gamma_{\text{outflow}})}. \end{aligned}$$

For the viscosity, we take $\nu = 0.001$. The inflow velocity is given by $\hat{u}(x,y) \coloneqq (1.2y(H-y)/H^2, 0)$. The pressure is uniquely determined due to the condition prescribed on Γ_{outflow} [23]. The Reynold's number of this problem is then $\text{Re} = 20$ using the mean velocity. We take the goal functional as the lift on the cylinder, defined by

$$J(u) \coloneqq 500\int_{\partial\mathcal{B}}\left(\nu\frac{\partial u}{\partial n} - pn\right)\cdot e_2 \,\mathrm{d}s, \tag{3.10}$$

where $e_2 = (0,1)^T$. For our reference value, we take $J(u) = 0.03616435$, computed with $[\text{CG}_5]^2\times\text{CG}_4$ elements on a fine mesh of 1,3400,432 degrees of freedom.

The Galerkin approximation of the variational formulation of (3.9) is given by the approximation of (2.35), that is: We seek $(v_h, p_h)\in V_h$ such that

$$A(u;\varphi) \coloneqq \nu(\nabla v,\nabla w) + (v\cdot\nabla v, w) - (p,\nabla\cdot w) + (q,\nabla\cdot v), \quad \forall (w,q)\in\hat{V}_h. \tag{3.11}$$

We discretize the space using Taylor-Hood elements, the velocity space is discretized using continuous piecewise quadratic vector elements $[\text{CG}_2]^2$, and the pressure space is discretized using continuous piecewise linear elements CG_1.

We applied Algorithm 1 to the lift problem. To do so required specific implementations to ensure convergence of the Newton algorithm. Firstly, the system would not solve with Newton iteration, with or without linesearch, if the solution field was initialised at 0 with $\nu = 10^{-3}$. Instead, parameter continuation was necessary, which was performed before calling the GoalAdaptiveNonlinearVariationalSolver solve method. The viscosity schedule (five values, logarithmically spaced from 0.05 to 0.001) was generated with NumPy [21].

The solution on the final mesh generated by Algorithm 1 is shown in Figure 3.8.

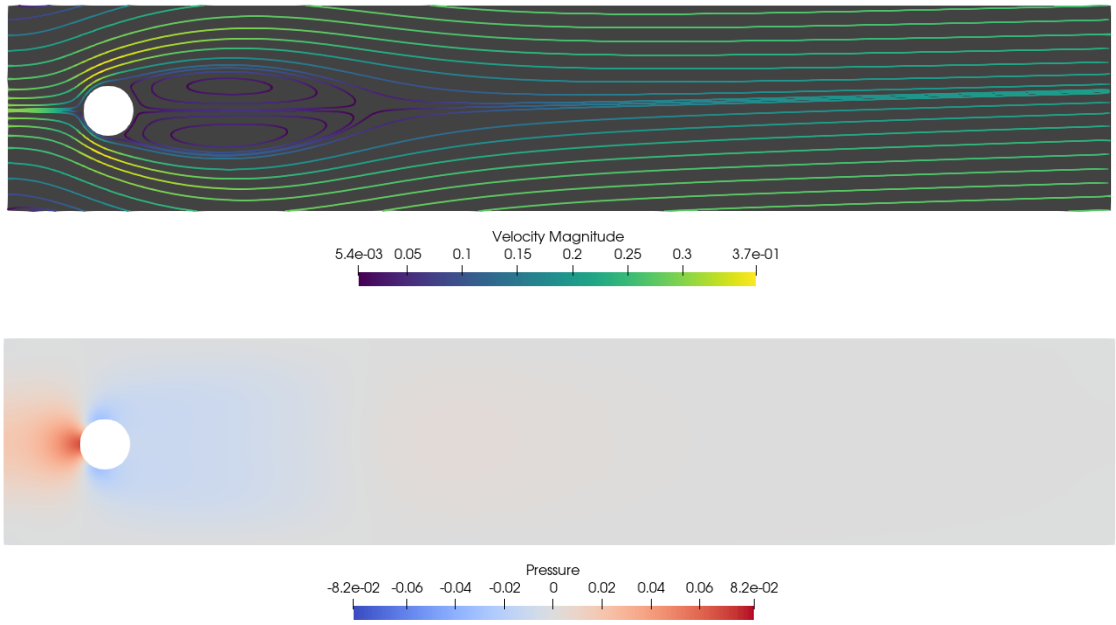


Figure 3.8: Velocity streamlines (top) and pressure field (bottom).

Figure 3.9 shows a selection of meshes generated by the solver. We observe that initial mesh refinement is concentrated very close to the cylinder boundary. Later refinements occur further from the cylinder boundary, indicating that the local error estimates are distributed more uniformly. We also note that refinement occurs at the inlet, and refinement around the cylinder is concentrated on the streamlines that pass above and below the cylinder, not directly behind it, which is interesting because these are the zones that most impact the lift around the cylinder. Refinement around the cylinder boundary is very fine, a close-up image of the mesh at the cylinder boundary is shown in Figure 3.10. This replicates the use of inflation layers by engineers when designing meshes for flow problems [35].

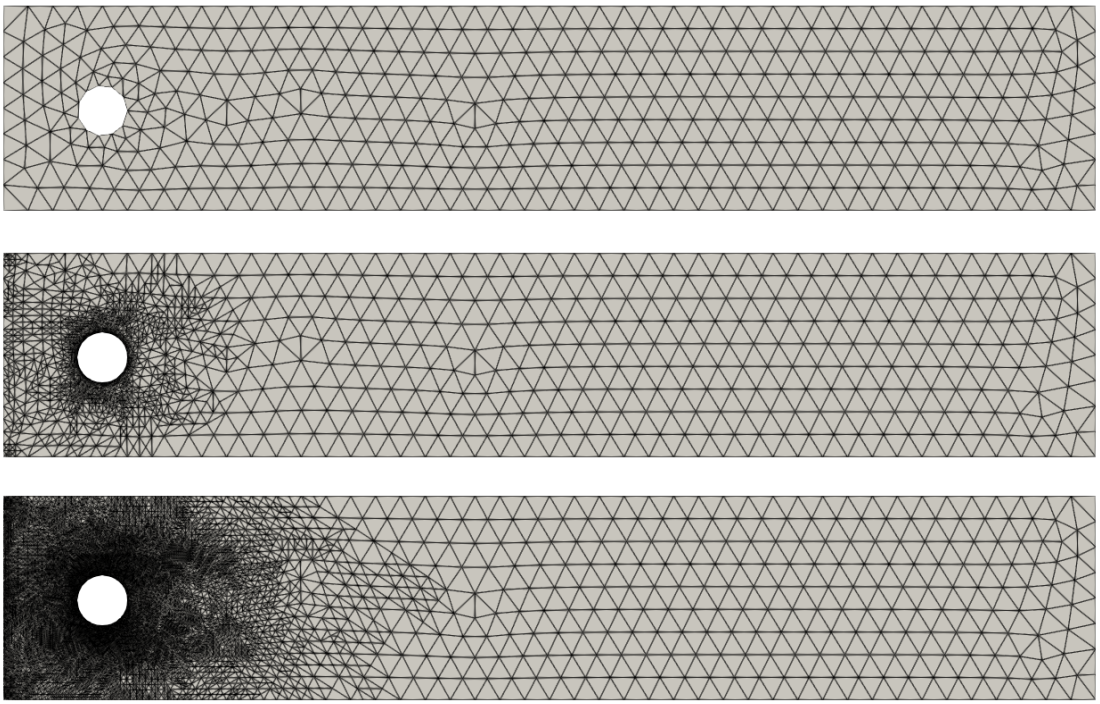


Figure 3.9: Progression of meshes for the Navier-Stokes problem. Top: Initial mesh. Middle: Mesh iteration 9. Bottom: Mesh iteration 18.

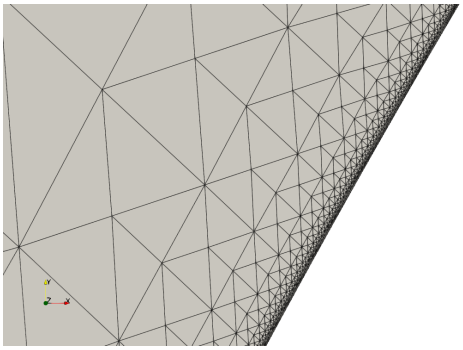
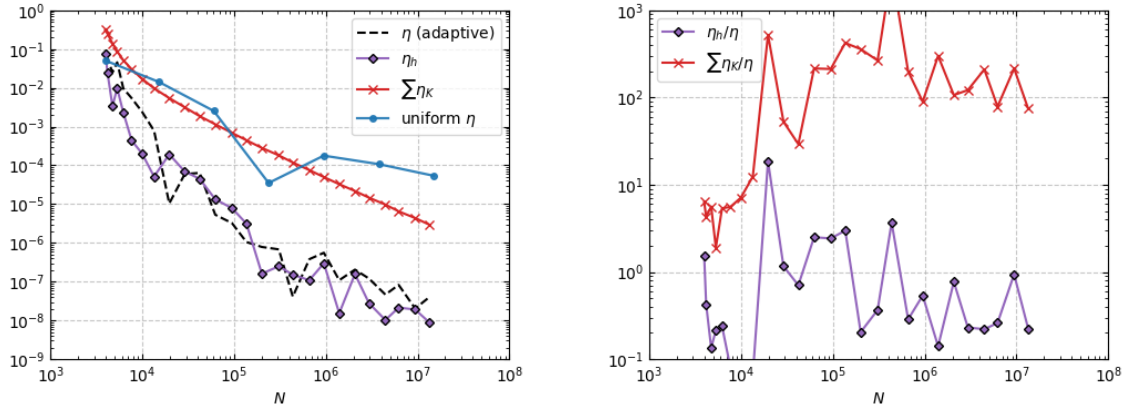


Figure 3.10: Close up of the cylinder boundary on mesh 18.

Errors and effectivity indices for the application of Algorithm 1 are shown in Figure 3.11. Firstly, we observe that the moving average of the error decreases with order of convergence $\approx \mathcal{O}(N^{-1})$, although the oscillatory nature of the error means there is no guaranteed decrease in the error from one refinement level to the next. The effectivity index η_h/η is highly oscillatory on earlier meshes, but appears to settle between 0.3 and 1 as the solver progresses. This is still good, particularly for a problem with Reynold's number 20.I It is similar to the effectivity we see in Rognes and Logg's example of with Reynold's number ≈ 20[1], although their problem

[1]Estimated using the stated outflux $\int_{\Gamma_{\text{output}}} u \cdot n \, \mathrm{d}s \approx 0.40863917 = U_{\text{mean}}$, the channel height

has reentrant corners, and our results are more oscillatory. The sum of the local error estimators is ≈ 100, which is large and suggests that the error in this problem is dominated by flux terms. Lastly, we include the error of a uniformly refined algorithm for comparison, and highlight the orders of magnitude better convergence achieved.

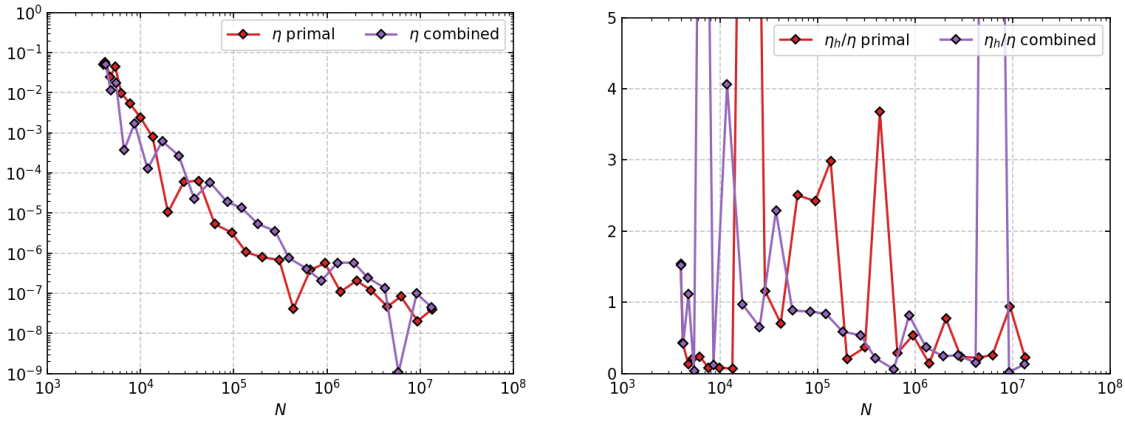


(a) Errors. (b) Effectivity indices.

Figure 3.11: Errors and effectivity indices for Algorithm 1 applied to the Navier-Stokes lift problem.

For this example, we also tested the performance of the combined residual Algorithm 2. Figure 3.12 shows the comparison between the exact errors and effectivity index η/η. Unfortunately, we do not see a noticeable improvement in error convergence when using the combined algorithm for this problem. This is likely because of the lack of sharpness of the error indicators as previously discussed, which also applies to the combined indicators. Lastly, we note that the computed solution is only reliable down to 7 decimal places, so errors and effectivities beyond this point cannot be used reliably. This is likely the cause of the outlier error and effectivity for the combined algorithm around $N = 6 \times 10^6$.

$H = 1$, and viscosity $\nu = 0.02$, using $\mathrm{Re} = U_{\mathrm{mean}} H/\nu$.



(a) Errors. (b) Effectivity index η_h/η.

Figure 3.12: Comparison of errors and effectivity index η_h/η of Algorithm 1 and Algorithm 2 applied to the Navier-Stokes lift problem.

3.4 p-Laplacian problem

To demonstrate Algorithm 1 and Algorithm 2 applied to a highly nonlinear problem, we tested them on the regularized p-Laplacian problem on a slit domain, used in [14].

The strong form of the equation seeks u such that

$$-\operatorname{div}((|\nabla u|^2+\epsilon^2)^{\frac{p-2}{p}}\nabla u) = 1 \qquad \text{in } \Omega, \tag{3.12}$$

$$u = 0 \qquad \text{on } \Gamma_D, \tag{3.13}$$

$$(|\nabla u|^2+\epsilon^2)^{\frac{p-2}{p}}\nabla u \cdot n = 0 \qquad \text{on } \Gamma_N. \tag{3.14}$$

The natural function space is the Sobolev space $W^{1,p}(\Omega)$ of L^p functions with L^p weak gradients, equipped with the standard norm

$$W^{1,p}(\Omega) := \left\{ v \in L^p(\Omega) \ : \ \nabla v \in (L^p(\Omega))^2 \right\}, \qquad \|v\|_{1,p} := \left(\|v\|^p_{L^p(\Omega)} + \|\nabla v\|^p_{L^p(\Omega)} \right)^{1/p}. \tag{3.15}$$

To encode the Dirichlet condition, we take the closed subspace of functions with zero trace on Γ_D,

$$W := W^{1,p}_D(\Omega) := \left\{ v \in W^{1,p}(\Omega) \ : \ \operatorname{Tr} v = 0 \text{ on } \Gamma_D \right\} \ = \ \overline{C^\infty_c(\Omega \cup \Gamma_N)}^{W^{1,p}(\Omega)}. \tag{3.16}$$

For the discretization, we let $\mathbb{T}_h$ be a simplical tesselation of Ω and define the continuous, piecewise-polynomial space of degree 1:

$$V_h := \left\{ v_h \in C^0(\overline{\Omega}) \ : \ v_h|_K \in \mathcal{P}^1(K) \ \ \forall K \in \mathbb{T}_h \right\}. \tag{3.17}$$

Imposing the Dirichlet trace on Γ_D yields the discrete trial/test space:

$$W_h := \left\{ v_h \in \mathbb{V}_h \ : \ v_h|_{\Gamma_D} = 0 \right\} \subset W. \tag{3.18}$$

The Galerkin formulation of the regularized p-Laplacian then seeks $u_h \in W_h$ such that for all $v_h \in W_h$,

$$\int_\Omega \left(|\nabla u_h|^2 + \varepsilon^2\right)^{\frac{p-2}{p}} \nabla u_h \cdot \nabla v_h \, dx \ = \ \int_\Omega v_h \, dx, \quad \forall v_h \in W_h. \tag{3.19}$$

Note that the homogeneous Neumann condition ensures the boundary term produced by integration by parts vanishes.

For the problem in [14], we take $p = 4$, $\epsilon = 10^{-10}$. The domain posed in [14] is $\Omega := (-1,1)^2 \backslash \{0\} \times (-1,0)$. The ability to generate slits of zero thickness is not yet implemented in Netgen, so we approximate this problem with a slit of thickness 0.0000005. We take the left side of the slit (the side with inward normal pointing in the negative x-direction) as the Neumann boundary, and the remaining boundary as Dirichlet.

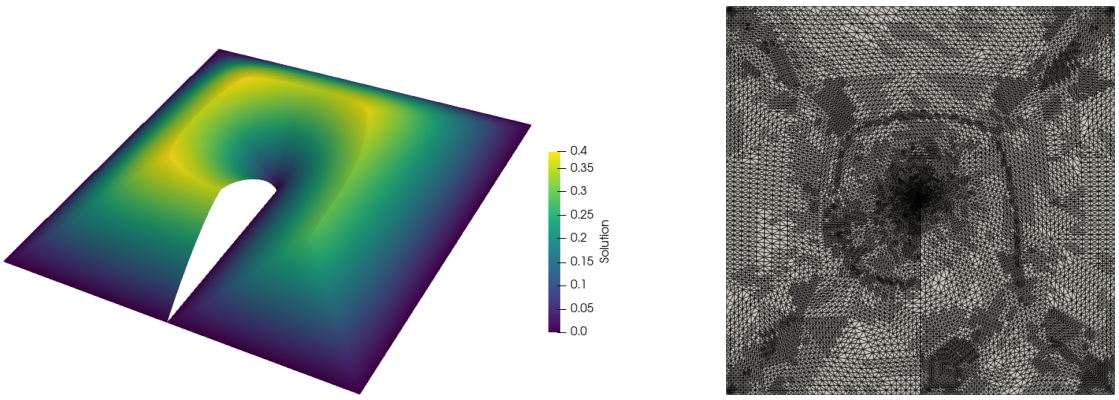
$$\partial\Omega = \Gamma_D \cup \Gamma_N, \qquad \Gamma_D \cap \Gamma_N = \emptyset.$$

A diagram is visible in [14]. For the goal functional, we take

$$J(u) := \int_\Omega u \, \mathrm{d}x, \tag{3.20}$$

which has the given solution 0.71755 [14].

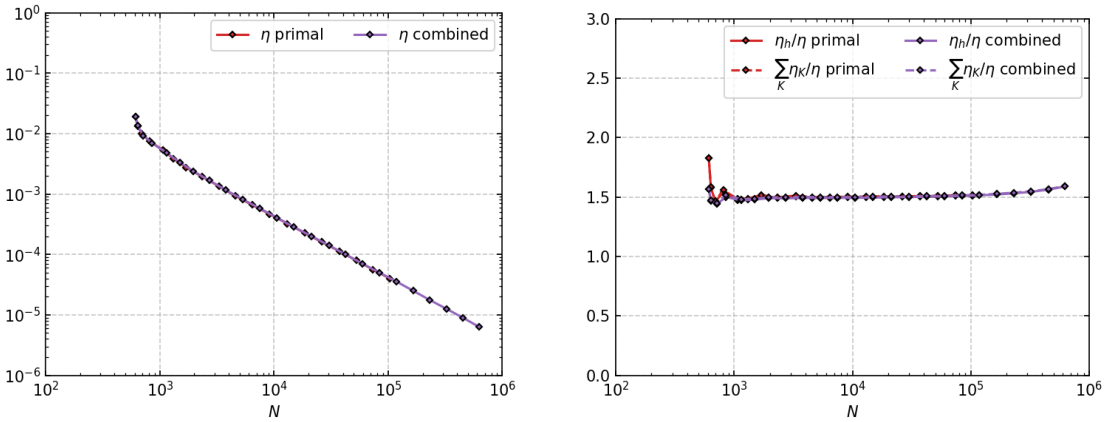
We apply Algorithm 1 to this problem, with an initial mesh of 611 degrees of freedom. The computed solution using Algorithm 1 on the final mesh and corresponding mesh are shown in Figure 3.13. We observe the mesh prioritises refinement at the slit corner and along ridges in the solution. Although, the goal functional being an average across the entire domain means that refinement still occurs throughout.



(a) Solution. (b) Mesh.

Figure 3.13: Final primal solution and mesh for Algorithm 1 applied to the p-Laplace problem.

A comparison of errors and effectivity index η_h/η of Algorithm 1 and Algorithm 2 applied to the p-Laplacian problem is shown in Figure 3.14. Minimal difference can be seen between the two algorithms. Regarding effectivities, all lines overlap, which indicates the local error estimators are sharp. The fact that they are all ≈ 1.5 is not a concern, as this is very likely to be a result of the approximate domain with finite width. This graph was primarily included to indicate the comparison between Algorithm 1 and 2, and between the local and global indicators.



(a) Errors. (b) Effectivity indices.

Figure 3.14: Comparison of errors and effectivity index η_h/η of Algorithm 1 and Algorithm 2 applied to the p-Laplacian problem.

Chapter 4

Extension to eigenproblems

4.1 Mathematical framework

4.1.1 FEM for eigenproblems

Let $\mathcal{A}$ be a uniformly elliptic operator defined on a bounded domain $\Omega \subset \mathbb{R}^2$. The eigenvalue problem for this operator is

$$\mathcal{A}v = \lambda v, \quad \text{in } \Omega, \quad v = \hat{v} \text{ on } \partial\Omega. \tag{4.1}$$

Before stating the variational formulation, we begin with some definitions. Let V and H be Hilbert spaces such that the embedding $V \hookrightarrow H$ is compact. We denote the inner product on H by $(\cdot, \cdot)$, with associated norm $\| \cdot \|$.

Let $a(\cdot, \cdot)$ be the sesquilinear form generated by the operator $\mathcal{A}$. A sesquilinear form is a form that is linear in the first slot and conjugate-linear in the second, that is, $a : V \times V \to \mathbb{C}$,

$$\begin{aligned} a(\alpha\varphi_1 + \beta\varphi_2,\, \psi) &= \alpha\, a(\varphi_1, \psi) + \beta\, a(\varphi_2, \psi), \\ a(\varphi,\, \alpha\psi_1 + \beta v_2) &= \overline{\alpha}\, a(\varphi, \psi_1) + \overline{\beta}\, a(\varphi, \psi_2), \end{aligned}$$

for all $\varphi, \varphi_1, \varphi_2, \psi, \psi_1, \psi_2 \in V$ and all $\alpha, \beta \in \mathbb{C}$ [25].

We assume that $a(\cdot, \cdot)$ is V-elliptic in the sense that there exist constants $\alpha, \gamma > 0$ and $\beta \geq 0$ such that

$$|a(\varphi, \psi)| \leq \alpha \|\varphi\|_V \|\psi\|_V, \qquad \gamma \|\varphi\|_V^2 \leq |a(\varphi, \varphi)| + \beta \|\varphi\|^2 \quad \forall \varphi, \psi \in V. \tag{4.2}$$

As in [22], for simplicity we assume $\beta = 0$, so that $0 \notin \sigma(A)$, where $\sigma(A)$ denotes the spectrum of the operator A associated with $a(\cdot, \cdot)$.

The abstract eigenvalue problem then seeks pairs $u \coloneqq (v, \lambda) \in V \times \mathbb{C}$ with $\|v\| = 1$ such that

$$a(v, \varphi) = \lambda(u, \varphi) \qquad \forall \varphi \in V. \tag{4.3}$$

The adjoint problem seeks pairs $u \coloneqq (v^*, \lambda^*) \in V \times \mathbb{C}$ such that

$$a(\psi, v^*) = \overline{\lambda^*}\,(\psi, v^*) \qquad \forall \psi \in V, \qquad (v, v^*) = 1. \tag{4.4}$$

Since the embedding $V \hookrightarrow H$ is compact, the classical Riesz–Schauder theory applies (see [25]). Consequently, both the primal and adjoint eigenproblems (4.3)–(4.4) possess discrete spectra consisting of countably many isolated eigenvalues of finite algebraic multiplicity, with no finite accumulation point [22]. Furthermore, $\lambda_i^* = \overline{\lambda_i}$ [22].

We denote the algebraic multiplicity of an eigenvalue λ as τ_λ, the geometric multiplicity as κ_λ. The 'ascent' α_λ is the smallest integer such that $\ker\{(\mathcal{A} - \lambda I)^{\alpha+1}\} = \ker\{(\mathcal{A} - \lambda I)^{\alpha}\}$ [22]. Henceforth, we assume all eigenvalues have ascent 1, and therefore the geometric and algebraic multiplicities are equal [22].

The Galerkin approximation of (4.3) and (4.4) is as follows. Let $V_h \subset V$ be a finite-dimensional finite element space, where, as before, h is a mesh size parameter. We seek nontrivial pairs $u_h = \{v_h, \lambda_h\}$ and $u_h^* = \{v_h^*, \lambda_h^*\}$ in $V_h \times \mathbb{C}$ such that

$$a(v_h, \varphi_h) = \lambda_h(v_h, \varphi_h) \quad \forall \varphi \in V_h, \quad \|v_h\| = 1, \tag{4.5}$$

$$a(\psi_h, v_h^*) = \overline{\lambda_h^*}(\psi_h, v_h^*) \quad \forall \psi_h \in V_h, \quad (v_h, v_h^*) = 1. \tag{4.6}$$

Again, the approximate primal and dual eigenvalues satisfy $\lambda_{h,i}^* = \overline{\lambda}_{h,i}$.

4.1.2 Dual-weighted residual theory applied to eigenproblems

We apply the dual-weighted residual theory by Becker and Rannacher [22, 8], presented in Section 2.1, to the eigenvalue setting presented above. We begin by introducing the mixed function spaces $V \coloneqq H \times \mathbb{C}$ and $V_h \coloneqq H_h \times \mathbb{C}$, where H is a Hilbert space and H_h is its Galerkin approximation. To do so, we let the test function pair $\Psi \coloneqq \{\psi, \chi\}$ and define the nonlinear form

$$A(u; \varphi) \coloneqq -a(v, \psi) + \lambda(v, \psi) + \overline{\chi}\{\|v\|^2 - 1\}. \tag{4.7}$$

The discrete eigenvalue problems (4.5) and (4.6) are equivalent to the nonlinear variational equations

$$A(u_h; \Psi_h) = 0 \quad \forall \Psi_h \in V_h, \tag{4.8}$$

and

$$A'(u_h;\Psi_h,z_h) = J'(u_h;\Psi_h) \quad \forall \Psi_h \in V_h. \tag{4.9}$$

The corresponding primal and adjoint residuals are

$$\rho(u_h;\cdot) \coloneqq a(v_h,\cdot) - \lambda_h(v_h,\cdot), \tag{4.10}$$

$$\rho^*(u_h^*;\cdot) \coloneqq a(\cdot,v_h^*) - \lambda_h^*(\cdot,v_h^*). \tag{4.11}$$

By selecting the goal functional $J(w) \coloneqq \chi\|v\|^2 = \chi$, the error in a particular target eigenvalue can be calculated *a posteriori* as

$$(\lambda-\lambda_h)(1-\sigma_h) = \frac{1}{2}\big\{\rho(v_h;v^*-\psi_h) + \rho^*(v^*,v-\varphi_h)\big\}, \tag{4.12}$$

for arbitrary $\varphi_h, \psi_h \in V_h$, where $\sigma_h \coloneqq \frac{1}{2}(v-v_h, v^*-v_h^*)$. The proof is given in [22]. We note that the factor $\sigma_h(\lambda-\lambda_h)$ is the remainder term, so this error representation is exact.

For symmetric problems, $a(w,v) = a(v,w)$, the global error representation (4.12) becomes

$$(\lambda-\lambda_h)(1-\sigma_h) = \rho(v_h;v-\varphi_h), \tag{4.13}$$

for arbitrary $\varphi_h \in V_h$, where $\sigma_h \coloneqq \frac{1}{2}\|v-v_h\|^2$.

A further remark (for general nonsymmetric problems) is that the adjoint residual is given by

$$\rho^*(v_h^*;v-v_h) = (\lambda-\lambda_h)(v,v_h^*) = (\lambda-\lambda_h)\big\{(v-v_h,v^*)+1\big\}, \tag{4.14}$$

one could eliminate the adjoint residual by replacing it with a second-order remainder term:

$$(\lambda-\lambda_h)(1-\sigma_h-\sigma_h^{(2)}) = \frac{1}{2}\rho(v_h,v^*-\psi_h), \tag{4.15}$$

where $\sigma_h^{(2)} \coloneqq \frac{1}{2}\{(v-v_h,v^*)+1\}$. Unfortunately, numerical experiments in [22] show that this is not suitable for transport dominated problems. The normalization $(v,v^*)=1$ forces v^* to be large, so that the term $(v-v_h,v^*)$ may be large on meshes constructed solely considering local error indicators generated by $\rho(v_h;v-v_h)$.

4.1.3 Practical error estimates

Except in special cases, we do not have access to the exact primal and adjoint eigenfunctions v, v^*, so they must be approximated. As before, we do this in the lowest space above V_h in which additional information can be obtained. As with PDEs,

we select our 'enriched' approximation space to be the space spanned by piecewise polynomials of degree $p+1$:

$$\tilde{V}_h \coloneqq \{v \in V : v|_K \in \mathcal{P}^{p+1}(K),\ K \in \mathbb{T}_h\}. \tag{4.16}$$

We then replace the exact eigenfunctions v and v^* by $\tilde{v}, \tilde{v^*} \in \tilde{V}_h$. The additional error introduced by this approximation is analysed in [14]. For arbitrary $\psi_h, \varphi_h \in V_h$, we take the interpolant of our enriched eigenfunction approximations, $\psi_h = \pi_h \tilde{v}$ and $\varphi_h = \pi_h \tilde{v^*}$, because this gives the sharpest error indicators in practice.

To localise the error representation, we take (4.12) and integrate the sesquilinear form $a(\cdot,\cdot)$ by parts, to obtain cell and facet residuals as before. As a model problem, we take the convection-diffusion problem of order $m = 2$,

$$\mathcal{A}v \coloneqq -\nabla \cdot (a\nabla v) + b \cdot \nabla v + cv = \lambda v \quad \text{in } \Omega,$$

with homogeneous Dirichlet boundary conditions. With our primal and adjoint error estimates, $e = \tilde{u} - \pi_h \tilde{u}$ and $e^* = \tilde{v^*} - \pi_h \tilde{v^*}$ Integrating the primal residual by parts gives

$$a(v_h, e^*) - \lambda_h(v_h, e^*) = \sum_{K \in \mathbb{T}_h} \Big\{ (\underbrace{\mathcal{A}v_h - \lambda_h v_h}_{R(u_h)}, e^*)_K - (\underbrace{a[\partial_n v_h]}_{r(u_h)}, e^*)_{\partial K} \Big\}, \tag{4.17}$$

where, as with PDEs, we have redistribute the facet fluxes by averaging the flux across both sides of the boundary. For the adjoint residual, we have

$$a(e, v_h^*) - \lambda_h^*(e, v_h^*) = \sum_{K \in \mathbb{T}_h} \Big\{ (e, \underbrace{\mathcal{A}v_h - \lambda_h v_h}_{R^*(u_h^*)})_K - (e, \underbrace{a[\partial_n v_h^*]}_{r^*(u_h^*)})_{\partial K} \Big\}. \tag{4.18}$$

This form of localised error estimate is consistent across a wide range of PDEs. The general combined form is

$$\begin{aligned} \eta_h = |\lambda - \lambda_h| = \frac{1}{(1-\sigma_h)} \sum_{K \in \mathbb{T}_h} \big| &(R(u_h), e^*)_K + (r(u_h, e^*) \\ &+ (e, R^*(u_h^*)) + (e, r^*(u_h^*)) \big| \end{aligned} \tag{4.19}$$

where $R(u_h)$ and $R^*(u_h^*)$ are the primal and adjoint cell residuals, and $r(u_h)$ and $r^*(u_h^*)$ are the primal and adjoint facet residuals, obtained by integrating by parts and averaging cell boundary fluxes across neighbouring cells.

It follows that one may use as the local error indicators

$$\eta_K \coloneqq \big| (R(u_h), e^*)_K + (r(u_h)), e^*)_{\partial K} + (e, R^*(u_h^*))_K + (e, r^*(u_h^*))_{\partial K} \big|, \tag{4.20}$$

or, for symmetric problems,

$$\eta_K \coloneqq \left|(R(u_h), e)_K + (r(u_h), e)_{\partial K}\right|. \tag{4.21}$$

Dörlfer marking is not impacted by the global scaling factor $(1-\sigma_h)$, so we may omit this term for local error indicators. However, when comparing $\sum_K \eta_K$ to η in the numerical examples, we reintroduce it to ensure the scalings are consistent.

As with PDEs, expertise is required to derive the forms of the local cell and facet residuals, which can become arbitrarily complicated depending on the problem at hand. To negate this, we employ the automated procedure described in Section 2.3, in which we approximate the local cell residuals $R(u_h), r(u_h), R^*(u_h^*), r^*(u_h^*)$ with the polynomial approximations $R_K, R_{\partial K}, R_K^*, R_{\partial K}^*$ by solving local problems on each cell.

To be precise, we now have the following error approximations. Global error:

$$\eta_h = |\lambda - \lambda_h| = \frac{1}{2}\left|\frac{\{\rho(v_h; e^*) + \rho^*(v_h^*, e)}{1 - \frac{1}{2}(\tilde{v} - v_h, \tilde{v^*} - v_h^*)}\right|. \tag{4.22}$$

If symmetric, we use the simplified form,

$$\eta_h = |\lambda - \lambda_h| = \left|\frac{\rho(v_h; e^*)}{1 - \frac{1}{2}\|\tilde{v} - v_h\|^2}\right|. \tag{4.23}$$

For local error estimators, we use

$$\eta_K \coloneqq \left|(R_K, e^*)_K + (R_{\partial K}, e^*)_{\partial K} + (e, R_K^*)_K + (e, R_{\partial K}^*)_{\partial K}\right|, \tag{4.24}$$

or, for symmetric problems,

$$\eta_K \coloneqq \left|(R_K, \lambda_h)), e)_K + (R_{\partial K}, e)_{\partial K}\right|. \tag{4.25}$$

4.1.4 Implementation

In our algorithm, the user defines an eigenvalue problem, provides a target eigenvalue, λ_{target}, and a tolerance ε. To implement the algorithm, we use Firedrake's `LinearEigenSolver` class, which uses SLEPc in the backend [40]. The `LinearEigenSolver` class takes in the argument `LinearEigenProblem`, which itself takes in two bilinear forms, a and m, where m is the representation of the inner product on V, and corresponding boundary conditions [17].

To compute the adjoint eigenvalue problem, we use Firedrake's `adjoint` function [19]. The enriched bilinear forms are obtained by using Firedrake p-multigrid's

`PMGPC.reconstruct_degree` function to reconstruct the general element in $\hat{V}$ [19]. We then replace the trial and test function arguments of the bilinear forms. This is the same procedure implemented for PDEs, and works for general elements of arbitrary order and shape.

The evaluation of the global and local error estimators for a single eigenvalue of a nonsymmetric problem requires the computation of four eigenpairs, $\tilde{u} \coloneqq \{\tilde{v}, \tilde{\lambda}\}$, $u_h \coloneqq \{v_h, \lambda_h\}$, $\tilde{u^*} \coloneqq \{\tilde{v^*}, \tilde{\lambda^*}\}$, $u_h^* \coloneqq \{v_h^*, \lambda_h^*\}$.

We compute each of these separately, using SLEPc's `target_eps` solver parameter, which enables one to target a particular eigenvalue [40]. However, there is no guarantee that the eigenpair returned for each problem is the corresponding eigenpair to the primal problem. (There is also no way to check that the computed eigenvalue for the primal problem is the correct one, but we take it and use refinement to ensure that it converges to the correct eigenvalue). To solve this problem, we compute a user-defined number of eigenpairs for each problem, for which the default is 5. Once computed, the respective eigenpairs are matched by considering the shape of the eigenfunctions. For simplicity, we assume now that all the eigenvalues are simple, that is, $\tau_\lambda = \kappa_\lambda = 1$. Eigenvalues of multiplicity greater than one are more complicated because one must consider the entire eigenspace and match basis functions accordingly.

Algorithm 3 Eigenvalue matching

Let $(\cdot, \cdot)$ denote the Hermitian mass inner product induced by the mass bilinear form $m(\cdot, \cdot)$ and $\|\phi\| = \sqrt{(\phi, \phi)}$ its norm. The user provides a reference mode $u \in V \setminus \{0\}$ and a finite set of candidate eigenfunctions $\{w_j\}_{i=j}^{N_{\text{ev}}} \subset W$ (e.g. from a degree-raised or adjoint solve), with associated eigenvalues $\{\lambda_j\}_{j=1}^{N_{\text{ev}}}$.

1. **Normalize.** Set $\hat{u} := u/\|u\|$ and $\hat{w}_j := w_j/\|w_j\|$ for $j = 1, \ldots, N_{\text{ev}}$.
2. **Correlate.** For each j, compute the complex overlap $c_j := (\hat{u}, \hat{w}_j)$ and the phase-invariant score $s_j := |c_j| \in [0, 1]$.
3. **Select.** Choose $j^\star \in \arg\max_{1 \le j \le m} s_j$. Set $w^\star \leftarrow \hat{w}_{j^\star}$ and $c^\star \leftarrow c_{j^\star}$.
4. **Phase align.** If $c^\star \neq 0$, define $\phi := \overline{c^\star}/|c^\star|$ and replace $w^\star \leftarrow \phi\, w^\star$ so that $(\hat{u}, w^\star) = |c^\star| \in \mathbb{R}_{\ge 0}$. (This removes the arbitrary complex phase/sign.)
5. Return phase-aligned eigenfunction w^* and associated eigenvalue $\lambda_{j^\star}$.

We are now in a position to state the entire adaptive eigenvalue algorithm.

Algorithm 4 Eigenproblem adaptive algorithm

Let $a : V \times V \to \mathbb{R}$ be a given bilinear form and $m : V \times V \to \mathbb{R}$ a given symmetric bilinear form, characterising an eigenproblem

$$a(\cdot, \cdot) = \lambda m(\cdot, \cdot). \tag{4.26}$$

The user provides a target eigenvalue λ_{target}, a tolerance $\epsilon > 0$, and a finite-element formulation of (4.26) on spaces $V_h \subset V$ and $\hat{V}_h \subset \hat{V}$, including the element family and degree, defined on an initial tessellation $\mathbb{T}_h$ of the domain Ω.

1. Using the provided target eigenvalue λ_{target}, compute the closest eigenpair $\{v_h, \lambda_h\}$ of the primal problem.
2. Using the computed λ_h as the new target, compute the closest N_{ev} enriched primal eigenpairs $\{\tilde{v}_i, \tilde{\lambda}_i\}_{i=1}^{N}$, and if the problem is nonsymmetric, compute the closest N_{ev} adjoint and enriched adjoint eigenpairs $\{v_{h,i}^*, \lambda_{h,i}^*\}_{i=1}^{N}$ $\{\tilde{v_i^*}, \tilde{\lambda_i^*}\}_{i=1}^{N}$.
3. Apply Algorithm 3 to correlate the eigenpair sets computed in Step 2 correlated with the primal eigenpair to obtain the corresponding eigenpair.
4. Evaluate the global error estimate, using (4.22) for a nonsymmetric problem or (4.23) for a symmetric problem.
5. If $|\eta_h| \leq \epsilon$, accept the solution u_h and break.
6. Compute the cell and facet residuals R_K and $R_{\partial K}$ of the residual representation (4.25) by solving the local problems (2.50) and (2.51), using the primal residual $\rho(u_h; \varphi)$. For nonsymmetric problems, also compute R_K^* and $R_{\partial K}^*$ using the adjoint residual.
7. Compute the error indicators using (4.24) for a nonsymmetric problem or (4.25) for a symmetric problem.
8. Sort the error indicators in order of magnitude and apply the Dörfler marking strategy discussed in Section 2.5.1.
9. Refine all cells marked for refinement.
10. Go back to step 1.

4.2 Numerical examples

4.2.1 Maxwell eigenproblem

We begin by investigating the Maxwell problem posed in [20], that is

$$\begin{aligned} \nabla \times (\nabla \times \mathbf{E}) - \lambda \mathbf{E} = 0, \quad & \text{in } \Omega, \\ n \times \mathbf{E} = 0, \quad & \text{on } \partial\Omega, \end{aligned} \tag{4.27}$$

where we seek the spatially varying electric field $\mathbf{E} = \mathbf{E}(x, y)$ and the eigenvalue (square of the wave-number) $\lambda = k_0^2$. Physical eigenpairs of (4.27) correspond to those with eigenvalues greater than zero. Eigenfunctions of positive eigenvalues automatically satisfy the divergence condition $\nabla \cdot \mathbf{E} = 0$. Spurious modes cluster exclusively around zero and may be eliminated in post-processing [12]. When reduced to a 2D cross-section, $\Omega \subset \mathbb{R}^2$, the strong form (4.27) admits the following variational formulation: We seek $\{u, \lambda\} \in H_0(\text{curl}; \Omega) \times \mathbb{R}^+$ such that

$$a(u, v) = \lambda m(u, v), \quad \forall v \in H_0(\text{curl}; \Omega), \tag{4.28}$$

with

$$\begin{aligned} a(u, v) &= (\nabla \times u, \nabla \times v) \\ m(u, v) &= (u, v), \end{aligned}$$

and

$$H_0(\text{curl}; \Omega) = \{u \in H(\text{curl}; \Omega) : n \times u = 0 \text{ on } \partial\Omega\}.$$

We discretize (4.28) with the Galerkin approximation $\{u_h, \lambda_h\}$, with $u_h \in V_h \subset H_0(\text{curl}; \Omega)$, using degree 3 Nédélec elements of the first kind [33]. We are interested in minimising the approximation error of the eigenvalue λ_h via adaptive refinement:

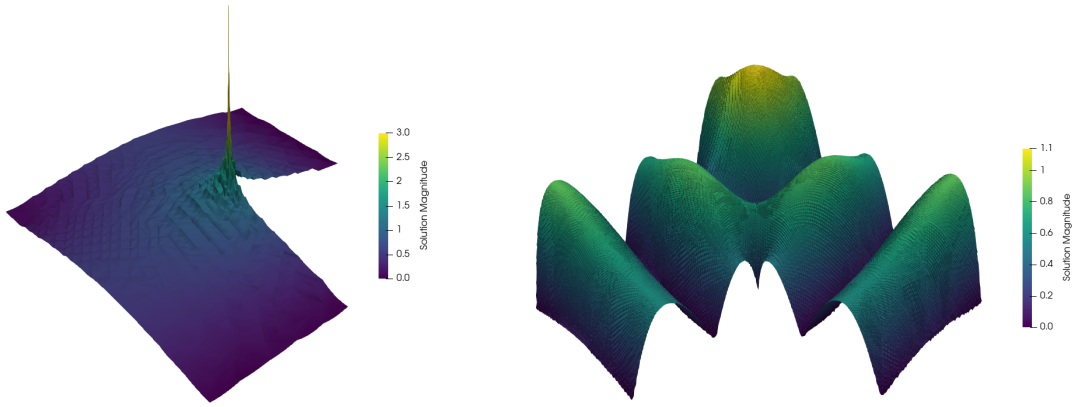
$$\text{error} = |\lambda - \lambda_h|. \tag{4.29}$$

The domain is $\Omega \coloneqq (-1, 1) \times (-1, 1) \backslash (0, 1) \times (-1, 0)$.

We note that the bilinear form $a(\cdot, \cdot)$ is symmetric, so we can use the simplified global error estimate (4.23) and local error estimate (4.25). Using Algorithm 4 we obtain solutions for the first eigenpair, third eigenvalue and eigenspace, and the eighth eigenpair. The first eigenvalue corresponds to a singular eigenfunction, with the singularity located at the reentrant corner at $(0, 0)$. The third eigenvalue is of multiplicity 2 and has an eigenspace which is spanned by two orthogonal eigenfunctions under the L^2 inner product, as induced by $m(\cdot, \cdot)$. The eighth eigenvalue is included to demonstrate convergence for a higher eigenvalue. We take the benchmark eigenvalues from [20], which are:

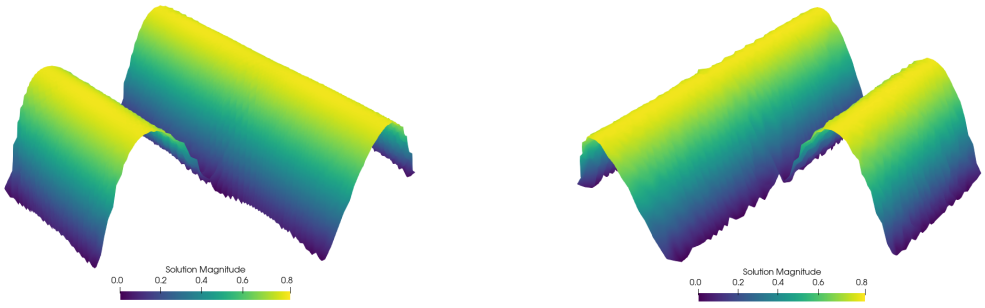
1. First eigenvalue (multiplicity 1): 1.47562182397
2. Third eigenvalue (multiplicity 2): π^2
3. Eighth eigenvalue (multiplicity 1): 23.344371957137

We note that [20] counts the multiple eigenvalue as two eigenvalues, so the eight eigenvalue is listed as the ninth in this paper. The computed eigenfunctions corresponding to the 1st and eighth eigenvalue are shown in Figure 4.1, and a computed basis for the eigenspace corresponding to the 3rd eigenvalue is shown in Figure 4.2.



(a) 1st eigenvalue (b) 8th eigenvalue

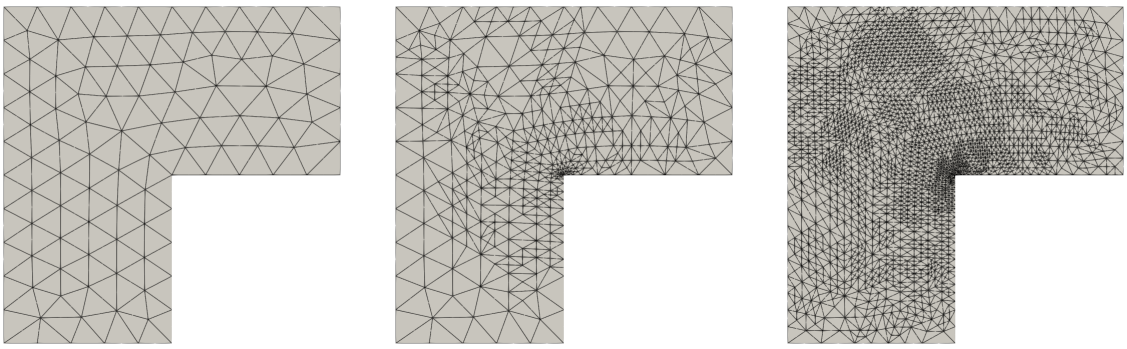
Figure 4.1: Eigenfunctions corresponding to the 1st and 8th eigenvalues.



(a) Basis 1 (b) Basis 2

Figure 4.2: Eigenspace basis functions of the π^2 eigenvalue.

The mesh refinement for the first eigenpair is shown in Figure 4.3. As expected, refinement is concentrated at the singularity located at $(0, 0)$. This matches with the refinement behaviour seen in [20].



(a) Initial (b) Iteration 6 (c) Iteration 12

Figure 4.3: Successive meshes for the 1st eigenpair.

Errors and effectivity indices are shown in Figure 4.4. We see much faster convergence than uniform refinement for both the first and eight (labelled 9th eigenvalue, in line with [20]) eigenvalue, but comparable convergence rates for the third eigenvalue. This is somewhat expected, because the eigenspace spans much of the domain. We have also applied a naive algorithm for eigenvalues of multiplicity 1 to an eigenvalue of multiplicity 2, which is why we see very large effectivity indices $\sum_K \eta_K$ for the third eigenvalue. SLEPc selects an arbitrary basis at each mesh refinement level, and the basis for the enriched eigenfunction $\tilde{v}_i$ may differ from the basis for $v_{h,i}$.

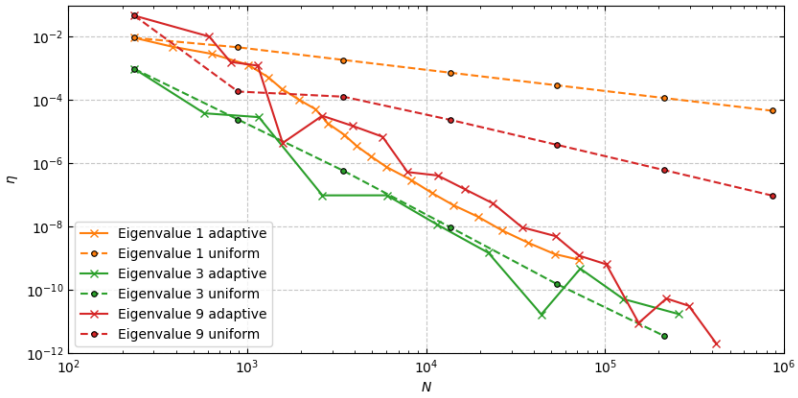
A proper treatment of eigenvalues of multiplicity > 1 is a straightforward extension to Algorithm 4. Firstly, the computed eigenbasis for the enriched eigenspace must be m-orthonormalized against the computed eigenbasis for the finite element solution. This will ensure that e (or e^* if nonsymmetric is used) is computed correctly. Secondly, we compute error indicators for the entire eigenspace, instead of flip-flopping between whatever basis function comes first at each iteration. The error indicators can then be combined on each cell as

$$\eta_K^2 = \sum_j^d \eta_{K,j}^2, \tag{4.30}$$

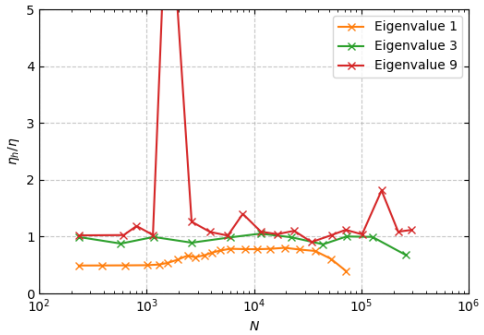
where d is the dimension of the eigenspace (and the multiplicity of the eigenvalue). This combined estimator remains consistent regardless of the basis returned by SLEPc.

Finally, we note that we achieve good effectivity index η_h/η for all three eigenvalues. The singular eigenfunction, the first eigenvalue, has the worst effectivity, although it i still within an acceptable order of accuracy. Additional error is introduced when approximating higher eigenvalues due to the nature of the iterative solver, combined with the fact that the corresponding eigenfunction is not globally smooth [20], explains the slightly more oscillatory behaviour for the eighth eigenvalue

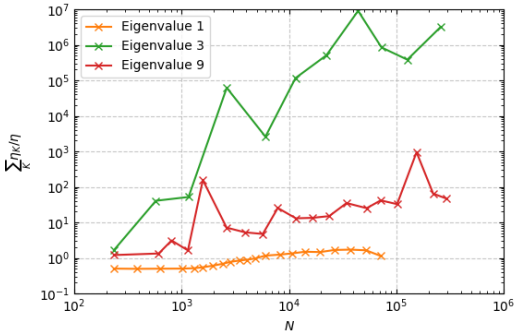
compared to the third, which is globally smooth.



(a) Errors, adaptive vs uniform refinement.



(b) Adaptive refinement η_h/η.



(c) Adaptive refinement $\sum_K \eta_K/\eta$.

Figure 4.4: Errors and effectivity indices for selected eigenvalues of the Maxwell problem.

4.2.2 Stokes eigenproblem

The next problem we investigate is a mixed problem with a mass matrix which is not the L^2 inner product. Consider the velocity-pressure Stokes eigenvalue problem [9]: Find an eigenpair (u, p, λ) with $u \neq 0$ such that

$$\begin{aligned} -\Delta u + \nabla p &= \lambda u \quad \text{in } \Omega \\ \operatorname{div} u &= 0 \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{4.31}$$

Before moving on to the weak formulation, we define the function spaces

$$V = H_0^1(\Omega; \mathbb{R}^2), \tag{4.32}$$

$$Q = L_0^2(\Omega) = \left\{ p \in L^2(\Omega) : \int_\Omega p = 0 \right\}. \tag{4.33}$$

The standard weak formulation of the Stokes eigenvalue problem (4.31) is obtained as follows: Find $(u, p, \lambda) \in V \times L_0^2(\Omega) \times \mathbb{R}^+$ with $\|u\|_0 = 1$ such that

$$\begin{aligned} a(u, v) + b(v, p) &= \lambda(u, v), \qquad && \forall v \in V \\ b(u, q) &= 0, && \forall q \in Q, \end{aligned} \tag{4.34}$$

where $a(u, v) = (\nabla u, \nabla v)$ and $b(v, p) = (-\operatorname{div} v, p)$.

We discretize (4.38) with the standard Taylor-Hood finite element scheme [46]. That is, we use the space of piecewise quadratic elements for the velocity components, and the space of piecewise linear elements for the pressure. The spaces $V_h \subset V$ and $Q_h \subset L^2(\Omega)$ are defined as

$$\mathcal{P}^p(\mathbb{T}_h) := \{v : H^1(\Omega) : v|_K \text{ is a polynomial with degree} \leq p, K \in \mathbb{T}_h\}, \tag{4.35}$$

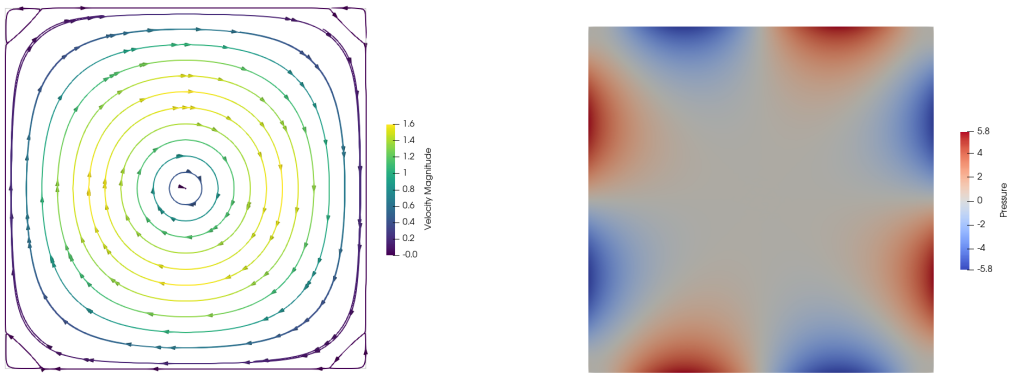
$$V_h := (\mathcal{P}^2(\mathbb{T}_h))^2 \cap (H_0^1(\Omega))^2, \tag{4.36}$$

$$Q_h := \mathcal{P}^1(\mathbb{T}_h) \cap L_0^2(\Omega). \tag{4.37}$$

The Galerkin formulation is then: Find $(u_h, p_h, \lambda_h) \in V_h \times Q_h \times \mathbb{R}^+$ with $\|u\|_0 = 1$ such that

$$\begin{aligned} a(u_h, v_h) + b(v_h, p_h) &= \lambda_h(u_h, v_h), \qquad && \forall v_h \in V_h \\ b(u_h, q_h) &= 0, && \forall q_h \in Q_h. \end{aligned} \tag{4.38}$$

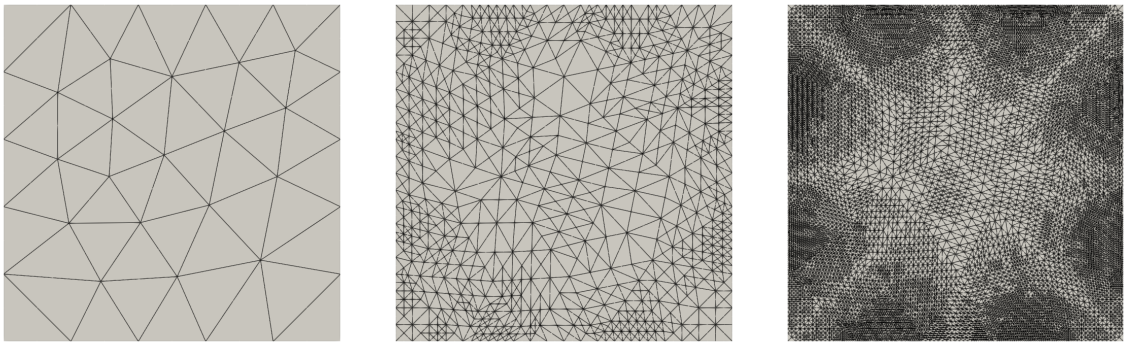
We target the first eigenvalue, which is known. Figure 4.5 shows the computed solution at the final mesh refinement level.



(a) Velocity (b) Pressure

Figure 4.5: Velocity and pressure eigenfunctions corresponding to the first eigenvalue of the Stokes eigenproblem.

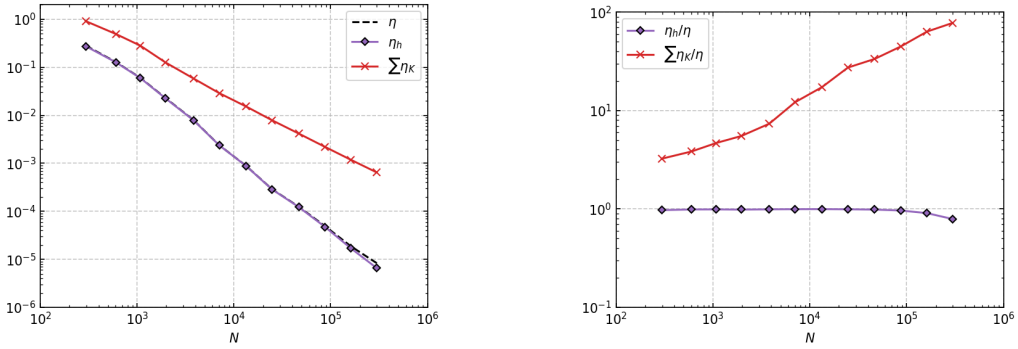
Figure 4.6 shows selected meshes generated by the adaptive algorithm. Refinement is concentrated where the pressure deviates from the baseline, suggesting that this field dominate contribution to the error in the eigenvalue.



(a) Initial (b) Iteration 6 (c) Iteration 12

Figure 4.6: Successive meshes for the first eigenvalue of the stokes eigenproblem.

The errors and effectivity indices are shown in Figure 4.7. The effectivity η_h/η is very good, remaining ≈ 1 throughout. We do observe some deviation on later refinement levels, but there is not enough data to draw conclusions from this. The effectivity $\sum_K \eta_K$ not particularly sharp and increase as the mesh is refined, which suggests that flux terms dominate the error of this problem.



(a) Errors. (b) Effectivity indices.

Figure 4.7: Errors and effectivity indices for the first eigenvalue of the Stokes eigenproblem.

Chapter 5

Conclusions and Future Work

5.1 Summary

To summarise, we have achieved the objectives set out at the start of the project. Firstly, we have implemented a goal-based adaptive solver which enables the user to pass in a variational problem, goal functional, and tolerance, with the solver doing the rest of the work automatically. It is the author's belief that this solver will be an incredibly useful implementation for Firedrake users seeking to estimate and control the error in finite element simulations henceforth. The solver can also perform uniform refinement, and careful selection of goal functionals can target norm-like errors, point errors, and errors on specific patches of the domain.

Secondly, by reinterpreting the automated portion of the code in the finite element spaces offered by FIAT, we have built an automated algorithm that automatically parallelizes, a significant advantage over the previous implementation by Rognes and Logg.

Thirdly, we have developed and implemented an eigenvalue targetting algorithm based on the DWR method, utilizing the automated local residual estimator. Once rough eigenvalues are known (which could easily be computed on a coarse mesh prior to calling `GoalAdaptiveEigenSolver`), this algorithm could be used to gain accurate solutions of said eigenvalues.

Both algorithms have been extensively tested, and numerical results show that the algorithms work well for a range of problems. The error estimation aspect is not always reliable, particularly for flux-dominated problems, but this is largely a feature of the DWR method and not a fault in the implementation. Stabilized forms offer a method of circumventing this issue, and the DWR theory extends to these as well [7].

5.2 Future Work

For future work, an immediate extension would be develop the eigenproblem solver to target the error in goal functionals of eigenfunctions. The theory is readily available [22], but the presence of the exact goal functional $J(v)$ in the estimator will require some thought to understand localized and automatic error estimators in a similar style to [39]. A fallback could be using the combined *a priori-a posteriori* error estimators favoured by Becker and Rannacher [7].

For PDEs, it would be beneficial to obtain a nonlinear example where the combined Algorithm 2 performs noticeably better than the primal-only Algorithm 1. This was attempted, but it is likely the problems considered in this report were not nonlinear enough, or the goal functional localised enough, for differences to become obvious. Although not stated in this report, the adjoint error estimate ρ^* was closer to the true error, but usually by a factor negligible relative to the error magnitude.

A further extension would be to develop a patchwise extrapolation procedure for efficient approximation of the enriched dual and primal solutions.

References

[1] Martin S. Alnæs, Anders Logg, Kristian B. Ølgaard, Marie E. Rognes, and Garth N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Trans. Math. Softw.*, 40(2), March 2014.

[2] Douglas N. Arnold, Arup Mukherjee, and Luc Pouly. Locally adapted tetrahedral meshes using bisection. *SIAM Journal on Scientific Computing*, 22(2):431–448, 2000.

[3] I. Babuvška and W. C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis*, 15(4):736–754, 1978.

[4] S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. Constantinescu, L. Dalcin, A. Dener, et al. Petsc/tao users manual revision 3.23. Technical report, Argonne National Laboratory (ANL), 03 2025.

[5] Wolfgang Bangerth and Rolf Rannacher. Adaptive finite element techniques for the acoustic wave equation. *Journal of Computational Acoustics*, 9(02):575–591, 2001.

[6] Igor A. Baratta, Joseph P. Dean, Jørgen S. Dokken, Michal Habera, Jack S. Hale, Chris N. Richardson, Marie E. Rognes, Matthew W. Scroggs, Nathan Sime, and Garth N. Wells. Dolfinx: The next generation fenics problem solving environment, December 2023.

[7] Roland Becker, Vincent Heuveline, and Rolf Rannacher. An optimal control approach to adaptivity in computational fluid mechanics. *International journal for numerical methods in fluids*, 40(1-2):105–120, 2002.

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

[9] Daniele Boffi and Arbaz Khan. Adaptive mixed fem for the stokes eigenvalue problem, 2025.

[10] Alexander N. Brooks and Thomas J. R. Hughes. 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(1):199–259, 1982.

[11] Lisandro D. Dalcin, Rodrigo R. Paz, Pablo A. Kler, and Alejandro Cosimo. Parallel distributed computing using python. *Advances in Water Resources*, 34(9):1124–1139, 2011. New Computational Methods and Software Tools.

[12] Patrik Daniel, Alexandre Ern, and Martin Vohralík. An adaptive hp-refinement strategy with inexact solvers and computable guaranteed bound on the error reduction factor. *Computer Methods in Applied Mechanics and Engineering*, 359:112607, 2020.

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

[14] Bernhard Endtmayer, Ulrich Langer, and Thomas Wick. Reliability and efficiency of dwr-type a posteriori error estimates with smart sensitivity weight recovering. *Computational Methods in Applied Mathematics*, 21(2):351–371, 2021.

[15] Donald Estep and Donald French. Global error control for the continuous Galerkin finite element method for ordinary differential equations. *ESAIM: Modélisation mathématique et analyse numérique*, 28(7):815–852, 1994.

[16] Donald J Estep, Mats G Larson, and Roy D Williams. *Estimating the error of numerical solutions of systems of reaction-diffusion equations*, volume 696. American Mathematical Soc., 2000.

[17] Firedrake Project. Source code for `firedrake.eigensolver`. `https://www.firedrakeproject.org/_modules/firedrake/eigensolver.html`. Accessed 2025-09-10.

[18] Andrew Gillette, Brendan Keith, and Socratis Petrides. Learning robust marking policies for adaptive mesh refinement. *SIAM Journal on Scientific Computing*, 46(1):A264–A289, 2024.

[19] David A. Ham, Paul H. J. Kelly, Lawrence Mitchell, Colin J. Cotter, Robert C. Kirby, Koki Sagiyama, Nacime Bouziani, Sophia Vorderwuelbecke, Thomas J. Gregory, Jack Betteridge, Daniel R. Shapero, Reuben W. Nixon-Hill, Connor J. Ward, Patrick E. Farrell, Pablo D. Brubeck, India Marsden, Thomas H. Gibson, Miklós Homolya, Tianjiao Sun, Andrew T. T. McRae, Fabio Luporini, Alastair Gregory, Michael Lange, Simon W. Funke, Florian Rathgeber, Gheorghe-Teodor Bercea, and Graham R. Markall. *Firedrake User Manual*. Imperial College London and University of Oxford and Baylor University and University of Washington, first edition edition, 5 2023.

[20] Jake J. Harmon and Branislav M. Notaroš. Adaptive hp-refinement for 2-d maxwell eigenvalue problems: Method and benchmarks. *IEEE Transactions on Antennas and Propagation*, 70(6):4663–4673, 2022.

[21] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[22] Vincent Heuveline and Rolf Rannacher. A posteriori error control for finite element approximations of elliptic eigenvalue problems. *Advances in Computational Mathematics*, 15(1):107–138, 2001.

[23] John G Heywood, Rolf Rannacher, and Stefan Turek. Artificial boundaries and flux and pressure conditions for the incompressible navier–stokes equations. *International Journal for numerical methods in fluids*, 22(5):325–352, 1996.

[24] Johan Hoffman. On duality-based a posteriori error estimation in various norms and linear functionals for large eddy simulation. *SIAM Journal on Scientific Computing*, 26(1):178–195, 2004.

[25] Tosio Kato. *Perturbation Theory for Linear Operators*. Classics in Mathematics. Springer, Berlin, reprint of the 2nd ed. edition, 1995.

[26] Robert C. Kirby. Algorithm 839: Fiat, a new paradigm for computing finite element basis functions. *ACM Trans. Math. Softw.*, 30(4):502–516, December 2004.

[27] Mats G. Larson and Timothy J. Barth. A posteriori error estimation for adaptive discontinuous galerkin approximations of hyperbolic systems. In Bernardo Cockburn, George E. Karniadakis, and Chi-Wang Shu, editors, *Discontinuous Galerkin Methods*, pages 363–368, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[28] Mats G Larson and Fredrik Bengzon. Adaptive finite element approximation of multiphysics problems. *Communications in Numerical Methods in Engineering*, 24(6):505–521, 2008.

[29] Mats G Larson and Axel Målqvist. Adaptive variational multiscale methods based on a posteriori error estimation: duality techniques for elliptic problems. In *Multiscale methods in science and engineering*, pages 181–193. Springer, 2005.

[30] Fredrik Larsson, Peter Hansbo, and Kenneth Runesson. Strategies for computing goal-oriented a posteriori error measures in non-linear elasticity. *International Journal for Numerical Methods in Engineering*, 55(8):879–894, 2002.

[31] A. Limache, S. Idelsohn, R. Rossi, and E. Oñate. The violation of objectivity in laplace formulations of the navier–stokes equations. *International Journal for Numerical Methods in Fluids*, 54(6-8):639–664, 2007.

[32] Pedro Morin, Ricardo H. Nochetto, and Kunibert G. Siebert. Convergence of adaptive finite element methods. *SIAM Review*, 44(4):631–658, 2002.

[33] J.-C. Nédélec. Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 35(3):315–341, 1980.

[34] Ricardo H. Nochetto, Andreas Veeser, and Marco Verani. A safeguarded dual weighted residual method. *IMA Journal of Numerical Analysis*, 29(1):126–140, 2009.

[35] OpenCFD Ltd. *OpenFOAM User Guide: Layer addition (`snappyHexMesh`)*, 2025. Accessed 10 Sep 2025.

[36] Rolf Rannacher and Franz-Theo Suttmeier. A posteriori error control in finite element methods via duality techniques: Application to perfect plasticity. *Computational Mechanics*, 21(2):123–133, 1998.

[37] Sabine Richling, Erik Meinköhn, N Kryzhevoi, and Guido Kanschat. Radiative transfer with finite elements-i. basic method and tests. *Astronomy & Astrophysics*, 380(2):776–788, 2001.

[38] Thomas Richter. Goal-oriented error estimation for fluid–structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 223-224:28–42, 2012.

[39] Marie E. Rognes and Anders Logg. Automated goal-oriented error control I: stationary variational problems. *SIAM Journal on Scientific Computing*, 35(3):C173–C193, 2013.

[40] J. E. Roman, C. Campos, L. Dalcin, E. Romero, and A. Tomas. SLEPc users manual. Technical Report DSIC-II/24/02 - Revision 3.23, D. Sistemes Informàtics i Computació, Universitat Politècnica de València, 2025.

[41] Robert Sandboge. Adaptive finite element methods for systems of reaction-diffusion equations. *Computer methods in applied mechanics and engineering*, 166(3-4):309–328, 1998.

[42] Robert Sandboge. Adaptive finite element methods for reactive compressible flow. *Mathematical Models and Methods in Applied Sciences*, 09(02):211–241, 1999.

[43] Joachim Schöberl. NETGEN: An advancing front 2d/3d-mesh generator based on abstract rules. *Computing and Visualization in Science*, 1(1):41–52, 1997.

[44] Kristoffer Selim. An adaptive finite element solver for fluid–structure interaction problems. In Anders Logg, Kent-Andre Mardal, and Garth Wells, editors, *Automated Solution of Differential Equations by the Finite Element Method*, volume 84 of *Lecture Notes in Computational Science and Engineering*. Springer, 2012.

[45] Rob Stevenson. Optimality of a standard adaptive finite element method. *Foundations of Computational Mathematics*, 7:245–269, 2007.

[46] C. Taylor and P. Hood. A numerical solution of the navier-stokes equations using the finite element technique. *Computers & Fluids*, 1(1):73–100, 1973.

[47] Rüdiger Verfürth. A review of a posteriori error estimation techniques for elasticity problems. *Computer Methods in Applied Mechanics and Engineering*, 176(1):419–440, 1999.

[48] Umberto Zerbinati and Patrick E. Farrell. Netgen integration in firedrake. `https://www.firedrakeproject.org/firedrake/demos/netgen_mesh.py.html`. Firedrake Project documentation, accessed September 9, 2025.