Adaptive Algorithms for Goal-Oriented Error Estimation: Comparison of Different Error Representations

Anne Marie Conway

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the
degree of Master of Science Mathematics and Statistics*

University of Ottawa

*The M.Sc. program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics

Abstract

This thesis focuses on approximating a quantity of interest $Q(u)$, where u is the solution to a Partial Differential Equation (PDE). With this goal, it compares various approaches to adaptive algorithms based on the Finite Element (FE) method. Specifically, it examines several globally equivalent error representations used in goal-oriented adaptivity. It borrows previously developed error representations and draws a comparison of their efficiency and accuracy with newly introduced representations. The error representations are first defined in the context of an abstract problem using duality techniques and Galerkin orthogonality properties. They are then explicitly defined in the context of two problems: a general 1D convection-diffusion-reaction problem, and a 2D Poisson problem. These representations are incorporated into h-adaptive algorithms for a 1D boundary-layer problem and a 2D Poisson problem, as well as a p-adaptive algorithm for the 1D Helmholtz problem. The adaptive algorithms are tested with different marking strategies and for different quantities of interest. Numerical results highlight the local variation of the error representations and demonstrate their influence on the performance of the algorithms. Finally, these results are used to classify the error representations according to their efficiency and accuracy.

Dedications

To my parents, Louise and Danny, and to my partner, Kareem, whose unwavering support encouraged me to stay motivated and focused throughout my academic journey.

Acknowledgements

I wish to thank my supervisor, Dr. Guignard, for her patience, trust, support, and exceptional teaching throughout my undergraduate and graduate studies. Taking on this project, which was built on the foundations of her work, was both an honour and a valuable learning experience. I am truly grateful for her academic guidance and personal advice during this process. I would also like to thank Dr. Prudhomme, Dr. Kergrene, and Dr. Vacher for their input and expertise during our biweekly meetings.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many problems in science and engineering involve PDEs for which closed-form solutions are not available. In such cases, numerical methods are essential for computing approximate solutions. To fix the idea, consider the weak formulation of a general elliptic boundary value problem, which in abstract form, reads:

$$\text{find } u \in V: \quad B(u,v) = F(v) \quad \forall v \in V.$$

Here, $V = V(\Omega) \subset H^1(\Omega)$ is a Hilbert space with $\Omega \subset \mathbb{R}^d$ an open bounded domain for $d \in \{1,2,3\}$, $B : V \times V \to \mathbb{R}$ is a continuous and coercive bilinear form, and $F : V \to \mathbb{R}$ is a continuous linear functional. The Galerkin *Finite Element* (FE) method is a numerical method designed for approximately solving such problems in various applications. It involves the discretization of the domain Ω with the construction of a mesh comprised of a finite number of subdomains, or *elements*. The solution u is approximated by a piecewise polynomial function $\tilde{u}$, which is obtained by computing $B(\cdot,\cdot)$ and $F(\cdot)$ on each element and solving a linear system. In FE error analysis, numerous techniques have been developed to approximate the discretization error $u - \tilde{u}$ in a global norm, for instance, the L^2 norm

$$\|u - \tilde{u}\|_{L^2(\Omega)} = \sqrt{\int_\Omega |u - \tilde{u}|^2 dx},$$

the H^1 norm

$$\|u - \tilde{u}\|_{H^1(\Omega)} = \sqrt{\|u - \tilde{u}\|^2_{L^2(\Omega)} + \|\nabla(u - \tilde{u})\|^2_{L^2(\Omega)}}$$

or the energy norm

$$\|u - \tilde{u}\|_B = \sqrt{B(u - \tilde{u}, u - \tilde{u})},$$

in the case where $B(\cdot,\cdot)$ is symmetric. *A priori* error estimates approximate the error using the norm of the exact solution u and the mesh size. They are used to provide bounds on the error and to assess the convergence rate of the numerical method, prior to computing the numerical solution. In most cases, these estimates are not computable as they involve the exact solution. Alternatively, *a posteriori* error estimates use the computed numerical solution $\tilde{u}$ to approximate the error, applying various techniques. These techniques include residual methods [2,5,7,16,23,30], recovery methods [8,31–34], and duality methods [18]. An extensive survey of these methods for linear elliptic diffusion problems can be found in [11].

In many scientific and engineering applications, there are problems where the full solution u is not of primary interest, and only a specific quantity of interest $Q(u)$ is considered relevant. One example is the transportation of pollution in water. Let E_1 and E_2 be two pollutant sources placed in a river characterized by a dry area (e.g., an island) localized in the center of the domain. The aim could be to evaluate the average value of the pollutant concentration u in a zone D of interest (e.g., a fish or beach area), computed by $Q(u) = \int_D u$ [15].

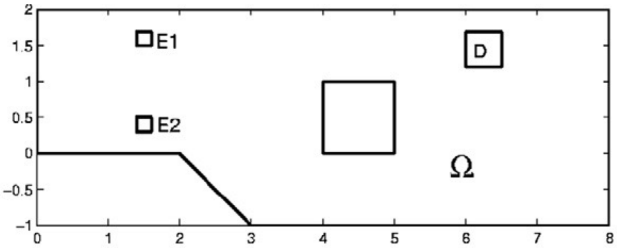


Figure 1.1: Domain and zone of interest for pollution problem

When dealing with these scenarios in mesh adaptivity, applying global a posteriori error estimates can be highly inefficient. Instead, it is more optimal to target the error in $Q(u)$ directly. This approach is known as *Goal-oriented error estimation*. It focuses on accurately approximating the solution in the quantity of interest $Q(u)$, and on quantifying the approximation error in this quantity. The method is designed to capture the primal error $e = u - \tilde{u}$ of the original problem weighted by the dual error ε of a well-chosen adjoint problem corresponding to the quantity of interest. In most applications, a single goal-oriented error representation is chosen and upper-bounded by localized estimators to drive an algorithm for mesh adaptivity.

One widely used error representation is the bilinear form $B(\cdot,\cdot)$, evaluated in the primal and dual errors. Estimators for this error representation are derived through various techniques, such as applying the Cauchy-Schwarz inequality [3]

$$B(e,\varepsilon) \leq \|e\|_B \|\varepsilon\|_B,$$

applying a scaling factor s to the energy norm of the primal and dual errors [25, 27]

$$B\left(e,\varepsilon\right) = B\left(se, \frac{\varepsilon}{s}\right) = \frac{1}{4}\|se + \frac{\varepsilon}{s}\|_B^2 - \frac{1}{4}\|se - \frac{\varepsilon}{s}\|_B^2,$$

and applying a gradient recovery operator to approximate e and ε inside $B(\cdot,\cdot)$ [26].

Another error representation is the *dual-weighted residual* (DWR), which expresses the error in the quantity of interest in terms of the primal residual $F(\cdot) - B(\tilde{u},\cdot)$ and the dual approximation error ε [9, 10]. The DWR method has been successfully applied to a variety of goal-oriented applications [6, 24].

A third and more recently developed error representation involves the Riesz representant of the adjoint residual, solved with respect to a chosen symmetric and positive definite bilinear form. The error in the quantity of interest is expressed as the alternative bilinear form, evaluated at the Riesz representant and the primal error e. This method was applied with different bilinear forms to the 1D [12] and multidimensional [13] Helmholtz equations. It was shown to provide sharper error estimations, driving a more efficient p-adaptive process. Similar conclusions were drawn from applying the method to a one-dimensional convection-dominated diffusion problem, where the alternative estimator successfully captured the boundary layer [14].

In an attempt to reduce the arbitrariness in the choice of error representations, this thesis builds upon these existing formulations and introduces new globally equivalent alternatives. It examines how the decomposition into local contributions differs from one representation to another. Drawing on their different local contributions, it presents an in-depth comparison of their efficiency and accuracy in the context of different one-dimensional and two-dimensional goal-oriented problems. The subsequent content of the thesis is divided into four chapters. Chapter 2 describes the goal-oriented error estimation problem at an abstract level and establishes some fundamental results leading to five globally equivalent error representations. Chapter 3 defines the error estimators in the context of two model problems: a general 1D diffusion-convection-reaction problem with homogeneous Dirichlet boundary conditions, and the 2D Poisson problem on an L-shaped domain. The estimators are explicitely defined for different boundary conditions, such as nonhomogeneous Dirichlet boundary conditions in the 2D case. Chapter 4 focuses on the implementation of the estimates in adaptive algorithms, and summarizes numerical tests conducted with different quantities of interest on a 1D boundary-layer problem, the 1D Helmholtz equation and a 2D Poisson problem. Chapter 5 summarizes the thesis and states its main conclusions.

Chapter 2

Abstract problem

In this chapter, we define the weak formulation of an abstract problem, and introduce five goal-oriented error representations.

For $d = 1, 2$ or 3, let $\Omega \subset \mathbb{R}^d$ be an open bounded domain with boundary $\partial\Omega$ and let $V = V(\Omega)$ be a Hilbert space. Consider the *primal problem*

$$\text{find } u \in V: \quad B(u, v) = F(v) \quad \forall v \in V, \tag{2.1}$$

with $B : V \times V \to \mathbb{R}$ a continuous and coercive bilinear form and $F : V \to \mathbb{R}$ a continuous linear functional. We are interested in computing the solution of this problem for a specific *Quantity of Interest* (QoI) $Q(u)$, where $Q : V \to \mathbb{R}$ is a given continuous functional that we assume to be linear. The QoI can be, for example, the solution $u(x^*)$ at a point $x^* \in \Omega$ (provided u is sufficiently smooth, i.e. $u \in C^0(\Omega)$), or the integral $\int_{\Omega^*} u(x)dx$ over a subdomain $\Omega^* \subset \bar{\Omega}$. This quantity is represented using the *dual* or *adjoint problem*

$$\text{find } z \in V: \quad B(v, z) = Q(v) \quad \forall v \in V. \tag{2.2}$$

By the Lax-Milgram Lemma (Theorem 5.1.1 in [29]), both the primal and dual problems are well-posed. Moreover, note that from the primal and dual solutions defined, we have

$$Q(u) = B(u, z) = F(z).$$

To discretize these problems, we use the FE method. We introduce a (conforming) mesh $\mathcal{T}_h$ of $\bar{\Omega}$ with finite elements K of size $h_K \leq h$ and polynomials of degree $p_K \leq p$. We use a hierarchical basis defined on a *reference element* $\hat{K}$ and based on the Legendre polynomials.

For one-dimensional problems, the degree 1 basis functions are defined on the reference interval

$$\hat{K} = \{\hat{x} : -1 \leq \hat{x} \leq 1\}$$

by

$$\left\{ \frac{1 - \hat{x}}{2}, \frac{1 + \hat{x}}{2} \right\}$$

and for the intervals K with $p_K \geq 2$, we add the polynomials $\phi_2(\hat{x}), ..., \phi_{p_K}(\hat{x})$, defined by

$$\phi_j(\hat{x}) = \frac{1}{\sqrt{2(2j-1)}}(P_j(\hat{x}) - P_{j-2}(\hat{x})),$$

where P_j is the Legendre polynomial of degree j.

For two-dimensional problems, the degree 1 vertex basis functions are defined in terms of the barycentric coordinates on the reference triangle

$$\hat{K} = \{(\hat{x}, \hat{y}) : -1 \leq \hat{x} \leq 1, 0 \leq \hat{y} \leq \sqrt{3} - \sqrt{3}|\hat{x}|\}$$

by

$$L_1(\hat{x}, \hat{y}) = \frac{1 - \hat{x} - \frac{\hat{y}}{\sqrt{3}}}{2}, \quad L_2(\hat{x}, \hat{y}) = \frac{1 + \hat{x} - \frac{\hat{y}}{\sqrt{3}}}{2}, \quad L_3(\hat{x}, \hat{y}) = \frac{\hat{y}}{\sqrt{3}}$$

and for the triangles K with $p_K = 2$, we add the 3 edge shape functions

$$\phi_{2,1} = -4\sqrt{6}L_1L_2, \quad \phi_{2,2} = -4\sqrt{6}L_2L_3, \quad \phi_{2,3} = -4\sqrt{6}L_3L_1.$$

We refer to [1, 4] for details.

We then consider

$$V_{h,p} = \{v \in V : v|_K \in \mathbb{P}_{p_K}, \forall K \in \mathcal{T}_h\},$$

where $\mathbb{P}_{p_K}$ is the space of polynomials of degree p_K on the element K. This is a subspace of V and is of finite dimension N_h. We approximate the primal problem by

$$\text{find } u_{h,p} \in V_{h,p} : \quad B(u_{h,p}, v_{h,p}) = F(v_{h,p}) \quad \forall v_{h,p} \in V_{h,p} \tag{2.3}$$

and the dual problem by

$$\text{find } z_{h,p} \in V_{h,p} : \quad B(v_{h,p}, z_{h,p}) = Q(v_{h,p}) \quad \forall v_{h,p} \in V_{h,p}. \tag{2.4}$$

Note that from the primal and dual numerical solutions defined, we have

$$Q(u_{h,p}) = B(u_{h,p}, z_{h,p}) = F(z_{h,p}).$$

Remark 2.1. In general, B and F will also be approximated by some operators $B_{h,p}$ and $F_{h,p}$, respectively, due for instance to the use of quadrature formulae to approximate the underlying integrals. This additional error is not accounted for in what follows, as its value is negligible for the cases examined.

Since we are not interested in the full solution, error estimation in the energy (or any global) norm is not suitable. Our goal is to estimate the error

$$Q(u) - Q(u_{h,p}) = Q(u - u_{h,p})$$

in the QoI by a quantity that is computable and localized. We want the error to be localized on each element of the mesh to allow for adaptivity, by refining the mesh (*h-refinement*), by increasing the polynomial degree on some elements (*p-refinement*) or by combining both approaches (*hp-refinement*). To achieve this, we first define the error in the primal and the dual, by, respectively,

$$e_u = u - u_{h,p} \quad \text{and} \quad e_z = z - z_{h,p}.$$

Noting that for any $v_{h,p} \in V_{h,p} \subset V$,

$$B(e_u, v_{h,p}) = B(u, v_{h,p}) - B(u_{h,p}, v_{h,p}) \overset{(2.1),\,(2.3)}{=} F(v_{h,p}) - F(v_{h,p}) = 0 \tag{2.5}$$

and

$$B(v_{h,p}, e_z) = B(v_{h,p}, z) - B(v_{h,p}, z_{h,p}) \overset{(2.2),\,(2.4)}{=} Q(v_{h,p}) - Q(v_{h,p}) = 0, \tag{2.6}$$

we obtain the *Galerkin orthogonality* for the primal and dual problems, respectively.
For any $v \in V$, we define the residuals for $u_{h,p}$ and $z_{h,p}$ by

$$R_{u_{h,p}}(v) = F(v) - B(u_{h,p}, v) \tag{2.7}$$

and

$$R_{z_{h,p}}(v) = Q(v) - B(v, z_{h,p}). \tag{2.8}$$

Thanks to the primal and dual numerical solutions defined, for all $v_{h,p} \in V_{h,p}$, these satisfy

$$R_{u_{h,p}}(v_{h,p}) \overset{(2.7)}{=} F(v_{h,p}) - B(u_{h,p}, v_{h,p}) \overset{(2.3)}{=} 0 \tag{2.9}$$

and

$$R_{z_{h,p}}(v_{h,p}) \overset{(2.8)}{=} Q(v_{h,p}) - B(v_{h,p}, z_{h,p}) \overset{(2.4)}{=} 0. \tag{2.10}$$

Using these results, we can now introduce several possible representations of the error $Q(e_u)$, which give the same value globally but have different local contributions. From one representation to another, the value of the error estimator on each element will differ. As a result, the performance of an adaptive algorithm will depend on the choice of the error representation.

Remark 2.2. As we shall see, each error representation is *localized* (namely it better captures the error) in either the primal solution, the dual solution, or both solutions.

Notice that all the estimates presented below still contain unknown functions, namely u, z or both, and are thus not computable. At the end of this chapter, we will give a strategy to get computable estimates for the error in the quantity of interest $Q(e_u)$.

First estimate

The first error estimate we introduce, which is widely used in practice (see for instance [27]), is given by

$$Q(e_u) \overset{(2.2)}{=} B(e_u, z) \overset{(2.5)}{=} B(e_u, z) - B(e_u, z_{h,p}) = \boxed{B(e_u, e_z)}. \tag{2.11}$$

The total error is localized by

$$Q(e_u) = \sum_{K \in \mathcal{T}_h} B_K(e_u, e_z),$$

where B_K involves the integral over the element K instead of the whole domain Ω. Similar notation will be used for the other forms, when we replace the integral over the domain Ω to an integral over a subdomain.

Remark 2.3. Notice that, thanks to the Galerkin orthogonality for the primal and the dual, we also have

$$Q(e_u) \overset{(2.11)}{=} B(e_u, e_z) = B(u, e_z) - B(u_{h,p}, e_z) \overset{(2.6)}{=} B(u, e_z) \overset{(2.6)}{=} B(u - \mathcal{I}_{h,p}u, e_z)$$

and

$$Q(e_u) \overset{(2.2)}{=} B(e_u, z) \overset{(2.5)}{=} B(e_u, z - \mathcal{I}_{h,p}z)$$

where $\mathcal{I}_{h,p} : V \to V_{h,p}$ is some interpolant (or projection) operator. More generally, we have

$$Q(e_u) = B(u - v_{h,p}, e_z) = B(e_u, z - v_{h,p})$$

for any $v_{h,p} \in V_{h,p}$.

Second estimate

The second error estimate is based on the residual for $u_{h,p}$. Indeed, we have

$$\begin{aligned} Q(e_u) &\overset{(2.11)}{=} B(e_u, e_z) = B(u, e_z) - B(u_{h,p}, e_z) \\ &\overset{(2.1)}{=} F(e_z) - B(u_{h,p}, e_z) \overset{(2.7)}{=} \boxed{R_{u_{h,p}}(e_z)}. \end{aligned} \tag{2.12}$$

As before, the total error is decomposed into local contributions as

$$Q(e_u) = \sum_{K \in \mathcal{T}_h} R_K^{u_{h,p}}(e_z) = \sum_{K \in \mathcal{T}_h} \left[F_K(e_z) - B_K(u_{h,p}, e_z) \right],$$

where the approximation error inside each element is expressed with respect to the strong form of the equation. We then perform integration by parts to split the residual into interior and jump residuals:

$$Q(e_u) = \sum_{K \in \mathcal{T}_h} R_K^{u_{h,p}}(e_z) = \sum_{K \in \mathcal{T}_h} r_K^{u_{h,p}}(e_z) + \sum_{e \in \mathcal{E}_h} j_e^{u_{h,p}}(e_z),$$

where $\mathcal{E}_h$ is the set of edges in the mesh (this is limited to the interior edges when we have homogeneous Dirichlet boundary conditions). This controls the approximation error of the solution flux across the edges of the elements. The terms are then regrouped per element or per edge. For instance, a choice often made in practice for element based error estimators is to split the jump term across an edge to each of the two adjacent elements, as follows:

$$\eta_K = r_K^{u_{h,p}}(e_z) + \frac{1}{2} \sum_{e \subset \partial K} j_e^{u_{h,p}}(e_z). \tag{2.13}$$

The estimator is then not unique since it involves an arbitrary choice.

Remark 2.4. We also have

$$\begin{aligned} R_{u_{h,p}}(e_z) &\overset{(2.12)}{=} B(e_u, e_z) = B(e_u, z) - B(e_u, z_{h,p}) \\ &\overset{(2.5)}{=} B(e_u, z) = B(u - u_{h,p}, z) \overset{(2.1)}{=} F(z) - B(u_{h,p}, z) \overset{(2.7)}{=} R_{u_{h,p}}(z). \end{aligned}$$

Proceeding in a similar way, we equally have

$$R_{u_{h,p}}(e_z) = R_{u_{h,p}}(z - \mathcal{I}_{h,p} z) = R_{u_{h,p}}(z - v_{h,p})$$

for any $v_{h,p} \in V_{h,p}$.

Third estimate

Similarly, we estimate the error in the quantity of interest using the residual for the dual with

$$Q(e_u) \overset{(2.5)}{=} Q(e_u) - B(e_u, z_{h,p}) \overset{(2.8)}{=} \boxed{R_{z_{h,p}}(e_u)}. \tag{2.14}$$

Locally, we have

$$Q(e_u) = \sum_{K \in \mathcal{T}_h} \left[Q_K(e_u) - B_K(e_u, z_{h,p}) \right].$$

Integrating by parts, we once again obtain a split into interior and jump residuals. This involves an arbitrary choice for the local contributions similar to (2.13), when defining an element (or edge-based) estimator.

Remark 2.5. We also have

$$R_{z_{h,p}}(e_u) \overset{(2.14),\,(2.11)}{=} B(e_u, e_z) = B(u, e_z) - B(u_{h,p}, e_z)$$
$$\overset{(2.6)}{=} B(u, e_z) = B(u, z - z_{h,p}) \overset{(2.2)}{=} Q(u) - B(u, z_{h,p}) \overset{(2.8)}{=} R_{z_{h,p}}(u).$$

Proceeding in a similar way, we equally have

$$R_{z_{h,p}}(e_u) = R_{z_{h,p}}(u - \mathcal{I}_{h,p}u) = R_{z_{h,p}}(u - v_{h,p})$$

for any $v_{h,p} \in V_{h,p}$.

Fourth estimate

Another method to estimate $Q(e_u)$ is to introduce the so-called Riesz representant of the residual $R_{u_{h,p}}$. To do so, let $\mathcal{A} : V \times V \to \mathbb{R}$ be a symmetric and coercive bilinear form, i.e. an inner product. Then, define $\varphi^u \in V$ as the Riesz representant of $R_{u_{h,p}}$ with respect to the scalar product induced by $\mathcal{A}$, namely the solution of

$$\text{find } \varphi^u \in V : \quad \mathcal{A}(\varphi^u, v) = R_{u_{h,p}}(v) \quad \forall v \in V. \tag{2.15}$$

We then have

$$Q(e_u) \overset{(2.12)}{=} R_{u_{h,p}}(e_z) \overset{(2.15)}{=} \boxed{\mathcal{A}(\varphi^u, e_z)} \tag{2.16}$$

and, locally,

$$Q(e_u) = \sum_{K \in \mathcal{T}_h} \mathcal{A}_K(\varphi^u, e_z).$$

Fifth estimate

Of course, we could also use the Riesz representant of the residual for $z_{h,p}$ as follows. We have

$$Q(e_u) \overset{(2.14)}{=} R_{z_{h,p}}(e_u) = \boxed{\mathcal{A}(e_u, \varphi^z)} \tag{2.17}$$

with φ^z the solution of

$$\text{find } \varphi^z \in V : \quad \mathcal{A}(v, \varphi^z) = R_{z_{h,p}}(v) \quad \forall v \in V. \tag{2.18}$$

For the local contributions, we get

$$Q(e_u) = \sum_{K \in \mathcal{T}_h} \mathcal{A}_K(e_u, \varphi^z).$$

Remark 2.6. The five error estimates presented above have all previously appeared in the literature, as mentioned in Chapter 1. In particular, estimates (2.11), (2.12), and (2.17) have been applied in various contexts, while estimates (2.14) and (2.16) have been mentioned more briefly. Building on these existing formulations, we can derive additional error estimates using a variety of techniques. One approach involves introducing the partition of unity. If $\{\varphi_i\}_{i=0}^{N+1}$ is a family of functions on Ω such that $\sum_{i=0}^{N+1} \varphi_i(x) = 1$ for all $x \in \Omega$, for example, the standard basis functions for linear ($\mathbb{P}1$) finite elements, then we can replace the second estimate $R_{u_{h,p}}(e_z)$ with $R_{u_{h,p}}(e_z \sum_i \varphi_i)$. This gives an estimate without arbitrary splitting that yields a patch-based error indicator, as defined in (3.2.17) below. The same trick can be used for any other error estimate. We can also combine some of the estimates presented above. For example, taking the average of the fourth (2.16) and fifth (2.17) estimates yields:

$$Q(e_u) = \frac{1}{2}\left[\mathcal{A}(\varphi^u, e_z) + \mathcal{A}(e_u, \varphi^z)\right].$$

Computable estimates

All the quantities introduced in (2.11), (2.12), (2.14), (2.16) and (2.17) are not computable since they depend on u, z or both. To have computable error estimates, we introduce a *richer* finite element subspace $\tilde{V}$ which satisfies $V_{h,p} \subset \tilde{V} \subset V$. Typically, $\tilde{V}$ is obtained by considering a refined mesh of $\mathcal{T}_h$ (e.g. $V_{h/2,p}$), by using the same mesh with a higher polynomial degree on each element (e.g. $V_{h,p+1}$), or by combining both (e.g. $V_{h/2,p+1}$).

Let $\tilde{u}$, $\tilde{z}$, $\tilde{\varphi}^u$ and $\tilde{\varphi}^z$ be the solutions of, respectively,

$$\text{find } \tilde{u} \in \tilde{V} \quad : \quad B(\tilde{u},\tilde{v}) = F(\tilde{v}) \quad \forall \tilde{v} \in \tilde{V} \tag{2.19}$$

$$\text{find } \tilde{z} \in \tilde{V} \quad : \quad B(\tilde{v},\tilde{z}) = Q(\tilde{v}) \quad \forall \tilde{v} \in \tilde{V} \tag{2.20}$$

$$\text{find } \tilde{\varphi}^u \in \tilde{V} \quad : \quad \mathcal{A}(\tilde{\varphi}^u,\tilde{v}) = R_{u_{h,p}}(\tilde{v}) \quad \forall \tilde{v} \in \tilde{V} \tag{2.21}$$

$$\text{find } \tilde{\varphi}^z \in \tilde{V} \quad : \quad \mathcal{A}(\tilde{v},\tilde{\varphi}^z) = R_{z_{h,p}}(\tilde{v}) \quad \forall \tilde{v} \in \tilde{V}. \tag{2.22}$$

Note that from the primal and dual numerical solutions defined, we have

$$Q(\tilde{u}) = B(\tilde{u},\tilde{z}) = F(\tilde{z}).$$

Additionally, from these problems, we can write $Q(e_u) = Q(u-\tilde{u}) + Q(\tilde{u}-u_{h,p})$, and approximate the error in the QoI by dropping the first term:

$$Q(e_u) \approx Q(\tilde{u}-u_{h,p}).$$

The estimates (2.11), (2.12), (2.14), (2.16) and (2.17) of $Q(e_u)$ are then replaced by, respectively,

$$\boxed{B(\tilde{u}-u_{h,p},\tilde{z}-z_{h,p})}, \boxed{R_{u_{h,p}}(\tilde{z}-z_{h,p})}, \boxed{R_{z_{h,p}}(\tilde{u}-u_{h,p})}, \boxed{\mathcal{A}(\tilde{\varphi}^u,\tilde{z}-z_{h,p})} \text{ and } \boxed{\mathcal{A}(\tilde{u}-u_{h,p},\tilde{\varphi}^z)}. \tag{2.23}$$

Proposition 2.7. All the quantities appearing in (2.23) have the same global value and are equal to $Q(\tilde{u}-u_{h,p})$.

Proof. To simplify the notation, we write

$$e_u = e_1 + e_2 \quad \text{with} \quad e_1 = u-\tilde{u} \text{ and } e_2 = \tilde{u}-u_{h,p}$$

and

$$e_z = \varepsilon_1 + \varepsilon_2 \quad \text{with} \quad \varepsilon_1 = z-\tilde{z} \text{ and } \varepsilon_2 = \tilde{z}-z_{h,p}.$$

The various *Galerkin orthogonalities* read

$$B(e_u, v_{h,p}) \overset{(2.5)}{=} 0 \quad \forall v_{h,p} \in V_{h,p} \tag{2.24}$$

$$B(e_1, \tilde{v}) \overset{(2.1),\,(2.19)}{=} 0 \quad \forall \tilde{v} \in \tilde{V} \text{ (particularly } \forall v_{h,p} \in V_{h,p} \subset \tilde{V}) \tag{2.25}$$

$$B(e_2, v_{h,p}) \overset{(2.19),\,(2.3)}{=} 0 \quad \forall v_{h,p} \in V_{h,p} \tag{2.26}$$

$$B(v_{h,p}, e_z) \overset{(2.6)}{=} 0 \quad \forall v_{h,p} \in V_{h,p} \tag{2.27}$$

$$B(\tilde{v}, \varepsilon_1) \overset{(2.2),\,(2.20)}{=} 0 \quad \forall \tilde{v} \in \tilde{V} \text{ (particularly } \forall v_{h,p} \in V_{h,p} \subset \tilde{V}) \tag{2.28}$$

$$B(v_{h,p}, \varepsilon_2) \overset{(2.20),\,(2.4)}{=} 0 \quad \forall v_{h,p} \in V_{h,p} \tag{2.29}$$

$$\mathcal{A}(\varphi^u - \tilde{\varphi}^u, \tilde{v}) \overset{(2.15),\,(2.21)}{=} 0 \quad \forall \tilde{v} \in \tilde{V} \tag{2.30}$$

$$\mathcal{A}(\tilde{v}, \varphi^z - \tilde{\varphi}^z) \overset{(2.18),(2.22)}{=} 0 \quad \forall \tilde{v} \in \tilde{V}. \tag{2.31}$$

We can easily show that in any case, the neglected part is the same, namely equal to $B(e_1,\varepsilon_1)$ which is nothing else than $Q(u-\tilde{u})$:

$$B(e_1,\varepsilon_1) = B(e_1,z) - B(e_1,\tilde{z}) \overset{(2.25)}{=} B(u-\tilde{u},z) \overset{(2.2)}{=} Q(u-\tilde{u}). \tag{2.32}$$

- For the first estimator (2.11), we have

$$Q(e_u) \overset{(2.11)}{=} B(e_u, e_z) = B(e_1, \varepsilon_1) + B(e_2, \varepsilon_2) \overset{(2.32)}{=} \underbrace{Q(u - \tilde{u})}_{\text{neglected term}} + \underbrace{B(e_2, \varepsilon_2)}_{\text{estimator}}.$$

  Therefore, we have

$$Q(e_u) = Q(u - \tilde{u}) + B(e_2, \varepsilon_2) \implies Q(\tilde{u} - u_{h,p}) = B(e_2, \varepsilon_2).$$

- For the second estimator (2.12), we have the splitting

$$Q(e_u) \overset{(2.12)}{=} R_{u_{h,p}}(e_z) = R_{u_{h,p}}(\varepsilon_1 + \varepsilon_2) = \underbrace{R_{u_{h,p}}(\varepsilon_1)}_{\text{neglected term}} + \underbrace{R_{u_{h,p}}(\varepsilon_2)}_{\text{estimator}}$$

  and we have

$$\begin{aligned} R_{u_{h,p}}(\varepsilon_1) &\overset{(2.7)}{=} F(\varepsilon_1) - B(u_{h,p}, \varepsilon_1) \\ &\overset{(2.1)}{=} B(u, \varepsilon_1) - B(u_{h,p}, \varepsilon_1) = B(u - \tilde{u}, \varepsilon_1) + B(\tilde{u} - u_{h,p}, \varepsilon_1) \\ &\overset{(2.28)}{=} B(e_1, \varepsilon_1) \overset{(2.32)}{=} Q(u - \tilde{u}). \end{aligned}$$

  Therefore, we get

$$Q(e_u) = Q(u - \tilde{u}) + R_{u_{h,p}}(\varepsilon_2) \implies Q(\tilde{u} - u_{h,p}) = R_{u_{h,p}}(\varepsilon_2).$$

- Similarly to the previous case, for the estimator (2.14) we have

$$Q(e_u) \overset{(2.14)}{=} R_{z_{h,p}}(e_u) = R_{z_{h,p}}(e_1 + e_2) = \underbrace{R_{z_{h,p}}(e_1)}_{\text{neglected term}} + \underbrace{R_{z_{h,p}}(e_2)}_{\text{estimator}}$$

  and

$$\begin{aligned} R_{z_{h,p}}(e_1) &\overset{(2.8)}{=} Q(e_1) - B(e_1, z_{h,p}) \\ &\overset{(2.2)}{=} B(e_1, z) - B(e_1, z_{h,p}) = B(e_1, z - \tilde{z}) + B(e_1, \tilde{z} - z_{h,p}) \\ &\overset{(2.25)}{=} B(e_1, \varepsilon_1) \overset{(2.32)}{=} Q(u - \tilde{u}). \end{aligned}$$

  Therefore, we have

$$Q(e_u) = Q(u - \tilde{u}) + R_{z_{h,p}}(e_2) \implies Q(\tilde{u} - u_{h,p}) = R_{z_{h,p}}(e_2).$$

- For (2.16), the estimator and the neglected terms are

$$\begin{aligned} Q(e_u) \overset{(2.16)}{=} \mathcal{A}(\varphi^u, e_z) &= \mathcal{A}(\varphi^u, \varepsilon_1 + \varepsilon_2) \\ &= \mathcal{A}(\varphi^u, \varepsilon_1) + \mathcal{A}(\varphi^u - \tilde{\varphi}^u + \tilde{\varphi}^u, \varepsilon_2) \end{aligned}$$

$$= \underbrace{\mathcal{A}(\varphi^u, \varepsilon_1) + \mathcal{A}(\varphi^u - \tilde{\varphi}^u, \varepsilon_2)}_{\text{neglected term}} + \underbrace{\mathcal{A}(\tilde{\varphi}^u, \varepsilon_2)}_{\text{estimator}}.$$

Then, using the proof for the second estimate we easily get

$$\mathcal{A}(\varphi^u, \varepsilon_1) + \mathcal{A}(\varphi^u - \tilde{\varphi}^u, \varepsilon_2) \overset{(2.30)}{=} \mathcal{A}(\varphi^u, \varepsilon_1) \overset{(2.15)}{=} R_{u_{h,p}}(\varepsilon_1) \overset{(2.7),(2.1),(2.28)}{=} B(e_1, \varepsilon_1) \overset{(2.32)}{=} Q(u - \tilde{u}).$$

Therefore, we have

$$Q(e_u) = Q(u - \tilde{u}) + \mathcal{A}(\tilde{\varphi}^u, \varepsilon_2) \implies Q(\tilde{u} - u_{h,p}) = \mathcal{A}(\tilde{\varphi}^u, \varepsilon_2).$$

- Finally, for (2.17) we get

$$\begin{aligned} Q(e_u) \overset{(2.17)}{=} \mathcal{A}(e_u, \varphi^z) &= \mathcal{A}(e_1 + e_2, \varphi^z) \\ &= \mathcal{A}(e_1, \varphi^z) + \mathcal{A}(e_2, \varphi^z - \tilde{\varphi}^z + \tilde{\varphi}^z) \\ &= \underbrace{\mathcal{A}(e_1, \varphi^z) + \mathcal{A}(e_2, \varphi^z - \tilde{\varphi}^z)}_{\text{neglected term}} + \underbrace{\mathcal{A}(e_2, \tilde{\varphi}^z)}_{\text{estimator}} \end{aligned}$$

and, similarly to the previous case,

$$\mathcal{A}(e_1, \varphi^z) + \mathcal{A}(e_2, \varphi^z - \tilde{\varphi}^z) \overset{(2.31)}{=} \mathcal{A}(e_1, \varphi^z) \overset{(2.18)}{=} R_{z_{h,p}}(e_1) \overset{(2.8),(2.2),(2.25)}{=} B(e_1, \varepsilon_1) \overset{(2.32)}{=} Q(u - \tilde{u}).$$

Therefore, we have

$$Q(e_u) = Q(u - \tilde{u}) + \mathcal{A}(e_2, \tilde{\varphi}^z) \implies Q(\tilde{u} - u_{h,p}) = \mathcal{A}(e_2, \tilde{\varphi}^z).$$

∎

From what precedes, we see that there are many different ways to estimate the error in the quantity of interest $Q(e_u)$. Even though all the error representations are the same globally, the situation is different when looking at the local contributions. Indeed, as illustrated in the numerical results presented in Chapter 4, the local error indicators are different from one representation to another, obviously yielding different results when using adaptive algorithms. More precisely, the error indicators will be localized in the primal, in the dual or in both depending on the choice of representation of the error in the QoI.

We now consider specific problems, namely a one-dimensional diffusion-convection-reaction problem and a two-dimensional Poisson problem, in order to give more details on the various error estimates introduced above for the abstract problem (2.1).

Chapter 3

Specific Problems

3.1 Notations and fundamental results

For $d = 1, 2$ or 3, let $\Omega \subset \mathbb{R}^d$ be an open bounded domain with boundary $\partial\Omega$. Throughout this chapter, we will use the following notations and results.

Definition 3.1.1. For $p \geq 1$, we define the Lebesgue space

$$L^p(\Omega) = \left\{ f : \Omega \to \mathbb{R} : \int_\Omega |f(x)|^p d\Omega < \infty \right\}$$

and the norm

$$\|v\|_{L^p(\Omega)} = \left(\int_\Omega |v(x)|^p d\Omega \right)^{1/p}.$$

Definition 3.1.2. Let k be a positive integer.

1. The Sobolev space of order k on Ω is defined as :

   $$H^k(\Omega) = \{ f \in L^2(\Omega) : D^\alpha f \in L^2(\Omega) \quad \forall \alpha : |\alpha| \leq k \},$$

   with the scalar product

   $$(f, g)_k = \sum_{|\alpha| \leq k} \int_\Omega (D^\alpha f)(D^\alpha g) d\Omega,$$

   the norm

   $$\|f\|_k = \|f\|_{H^k(\Omega)} = \sqrt{(f, f)_k} = \sqrt{\sum_{|\alpha| \leq k} \int_\Omega (D^\alpha f)^2 d\Omega}$$

   and the semi-norm

   $$|f|_k = |f|_{H^k(\Omega)} = \sqrt{\sum_{|\alpha| = k} \int_\Omega (D^\alpha f)^2 d\Omega}.$$

2. If $1 \leq p \leq \infty$, we define the Sobolev space

   $$W^{k,p}(\Omega) = \{ v \in L^p(\Omega) : D^\alpha v \in L^p(\Omega) \quad \forall \alpha : |\alpha| \leq k \},$$

   with the norm

   $$\|v\|_{W^{k,p}(\Omega)} = \left(\sum_{|\alpha| \leq k} \|D^\alpha v\|^p_{L^p(\Omega)} \right)^{1/p}.$$

Here,

$$D^\alpha v = \frac{\partial^{|\alpha|} v}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \dots \partial x_n^{\alpha_n}}$$

is the derivative of v in the distribution sense for the multi-index $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$ of length $|\alpha| = \alpha_1 + \alpha_2 + \cdots + \alpha_n$.

Theorem 3.1.3. *(Hölder inequality)* Let $v \in L^p(\Omega)$ and $w \in L^{\bar{p}}(\Omega)$, with $1 \leq p \leq \infty$, $1 \leq \bar{p} \leq \infty$ and $\frac{1}{p} + \frac{1}{\bar{p}} = 1$. Then,

$$vw \in L^1(\Omega)$$

and

$$\int_\Omega |v(x)w(x)| d\Omega \leq ||v||_{L^p(\Omega)} ||w||_{L^{\bar{p}}(\Omega)}.$$

We refer to Theorem 5.5.2 in [22] for details.

Definition 3.1.4. Considering $k = 1$, we define

$$H_0^1(\Omega) = \{v \in H^1(\Omega) : v|_{\partial\Omega} = 0\}.$$

Theorem 3.1.5. *(Poincaré inequality)* There exists a constant $C_p > 0$ such that

$$\|v\|_{L^2(\Omega)} \leq C_p |v|_{H^1(\Omega)}, \quad \forall v \in H_0^1(\Omega).$$

We refer to Property 2.4 in [28] for details.

Remark 3.1.6. From Theorem 3.1.5, we have that for any $v \in H_0^1(\Omega)$,

$$\|v\|^2_{H^1(\Omega)} = \|v\|^2_{L^2(\Omega)} + |v|^2_{H^1(\Omega)} \leq C_p^2 |v|^2_{H^1(\Omega)} + |v|^2_{H^1(\Omega)} = (C_p^2 + 1)|v|^2_{H^1(\Omega)}.$$

Therefore,

$$|v|_{H^1(\Omega)} \leq \|v\|_{H^1(\Omega)} \leq \sqrt{C_p^2 + 1} |v|_{H^1(\Omega)},$$

meaning $|v|_{H^1(\Omega)}$ and $\|v\|_{H^1(\Omega)}$ are equivalent and the space $H_0^1(\Omega)$ is equipped with the norm

$$\|v\|_{H_0^1(\Omega)} = |v|_{H^1(\Omega)}.$$

3.2 Specific 1D Problem

Let $\Omega = (0, 1)$. We consider the diffusion-convection-reaction (primal) problem

$$\begin{cases} -(au')' + bu' + cu = f \text{ in } (0,1) \\ u(0) = u(1) = 0, \end{cases}$$

where $f \in L^2(\Omega)$ is a given function, and where for well-posedness, the coefficients satisfy

- $a, c \in L^\infty(\Omega)$;
- $b \in W^{1,\infty}(\Omega)$;
- $a(x) \geq a_0 > 0 \quad \forall x \in (0,1)$;
- $c(x) - \frac{1}{2} b'(x) \geq 0 \quad \forall x \in (0,1)$.

For the quantity of interest, we choose

$$Q(u) = u(x^*)$$

with x^* some given point in $(0,1)$. Setting $V = H_0^1(\Omega)$ and multiplying the PDE by a test function $v \in V$, we have

$$\begin{aligned}\int_0^1 fvdx &= \int_0^1 (-(au')'v + bu'v + cuv)dx \\ &= \int_0^1 au'v'dx - [au'v]_0^1 + \int_0^1 bu'vdx + \int_0^1 cuvdx \qquad = \int_0^1 (au'v' + bu'v + cuv)dx.\end{aligned}$$

Therefore, the problem under consideration can be put in the abstract formulation of Chapter 2 with

$$B(u,v) = \int_0^1 (au'v' + bu'v + cuv)dx, \tag{3.2.1}$$

$$F(v) = \int_0^1 fvdx \tag{3.2.2}$$

and

$$Q(v) = \langle \delta_{x^*}, v \rangle_{H^{-1}(\Omega)\times H_0^1(\Omega)}, \tag{3.2.3}$$

where δ_{x^*} is the Delta Dirac function centered at x^*.

Remark 3.2.1. Since $\delta_{x^*} \notin L^2(\Omega)$, we cannot express $Q(v)$ as an integral over Ω. However, we have that $\delta_{x^*} \in H^{-1}(\Omega)$ (see Section 4.2 in [19] for details). This is the dual of $H_0^1(\Omega)$, equipped with the norm $\|\cdot\|_{H^{-1}(\Omega)}$ defined by

$$\|h\|_{H^{-1}(\Omega)} = \sup_{v\in H_0^1(\Omega)} \frac{\langle h, v\rangle_{H^{-1}(\Omega)\times H_0^1(\Omega)}}{\|v\|_{H_0^1(\Omega)}}.$$

Therefore, we express the quantity of interest as

$$Q(v) = v(x^*) = \langle \delta_{x^*}, v \rangle_{H^{-1}(\Omega)\times H_0^1(\Omega)}.$$

According to Property 2.3 in [28], one-dimensional functions in $H^1(\Omega)$ are continuous. Therefore, by this property, the quantity $Q(v)$ is well-defined in $H_0^1(\Omega)$.

For the dual problem, for all $v \in V$, we look for $z \in V$ such that $B(v,z) = Q(v)$, i.e.

$$\int_0^1 (av'z' + bv'z + cvz)dx = \langle \delta_{x^*}, v \rangle_{H^{-1}(\Omega)\times H_0^1(\Omega)}, \quad \forall v \in V.$$

Applying integration by parts, we have

$$\begin{aligned}\int_0^1 (av'z' + bv'z + cvz)dx &= [avz']_0^1 - \int_0^1 (az')'vdx + [bvz]_0^1 - \int_0^1 (bz)'vdx + \int_0^1 cvzdx \\ &= \int_0^1 (-(az')' - (bz)' + cz)vdx,\end{aligned}$$

from which we derive the strong formulation of the problem:

$$\begin{cases} -(az')' - (bz)' + cz = \delta_{x^*} \text{ in } (0,1) \\ z(0) = z(1) = 0. \end{cases}$$

Remark 3.2.2. According to the Lax-Milgram Lemma (Theorem 5.1.1 in [29]), we can validate the existence and uniqueness of a solution to the primal problem and a solution to the dual problem, by verifying the following hypotheses:

1. $B(u,v)$ is bilinear,
2. $B(u,v)$ is continuous,
3. $B(u,v)$ is coercive,
4. $F(v)$ and $Q(v)$ are linear,
5. $F(v)$ and $Q(v)$ are continuous.

The bilinearity of B and linearity of F and Q are clear, so we focus on verifying 2, 3 and 5. We choose the norm $\|.\|_{H_0^1(\Omega)}$.

2. For all $u, v \in H_0^1(\Omega)$, we have

$$\begin{aligned}
B(u,v) &= \int_0^1 (au'v' + bu'v + cuv)dx = \int_0^1 au'v'dx + \int_0^1 bu'vdx + \int_0^1 cuvdx \\
&\overset{\text{Thm 3.1.3}}{\leq} \|a\|_{L^\infty(\Omega)}\|u'\|_{L^2(\Omega)}\|v'\|_{L^2(\Omega)} + \|b\|_{L^\infty(\Omega)}\|u'\|_{L^2(\Omega)}\|v\|_{L^2(\Omega)} \\
&\quad + \|c\|_{L^\infty(\Omega)}\|u\|_{L^2(\Omega)}\|v\|_{L^2(\Omega)} \\
&= \|a\|_{L^\infty(\Omega)}\|u\|_{H_0^1(\Omega)}\|v\|_{H_0^1(\Omega)} + \|b\|_{L^\infty(\Omega)}\|u\|_{H_0^1(\Omega)}\|v\|_{L^2(\Omega)} \\
&\quad + \|c\|_{L^\infty(\Omega)}\|u\|_{L^2(\Omega)}\|v\|_{L^2(\Omega)} \\
&\overset{\text{Thm 3.1.5}}{\leq} \|a\|_{L^\infty(\Omega)}\|u\|_{H_0^1(\Omega)}\|v\|_{H_0^1(\Omega)} + C_p\|b\|_{L^\infty(\Omega)}\|u\|_{H_0^1(\Omega)}\|v\|_{H_0^1(\Omega)} \\
&\quad + C_p^2\|c\|_{L^\infty(\Omega)}\|u\|_{H_0^1(\Omega)}\|v\|_{H_0^1(\Omega)} \\
&= \left(\|a\|_{L^\infty(\Omega)} + C_p\|b\|_{L^\infty(\Omega)} + C_p^2\|c\|_{L^\infty(\Omega)}\right)\|u\|_{H_0^1(\Omega)}\|v\|_{H_0^1(\Omega)} \\
&=: C\|u\|_{H_0^1(\Omega)}\|v\|_{H_0^1(\Omega)}.
\end{aligned}$$

   $\Rightarrow B(u,v)$ is continuous.

3. For any $u \in H_0^1(\Omega)$,

$$\begin{aligned}
B(u,u) &= \int_0^1 a(u')^2dx + \int_0^1 bu'udx + \int_0^1 cu^2dx \\
&\geq a_0\|u'\|_{L^2(\Omega)} + \int_0^1 bu'udx + \int_0^1 cu^2dx \\
&= a_0\|u'\|_{L^2(\Omega)} + \frac{1}{2}\int_0^1 b(u^2)'dx + \int_0^1 cu^2dx \\
&= a_0\|u'\|_{L^2(\Omega)} + \left[\frac{1}{2}b(u^2)\right]_0^1 - \frac{1}{2}\int_0^1 b'u^2dx + \int_0^1 cu^2dx \\
&= a_0\|u'\|_{L^2(\Omega)} - \frac{1}{2}\int_0^1 b'u^2dx + \int_0^1 cu^2dx \\
&= a_0\|u'\|_{L^2(\Omega)} + \int_0^1 \left(c - \frac{1}{2}b'\right)u^2dx \\
&\geq a_0\|u'\|_{L^2(\Omega)} = a_0\|u\|_{H_0^1(\Omega)}.
\end{aligned}$$

   $\Rightarrow B(u,v)$ is coercive.

5. For all $v \in H_0^1(\Omega)$, we have

$$\begin{aligned}
\int_0^1 fvdx &\overset{\text{Thm 3.1.3}}{\leq} \|f\|_{L^2(\Omega)}\|v\|_{L^2(\Omega)} \\
&\overset{\text{Thm 3.1.5}}{\leq} C_p\|f\|_{L^2(\Omega)}\|v\|_{H_0^1(\Omega)}
\end{aligned}$$

$$=: C\|v\|_{H_0^1(\Omega)}.$$

For $Q(\cdot)$, we have

$$\begin{aligned}\langle \delta_{x^*}, v\rangle_{H^{-1}(\Omega)\times H_0^1(\Omega)} &\leq \|\delta_{x^*}\|_{H^{-1}(\Omega)}\|v\|_{H_0^1(\Omega)}\\ &=: C\|v\|_{H_0^1(\Omega)}.\end{aligned}$$

$\Rightarrow F(v)$ and $Q(v)$ are linear and bounded, therefore continuous.

The hypotheses of the Lax-Milgram Lemma are all satisfied, and we can conclude that the primal and dual problems have unique solutions.

Remark 3.2.3. In the case where the QoI is an integral over a subdomain $\Omega^* \subset \Omega$ rather than a solution at a point, we use $Q(v) = \int_{\Omega^*} v dx$. Consequently, the right-hand side of the PDE in the dual problem will be $\mathbb{1}_{\Omega^*}$.

Remark 3.2.4. In the nonhomogeneous Neumann case, the boundary conditions will appear in the linear functional $F(v)$ and in the strong formulation of the dual problem. For instance, if we impose the Neumann boundary condition $u'(1) = g$, then we will have

$$F(v) = \int_0^1 fv dx + agv(1)$$

and

$$\begin{cases} -(az')' - (bz)' + cz = \delta_{x^*} \text{ in } (0,1)\\ z(0) = 0\\ az'(1) + bz(1) = 0. \end{cases}$$

For $\mathcal{A}$, we consider the three following cases :

$$\mathcal{A}_1(u,v) = \int_0^1 a(x)u'(x)v'(x)dx, \tag{3.2.4}$$

$$\mathcal{A}_2(u,v) = \mathcal{A}_1(u,v) + \frac{1}{2}\int_0^1 b(x)\left(u'(x)v(x) + u(x)v'(x)\right)dx, \tag{3.2.5}$$

$$\mathcal{A}_3(u,v) = \mathcal{A}_2(u,v) + \int_0^1 c(x)u(x)v(x)dx, \tag{3.2.6}$$

but (infinitely many) other options are possible. We now give the five error estimates corresponding to the ones introduced in (2.23), that can all be written under the form

$$\eta = \sum_{K\in\mathcal{T}_h} \eta_K$$

with $|\eta_K|$ an indicator of the error on element $K \in \mathcal{T}_h$.

Remark 3.2.5. Notice that we have

$$|Q(e_u)| \approx |Q(\tilde{u} - u_{h,p})| = \left|\sum_{K\in\mathcal{T}_h} \eta_K\right| \leq \sum_{K\in\mathcal{T}_h} |\eta_K|. \tag{3.2.7}$$

When running an adaptive algorithm (based on an element error estimator), the quantities $|\eta_K|$ are required as local error indicators for the elements $K \in \mathcal{T}_h$ to know if such elements should be refined or not. However, the quantity $\sum_{K\in\mathcal{T}_h} |\eta_K|$ is not of primary interest since it might highly overestimate the error in the QoI due to the use of the triangle inequality. One might assume that in the case of a sharp inequality, we have a good error representation. This is not the case. If we consider an error that is equi-distributed among the elements with local estimators of the same sign, we have an equality, but such a case is not suitable for a problem with localized error as it does not narrow down the number of elements to be refined.

Since we are considering the 1D case, to define the various error representations in a precise way, let us introduce the following (non-necessarily uniform) partition of the unit interval

$$0 = x_0 < x_1 < \cdots < x_N < x_{N+1} = 1$$

that consists of $N+2$ nodes and $N+1$ elements $[x_i, x_{i+1}]$, $i = 0, ..., N$, of length $h_i = x_{i+1} - x_i$.

Let $\{\varphi_i\}_{i=0}^{N+1}$ be the standard basis functions for linear ($\mathbb{P}_1$) FE. We consider hierarchical FE because in this framework, it is easy to increase the polynomial degree on each subinterval. We characterize the FE space $V_{h,p}$ by the vectors $h \in \mathbb{R}_+^{N+1}$ and $p \in \mathbb{N}_+^{N+1}$, which indicate the length size of and polynomial degree on each subinterval $[x_i, x_{i+1}]$ for $i = 0, \ldots, N$, respectively.

As we have set $V = H_0^1(\Omega)$, the dimension N_h of the space $V_{h,p}$ is the number of internal nodes plus the additional degrees of freedom introduced by the polynomial degrees on each element:

$$N_h = N + \sum_{i=0}^{N}(p_i - 1) = N - (N+1) + \sum_{i=0}^{N} p_i = -1 + \sum_{i=0}^{N} p_i.$$

Denoting $\{\psi_i\}_{i=1}^{N_h}$ as the basis for $V_{h,p}$, we have $\{\varphi_i\}_{i=1}^{N} \subset \{\psi_i\}_{i=1}^{N_h}$. Associating φ_0 and φ_{N+1} to the boundary nodes x_0 and x_{N+1}, we set $\psi_0 = \varphi_0$ and $\psi_{N_h+1} = \varphi_{N+1}$. These two functions are excluded from the basis of $V_{h,p}$ because of the homogeneous Dirichlet boundary conditions.

First estimator

In the first case based on (2.11), we have

$$\boxed{\eta^{(1)} = \sum_{i=0}^{N} \eta_i^{(1)} \quad \text{with} \quad \eta_i^{(1)} = B_i(\tilde{u} - u_{h,p}, \tilde{z} - z_{h,p})} \tag{3.2.8}$$

where for $u, v \in H^1(\Omega)$,

$$B_i(u,v) = \int_{x_i}^{x_{i+1}} a(x)u'(x)v'(x)dx + \int_{x_i}^{x_{i+1}} b(x)u'(x)v(x)dx + \int_{x_i}^{x_{i+1}} c(x)u(x)v(x)dx.$$

Second estimator

Before giving the error estimator, let us define the right and left derivatives of a function g at a point x_i, denoted $g'(x_i)^+$ and $g'(x_i)^-$ respectively, by

$$g'(x_i)^+ = \lim_{s \to 0^+} \frac{g(x_i + s) - g(x_i)}{s} \quad \text{and} \quad g'(x_i)^- = \lim_{s \to 0^+} \frac{g(x_i) - g(x_i - s)}{s}.$$

Moreover, we introduce the jump of g' across x_i for $i = 1, ..., N$, denoted $[g']_i$, by

$$[g']_i = g'(x_i)^+ - g'(x_i)^- \tag{3.2.9}$$

and we set $[g']_0 = [g']_{N+1} = 0$, which is consistent with the homogeneous Dirichlet boundary conditions. The second error estimator, based on (2.12), then reads

$$\boxed{\eta^{(2)} = \sum_{i=0}^{N} \eta_i^{(2)} \quad \text{with} \quad \eta_i^{(2)} = R^i_{u_{h,p}}(\tilde{z} - z_{h,p})} \tag{3.2.10}$$

where, for $v \in V$,

$$R^i_{u_{h,p}}(v) = \int_{x_i}^{x_{i+1}} \left(f(x) + (a(x)u'_{h,p}(x))' - b(x)u'_{h,p}(x) - c(x)u_{h,p}(x)\right) v(x)dx + \frac{1}{2}\left(a(x_i)[u'_{h,p}]_i v(x_i) + a(x_{i+1})[u'_{h,p}]_{i+1} v(x_{i+1})\right).$$

This is derived from the fact that

$$\begin{aligned} R_{u_{h,p}}(v) &= \sum_{i=0}^{N} \left\{ \int_{x_i}^{x_{i+1}} f(x)v(x)dx - \int_{x_i}^{x_{i+1}} \left(a(x)u'_{h,p}(x)v'(x) + b(x)u'_{h,p}(x)v(x) + c(x)u_{h,p}(x)v(x)\right) dx \right\} \\ &= \sum_{i=0}^{N} \left\{ \int_{x_i}^{x_{i+1}} \left(f(x) + (a(x)u'_{h,p}(x))' - b(x)u'_{h,p}(x) - c(x)u_{h,p}(x)\right) v(x)dx - \left[a(x)u'_{h,p}(x)v(x)\right]_{x_i}^{x_{i+1}} \right\} \end{aligned}$$

and

$$\begin{aligned} \sum_{i=0}^{N} \left[a(x)u'_{h,p}(x)v(x)\right]_{x_i}^{x_{i+1}} &= \sum_{i=0}^{N} \left(a(x_{i+1})v(x_{i+1})u'_{h,p}(x_{i+1})^- - a(x_i)v(x_i)u'_{h,p}(x_i)^+\right) \\ &= -a(x_0)v(x_0)u'_{h,p}(x_0)^+ + a(x_{N+1})v(x_{N+1})u'_{h,p}(x_{N+1})^- \\ &\quad + \sum_{i=1}^{N} \left(a(x_i)v(x_i)u'_{h,p}(x_i)^- - a(x_i)v(x_i)u'_{h,p}(x_i)^+\right) \\ &= -\sum_{i=1}^{N} a(x_i)[u'_{h,p}]_i v(x_i) \\ &= -\frac{1}{2}\sum_{i=0}^{N} \left(a(x_i)[u'_{h,p}]_i v(x_i) + a(x_{i+1})[u'_{h,p}]_{i+1} v(x_{i+1})\right). \end{aligned}$$

We mention that the choice to split the jump for vertex x_i at one half to each of the two adjacent elements, namely $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$, is arbitrary but the most often used in practice.

Remark 3.2.6. When the polynomial degree is set to 1 on each element, for $x \in [x_i, x_{i+1}]$, $i = 0, \ldots, N$, we have $u''_{h,p}(x) = 0$, so we can write $(a(x)u'_{h,p}(x))' = a'(x)u'_{h,p}(x)$.

Third estimator

For this case, based on (2.14), we distinguish the case where x^* is a discretization node or not. In the former case, let j be the index for which $x^* = x_j$. We then define, for $v \in V$,

$$R^i_{z_{h,p}}(v) = \begin{cases} \frac{1}{2}v(x^*) + \int_{x_i}^{x_{i+1}} \left((a(x)z'_{h,p}(x))' + b(x)z'_{h,p}(x) - c(x)z_{h,p}(x)\right) v(x)dx \\ \quad + \frac{1}{2}(a(x_i)[z'_{h,p}]_i v(x_i) + a(x_{i+1})[z'_{h,p}]_{i+1} v(x_{i+1})) & \text{if } i \in \{j-1, j\} \\ \int_{x_i}^{x_{i+1}} \left((a(x)z'_{h,p}(x))' + b(x)z'_{h,p}(x) - c(x)z_{h,p}(x)\right) v(x)dx \\ \quad + \frac{1}{2}(a(x_i)[z'_{h,p}]_i v(x_i) + a(x_{i+1})[z'_{h,p}]_{i+1} v(x_{i+1})) & \text{otherwise.} \end{cases} \tag{3.2.11}$$

If x^* belongs to the interior of a subinterval, we set j the index for which $x^* \in (x_j, x_{j+1})$ and we then define

$$R^i_{z_{h,p}}(v) = \begin{cases} v(x^*) + \displaystyle\int_{x_i}^{x_{i+1}} \left((a(x)z'_{h,p}(x))' + b(x)z'_{h,p}(x) - c(x)z_{h,p}(x)\right) v(x)dx \\ \quad + \frac{1}{2}(a(x_i)[z'_{h,p}]_i v(x_i) + a(x_{i+1})[z'_{h,p}]_{i+1} v(x_{i+1})) & \text{if } i = j \\ \displaystyle\int_{x_i}^{x_{i+1}} \left((a(x)z'_{h,p}(x))' + b(x)z'_{h,p}(x) - c(x)z_{h,p}(x)\right) v(x)dx \\ \quad + \frac{1}{2}(a(x_i)[z'_{h,p}]_i v(x_i) + a(x_{i+1})[z'_{h,p}]_{i+1} v(x_{i+1})) & \text{otherwise.} \end{cases} \tag{3.2.12}$$

The error estimator then reads

$$\boxed{\eta^{(3)} = \sum_{i=0}^{N} \eta_i^{(3)} \quad \text{with} \quad \eta_i^{(3)} = R^i_{z_{h,p}}(\tilde{u} - u_{h,p})} \tag{3.2.13}$$

where $R^i_{z_{h,p}}$ is defined in (3.2.11) or (3.2.12) depending on the choice of x^* (interior discretization node or not, respectively). Both cases are derived by applying integration by parts, as done for (3.2.10). We mention that we have two arbitrary choices in this case, namely the splitting of the jump terms and the splitting of $(\tilde{u} - u_{h,p})(x^*)$ when x^* is a discretization point.

Fourth estimator

In the fourth case based on (2.16), we have

$$\boxed{\eta^{(4)} = \sum_{i=0}^{N} \eta_i^{(4)} \quad \text{with} \quad \eta_i^{(4)} = \mathcal{A}^i(\tilde{\varphi}^u, \tilde{z} - z_{h,p})} \tag{3.2.14}$$

where, choosing (3.2.4), for $u, v \in H^1(\Omega)$,

$$\mathcal{A}^i(u, v) = \int_{x_i}^{x_{i+1}} a(x)u'(x)v'(x)dx.$$

Fifth estimator

In the fifth case based on (2.17), we have

$$\boxed{\eta^{(5)} = \sum_{i=0}^{N} \eta_i^{(5)} \quad \text{with} \quad \eta_i^{(5)} = \mathcal{A}^i(\tilde{u} - u_{h,p}, \tilde{\varphi}^z)} \tag{3.2.15}$$

and $\mathcal{A}^i$ as above.

Sixth and seventh estimators

Based on Remark 2.6, we introduce two more estimators. The sixth one is a combination of the two estimators that use the Riesz representation theorem (i.e. fourth and fifth), namely we consider

$$\boxed{\eta^{(6)} = \sum_{i=0}^{N} \eta_i^{(6)} \quad \text{with} \quad \eta_i^{(6)} = \frac{1}{2}\left[\mathcal{A}^i(\tilde{u} - u_{h,p}, \tilde{\varphi}^z) + \mathcal{A}^i(\tilde{\varphi}^u, \tilde{z} - z_{h,p})\right]} \tag{3.2.16}$$

and $\mathcal{A}^i$ as above.

The seventh estimator we consider here is similar to the second estimator based on $R_{u_{h,p}}(\tilde{z} - z_{h,p})$, but with the additional use of the partition of unity. In this case, we obtain a node-based error estimator instead of an element-based one. Compared to (3.2.10), it has the advantage that no arbitrary choice is required for the jump term.

Recall that $\{\varphi_i\}_{i=0}^{N+1}$ denotes the standard basis functions for the $\mathbb{P}_1$ FE space (hat functions), including the basis functions associated to the boundary nodes.

For each node x_i, $i = 0, \ldots, N+1$, we define the patch ω_i as the support of φ_i, that is, we have here $\omega_i = [x_{i-1}, x_{i+1}]$ for each interior node while $w_0 = [x_0, x_1]$ and $w_{N+1} = [x_N, x_{N+1}]$ (see Figure 3.1). In the adaptive strategies considered below, if an interior node x_i is marked for refinement, then the two elements $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$ are refined.

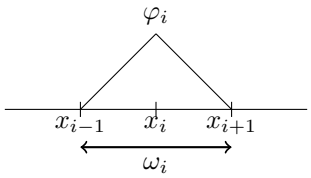


Figure 3.1: φ_i supported on the patch ω_i

For $i = 0, ..., N$, let r_i be the restriction of $(f + (au'_{h,p})' - bu'_{h,p} - cu_{h,p})$ on the subinterval $[x_i, x_{i+1}]$. We then have

$$\boxed{\eta^{(7)} = \sum_{i=0}^{N+1} \eta_i^{(7)} \quad \text{with} \quad \eta_i^{(7)} = \mathcal{R}^i_{u_{h,p}}(\tilde{z} - z_{h,p})} \tag{3.2.17}$$

where, for $v \in H^1(\Omega)$,

$$\mathcal{R}^i_{u_{h,p}}(v) = \begin{cases} \displaystyle\int_{x_0}^{x_1} r_0(x)\varphi_0(x)v(x)dx & \text{if } i = 0 \\ \displaystyle\int_{x_{i-1}}^{x_i} r_{i-1}(x)\varphi_i(x)v(x)dx + \int_{x_i}^{x_{i+1}} r_i(x)\varphi_i(x)v(x)dx + a(x_i)[u'_{h,p}]_i v(x_i) & \text{if } 2 \le i \le N \\ \displaystyle\int_{x_N}^{x_{N+1}} r_N(x)\varphi_{N+1}(x)v(x)dx & \text{if } i = N+1 \end{cases}$$

recalling that the jump is defined in (3.2.9).

Eighth and ninth estimators

Finally, using the representation of $\tilde{e}_u = \tilde{u} - u_{h,p}$ and $\tilde{e}_z = \tilde{z} - z_{h,p}$ in the basis of the FE space $\tilde{V}$, we give two more (*patch-based*) estimators based on $R_{u_{h,p}}(\tilde{e}_z)$ and $R_{z_{h,p}}(\tilde{e}_u)$. We write

$$\tilde{e}_z = \sum_{i=1}^{\tilde{N}_h} e_{z,i}\tilde{\psi}_i \quad \text{and} \quad \tilde{e}_u = \sum_{i=1}^{\tilde{N}_h} e_{u,i}\tilde{\psi}_i,$$

where $\{\tilde{\psi}_i\}_{i=1}^{\tilde{N}_h}$ are the basis functions for $\tilde{V}$ with $\tilde{N}_h = \dim(\tilde{V})$, and $e_{u,i}, e_{z,i}$ correspond to the coefficients in the basis for $\tilde{V}$. We set the first basis functions to be the $\mathbb{P}_1$ basis functions as above, i.e. $\tilde{\psi}_i = \varphi_i$ for

$i = 1, \ldots, N$. Note that by setting $\tilde{\psi}_0 = \varphi_0$ and $\tilde{\psi}_{\tilde{N}_h+1} = \varphi_{N+1}$, we can write

$$\tilde{e}_z = \sum_{i=1}^{\tilde{N}_h} e_{z,i}\tilde{\psi}_i = \sum_{i=0}^{\tilde{N}_h+1} e_{z,i}\tilde{\psi}_i \quad \text{and} \quad \tilde{e}_u = \sum_{i=1}^{\tilde{N}_h} e_{u,i}\tilde{\psi}_i = \sum_{i=0}^{\tilde{N}_h+1} e_{u,i}\tilde{\psi}_i$$

where we define $e_{u,i} = e_{z,i} = 0$ for $i \in \{0, \tilde{N}_h + 1\}$.

For each node x_i, $i = 0, ..., N+1$, we define the patch ω_i as the support of $\tilde{\psi}_i$. We let $\tilde{p} \in \mathbb{N}_+^{N+1}$ be the vector characterizing the polynomial degree on each element for $\tilde{V}$ and let $i_2, ..., i_{\tilde{p}_i}$ be the indices of the basis functions of degree ≥ 2 having support on $[x_i, x_{i+1}]$, $i = 0, ..., N$.

The eighth error estimator is then given by

$$\boxed{\begin{aligned} \eta^{(8)} &= \sum_{i=0}^{N+1} \eta_i^{(8)} \quad \text{with} \quad \eta_i^{(8)} = R_{u_{h,p}}(\varphi_i)e_{z,i} + \sum_{j=i-1}^{i} D_j^{(8)}, \\ &\text{where } D_j^{(8)} = \begin{cases} \dfrac{1}{2}\displaystyle\sum_{k=2}^{\tilde{p}_j} R_{u_{h,p}}(\tilde{\psi}_{j_k})e_{z,j_k} & \text{if } 0 \leq j \leq N, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}} \tag{3.2.18}$$

We sum over j to consider all the elements of the patch ω_i and we introduce the factor $\frac{1}{2}$ to take into account that each element belongs to 2 patches (see Figure 3.2). This is because we arbitrarily choose to split the contribution of the functions of degree $k \geq 2$ of an element to each corresponding patch.

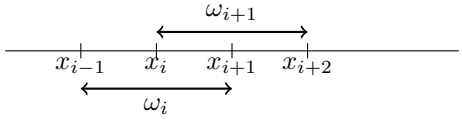


Figure 3.2: Element $[x_i, x_{i+1}]$ shared between the patches ω_i and ω_{i+1}

Remark 3.2.7. In higher dimensions, the error indicator for patch ω_i associated to node x_i and constituted of the elements $K_1, ..., K_n$ with degree $\tilde{p}_1, ...\tilde{p}_n$, respectively, would read

$$\eta_i^{(8)} = R_{u_{h,p}}(\varphi_i)e_{z,i} + \sum_{j=1}^{n} \frac{1}{m_j} \sum_{k=2}^{\tilde{p}_j} R_{u_{h,p}}(\tilde{\psi}_{j_k})e_{z,j_k},$$

where m_j is the number of patches in which K_j belongs to and $j_2, ..., j_{\tilde{p}_j}$ are the indices of the basis functions of degree ≥ 2 having support on K_j.

Similarly, we define the ninth (and last) error estimator by

$$\boxed{\begin{aligned} \eta^{(9)} &= \sum_{i=0}^{N+1} \eta_i^{(9)} \quad \text{with} \quad \eta_i^{(9)} = R_{z_{h,p}}(\varphi_i)e_{u,i} + \sum_{j=i-1}^{i} D_j^{(9)}, \\ &\text{where } D_j^{(9)} = \begin{cases} \dfrac{1}{2}\displaystyle\sum_{k=2}^{\tilde{p}_j} R_{z_{h,p}}(\tilde{\psi}_{j_k})e_{u,j_k} & \text{if } 0 \leq j \leq N, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}} \tag{3.2.19}$$

3.3 Specific 2D Problem

In this section, we define the same error estimators in the context of a general two-dimensional Poisson problem with Dirichlet boundary conditions. We begin with the homogeneous case, and then extend the approach to accommodate the slightly modified estimators required for nonhomogeneous conditions.

Homogeneous Dirichlet boundary conditions

Consider the L-shaped domain $\Omega = (-1,1) \times (-1,1) \setminus [0,1) \times (-1,0]$ and the Poisson (primal) problem

$$\begin{cases} -\Delta u = f \text{ in } \Omega \\ u = 0 \text{ on } \partial\Omega, \end{cases}$$

where $f \in L^2(\Omega)$ is a given function.

The chosen quantity of interest is

$$Q(u) = \int_{\Omega^*} u(x)dx$$

with Ω^* some subset of Ω.

Let n be the outward unit vector orthogonal to $\partial\Omega$. Setting $V_0 = H_0^1(\Omega)$ and multiplying the PDE by a test function $v \in V_0$, we have

$$\int_\Omega fvdx = \int_\Omega (-\Delta u)vdx = \int_\Omega \nabla u \cdot \nabla v dx - \int_\Omega (\nabla u \cdot n)vds = \int_\Omega \nabla u \cdot \nabla v dx,$$

meaning that we look for $u \in V_0$ such that

$$\int_\Omega \nabla u \cdot \nabla v dx = \int_\Omega fvdx, \quad \forall v \in V_0.$$

The problem under consideration can be put in the abstract formulation of Chapter 2 with

$$B(u,v) = \int_\Omega \nabla u \cdot \nabla v dx$$

and

$$F(v) = \int_\Omega fvdx.$$

For the dual problem, for all $v \in V_0$, we look for $z \in V_0$ such that $B(v,z) = Q(v)$, i.e.

$$\int_\Omega \nabla v \cdot \nabla z dx = \int_{\Omega^*} vdx, \quad \forall v \in V_0.$$

Applying integration by parts, we have

$$\int_{\Omega^*} vdx = \int_\Omega (-\Delta z)vdx + \int_{\partial\Omega} (\nabla z \cdot n)vds = \int_\Omega (-\Delta z)vdx,$$

from which we derive the strong formulation of the problem:

$$\begin{cases} -\Delta z = \mathbb{1}_{\Omega^*} \text{ in } \Omega \\ z = 0 \text{ on } \partial\Omega. \end{cases}$$

Remark 3.3.1. As in the 1D case, we can validate the existence and uniqueness of a solution to the primal problem and to the dual problem, by verifying the five hypotheses mentioned in Remark 3.2.2. We once again focus on verifying 2, 3 and 5 using $\|.\|_{H^1(\Omega)}$, defined by

$$\|v\|_{H^1(\Omega)} = \sqrt{\|v\|^2_{L^2(\Omega)} + |v|^2_{H^1(\Omega)}}.$$

2. For all $u, v \in H^1_0(\Omega)$,

$$\begin{aligned} B(u,v) &= \int_\Omega \nabla u \cdot \nabla v dx \\ &\overset{\text{Thm 3.1.3}}{\leq} \|\nabla u\|_{L^2(\Omega)} \|\nabla v\|_{L^2(\Omega)} \\ &\leq \left(\|u\|^2_{L^2(\Omega)} + \|\nabla u\|^2_{L^2(\Omega)}\right)^{\frac{1}{2}} \left(\|v\|^2_{L^2(\Omega)} + \|\nabla v\|^2_{L^2(\Omega)}\right)^{\frac{1}{2}} \\ &= \|u\|_{H^1(\Omega)} \|v\|_{H^1(\Omega)}. \end{aligned}$$

   $\Rightarrow B(u,v)$ is continuous.

3. For any $u \in H^1_0(\Omega)$,

$$\begin{aligned} B(u,u) &= \int_\Omega |\nabla u|^2 dx \\ &= \frac{1}{2}|u|^2_{H^1(\Omega)} + \frac{1}{2}|u|^2_{H^1(\Omega)} \\ &\overset{\text{Thm 3.1.5}}{\geq} \frac{1}{2}|u|^2_{H^1(\Omega)} + \frac{1}{2C_p^2}\|u\|^2_{L^2(\Omega)} \\ &\geq \min\left(\frac{1}{2}, \frac{1}{2C_p^2}\right) \|u\|^2_{H^1(\Omega)} \\ &=: C\|u\|^2_{H^1(\Omega)}. \end{aligned}$$

   $\Rightarrow B(u,v)$ is coercive.

5. Finally, for all $v \in H^1_0(\Omega)$,

$$\begin{aligned} \int_\Omega f v dx &\overset{\text{Thm 3.1.3}}{\leq} \|f\|_{L^2(\Omega)} \|v\|_{L^2(\Omega)} \\ &\leq \|f\|_{L^2(\Omega)} \|v\|_{H^1(\Omega)} \\ &=: C\|v\|_{H^1(\Omega)} \end{aligned}$$

   and

$$\begin{aligned} \int_\Omega \mathbb{1}_{\Omega^*} v dx &\overset{\text{Thm 3.1.3}}{\leq} \|\mathbb{1}_{\Omega^*}\|_{L^2(\Omega)} \|v\|_{L^2(\Omega)} \\ &\leq \|\mathbb{1}_{\Omega^*}\|_{L^2(\Omega)} \|v\|_{H^1(\Omega)} \\ &=: C\|v\|_{H^1(\Omega)}. \end{aligned}$$

   $\Rightarrow F(v)$ and $Q(v)$ are linear and bounded, therefore continuous.

The hypotheses of the Lax-Milgram Lemma are all satisfied, and we can conclude that the primal and dual problems have unique solutions.

For $\mathcal{A}$, we consider

$$\mathcal{A}(u,v) = B(u,v) = \int_\Omega \nabla u \cdot \nabla v dx,$$

but other options are possible.

To define the error representations in the 2D case, we consider a triangulation

$$\mathcal{T}_h = \bigcup_{K\in\mathcal{T}_h} K$$

of the L-shaped $\bar{\Omega}$ that consists of N_v^0 interior nodes, N_v^b boundary nodes, N_e^0 interior edges, N_e^b boundary edges and N_t triangular elements K of diameter $h_K \le h$, where

$$h_K = \mathrm{diam}(K) = \max_{x,y\in K} |x-y|.$$

For this specific problem, we will only implement h-adaptation, meaning that we fix the polynomial degree to 1 for the FE approximation, and the polynomial degree to 2 for the finer FE space. We then define

$$V_h = \{v \in C^0(\bar{\Omega}) : v|_K \in \mathbb{P}_1 \quad \forall K \in \mathcal{T}_h\}, \qquad \tilde{V} = \{v \in C^0(\bar{\Omega}) : v|_K \in \mathbb{P}_2 \quad \forall K \in \mathcal{T}_h\},$$
$$V_{h,0} = V_h \cap H_0^1(\Omega), \qquad \tilde{V}_0 = \tilde{V} \cap H_0^1(\Omega),$$

and have

$$N_h = \dim(V_h) = N_v^0 + N_v^b, \qquad \tilde{N}_h = \dim(\tilde{V}) = N_v^0 + N_v^b + N_e^0 + N_e^b,$$
$$N_{h,0} = \dim(V_{h,0}) = N_v^0, \qquad \tilde{N}_{h,0} = \dim(\tilde{V}_0) = N_v^0 + N_e^0.$$

Since we are working with $V_0 = H_0^1(\Omega)$ (equivalent of V in previous sections), we take $V_{h,0}$ as the FE approximation of V_0 (equivalent of $V_{h,p}$ in previous sections), and similarly use $\tilde{V}_0$ for the refined space (equivalent of $\tilde{V}$ in previous sections). We denote elements of $V_{h,0}$ by v_h (instead of $v_{h,p}$) and elements of $\tilde{V}_0$ will be written as $\tilde{v}$ (as before). The basis of $V_{h,0}$ is denoted by $\{\varphi_i\}_{i=1}^{N_{h,0}}$ and the basis for $\tilde{V}_0$ is denoted by $\{\tilde{\psi}_i\}_{i=1}^{\tilde{N}_{h,0}}$.

First estimator

In the first case based on (2.11), we have

$$\boxed{\eta^{(1)} = \sum_{K\in\mathcal{T}_h} \eta_K^{(1)} \quad \text{with} \quad \eta_K^{(1)} = B_K(\tilde{u}-u_h, \tilde{z}-z_h)} \tag{3.3.1}$$

where, for $u, v \in H^1(\Omega)$,

$$B_K(u,v) = \int_K \nabla u(x)\cdot\nabla v(x)dx.$$

Second estimator

For an edge e, let n_e be a unit vector orthogonal to e of arbitrary (but fixed) direction, set to be outward when e is a boundary edge. Before giving the error estimator, let us define the jump of a function g across an interior edge e,

$$[g(x)]_e = \lim_{s\to 0^+} (g(x+sn_e) - g(x-sn_e)).$$

If e is a boundary edge, we set $[g(x)]_e = 0$. Applying Proposition 2.7 and integration by parts, we derive the second estimator.

$$\begin{aligned} Q(\tilde{u}-u_h) &= R_{u_h}(\tilde{z}-z_h) \\ &= \int_\Omega f(\tilde{z}-z_h)dx - \int_\Omega \nabla u_h\cdot\nabla(\tilde{z}-z_h)dx \end{aligned}$$

$$= \sum_{K\in\mathcal{T}_h} \left\{ \int_K (f+\Delta u_h)(\tilde{z}-z_h)dx - \int_{\partial K} (\nabla u_h \cdot n)(\tilde{z}-z_h)ds \right\}$$

$$= \sum_{K\in\mathcal{T}_h} \left\{ \int_K (f+\Delta u_h)(\tilde{z}-z_h)dx - \sum_{e\subset\partial K} \int_e (\nabla u_h \cdot n_e)(\tilde{z}-z_h)ds \right\}$$

(Suppose each interior edge e is shared between elements $K_{e,1}$ and $K_{e,2}$, with corresponding outward unit vectors $n_{K_{e,1}}$ and $n_{K_{e,2}}$. Let $\mathcal{E}_h^0$ be the set of interior edges in $\mathcal{T}_h$, $\mathcal{E}_h^b$ the set of boundary edges in $\mathcal{T}_h$, and $\mathcal{E}_h = \mathcal{E}_h^0 \cup \mathcal{E}_h^b$.)

$$= \sum_{K\in\mathcal{T}_h} \int_K (f+\Delta u_h)(\tilde{z}-z_h)dx$$

$$- \sum_{e\in\mathcal{E}_h^0} \int_e (\nabla u_h|_{K_{e,1}} \cdot n_{K_{e,1}} + \nabla u_h|_{K_{e,2}} \cdot n_{K_{e,2}})(\tilde{z}-z_h)ds$$

$$- \sum_{e\in\mathcal{E}_h^b} (\nabla u_h \cdot n_e)(\tilde{z}-z_h)ds$$

(We choose $n_e = n_{K_{e,1}} = -n_{K_{e,2}}$.)

$$= \sum_{K\in\mathcal{T}_h} \int_K (f+\Delta u_h)(\tilde{z}-z_h)dx - \sum_{e\in\mathcal{E}_h^0} \int_e (\nabla u_h|_{K_{e,1}} \cdot n_e - \nabla u_h|_{K_{e,2}} \cdot n_e)(\tilde{z}-z_h)ds$$

$$- \sum_{e\in\mathcal{E}_h^b} \int_e (\nabla u_h \cdot n_e)(\tilde{z}-z_h)ds$$

(Evaluating $\nabla u_h|_{K_{e,1}} \cdot n_e - \nabla u_h|_{K_{e,2}} \cdot n_e$ at $s \in e$, we have $\lim_{s\to 0^+} (\nabla u_h(x-sn_e) - \nabla u_h(x+sn_e)) \cdot n_e$.)

$$= \sum_{K\in\mathcal{T}_h} \int_K (f+\Delta u_h)(\tilde{z}-z_h)dx - \sum_{e\in\mathcal{E}_h^0} \int_e (-[\nabla u_h \cdot n_e]_e)(\tilde{z}-z_h)ds$$

$$- \sum_{e\in\mathcal{E}_h^b} \int_e (\nabla u_h \cdot n_e)(\tilde{z}-z_h)ds$$

$$= \sum_{K\in\mathcal{T}_h} \int_K (f+\Delta u_h)(\tilde{z}-z_h)dx + \sum_{e\in\mathcal{E}_h^0} \int_e ([\nabla u_h \cdot n_e]_e)(\tilde{z}-z_h)ds$$

$$- \sum_{e\in\mathcal{E}_h^b} \int_e (\nabla u_h \cdot n_e)(\tilde{z}-z_h)ds.$$

The second error estimator, based on (2.12), then reads

$$\boxed{\eta^{(2)} = \sum_{K\in\mathcal{T}_h} \eta_K^{(2)} \quad \text{with} \quad \eta_K^{(2)} = R_{u_h}^K(\tilde{z}-z_h)} \tag{3.3.2}$$

where, for $v \in H^1(\Omega)$,

$$R_{u_h}^K(v) = \int_K (f(x)+\Delta u_h(x))v(x)dx + \frac{1}{2} \sum_{e\subset\partial K\setminus\partial\Omega} \int_e [\nabla u_h(s)\cdot n_e]_e v(s)ds$$

$$- \sum_{e\subset\partial K\cap\partial\Omega} \int_e (\nabla u_h(s)\cdot n_e)v(s)ds.$$

As in the previous problem, the choice to split the jump for each interior edge at one half to each of the two adjacent elements is arbitrary, but the most often used in practice. Also, note that since $\tilde{z} - z_h = 0$ on $\partial\Omega$,

$$\sum_{e \subset \partial K \cap \partial\Omega} \int_e (\nabla u_h(s) \cdot n_e)(\tilde{z} - z_h)(s) ds = 0.$$

Finally, note that for any element K, $\Delta u_h|_K \equiv 0$ since u_h is a $\mathbb{P}_1$ FE function.

Third estimator

In the third case, based on (2.14), we have

$$\boxed{\eta^{(3)} = \sum_{K \in \mathcal{T}_h} \eta_K^{(3)} \quad \text{with} \quad \eta_K^{(3)} = R_{z_h}^K(\tilde{u} - u_h)} \tag{3.3.3}$$

where, for $v \in H^1(\Omega)$,

$$R_{z_h}^K(v) = \int_K \left(\mathbb{1}_{\Omega^*}(x) + \Delta z_h(x)\right) v(x) dx + \frac{1}{2} \sum_{e \subset \partial K \setminus \partial\Omega} \int_e [\nabla z_h(s) \cdot n_e]_e v(s) ds - \sum_{e \subset \partial K \cap \partial\Omega} \int_e (\nabla z_h(s) \cdot n_e) v(s) ds.$$

This estimator once again involves the splitting of the jump term, and since $\tilde{u} - u_h = 0$ on $\partial\Omega$,

$$\sum_{e \subset \partial K \cap \partial\Omega} \int_e (\nabla z_h(s) \cdot n_e)(\tilde{u} - u_h)(s) ds = 0.$$

Additionally, for any element K, we also have $\Delta z_h|_K \equiv 0$ since z_h is a $\mathbb{P}_1$ FE function.

Fourth estimator

In the fourth case, based on (2.16), we first define $\tilde{\varphi}^u \in \tilde{V}_0$ as the Riesz representant of R_{u_h} with respect to the scalar product induced by $\mathcal{A}$, namely the solution of

$$\text{find } \tilde{\varphi}^u \in \tilde{V}_0 : \quad \mathcal{A}(\tilde{\varphi}^u, \tilde{v}) = R_{u_h}(\tilde{v}) \quad \forall \tilde{v} \in \tilde{V}_0. \tag{3.3.4}$$

The estimator then reads

$$\boxed{\eta^{(4)} = \sum_{K \in \mathcal{T}_h} \eta_K^{(4)} \quad \text{with} \quad \eta_K^{(4)} = \mathcal{A}_K(\tilde{\varphi}^u, \tilde{z} - z_h)} \tag{3.3.5}$$

where, for $u, v \in H^1(\Omega)$,

$$\mathcal{A}_K(u, v) = \int_K \nabla u(x) \cdot \nabla v(x) dx.$$

Fifth estimator

In the fifth case, based on (2.17), we define $\tilde{\varphi}^z \in \tilde{V}_0$ as the Riesz representant of R_{z_h} with respect to the scalar product induced by $\mathcal{A}$, namely the solution of

$$\text{find } \tilde{\varphi}^z \in \tilde{V}_0 : \quad \mathcal{A}(\tilde{v}, \tilde{\varphi}^z) = R_{z_h}(\tilde{v}) \quad \forall \tilde{v} \in \tilde{V}_0. \tag{3.3.6}$$

The estimator then reads

$$\boxed{\eta^{(5)} = \sum_{K \in \mathcal{T}_h} \eta_K^{(5)} \quad \text{with} \quad \eta_K^{(5)} = \mathcal{A}_K(\tilde{u} - u_h, \tilde{\varphi}^z),} \tag{3.3.7}$$

with $\mathcal{A}_K$ as above.

Sixth and seventh estimators

We once again use Remark 2.6 to introduce two more estimators.

As in the 1D case, the sixth estimator is an average of the fourth and fifth estimators.

$$\boxed{\eta^{(6)} = \sum_{K\in\mathcal{T}_h} \eta_K^{(6)} \quad \text{with} \quad \eta_K^{(6)} = \frac{1}{2}(\eta_K^{(4)} + \eta_K^{(5)}).} \tag{3.3.8}$$

For the seventh estimator, let $\{\varphi_i\}_{i=1}^{N_h}$ denote the standard basis functions for the $\mathbb{P}_1$ FE. For $i = 1, \dots, N_h$, we define the patch ω_i as the support of φ_i (see Figure 3.3).

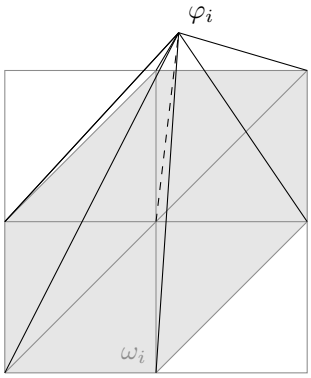


Figure 3.3: φ_i supported on the patch ω_i

Finally, we let r_K be the restriction of $(f + \Delta u_h)$ on the element K. We have

$$\begin{aligned} Q(\tilde{u} - u_h) &= R_{u_h}(\tilde{z} - z_h) \\ &= R_{u_h}\left(\sum_{i=1}^{N_h}(\tilde{z} - z_h)\varphi_i\right) = \sum_{i=1}^{N_h} R_{u_h}((\tilde{z} - z_h)\varphi_i) \\ &= \sum_{i=1}^{N_h}\sum_{K\in\omega_i}\left\{\int_K r_K(\tilde{z} - z_h)\varphi_i dx + \sum_{e\subset\partial K}\int_e [\nabla u_h \cdot n_e]_e(\tilde{z} - z_h)\varphi_i ds\right\}. \end{aligned}$$

The seventh (patch-based) estimator is given by

$$\boxed{\eta^{(7)} = \sum_{i=1}^{N_h} \eta_i^{(7)} \quad \text{with} \quad \eta_i^{(7)} = \mathcal{R}_{u_h}^i(\tilde{z} - z_h),} \tag{3.3.9}$$

where, for $v \in H^1(\Omega)$,

$$\mathcal{R}_{u_h}^i(v) = \sum_{K\in\omega_i}\left\{\int_K r_K(x)v(x)\varphi_i(x)dx + \sum_{e\subset\partial K}\int_e [\nabla u_h(s) \cdot n_e]_e v(s)\varphi_i(s)ds\right\}.$$

Remark 3.3.2. For $e \subset \partial K \cap \partial\omega_i$, we have $\varphi_i|_e = 0$, so

$$\int_e [\nabla u_h(s) \cdot n_e]_e \varphi_i(s)v(s)ds = 0.$$

Eighth and ninth estimators

Finally, using the representation of $\tilde{e}_u = \tilde{u} - u_h$ and $\tilde{e}_z = \tilde{z} - z_h$ in the basis of the FE space $\tilde{V}_0$, we give two more (patch-based) estimators based on $R_{u_h}(\tilde{e}_u)$ and $R_{z_h}(\tilde{e}_z)$. We write

$$\tilde{e}_u = \sum_{i=1}^{\tilde{N}_{h,0}} e_{u,i}\tilde{\psi}_i \quad \text{and} \quad \tilde{e}_z = \sum_{i=1}^{\tilde{N}_{h,0}} e_{z,i}\tilde{\psi}_i,$$

where $\{\tilde{\psi}_i\}_{i=1}^{\tilde{N}_{h,0}}$ are the basis functions for $\tilde{V}_0$, and $e_{u,i}, e_{z,i}$ correspond to the coefficients in the basis for $\tilde{V}_0$. Referring to the FE space dimensions listed above, we can write out the degrees of freedom of $\tilde{V}$ as

$$\{\underbrace{1,\dots,N_{h,0}}_{\text{interior nodes}},\underbrace{N_{h,0}+1,\dots,N_h}_{\text{boundary nodes}},\underbrace{N_h+1,\dots,N_h+N_e^0}_{\text{interior edges}},\underbrace{N_h+N_e^0+1,\dots,\tilde{N}_h}_{\text{boundary edges}}\}.$$

We set the first basis functions of $\tilde{V}$ to be the $\mathbb{P}_1$ basis functions as above, i.e. $\tilde{\psi}_i = \varphi_i$ for $i = 1,\dots,N_h$. Note that by setting $e_{z,i} = e_{u,i} = 0$ for $i \in \{N_{h,0}+1,\dots,N_h\} \cup \{N_h + N_e^0 + 1, \cdots, \tilde{N}_h\}$, we can write

$$\tilde{e}_u = \sum_{i=1}^{\tilde{N}_{h,0}} e_{u,i}\tilde{\psi}_i = \sum_{i=1}^{\tilde{N}_h} e_{u,i}\tilde{\psi}_i \quad \text{and} \quad \tilde{e}_z = \sum_{i=1}^{\tilde{N}_{h,0}} e_{z,i}\tilde{\psi}_i = \sum_{i=1}^{\tilde{N}_h} e_{z,i}\tilde{\psi}_i.$$

For each vertex x_i, $i = 1,\dots,N_h$, we define the patch ω_i as the support of the corresponding basis function φ_i (basis function equal to 1 at x_i) and let $i_1,\dots,i_{n_i}$ be the indices of the $\mathbb{P}_2$ basis functions associated to the edges having x_i as an endpoint. We have

$$\begin{aligned} R_{u_h}(\tilde{e}_z) &= R_{u_h}\left(\sum_{i=1}^{N_h}\left[\varphi_i e_{z,i} + \frac{1}{2}\sum_{j=1}^{n_i}\psi_{i_j}e_{z,i_j}\right]\right) \\ &= \sum_{i=1}^{N_h}\left(R_{u_h}(\varphi_i)e_{z,i} + \frac{1}{2}\sum_{j=1}^{n_i}R_{u_h}(\psi_{i_j})e_{z,i_j}\right) \end{aligned}$$

and

$$\begin{aligned} R_{z_h}(\tilde{e}_u) &= R_{z_h}\left(\sum_{i=1}^{N_h}\left[\varphi_i e_{u,i} + \frac{1}{2}\sum_{j=1}^{n_i}\psi_{i_j}e_{u,i_j}\right]\right) \\ &= \sum_{i=1}^{N_h}\left(R_{z_h}(\varphi_i)e_{u,i} + \frac{1}{2}\sum_{j=1}^{n_i}R_{u_h}(\psi_{i_j})e_{u,i_j}\right). \end{aligned}$$

The eighth error estimator is given by

$$\boxed{\eta^{(8)} = \sum_{i=1}^{N_h}\eta_i^{(8)} \quad \text{with} \quad \eta_i^{(8)} = R_{u_h}(\varphi_i)e_{z,i} + \frac{1}{2}\sum_{j=1}^{n_i}R_{u_h}(\psi_{i_j})e_{z,i_j}} \tag{3.3.10}$$

and we define the ninth estimator by

$$\boxed{\eta^{(9)} = \sum_{i=1}^{N_h}\eta_i^{(9)} \quad \text{with} \quad \eta_i^{(9)} = R_{z_h}(\varphi_i)e_{u,i} + \frac{1}{2}\sum_{j=1}^{n_i}R_{u_h}(\psi_{i_j})e_{u,i_j}.} \tag{3.3.11}$$

Here, φ_i are the $\mathbb{P}_1$ basis functions and ψ_{i_j} are the hierarchical $\mathbb{P}_2$ basis functions on each edge belonging to ω_i. The factor $\frac{1}{2}$ is introduced to take into account the fact that each edge belongs to two patches (see Figure 3.4). This is because we arbitrarily choose to evenly split the edge-based contributions of the shape functions to each corresponding patch.

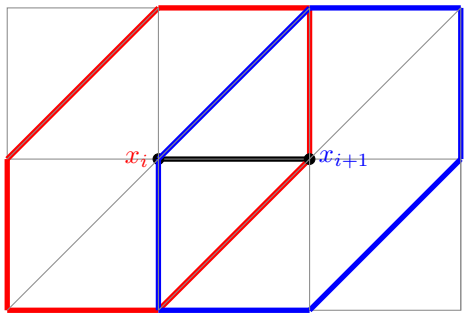


Figure 3.4: Edge shared between the patches ω_i and ω_{i+1}

Nonhomogeneous Dirichlet boundary conditions

We now consider the case of nonhomogeneous Dirichlet boundary conditions, for which the estimators are adjusted to account for the nonzero boundary values. Here, V_0, $V_{h,0}$, V_h, $\tilde{V}_0$ and $\tilde{V}$ remain unchanged.

Consider the Poisson (primal) problem

$$\begin{cases} -\Delta u = f \text{ in } \Omega \\ u = g \text{ on } \partial\Omega, \end{cases}$$

where $f \in L^2(\Omega)$ and $g \in H^{\frac{1}{2}}(\partial\Omega)$ are given functions.

We define

$$\gamma : H^1(\Omega) \to H^{\frac{1}{2}}(\partial\Omega)$$

as the trace operator such that

$$\gamma(u) = u\big|_{\partial\Omega}.$$

As before, multiplying the PDE by a test function $v \in V_0$, we have

$$\int_\Omega f v dx = \int_\Omega \nabla u \cdot \nabla v dx,$$

meaning that we look for $u \in H^1(\Omega)$ such that $\gamma(u) = g$ in $H^{\frac{1}{2}}(\partial\Omega)$ and

$$B(u, v) = F(v) \quad \forall v \in V_0. \tag{3.3.12}$$

From Theorem 3.10 in [19], we have that there exists $u_g \in H^1(\Omega)$, an *extension* (or *lifting*) of the boundary data $g \in H^{\frac{1}{2}}(\partial\Omega)$ such that $\gamma(u_g) = g$ and

$$||u_g||_{H^1(\Omega)} \leq \mathcal{C}||g||_{H^{\frac{1}{2}}(\partial\Omega)},$$

where $\mathcal{C} > 0$ is a constant, independent from g. We can for instance consider the solution to the problem

$$\begin{cases} -\Delta u_g = 0 \text{ in } \Omega \\ u_g = g \text{ on } \partial\Omega, \end{cases}$$

but other options are possible. Note that with this choice of u_g, multiplying the PDE by a test function $v \in V_0$, we have

$$0 = \int_\Omega (-\Delta u_g) v dx = \int_\Omega \nabla u_g \cdot \nabla v dx - \int_{\partial\Omega} (\nabla u_g \cdot n) v ds = \int_\Omega \nabla u_g \cdot \nabla v dx,$$

meaning u_g can be obtained by considering

$$V_g = \{v \in H^1(\Omega) : \gamma(v) = g\}$$

and solving the well-posed problem

$$\text{find } u_g \in V_g : B(u_g, v) = 0 \quad \forall v \in V_0. \tag{3.3.13}$$

Recall that $\{\varphi_i\}_{i=1}^{N_h}$ denotes the standard basis functions of V_h for the $\mathbb{P}_1$ FE. Let $\{a_i\}_{i=1}^{N_h}$ be the associated nodes. To derive the computable error estimates for the 2D problem with nonhomogeneous Dirichlet conditions, we consider the spaces

$$V_{h,g} = \{v_h \in V_h : \gamma(v_h) = g_h\}$$

and

$$\tilde{V}_g = \{\tilde{v} \in \tilde{V} : \gamma(\tilde{v}) = \tilde{g}\}.$$

Here, $g_h \in V_h$ and $\tilde{g} \in \tilde{V}$ are FE approximations of g. We can for instance consider the Lagrange interpolant

$$g_h = \sum_{a_i \in \partial\Omega} g(a_i)\gamma(\varphi_i),$$

assuming g is sufficiently smooth, namely at least continuous.

Note that we have $\dim(V_{h,g}) = \dim(V_{h,0}) = N_{h,0}$, $\dim(\tilde{V}_g) = \dim(\tilde{V}_0) = \tilde{N}_{h,0}$ and that if $g = 0$, $V_{h,g} = V_{h,0}$ and $\tilde{V}_g = \tilde{V}_0$.

We can approximate Problem (3.3.12) by

$$\text{find } u_h \in V_{h,g} : B(u_h, v_h) = F(v_h) \quad \forall v_h \in V_{h,0} \tag{3.3.14}$$

$$\text{find } \tilde{u} \in \tilde{V}_g : B(\tilde{u}, \tilde{v}) = F(\tilde{v}) \quad \forall \tilde{v} \in \tilde{V}_0 \tag{3.3.15}$$

and Problem (3.3.13) by

$$\text{find } u_{h,g} \in V_{h,g} : B(u_{h,g}, v_h) = 0 \quad \forall v_h \in V_{h,0} \tag{3.3.16}$$

$$\text{find } \tilde{u}_g \in \tilde{V}_g : B(\tilde{u}_g, \tilde{v}) = 0 \quad \forall \tilde{v} \in \tilde{V}_0. \tag{3.3.17}$$

Setting $u_0 := u - u_g \in V_0$, $u_{h,0} = u_h - u_{h,g} \in V_{h,0}$ and $\tilde{u}_0 = \tilde{u} - \tilde{u}_g \in \tilde{V}_0$, we have that

$$B(u_0, v) = F(v) - B(u_g, v), \quad \forall v \in V_0, \tag{3.3.18}$$

$$B(u_{h,0}, v_h) = F(v_h) - B(u_{h,g}, v_h), \quad \forall v_h \in V_{h,0}, \tag{3.3.19}$$

$$B(\tilde{u}_0, \tilde{v}) = F(\tilde{v}) - B(\tilde{u}_g, \tilde{v}), \quad \forall \tilde{v} \in \tilde{V}_0. \tag{3.3.20}$$

Since $\gamma(z) = 0$, the dual problem and its approximations are the same as in the homogeneous case:

$$\text{find } z \in V_0 : \quad B(v, z) = Q(v) \quad \forall v \in V_0, \tag{3.3.21}$$

$$\text{find } z_h \in V_{h,0} : \quad B(v_h, z_h) = Q(v_h) \quad \forall v_h \in V_{h,0}, \tag{3.3.22}$$

$$\text{find } \tilde{z} \in \tilde{V}_0 : \quad B(\tilde{v}, \tilde{z}) = Q(\tilde{v}) \quad \forall \tilde{v} \in \tilde{V}_0. \tag{3.3.23}$$

For the error in the QoI, we approximate $Q(u - u_h)$ by

$$Q(\tilde{u} - u_h) = Q(\tilde{u}_0 + \tilde{u}_g - u_{h,0} - u_{h,g})$$

$$= Q(\tilde{u}_0 - u_{h,0}) + Q(\tilde{u}_g - u_{h,g})$$
$$\overset{(3.3.21)}{=} B(\tilde{u}_0 - u_{h,0}, z) + Q(\tilde{u}_g - u_{h,g}). \tag{3.3.24}$$

Additionally, applying the strong formulation of the dual problem and integration by parts, we have

$$\begin{aligned} Q(\tilde{u}_g - u_{h,g}) &= \int_\Omega \mathbb{1}_{\Omega^*}(\tilde{u}_g - u_{h,g})dx \\ &= -\int_\Omega \Delta z(\tilde{u}_g - u_{h,g})dx \\ &= \int_\Omega \nabla z \cdot \nabla(\tilde{u}_g - u_{h,g})dx - \int_{\partial\Omega} (\nabla z \cdot n)(\tilde{u}_g - u_{h,g})ds \\ &= B(\tilde{u}_g - u_{h,g}, z) - \int_{\partial\Omega} (\nabla z \cdot n)(\tilde{u}_g - u_{h,g})ds, \end{aligned} \tag{3.3.25}$$

since the jump of $\nabla z \cdot n$ is 0 across interior edges. We have two orthogonality results for any $v_h \in V_{h,0}$ by noting that

$$\begin{aligned} B(\tilde{u} - u_h, v_h) &= B(\tilde{u}, v_h) - B(u_h, v_h) \\ &\overset{(3.3.14),(3.3.15)}{=} F(v_h) - F(v_h) = 0 \end{aligned} \tag{3.3.26}$$

and

$$\begin{aligned} B(\tilde{u}_0 - u_{h,0}, v_h) &= B(\tilde{u}_0, v_h) - B(u_{h,0}, v_h) \\ &\overset{(3.3.19),(3.3.20)}{=} F(v_h) - B(\tilde{u}_g, v_h) - (F(v_h) - B(u_{h,g}, v_h)) \\ &= -B(\tilde{u}_g, v_h) + B(u_{h,g}, v_h) \overset{(3.3.16),(3.3.17)}{=} 0. \end{aligned} \tag{3.3.27}$$

Note that (3.3.27) holds for the specific choice of $u_{h,g}$ and $\tilde{u}_g$ stated above, but not in general.

First estimator

Using the equalities above, we have

$$\begin{aligned} Q(\tilde{u} - u_h) &\overset{(3.3.24)}{=} B(\tilde{u}_0 - u_{h,0}, z) + Q(\tilde{u}_g - u_{h,g}) \\ &\overset{(3.3.27)}{=} B(\tilde{u}_0 - u_{h,0}, z - z_h) + Q(\tilde{u}_g - u_{h,g}) \\ &\overset{(3.3.25)}{=} B(\tilde{u}_0 - u_{h,0}, z - z_h) + B(\tilde{u}_g - u_{h,g}, z) - \int_{\partial\Omega} (\nabla z \cdot n)(\tilde{u}_g - u_{h,g})ds \\ &\overset{(3.3.16),(3.3.17)}{=} B(\tilde{u}_0 - u_{h,0}, z - z_h) + B(\tilde{u}_g - u_{h,g}, z - z_h) - \int_{\partial\Omega} (\nabla z \cdot n)(\tilde{u}_g - u_{h,g})ds \\ &= B(\tilde{u} - u_h, z - z_h) - \int_{\partial\Omega} (\nabla z \cdot n)(\tilde{u}_g - u_{h,g})ds \\ &\approx B(\tilde{u} - u_h, \tilde{z} - z_h) - \int_{\partial\Omega} (\nabla \tilde{z} \cdot n)(\tilde{u}_g - u_{h,g})ds. \end{aligned} \tag{3.3.28}$$

Therefore, the first estimator based on (2.11) is given by

$$\boxed{\eta^{(1)} = \sum_{K \in \mathcal{T}_h} \eta_K^{(1)} \quad \text{with} \quad \eta_K^{(1)} = B_K(\tilde{u} - u_h, \tilde{z} - z_h) - D_K(\tilde{u}_g - u_{h,g}),} \tag{3.3.29}$$

with B_K as defined in the homogeneous case, and where for $v \in H^1(\Omega)$,

$$D_K(v) = \sum_{e \subset \partial K \cap \partial\Omega} \int_e (\nabla \tilde{z}(s) \cdot n_e) v(s) ds.$$

Second estimator

The second case is based on (2.12). Recall that the residual for u_h is defined for any $v \in V_0$. Since $\gamma(\tilde{z} - z_h) = 0$, we have

$$\begin{aligned} Q(\tilde{u} - u_h) &\overset{(3.3.28)}{\approx} B(\tilde{u} - u_h, \tilde{z} - z_h) - \int_{\partial\Omega} (\nabla\tilde{z} \cdot n)(\tilde{u}_g - u_{h,g}) ds \\ &\overset{\text{Prop. } 2.7}{=} R_{u_h}(\tilde{z} - z_h) - \int_{\partial\Omega} (\nabla\tilde{z} \cdot n)(\tilde{u}_g - u_{h,g}) ds. \end{aligned} \tag{3.3.30}$$

Therefore,

$$\boxed{\eta^{(2)} = \sum_{K \in \mathcal{T}_h} \eta_K^{(2)} \quad \text{with} \quad \eta_K^{(2)} = R_{u_h}^K(\tilde{z} - z_h) - D_K(\tilde{u}_g - u_{h,g}),} \tag{3.3.31}$$

with $R_{u_h}^K$ as defined in the homogeneous case, and D_K as above.

Third estimator

The third case is based on (2.14). For any $v \in H^1(\Omega)$, we first define

$$\bar{R}_{z_h}(v) = Q(v) - B(v, z_h).$$

Then, we have

$$\begin{aligned} Q(\tilde{u} - u_h) &\overset{(3.3.26)}{=} Q(\tilde{u} - u_h) - B(\tilde{u} - u_h, z_h) \\ &= \bar{R}_{z_h}(\tilde{u} - u_h). \end{aligned} \tag{3.3.32}$$

Recall that the residual for z_h is defined for any $v \in V_0$. Since $\tilde{u} - u_h \notin V_0$ in general, $\bar{R}_{z_h}(\tilde{u} - u_h) \neq R_{z_h}(\tilde{u} - u_h)$. Therefore,

$$\boxed{\eta^{(3)} = \sum_{K \in \mathcal{T}_h} \eta_K^{(3)} \quad \text{with} \quad \eta_K^{(3)} = \bar{R}_{z_h}^K(\tilde{u} - u_h),} \tag{3.3.33}$$

where for $v \in H^1(\Omega)$,

$$\begin{aligned} \bar{R}_{z_h}^K(v) = \int_K \left(\mathbb{1}_{\Omega^*}(x) + \Delta z_h(x)\right) v(x) dx + \frac{1}{2} \sum_{e \subset \partial K \setminus \partial\Omega} \int_e [\nabla z_h(s) \cdot n_e]_e v(s) ds \\ - \sum_{e \subset \partial K \cap \partial\Omega} \int_e (\nabla z_h(s) \cdot n_e)\, v(s) ds. \end{aligned}$$

Fourth estimator

In the fourth case, based on (2.16), we have

$$\begin{aligned} Q(\tilde{u} - u_h) &\overset{(3.3.30)}{\approx} R_{u_h}(\tilde{z} - z_h) - \int_{\partial\Omega} (\nabla\tilde{z} \cdot n)(\tilde{u}_g - u_{h,g}) ds \\ &\overset{(3.3.4)}{=} \mathcal{A}(\tilde{\varphi}^u, \tilde{z} - z_h) - \int_{\partial\Omega} (\nabla\tilde{z} \cdot n)(\tilde{u}_g - u_{h,g}) ds. \end{aligned}$$

Therefore,

$$\boxed{\eta^{(4)} = \sum_{K \in \mathcal{T}_h} \eta_K^{(4)} \quad \text{with} \quad \eta_K^{(4)} = \mathcal{A}_K(\tilde{\varphi}^u, \tilde{z} - z_h) - D_K(\tilde{u}_g - u_{h,g}),} \tag{3.3.34}$$

with $\mathcal{A}_K$ as defined in the homogeneous problem.

Fifth estimator

In the fifth case, based on (2.17), we have

$$\begin{aligned}
Q(\tilde{u}-u_h) &\overset{(3.3.24)}{=} B(\tilde{u}_0-u_{h,0},z)+Q(\tilde{u}_g-u_{h,g})\\
&\overset{(3.3.27)}{=} B(\tilde{u}_0-u_{h,0},z)-B(\tilde{u}_0-u_{h,0},z_h)+Q(\tilde{u}_g-u_{h,g})\\
&\overset{(3.3.21)}{=} Q(\tilde{u}_0-u_{h,0})-B(\tilde{u}_0-u_{h,0},z_h)+Q(\tilde{u}_g-u_{h,g})\\
&= R_{z_h}(\tilde{u}_0-u_{h,0})+Q(\tilde{u}_g-u_{h,g})\\
&\overset{(3.3.25)}{=} R_{z_h}(\tilde{u}_0-u_{h,0})+B(\tilde{u}_g-u_{h,g},z)-\int_{\partial\Omega}(\nabla z\cdot n)(\tilde{u}_g-u_{h,g})ds\\
&\approx R_{z_h}(\tilde{u}_0-u_{h,0})+B(\tilde{u}_g-u_{h,g},\tilde{z})-\int_{\partial\Omega}(\nabla \tilde{z}\cdot n)(\tilde{u}_g-u_{h,g})ds.
\end{aligned}$$

Therefore,

$$\boxed{\eta^{(5)}=\sum_{K\in\mathcal{T}_h}\eta_K^{(5)} \quad \text{with} \quad \eta_K^{(5)}=\mathcal{A}_K(\tilde{u}_0-u_{h,0},\tilde{\varphi}^z)+B_K(\tilde{u}_g-u_{h,g},\tilde{z})-D_K(\tilde{u}_g-u_{h,g}).} \tag{3.3.35}$$

Sixth and seventh estimators

In the sixth case, we have

$$\boxed{\eta^{(6)}=\sum_{K\in\mathcal{T}_h}\eta_K^{(6)} \quad \text{with} \quad \eta_K^{(6)}=\frac{1}{2}(\eta_K^{(4)}+\eta_K^{(5)}).} \tag{3.3.36}$$

For the seventh estimator, we recall that $\{\varphi_i\}_{i=1}^{N_h}$ denotes the standard basis functions for the $\mathbb{P}_1$ FE space. For $i=1,\dots,N_h$, we define the patch ω_i as the support of φ_i and let r_K be the restriction of $(f+\Delta u_h)$ on K. We then have

$$\begin{aligned}
Q(\tilde{u}-u_h) &\overset{(3.3.30)}{\approx} R_{u_h}(\tilde{z}-z_h)-\int_{\partial\Omega}(\nabla\tilde{z}\cdot n)(\tilde{u}_g-u_{h,g})ds\\
&= R_{u_h}\left(\left(\sum_{i=1}^{N_h}\varphi_i\right)(\tilde{z}-z_h)\right)-\int_{\partial\Omega}(\nabla\tilde{z}\cdot n)\left(\sum_{i=1}^{N_h}\varphi_i\right)(\tilde{u}_g-u_{h,g})ds\\
&= \sum_{i=1}^{N_h}\left(R_{u_h}((\tilde{z}-z_h)\varphi_i)-\int_{\partial\Omega}(\nabla\tilde{z}\cdot n)(\tilde{u}_g-u_{h,g})\varphi_i ds\right)\\
&= \sum_{i=1}^{N_h}\sum_{K\in\omega_i}\Bigg\{\int_K(r_K(\tilde{z}-z_h)\varphi_i dx+\sum_{e\subset\partial K}\int_e[\nabla u_h\cdot n_e]_e(\tilde{z}-z_h)\varphi_i ds\\
&\quad -\sum_{e\subset\partial K\cap\partial\Omega}\int_e(\nabla\tilde{z}\cdot n_e)(\tilde{u}_g-u_{h,g})\varphi_i ds\Bigg\}.
\end{aligned}$$

Therefore,

$$\boxed{\eta^{(7)}=\sum_{i=1}^{N_h}\eta_i^{(7)} \quad \text{with} \quad \eta_i^{(7)}=\mathcal{R}_{u_h}^i(\tilde{z}-z_h)-\sum_{K\in\omega_i}D_K\left((\tilde{u}_g-u_{h,g})\varphi_i\right),} \tag{3.3.37}$$

with $\mathcal{R}_{u_h}^i$ as defined in the homogeneous problem.

Remark 3.3.3. For $e \subset \partial K \cap \partial\omega_i$, we will have $\varphi_i|_e = 0$, so

$$\int_e [\nabla u_h(s) \cdot n_e]_e v(s)\varphi_i(s)ds = 0$$

and

$$\int_e (\nabla \tilde{z}(s) \cdot n_e) v(s)\varphi_i(s)ds = 0.$$

Eighth and ninth estimators

Let $\{\tilde{\psi}_i\}_{i=1}^{\tilde{N}_h}$ be the basis functions for $\tilde{V}$. We write

$$\tilde{u} - u_h = \sum_{i=1}^{\tilde{N}_h} e_{u,i}\tilde{\psi}_i, \quad \tilde{z} - z_h = \sum_{i=1}^{\tilde{N}_h} e_{z,i}\tilde{\psi}_i, \quad \text{and} \quad \tilde{u}_g - u_{h,g} = \sum_{i=1}^{\tilde{N}_h} e_{g,i}\tilde{\psi}_i.$$

As in the homogeneous case, we set the first basis functions to be the $\mathbb{P}_1$ basis functions, i.e. $\tilde{\psi}_i = \varphi_i$ for $i = 1, \ldots, N_h$. Recall that for each vertex x_i, ω_i is the support of the corresponding basis function φ_i, and $i_1, \ldots, i_{n_i}$ are the indices of the $\mathbb{P}_2$ basis functions defined on the edges having x_i as an endpoint. We have

$$\begin{aligned}
Q(\tilde{u} - u_h) &\overset{(3.3.30)}{\approx} R_{u_h}(\tilde{z} - z_h) - \int_{\partial\Omega} (\nabla\tilde{z} \cdot n)(\tilde{u}_g - u_{h,g})ds \\
&= R_{u_h}\left(\sum_{i=1}^{N_h}\left[\varphi_i e_{z,i} + \frac{1}{2}\sum_{j=1}^{n_i}\psi_{i_j}e_{z,i_j}\right]\right) - \int_{\partial\Omega}(\nabla\tilde{z}\cdot n)\left(\sum_{i=1}^{N_h}\left[\varphi_i e_{g,i} + \frac{1}{2}\sum_{j=1}^{n_i}\psi_{i_j}e_{g,i_j}\right]\right)ds \\
&= \sum_{i=1}^{N_h}\left(R_{u_h}(\varphi_i)e_{z,i} + \frac{1}{2}\sum_{j=1}^{n_i}R_{u_h}(\psi_{i_j})e_{z,i_j}\right. \\
&\quad \left. - e_{g,i}\int_{\partial\Omega}(\nabla\tilde{z}\cdot n)\varphi_i ds + \frac{1}{2}\sum_{j=1}^{n_i}e_{g,i_j}\int_{\partial\Omega}(\nabla\tilde{z}\cdot n)\psi_{i_j}ds\right)
\end{aligned}$$

and

$$\begin{aligned}
Q(\tilde{u} - u_h) &\overset{(3.3.32)}{\approx} \bar{R}_{z_h}(\tilde{u} - u_h) \\
&= \bar{R}_{z_h}\left(\sum_{i=1}^{N_h}\left[\varphi_i e_{u,i} + \frac{1}{2}\sum_{j=1}^{n_i}\psi_{i_j}e_{u,i_j}\right]\right) \\
&= \sum_{i=1}^{N_h}\left(\bar{R}_{z_h}(\varphi_i)e_{u,i} + \frac{1}{2}\sum_{j=1}^{n_i}\bar{R}_{z_h}(\psi_{i_j})e_{u,i_j}\right).
\end{aligned}$$

The eighth error estimator is then given by

$$\boxed{\begin{aligned}
\eta^{(8)} = \sum_{i=1}^{N_h}\eta_i^{(8)} \quad \text{with} \quad \eta_i^{(8)} =& R_{u_h}(\varphi_i)e_{z,i} + \frac{1}{2}\sum_{j=1}^{n_i}R_{u_h}(\psi_{i_j})e_{z,i_j} \\
&- e_{g,i}\int_{\partial\Omega}(\nabla\tilde{z}\cdot n)\varphi_i ds - \frac{1}{2}\sum_{j=1}^{n_i}e_{g,i_j}\int_{\partial\Omega}(\nabla\tilde{z}\cdot n)\psi_{i_j}ds,
\end{aligned}} \tag{3.3.38}$$

and the ninth estimator is given by

$$\boxed{\eta^{(9)} = \sum_{i=1}^{N_h} \eta_i^{(9)} \quad \text{with} \quad \eta_i^{(9)} = \bar{R}_{z_h}(\varphi_i) e_{u,i} + \frac{1}{2} \sum_{j=1}^{n_i} \bar{R}_{z_h}(\psi_{i_j}) e_{u,i_j}.} \tag{3.3.39}$$

Remark 3.3.4. Notice that the sum of the local contributions of estimators η^{1-2}, η^{4-8} all yield the same value, namely

$$|Q(\tilde{u} - u_h)| \approx \left| \sum_i \eta_i^k \right| \quad \forall k \in \{1, 2, 4, 5, 6, 7, 8\},$$

while the sum of the local contributions of estimators $\eta^{(3)}$ and $\eta^{(9)}$ are equal to each other, and satisfy

$$|Q(\tilde{u} - u_h)| = \left| \sum_i \eta_i^k \right| \quad \forall k \in \{3, 9\}.$$

That is, the first group of estimators approximate the quantity $Q(\tilde{u} - u_h)$ with the same global value, whereas the second group computes $Q(\tilde{u} - u_h)$ exactly. However, as we will see in the numerical tests below, the local error indicators can differ from one representation to another.

Remark 3.3.5. Estimators involving the local contributions of

$$\int_{\partial\Omega} (\nabla \tilde{z} \cdot n)(\tilde{u}_g - u_{h,g}) ds,$$

namely $\eta^{(1)}$ in (3.3.29), $\eta^{(2)}$ in (3.3.31), $\eta^{(4)}$ in (3.3.34), $\eta^{(5)}$ in (3.3.35), $\eta^{(6)}$ in (3.3.36), $\eta^{(7)}$ in (3.3.37) and $\eta^{(8)}$ in (3.3.38), have an additional approximation error as they replace z with its FE approximation $\tilde{z}$. This is why the sum of their local contributions does not equal the error in the QoI $Q(\tilde{u} - u_h)$. It is possible to eliminate this error and to obtain an equality by rewriting the estimators. For all $v \in V_0$, define

$$R_{u_{h,0}}(v) = F(v) - B(u_{h,0}, v)$$

and define $\tilde{\varphi}^{u_0} \in \tilde{V}_0$ as the Riesz representant of $R_{u_{h,0}}$ with respect to the scalar product induced by $\mathcal{A}$, namely the solution of

$$\text{find } \tilde{\varphi}^{u_0} \in \tilde{V}_0 : \quad \mathcal{A}(\tilde{\varphi}^{u_0}, \tilde{v}) = R_{u_{h,0}}(\tilde{v}) \quad \forall \tilde{v} \in \tilde{V}_0. \tag{3.3.40}$$

We have

$$\begin{aligned}
Q(\tilde{u} - u_h) &= Q(\tilde{u}_0 - u_{h,0}) + Q(\tilde{u}_g - u_{h,g}) \\
&\overset{(3.3.23)}{=} B(\tilde{u}_0 - u_{h,0}, \tilde{z}) + Q(\tilde{u}_g - u_{h,g}) \\
&\overset{(3.3.27)}{=} B(\tilde{u}_0 - u_{h,0}, \tilde{z} - z_h) + Q(\tilde{u}_g - u_{h,g}) \\
&\overset{(3.3.20)}{=} F(\tilde{z} - z_h) - B(\tilde{u}_g, \tilde{z} - z_h) - B(u_{h,0}, \tilde{z} - z_h) + Q(\tilde{u}_g - u_{h,g}) \\
&\overset{(3.3.17)}{=} R_{u_{h,0}}(\tilde{z} - z_h) + Q(\tilde{u}_g - u_{h,g}) \\
&\overset{(3.3.40)}{=} \mathcal{A}(\tilde{\varphi}^{u_0}, \tilde{z} - z_h) + Q(\tilde{u}_g - u_{h,g})
\end{aligned}$$

and

$$\begin{aligned}
Q(\tilde{u} - u_h) &= Q(\tilde{u}_0 - u_{h,0}) + Q(\tilde{u}_g - u_{h,g}) \\
&\overset{(3.3.27)}{=} Q(\tilde{u}_0 - u_{h,0}) - B(\tilde{u}_0 - u_{h,0}, z_h) + Q(\tilde{u}_g - u_{h,g}) \\
&= R_{z_h}(\tilde{u}_0 - u_{h,0}) + Q(\tilde{u}_g - u_{h,g})
\end{aligned}$$

$$\overset{(3.3.6)}{=} \mathcal{A}(\tilde{u}_0 - u_{h,0}, \tilde{\varphi}^z) + Q(\tilde{u}_g - u_{h,g}).$$

Therefore, we can write

$$\eta_K^{(1)} = B_K(\tilde{u}_0 - u_{h,0}, \tilde{z} - z_h) + Q_K(\tilde{u}_g - u_{h,g}), \tag{3.3.41}$$

$$\eta_K^{(2)} = R^K_{u_{h,0}}(\tilde{z} - z_h) + Q_K(\tilde{u}_g - u_{h,g}), \tag{3.3.42}$$

$$\eta_K^{(4)} = \mathcal{A}_K(\tilde{\varphi}^{u_0}, \tilde{z} - z_h) + Q_K(\tilde{u}_g - u_{h,g}), \tag{3.3.43}$$

$$\eta_K^{(5)} = \mathcal{A}_K(\tilde{u}_0 - u_{h,0}, \tilde{\varphi}^z) + Q_K(\tilde{u}_g - u_{h,g}), \tag{3.3.44}$$

$$\eta_i^{(7)} = \mathcal{R}^i_{u_{h,0}}(\tilde{z} - z_h) + \sum_{K \in \omega_i} Q_K(\varphi_i(\tilde{u}_g - u_{h,g})), \tag{3.3.45}$$

$$\eta_i^{(8)} = R_{u_{h,0}}(\varphi_i)e_{z,i} + Q(\varphi_i)e_{g,i} + \frac{1}{2}\sum_{j=1}^{n_i}\left[R_{u_{h,0}}(\psi_{i_j})e_{z,i_j} + Q(\psi_{i_j})e_{g,i_j}\right] \tag{3.3.46}$$

where, for $v \in H^1(\Omega)$,

$$R^K_{u_{h,0}}(v) = \int_K (f(x) + \Delta u_{h,0}(x))v(x)dx + \frac{1}{2}\sum_{e \subset \partial K}\int_e [\nabla u_{h,0}(s) \cdot n_e]_e v(s)ds,$$

and

$$\mathcal{R}^i_{u_{h,0}}(v) = \sum_{K \in \omega_i}\left[\int_K (f(x) + \Delta u_{h,0}(x))v(x)\varphi_i(x)dx + \sum_{e \subset \partial K}\int_e [\nabla u_{h,0}(s) \cdot n_e]_e v(s)\varphi_i(s)ds\right].$$

We note that while these formulations are exact representations of $Q(\tilde{u} - u_h)$, they replace the local contributions of $\int_{\partial\Omega}(\nabla\tilde{z} \cdot n)(\tilde{u}_g - u_{h,g})ds$ with the local contributions Q_K of $Q(\tilde{u}_g - u_{h,g})$, which do not capture the approximation error near the boundary. For this reason, we opt for the formulations stated above, which involve the local contributions of

$$\int_{\partial\Omega}(\nabla\tilde{z} \cdot n)(\tilde{u}_g - u_{h,g})ds.$$

Chapter 4

Numerical Results

This chapter focuses on the implementation of the error estimates within both h-adaptive and p-adaptive algorithms. It presents the results of numerical tests applied to three problems: a 1D boundary-layer problem, the 1D Helmholtz equation and the 2D Poisson problem defined above.

4.1 Adaptive Algorithm

A standard adaptive algorithm is applied to two specific cases of the problem defined in Section 3.2 and two specific cases of the problem defined in Section 3.3. The adaptive loop is illustrated in Figure 4.1.

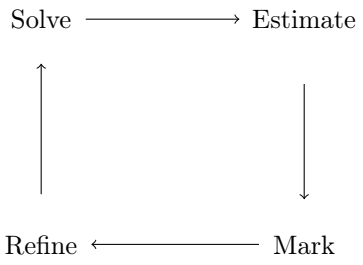


Figure 4.1: Standard adaptive loop

1. An initial mesh $\mathcal{T}_h$ is defined and the polynomial degree 1 is set on each element.
2. At each iteration,
   - The primal and dual problems are solved;
   - For element-based indicators:
     - The local error indicators η_K are calculated on each element $K \in \mathcal{T}_h$.
     - For a fixed value $\theta \in (0, 1)$, the elements are selected according to either the *Maximum criterion*, which selects all K satisfying

$$|\eta_K| \geq \theta \max_{K \in \mathcal{T}_h} |\eta_K|, \tag{4.1.1}$$

     or the *Dörfler criterion* [17], which selects the subset I_h of $\mathcal{T}_h$ of minimal cardinality such that

$$\sum_{K \in I_h} |\eta_K| \geq (1 - \theta) \sum_{K \in \mathcal{T}_h} |\eta_K|. \tag{4.1.2}$$

  - The selected elements are then refined or assigned a higher polynomial degree.
- For patch-based indicators, denoting N_v as the number of vertices in the mesh ($N_v = N + 2$ in 1D and $N_v = N_h$ in 2D):
  - The local error indicators η_i are calculated at each patch.
  - For a fixed value $\theta \in (0, 1)$, the patches are selected according to either the *Maximum criterion*, which selects all i satisfying
$$|\eta_i| \geq \theta \max_{1 \leq j \leq N_v} |\eta_j|, \tag{4.1.3}$$
    or the *Dörfler criterion*, which selects the subset I_v of $\{1, \ldots, N_v\}$ of minimal cardinality such that
$$\sum_{i \in I_v} |\eta_i| \geq (1 - \theta) \sum_{j=1}^{N_v} |\eta_j|. \tag{4.1.4}$$
  - The elements in the selected patches are then refined or assigned a higher polynomial degree.

3. The algorithm stops when
$$\left| \frac{\eta}{Q(\tilde{u})} \right| < Tol,$$
where Tol is a given tolerance, $\eta = \sum_{K \in \mathcal{T}_h} \eta_K$ for element-based indicators and $\eta = \sum_{j=1}^{N_v} \eta_i$ for patch-based indicators.

Remark 4.1.1. For smaller θ values, both strategies will mark more elements for refinement. Conversely, as θ approaches 1, the strategies select fewer elements. In the numerical results below, the value of θ is fixed, and thus its effect on the performance of the algorithms is not examined.

The numerical results that follow will vary according to the chosen local error indicator and marking strategy. It is important to note that the h-adaptive algorithms use different refinement strategies in the 1D and 2D settings.

4.2 Numerical Results in 1D

In the first set of numerical experiments, we implement h-adaptive and p-adaptive algorithms for one-dimensional problems in Octave. In the h-adaptive algorithms, marked elements are split into two equal parts, and in the p-adaptive approach, the polynomial degree assigned to each marked element is increased by one. Gaussian quadrature is used with suitably chosen polynomial degrees so that the underlying integrals are computed exactly.

4.2.1 First Problem

We first consider a boundary-layer problem by setting $a = \varepsilon$, $b = 1$, $c = 0$ and $f = 1$. We impose homogeneous Dirichlet boundary conditions and consider the quantity of interest

$$Q(u) = u(x^*).$$

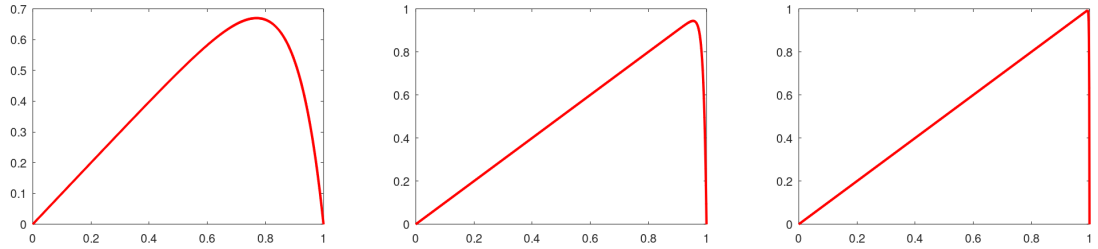
The primal and dual problems become

$$\begin{cases} -\varepsilon u'' + u' = 1 \text{ in } (0, 1) \\ u = 0 \text{ for } x = 0, 1 \end{cases} \quad \text{and} \quad \begin{cases} -\varepsilon z'' - z' = \delta_{x^*} \text{ in } (0, 1) \\ z = 0 \text{ for } x = 0, 1. \end{cases}$$

The exact solutions to these problems are, respectively,

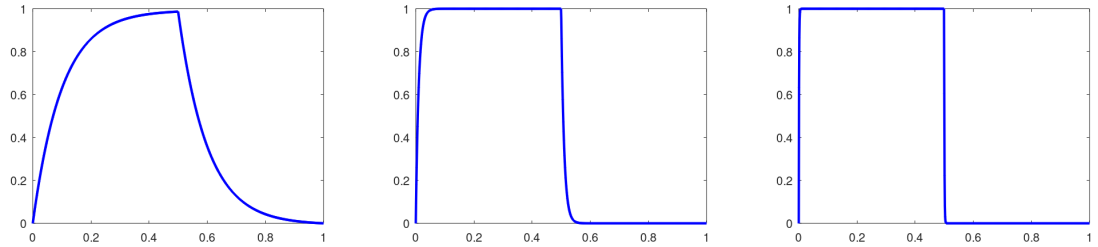
$$u(x) = x - \frac{e^{-1/\varepsilon} - e^{(x-1)/\varepsilon}}{e^{-(1/\varepsilon)} - 1} \quad \text{and} \quad z(x) = \frac{1 - e^{-x/\varepsilon}}{1 - e^{-1/\varepsilon}} \times (1 - e^{(x^*-1)/\varepsilon}) - H(x - x^*) \times (1 - e^{(x^*-x)/\varepsilon}).$$

Figure 4.2 shows the exact primal solution, Figure 4.3 shows the exact dual solution for $x^* = 0.5$, and Figure 4.4 shows the exact dual solution for $x^* = 0.1$.



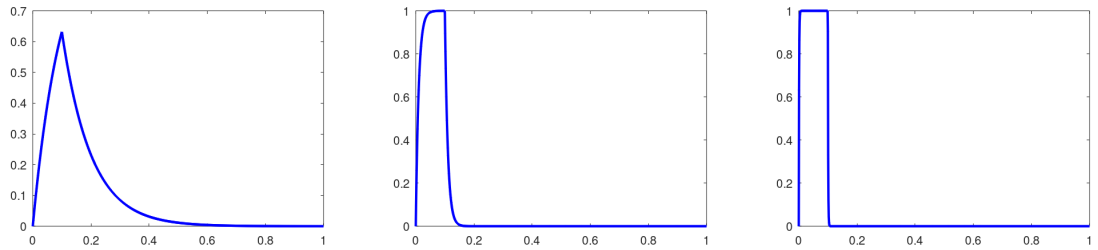
(a) $\varepsilon = 10^{-1}$ (b) $\varepsilon = 10^{-2}$ (c) $\varepsilon = 10^{-3}$

Figure 4.2: Primal solution of the boundary-layer problem for various ε values



(a) $\varepsilon = 10^{-1}$ (b) $\varepsilon = 10^{-2}$ (c) $\varepsilon = 10^{-3}$

Figure 4.3: Dual solution of the boundary-layer problem for $x^* = 0.5$ and various ε values

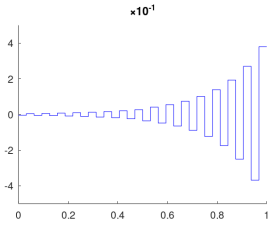
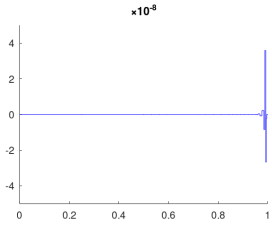
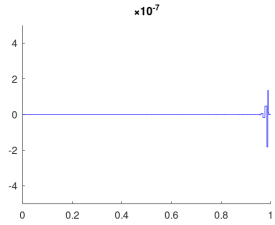
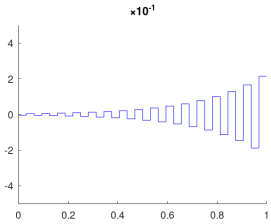
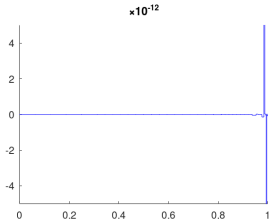
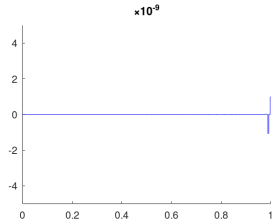
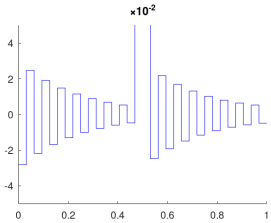
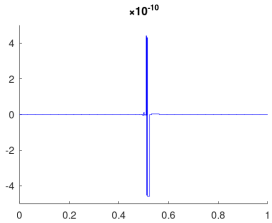
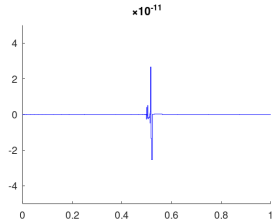
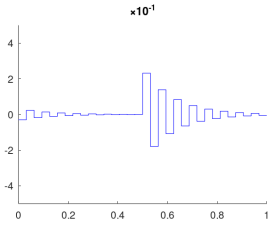
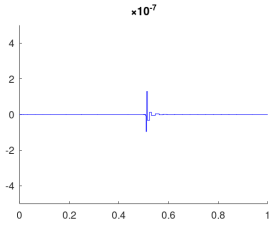
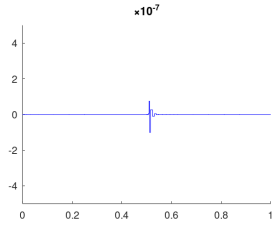
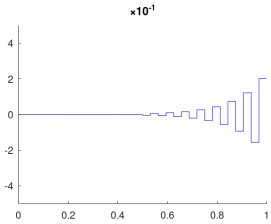
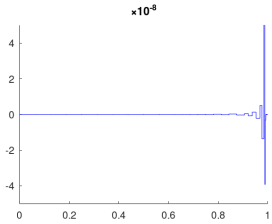
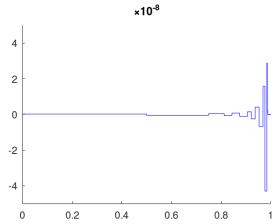
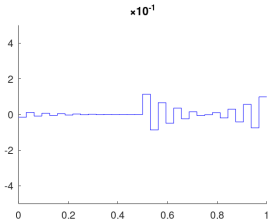
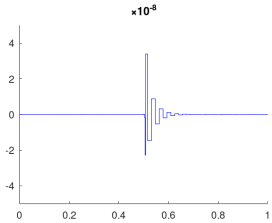
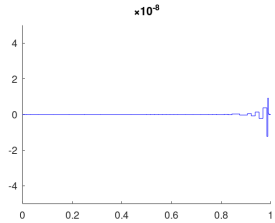
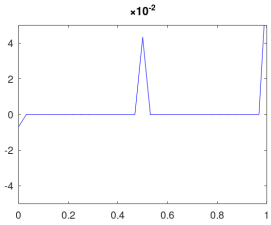
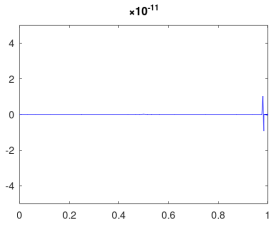
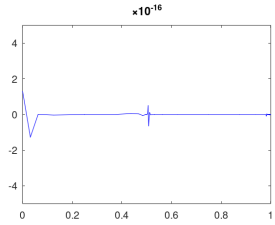


(a) $\varepsilon = 10^{-1}$ (b) $\varepsilon = 10^{-2}$ (c) $\varepsilon = 10^{-3}$

Figure 4.4: Dual solution of the boundary-layer problem for $x^* = 0.1$ and various ε values

First case: $x^* = 0.5$

Using h-adaptation, we run the algorithm for $\varepsilon = 10^{-3}$, $\theta = 0.5$ and $x^* = 0.5$ with a set tolerance of 10^{-10} and an initial mesh of two elements. We compare the distribution of the error estimators on a coarse uniform mesh of 32 elements with the distribution obtained after running the algorithm with each marking strategy.

Estimator	Uniform mesh	Maximum	Dörfler
$\eta^{(1)}$ in (3.2.8)			
$\eta^{(2)}$ in (3.2.10)			
$\eta^{(3)}$ in (3.2.13)			
$\eta^{(4)}$ in (3.2.14)			
$\eta^{(5)}$ in (3.2.15)			
$\eta^{(6)}$ in (3.2.16)			
$\eta^{(7)}$ in (3.2.17)			

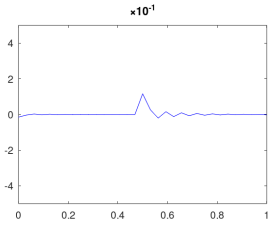
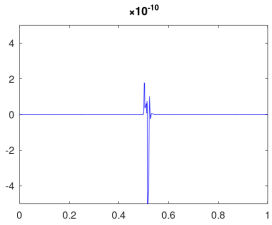
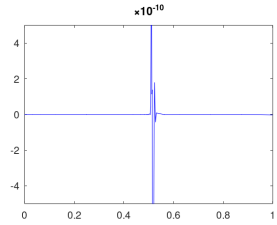
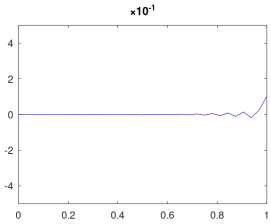
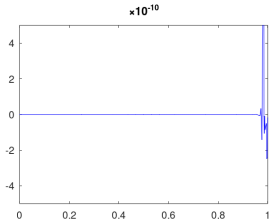
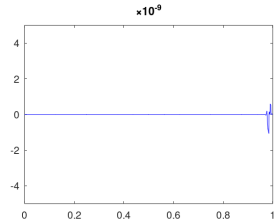
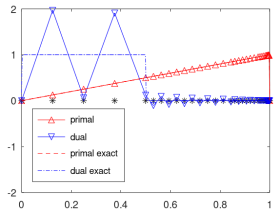
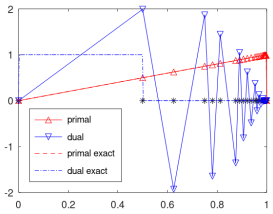
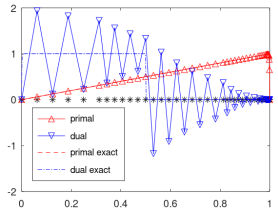
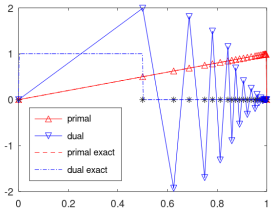
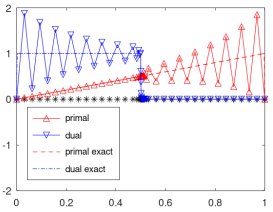
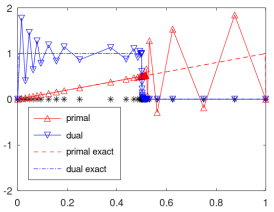
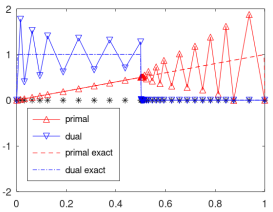
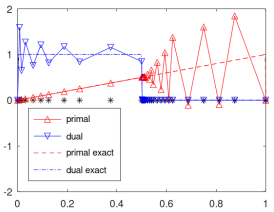
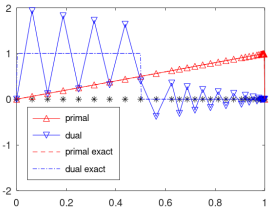
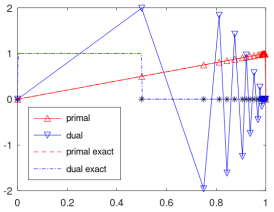
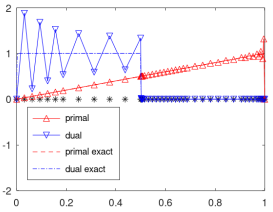
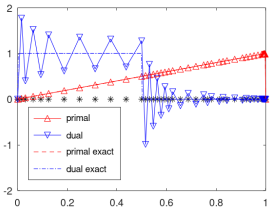
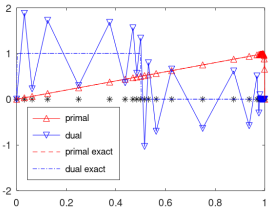
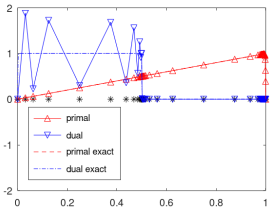
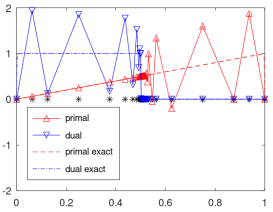
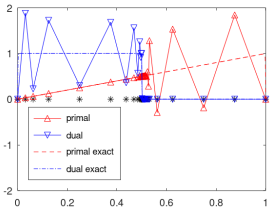
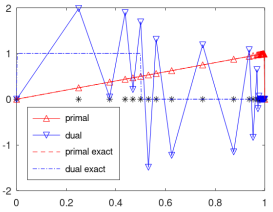
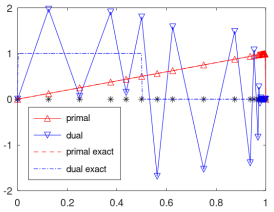
Estimator	Uniform mesh	Maximum	Dörfler
$\eta^{(8)}$ in (3.2.18)			
$\eta^{(9)}$ in (3.2.19)			

Table 4.1: Local error estimators before and after running the adaptive algorithm for $x^* = 0.5$

From the plots shown in Table 4.1, we can see that estimators $\eta^{(1)}$ in (3.2.8), $\eta^{(2)}$ in (3.2.10), $\eta^{(5)}$ in (3.2.15) and $\eta^{(9)}$ in (3.2.19) are localized in the primal, and that estimators $\eta^{(3)}$ in (3.2.13), $\eta^{(4)}$ in (3.2.14) and $\eta^{(8)}$ in (3.2.18) are localized in the dual. Estimators $\eta^{(6)}$ in (3.2.16) and $\eta^{(7)}$ in (3.2.17) capture the error in both problems. Note that estimators localized in both solutions, particularly $\eta^{(6)}$ in (3.2.16), have different post-adaptation distributions according to the marking strategy.

We next observe the primal numerical solution, the dual numerical solution and the final mesh nodes obtained for each marking strategy and error estimator.

Estimator	Maximum	Dörfler
$\eta^{(1)}$ in (3.2.8)		
$\eta^{(2)}$ in (3.2.10)		

Estimator	Maximum	Dörfler
$\eta^{(3)}$ in (3.2.13)		
$\eta^{(4)}$ in (3.2.14)		
$\eta^{(5)}$ in (3.2.15)		
$\eta^{(6)}$ in (3.2.16)		
$\eta^{(7)}$ in (3.2.17)		
$\eta^{(8)}$ in (3.2.18)		
$\eta^{(9)}$ in (3.2.19)		

Estimator	Maximum	Dörfler

Table 4.2: Numerical solutions obtained for $x^* = 0.5$

From the numerical solutions obtained in Table 4.2, we see that the estimators better approximate the solution in which they are localized. For example, the estimator $\eta^{(1)}$ in (3.2.8) captures the error in the primal problem, and when applied in the adaptive algorithm, the resulting primal numerical solution is significantly more accurate than the resulting dual solution. That being said, for each estimator, we can see that both solutions are accurate near $x^* = 0.5$. Finally, we compare the final meshes, number of iterations and global errors.

Estimator	$Q(e_u)$		$N+2$		No. of iterations	
	Maximum	Dörfler	Maximum	Dörfler	Maximum	Dörfler
$\eta^{(1)}$ in (3.2.8)	1.2235×10^{-11}	-4.1123×10^{-12}	34	25	10	11
$\eta^{(2)}$ in (3.2.10)	4.0090×10^{-13}	-4.8341×10^{-11}	44	25	11	10
$\eta^{(3)}$ in (3.2.13)	-2.1289×10^{-11}	-1.1036×10^{-12}	52	53	14	24
$\eta^{(4)}$ in (3.2.14)	3.9496×10^{-11}	-1.9265×10^{-12}	39	34	12	13
$\eta^{(5)}$ in (3.2.15)	1.2396×10^{-11}	-6.0640×10^{-13}	36	25	12	13
$\eta^{(6)}$ in (3.2.16)	2.8986×10^{-11}	-5.8216×10^{-12}	45	46	10	15
$\eta^{(7)}$ in (3.2.17)	7.0610×10^{-13}	7.7716×10^{-16}	44	53	18	18
$\eta^{(8)}$ in (3.2.18)	-2.2202×10^{-12}	-3.1641×10^{-14}	39	34	12	11
$\eta^{(9)}$ in (3.2.19)	3.0990×10^{-12}	-4.0512×10^{-13}	34	32	12	12

Table 4.3: Global error values, number of vertices and number of iterations obtained with each estimator, using the Maximum marking strategy and Dörfler marking strategy for $x^* = 0.5$

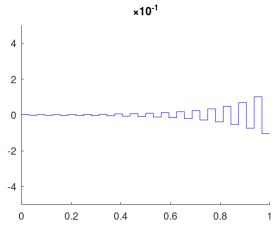
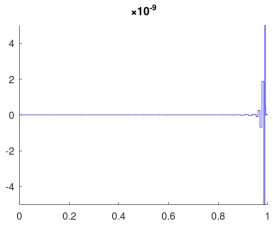
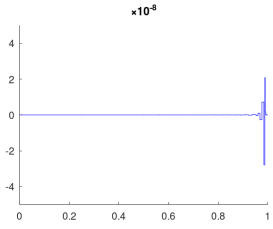
From the values displayed in Table 4.3, we see that in this case, the Dörfler strategy applied with estimators $\eta^{(1)}$ in (3.2.8), $\eta^{(2)}$ in (3.2.10), and $\eta^{(5)}$ in (3.2.15), requires the fewest nodes to reach the prescribed tolerance. Additionally, this strategy minimizes the global error with estimator $\eta^{(7)}$ in (3.2.17) as the local refinement indicator.

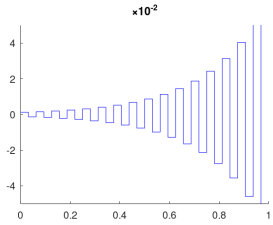
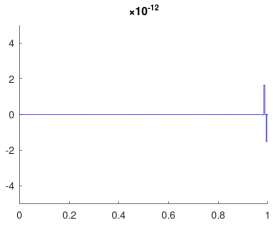
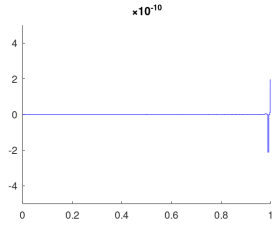
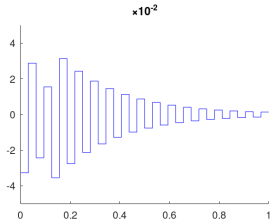
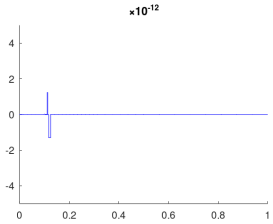
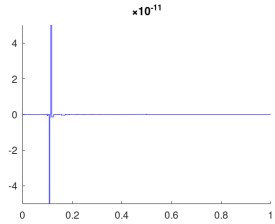
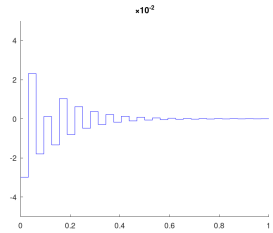
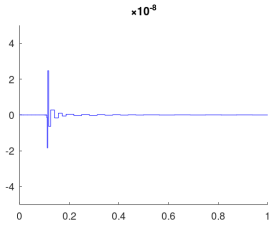
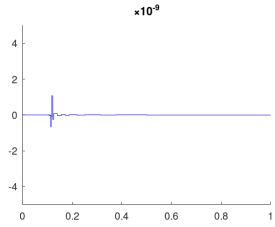
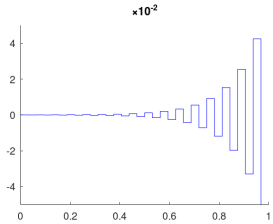
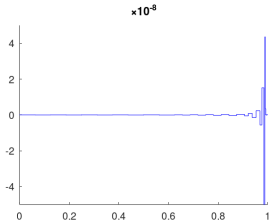
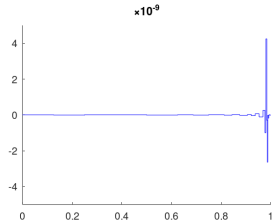
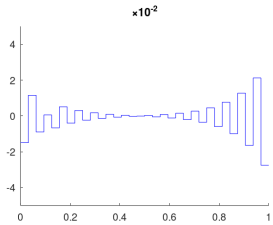
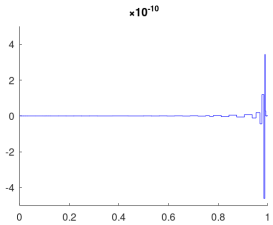
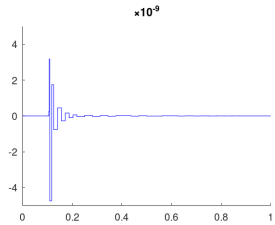
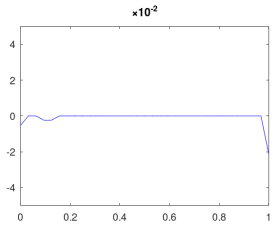
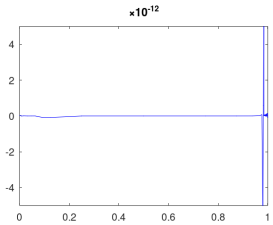
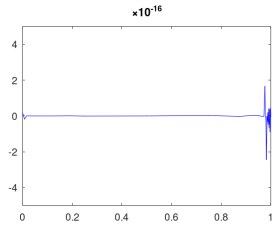
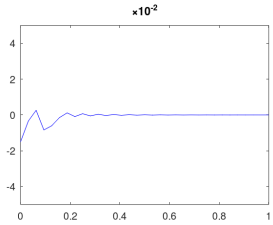
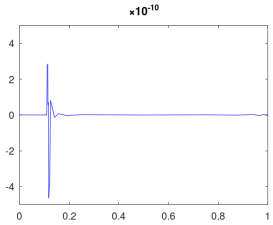
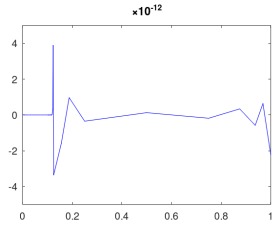
Examining the last column, we observe that the choice of marking strategy has little impact on the number of iterations required to reach the prescribed tolerance, with the exception of estimator $\eta^{(3)}$ in (3.2.13).

Note that applying an algorithm with uniform refinement, prescribing the same tolerance, we obtain $Q(e_u) = 2.1094 \times 10^{-15}$ with a final mesh of 129 nodes.

Second case: $x^* = 0.1$

In Tables 4.4, 4.5 and 4.6, we present the results obtained from repeating the numerical tests, now with $x^* = 0.1$.

Estimator	Uniform mesh	Maximum	Dörfler
$\eta^{(1)}$ in (3.2.8)			

Estimator	Uniform mesh	Maximum	Dörfler
$\eta^{(2)}$ in (3.2.10)			
$\eta^{(3)}$ in (3.2.13)			
$\eta^{(4)}$ in (3.2.14)			
$\eta^{(5)}$ in (3.2.15)			
$\eta^{(6)}$ in (3.2.16)			
$\eta^{(7)}$ in (3.2.17)			
$\eta^{(8)}$ in (3.2.18)			

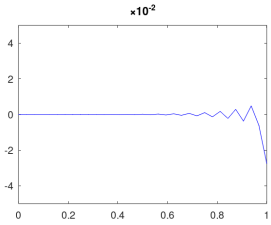
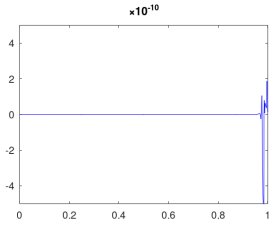
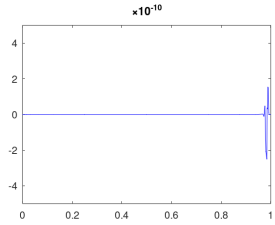
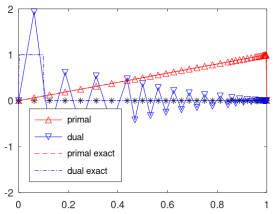
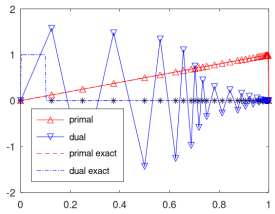
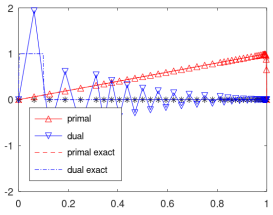
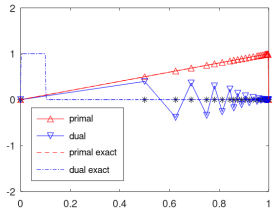
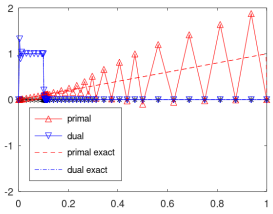
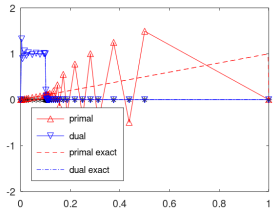
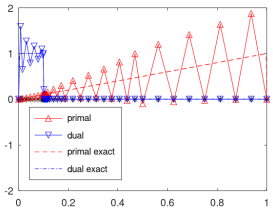
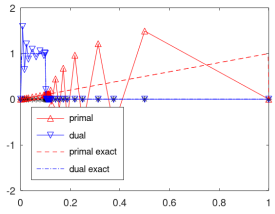
Estimator	Uniform mesh	Maximum	Dörfler
$\eta^{(9)}$ in (3.2.19)			

Table 4.4: Local error estimators before and after running the adaptive algorithm for $x^* = 0.1$

From the results in Table 4.4, we once again observe that estimators localized in both solutions, particularly $\eta^{(6)}$ in (3.2.16), have different post-adaptation distributions according to the marking strategy. However, while with $x^* = 0.5$, the Maximum strategy produced a localization in the dual solution and the Dörfler strategy produced a localization in the primal solution, the opposite is the case for $x^* = 0.1$.

Estimator	Maximum	Dörfler
$\eta^{(1)}$ in (3.2.8)		
$\eta^{(2)}$ in (3.2.10)		
$\eta^{(3)}$ in (3.2.13)		
$\eta^{(4)}$ in (3.2.14)		

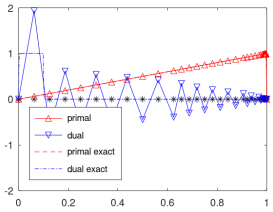
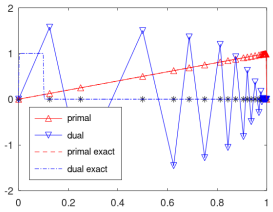
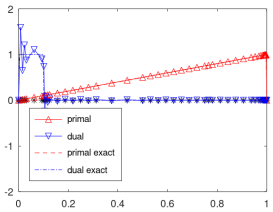
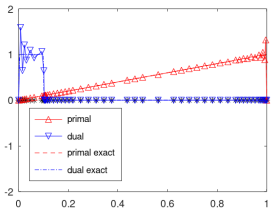
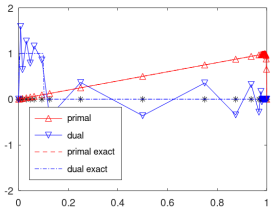
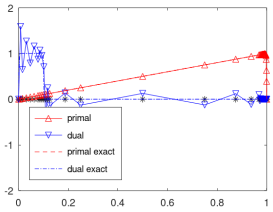
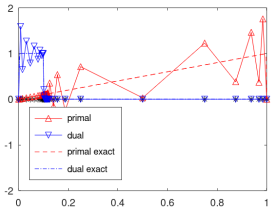
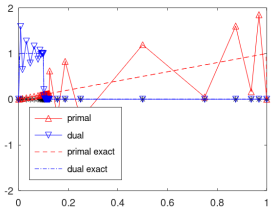
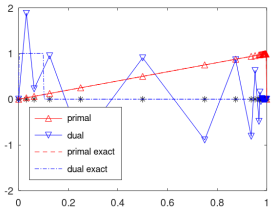
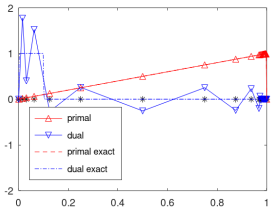
Estimator	Maximum	Dörfler
$\eta^{(5)}$ in (3.2.15)		
$\eta^{(6)}$ in (3.2.16)		
$\eta^{(7)}$ in (3.2.17)		
$\eta^{(8)}$ in (3.2.18)		
$\eta^{(9)}$ in (3.2.19)		

Table 4.5: Numerical solutions obtained for $x^* = 0.1$

Estimator	$Q(e_u)$		$N+2$		No. of iterations	
	Maximum	Dörfler	Maximum	Dörfler	Maximum	Dörfler
$\eta^{(1)}$ in (3.2.8)	-1.6359×10^{-13}	-6.3249×10^{-13}	41	35	11	13
$\eta^{(2)}$ in (3.2.10)	7.4593×10^{-14}	-9.6682×10^{-12}	44	25	11	10
$\eta^{(3)}$ in (3.2.13)	-6.0590×10^{-14}	3.9800×10^{-12}	53	38	16	17
$\eta^{(4)}$ in (3.2.14)	6.5522×10^{-12}	1.9602×10^{-13}	44	37	13	15
$\eta^{(5)}$ in (3.2.15)	-1.3797×10^{-11}	6.9234×10^{-13}	36	31	12	15
$\eta^{(6)}$ in (3.2.16)	-2.1733×10^{-13}	-5.2941×10^{-12}	50	51	14	14
$\eta^{(7)}$ in (3.2.17)	-3.9388×10^{-13}	1.3878×10^{-16}	40	50	16	18
$\eta^{(8)}$ in (3.2.18)	-1.9106×10^{-12}	-2.7756×10^{-17}	40	44	14	16
$\eta^{(9)}$ in (3.2.19)	-2.3234×10^{-12}	-9.6867×10^{-15}	31	27	12	10

Table 4.6: Global error values, number of vertices and number of iterations obtained with each estimator, using the Maximum marking strategy and Dörfler marking strategy for $x^* = 0.1$.

From the values in Table 4.6, we see that in this case, the Dörfler strategy used with estimator $\eta^{(2)}$ in (3.2.10), requires the least amount of nodes to reach the set tolerance. The global error is also minimized with this strategy, applying estimator $\eta^{(8)}$ in (3.2.18) as the local refinement indicator.

Note that applying an algorithm with uniform refinement, prescribing the same tolerance, we obtain $Q(e_u) = -6.1062 \times 10^{-16}$ with a final mesh of 129 nodes.

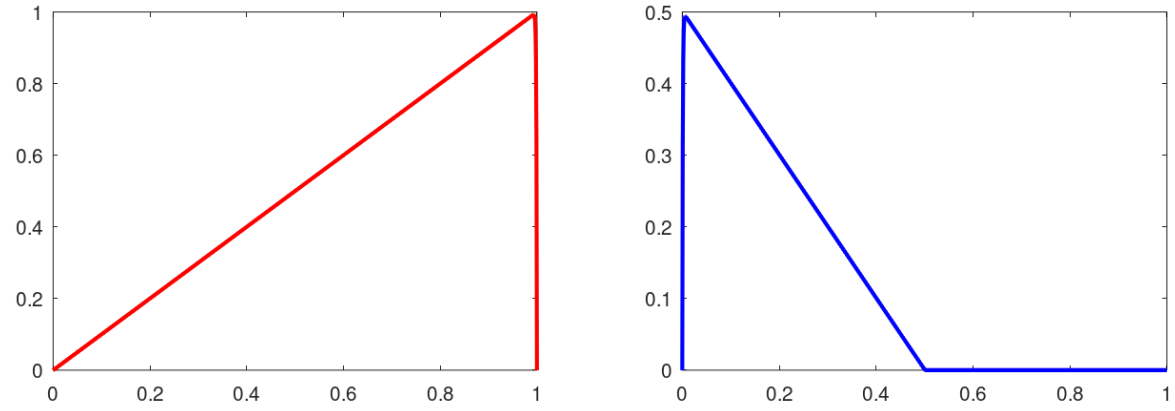
Remark 4.2.1. Running the results with a different set tolerance will affect the global error and final mesh. However, it will not significantly change the localization (as defined in Remark 2.2) or the relative ranking of the different error estimators (as shown below).

Third case: $\Omega^* = (0, 0.5)$

We now consider the boundary-layer problem with the integral of the solution over a subdomain as the QoI. Taking $\Omega^* = (0, 0.5)$, the primal and dual problems become

$$\begin{cases} -\varepsilon u'' + u' = 1 \text{ in } (0,1) \\ u = 0 \text{ for } x = 0, 1 \end{cases} \quad \text{and} \quad \begin{cases} -\varepsilon z'' - z' = \mathbb{1}_{[0,0.5]} \text{ in } (0,1) \\ z = 0 \text{ for } x = 0, 1. \end{cases}$$

Figure 4.5 shows the exact primal and dual solutions.

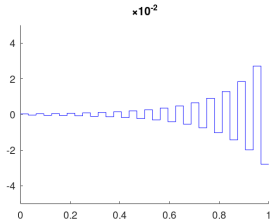
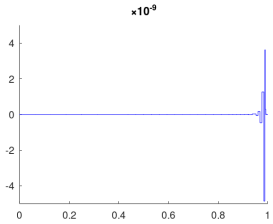
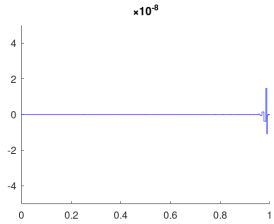
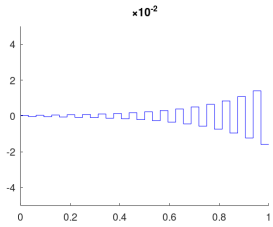
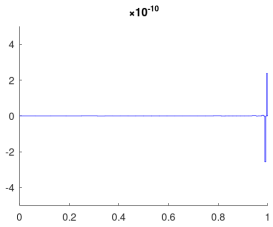
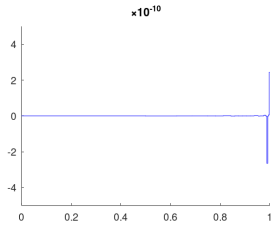
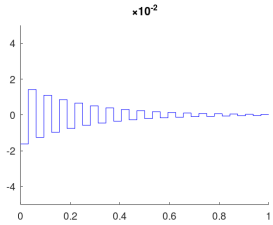
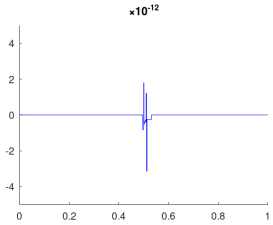
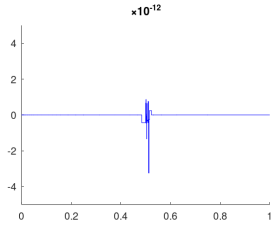
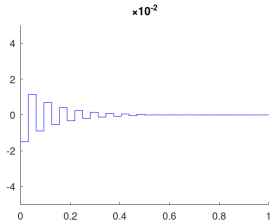
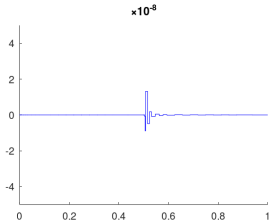
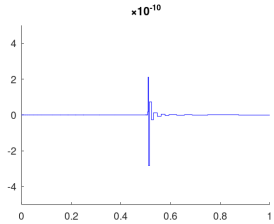
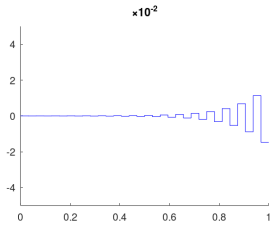
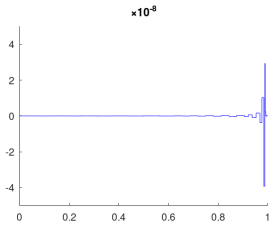
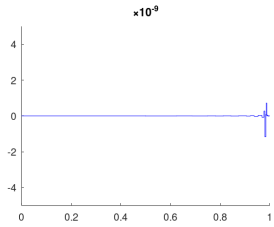
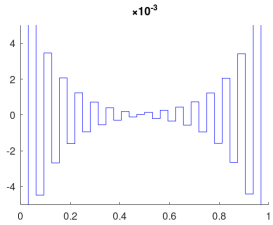
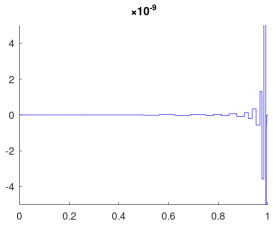
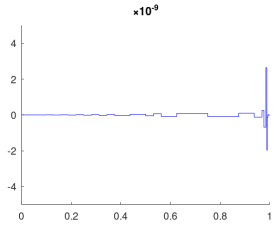


(a) Primal solution

(b) Dual solution

Figure 4.5: Primal and dual solutions of boundary-layer problem for $\Omega_* = (0, 0.5)$ and $\varepsilon = 10^{-3}$

Once again using h-adaptation, we run the algorithm for $\varepsilon = 10^{-3}$, $\theta = 0.5$, a set tolerance of 10^{-10} and an initial mesh of two elements. We compare the distribution of the error estimators on a coarse uniform mesh of 32 elements with the distribution obtained after running the algorithm with each marking strategy.

Estimator	Uniform mesh	Maximum	Dörfler
$\eta^{(1)}$ in (3.2.8)			
$\eta^{(2)}$ in (3.2.10)			
$\eta^{(3)}$ in (3.2.13)			
$\eta^{(4)}$ in (3.2.14)			
$\eta^{(5)}$ in (3.2.15)			
$\eta^{(6)}$ in (3.2.16)			

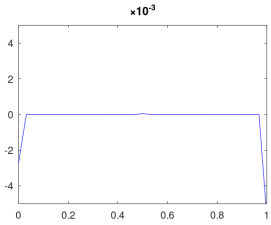
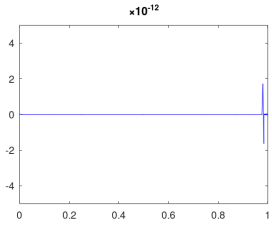
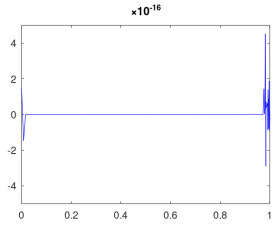
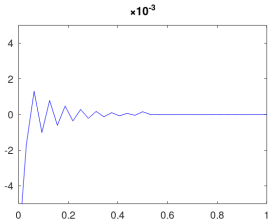
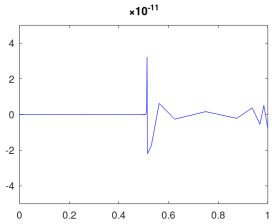
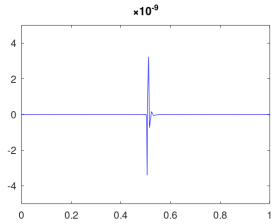
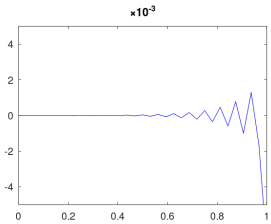
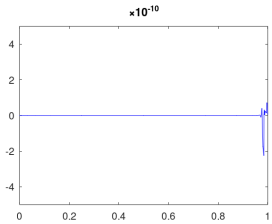
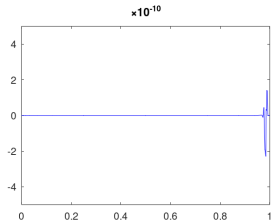
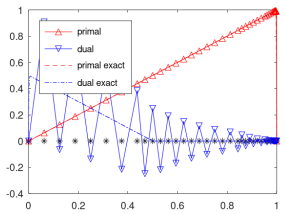
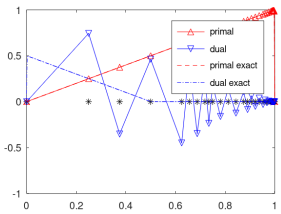
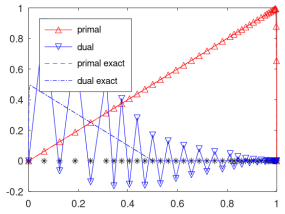
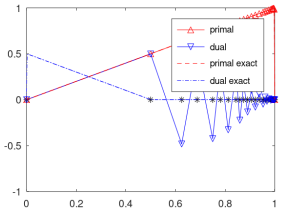
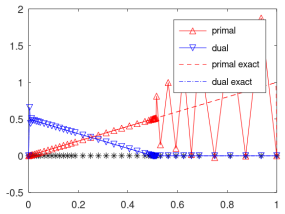
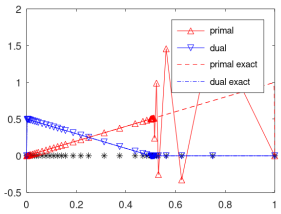
Estimator	Uniform mesh	Maximum	Dörfler
$\eta^{(7)}$ in (3.2.17)			
$\eta^{(8)}$ in (3.2.18)			
$\eta^{(9)}$ in (3.2.19)			

Table 4.7: Local error estimators before and after running the adaptive algorithm for $\Omega^* = (0, 0.5)$

Estimator	Maximum	Dörfler
$\eta^{(1)}$ in (3.2.8)		
$\eta^{(2)}$ in (3.2.10)		
$\eta^{(3)}$ in (3.2.13)		

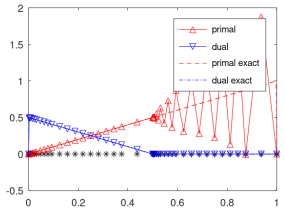
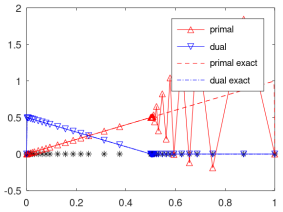
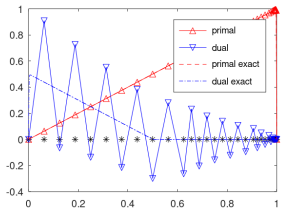
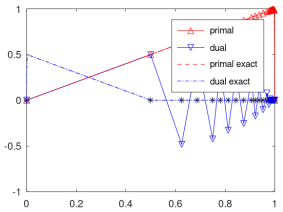
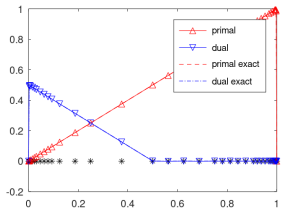
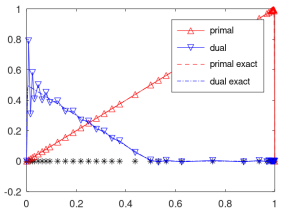
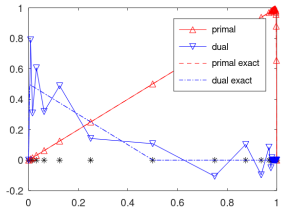
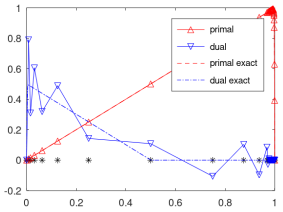
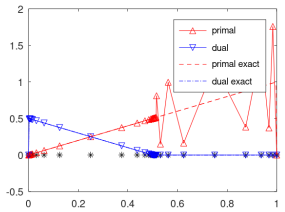
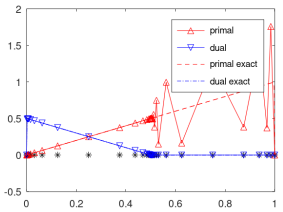
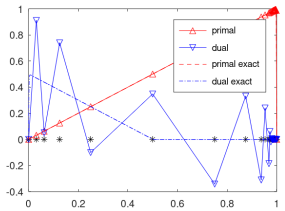
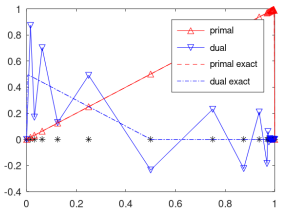
Estimator	Maximum	Dörfler
$\eta^{(4)}$ in (3.2.14)		
$\eta^{(5)}$ in (3.2.15)		
$\eta^{(6)}$ in (3.2.16)		
$\eta^{(7)}$ in (3.2.17)		
$\eta^{(8)}$ in (3.2.18)		
$\eta^{(9)}$ in (3.2.19)		

Table 4.8: Numerical solutions obtained for $\Omega_* = (0, 0.5)$

From Tables 4.7 and 4.8, as before, we see that the estimators better approximate the solution in which they are localized. However, estimators localized in both solutions no longer have different post-adaptation distributions according to the marking strategy. Both $\eta^{(6)}$ in (3.2.16) and $\eta^{(7)}$ (3.2.17) are more refined in the primal solution following the adaptive algorithm. That being said, these two estimators provide accurate

numerical solutions in both the primal and dual, with close to no oscillations. We now compare the final meshes, number of iterations and global errors.

Estimator	$Q(e_u)$		$N+2$		No. of iterations	
	Maximum	Dörfler	Maximum	Dörfler	Maximum	Dörfler
$\eta^{(1)}$ in (3.2.8)	1.1033×10^{-13}	3.3100×10^{-13}	41	32	11	12
$\eta^{(2)}$ in (3.2.10)	1.1731×10^{-11}	1.2085×10^{-11}	42	25	10	10
$\eta^{(3)}$ in (3.2.13)	1.1324×10^{-14}	1.2629×10^{-14}	55	49	16	23
$\eta^{(4)}$ in (3.2.14)	4.0633×10^{-12}	6.7002×10^{-14}	45	41	14	19
$\eta^{(5)}$ in (3.2.15)	9.2535×10^{-12}	1.8965×10^{-13}	36	30	12	14
$\eta^{(6)}$ in (3.2.16)	5.9740×10^{-12}	1.2468×10^{-12}	37	39	10	12
$\eta^{(7)}$ in (3.2.17)	1.1638×10^{-13}	1.2768×10^{-15}	38	43	16	17
$\eta^{(8)}$ in (3.2.18)	8.3267×10^{-17}	2.7567×10^{-13}	38	35	17	17
$\eta^{(9)}$ in (3.2.19)	8.9126×10^{-13}	8.2989×10^{-12}	31	27	12	11

Table 4.9: Global error values, number of vertices and number of iterations obtained with each estimator, using the Maximum marking strategy and Dörfler marking strategy for $\Omega_* = (0, 0.5)$.

From the values in Table 4.9, we see that as before, the Dörfler strategy used with estimator $\eta^{(2)}$ in (3.2.10) requires the least amount of nodes to reach the set tolerance. The global error is still minimized with estimator $\eta^{(8)}$ in (3.2.18), this time applying the Maximum marking strategy.

Note that applying an algorithm with uniform refinement, prescribing the same tolerance, we obtain $Q(e_u) = 2.7756 \times 10^{-16}$ with a final mesh of 129 nodes.

Classification of estimators

As an attempt to classify the nine estimators, we sum up the degrees of freedom required to reach the tolerance for each quantity of interest.

Estimator	$x^* = 0.5$		$x^* = 0.1$		$\Omega_* = (0, 0.5)$		Sum	
	Max	Dörfler	Max	Dörfler	Max	Dörfler	Max	Dörfler
$\eta^{(1)}$ in (3.2.8)	34	25	41	35	41	32	116	92
$\eta^{(2)}$ in (3.2.10)	44	25	44	25	42	25	130	75
$\eta^{(3)}$ in (3.2.13)	52	53	53	38	55	49	160	140
$\eta^{(4)}$ in (3.2.14)	39	34	44	37	45	41	128	112
$\eta^{(5)}$ in (3.2.15)	36	25	36	31	36	30	108	86
$\eta^{(6)}$ in (3.2.16)	45	46	50	51	37	39	132	136
$\eta^{(7)}$ in (3.2.17)	44	53	40	50	38	43	122	146
$\eta^{(8)}$ in (3.2.18)	39	34	40	44	38	35	117	113
$\eta^{(9)}$ in (3.2.19)	34	32	31	27	31	27	96	86

Table 4.10: Sum of degrees of freedom required with each estimator and marking strategy, for each quantity of interest

Rank	Sum of d.o.f.	Estimator	Strategy
1	75	$\eta^{(2)}$ in (3.2.10)	Dörfler
2	86	$\eta^{(9)}$ in (3.2.19)	Dörfler
3	86	$\eta^{(5)}$ in (3.2.15)	Dörfler
4	92	$\eta^{(1)}$ in (3.2.8)	Dörfler
5	96	$\eta^{(9)}$ in (3.2.19)	Max
6	108	$\eta^{(5)}$ in (3.2.15)	Max
7	112	$\eta^{(4)}$ in (3.2.14)	Dörfler
8	113	$\eta^{(8)}$ in (3.2.18)	Dörfler
9	116	$\eta^{(1)}$ in (3.2.8)	Max
10	117	$\eta^{(8)}$ in (3.2.18)	Max
11	122	$\eta^{(7)}$ in (3.2.17)	Max
13	128	$\eta^{(4)}$ in (3.2.14)	Max
14	130	$\eta^{(2)}$ in (3.2.10)	Max
15	132	$\eta^{(6)}$ in (3.2.16)	Max
16	136	$\eta^{(6)}$ in (3.2.16)	Dörfler
17	140	$\eta^{(3)}$ in (3.2.13)	Dörfler
18	146	$\eta^{(7)}$ in (3.2.17)	Dörfler
20	160	$\eta^{(3)}$ in (3.2.13)	Max

Table 4.11: Ranking of estimators according to the number of degrees of freedom required to reach the set tolerance. Estimators with the same number of degrees of freedom are ranked by smallest $Q(e_u)$ value.

The results in Tables 4.10 and 4.11 indicate that, among the tested approaches for the boundary layer problem, the primal-localized estimator $\eta^{(2)}$ in (3.2.10) provides the most efficient error estimation when incorporated into an h-adaptive refinement procedure with the Dörfler marking strategy.

4.2.2 Second Problem

We next consider the Helmholtz problem tested in [12] by setting $a = 1$, $b = 0$, $c = -k^2$ and $f = 1$. Referring to Remark 3.2.4, we impose the Neumann boundary condition $u'(1) = 0.5$ and consider the quantity of interest

$$Q(u) = \int_{\Omega^*} u(x),$$

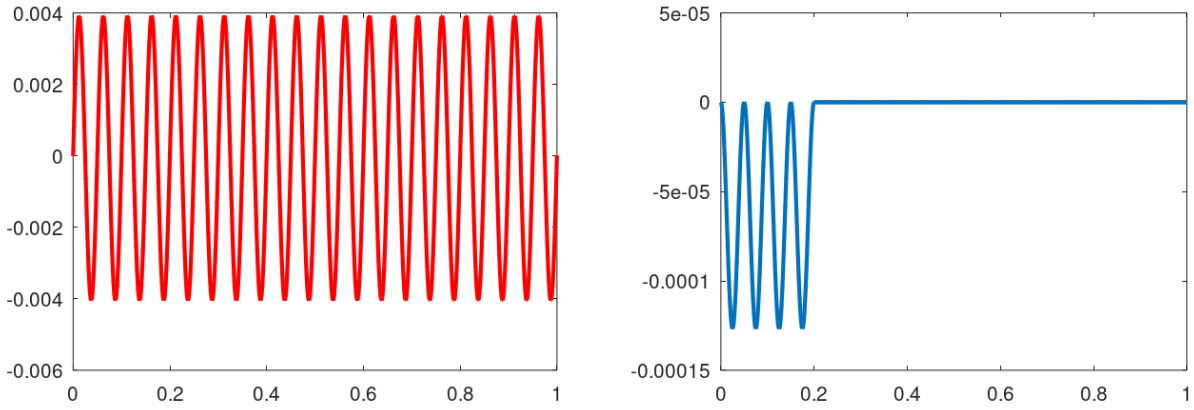
where $\Omega^* = (0, 0.2)$. The primal and dual problems become

$$\begin{cases} -u'' - k^2 u = 1 \text{ in } (0,1) \\ u(0) = 0 \\ u'(1) = 0.5 \end{cases}$$

and

$$\begin{cases} -z'' - k^2 z = \mathbb{1}_{[0,0.2]} \text{ in } (0,1) \\ z(0) = 0 \\ z'(1) = 0. \end{cases}$$

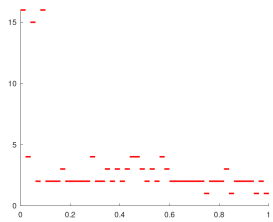
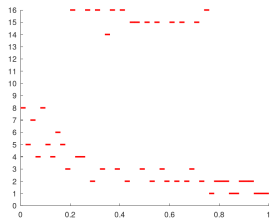
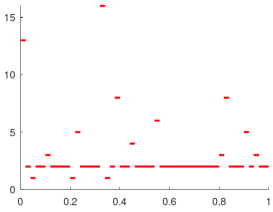
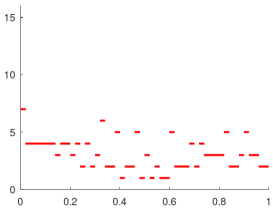
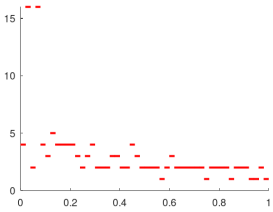
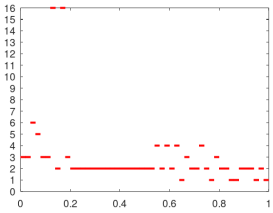
In this case, the bilinear form for the primal problem is not coercive. Therefore, we cannot apply the Lax-Milgram Lemma. We refer to Chapter 35 in [20] for details on the well-posedness of this problem. Figure 4.6 shows the exact primal and dual solutions for $k = 40\pi$.



(a) Primal solution (b) Dual solution

Figure 4.6: Primal and dual solutions of Helmholtz problem for $k = 40\pi$

Using p-adaptation, we run the algorithm with a set tolerance of 10^{-3} and a maximum polynomial degree of 16. The problems are solved on a mesh of 50 elements, where the enriched space $\tilde{V}$ is constructed by increasing the polynomial degree of each element in $V_{h,p}$ by two. We observe the polynomial degree reached on each element with each marking strategy and local error indicator, with $\theta = 0.5$.

Estimator	Maximum	Dörfler
$\eta^{(1)}$ in (3.2.8)		
$\eta^{(2)}$ in (3.2.10)		
$\eta^{(3)}$ in (3.2.13)		

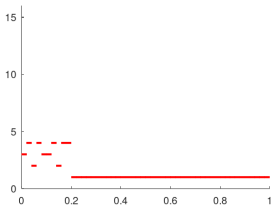
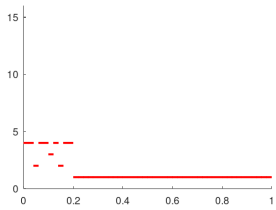
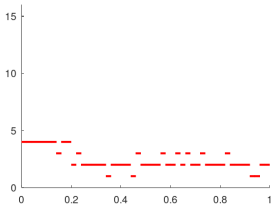
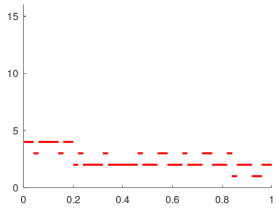
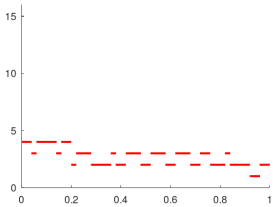
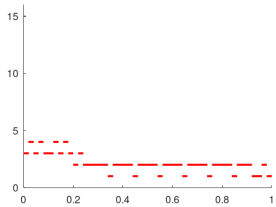
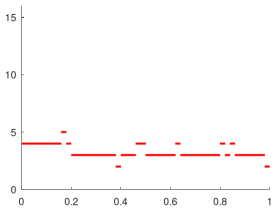
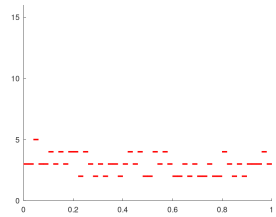
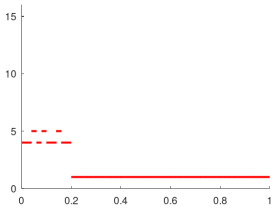
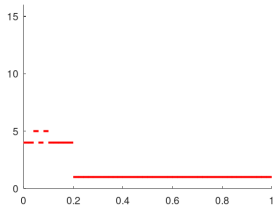
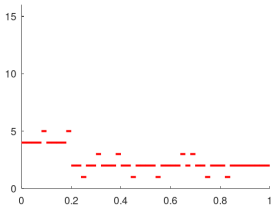
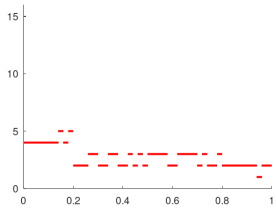
Estimator	Maximum	Dörfler
$\eta^{(4)}$ in (3.2.14)		
$\eta^{(5)}$ in (3.2.15)		
$\eta^{(6)}$ in (3.2.16)		
$\eta^{(7)}$ in (3.2.17)		
$\eta^{(8)}$ in (3.2.18)		
$\eta^{(9)}$ in (3.2.19)		

Table 4.12: Polynomial degrees obtained on each element for Helmholtz problem

For the Helmholtz problem, with both marking strategies, estimators $\eta^{(1)}$ in (3.2.8) and $\eta^{(3)}$ in (3.2.13) reach the maximal polynomial degree 16 before reaching the tolerance. The same can be said about estimator $\eta^{(2)}$ in (3.2.10) with the Maximum marking strategy. From the results in Table 4.12, we note that with estimators $\eta^{(4)}$ in (3.2.14) and $\eta^{(8)}$ in (3.2.18), the algorithm selectively increases the polynomial degree in elements

close to the region associated with the quantity of interest, therefore reaching the prescribed tolerance with fewer degrees of freedom. We next compare the final number of degrees of freedom and global errors.

Estimator	$Q(e_u)$		$N_h + 2$ (max(p))	
	Maximum	Dörfler	Maximum	Dörfler
$\eta^{(1)}$ in (3.2.8)	9.1580×10^{-7}	4.0324×10^{-8}	156 (16)	326 (16)
$\eta^{(2)}$ in (3.2.10)	3.7617×10^{-6}	2.0022×10^{-9}	150 (16)	155 (7)
$\eta^{(3)}$ in (3.2.13)	-1.8041×10^{-8}	-3.4858×10^{-8}	149 (16)	145 (16)
$\eta^{(4)}$ in (3.2.14)	-4.7150×10^{-9}	-9.8972×10^{-9}	74 (4)	76 (4)
$\eta^{(5)}$ in (3.2.15)	2.8907×10^{-9}	5.3269×10^{-9}	123 (4)	125 (4)
$\eta^{(6)}$ in (3.2.16)	4.5284×10^{-9}	1.1631×10^{-8}	133 (4)	107 (4)
$\eta^{(7)}$ in (3.2.17)	-1.5747×10^{-9}	9.4402×10^{-10}	165 (5)	149 (5)
$\eta^{(8)}$ in (3.2.18)	1.9381×10^{-9}	1.9389×10^{-9}	84 (5)	83 (5)
$\eta^{(9)}$ in (3.2.19)	2.1383×10^{-9}	6.2628×10^{-9}	122 (5)	138 (5)

Table 4.13: Global error values, number of degrees of freedom and maximal polynomial degree obtained with each estimator, using the Maximum marking strategy and Dörfler marking strategy.

From the values in Table 4.13, we see that in this case, the Maximum strategy used with estimator $\eta^{(4)}$ in (3.2.14) requires the fewest degrees of freedom to reach the prescribed tolerance. However, the global error is minimized by using the Dörfler strategy with estimator $\eta^{(7)}$ in (3.2.17) as the local refinement indicator.

Note that applying an algorithm with uniform enrichment (i.e. an algorithm that increases the polynomial degree on every element at each iteration), prescribing the same tolerance, we obtain $Q(e_u) = 2.6176 \times 10^{-9}$ with a maximum polynomial degree of 3, and $N_h = 149$ degrees of freedom. Therefore, an adaptive algorithm is only more efficient applying some of the error representations listed above.

Also note that in [12], Darrigrand et al. conducted similar tests with this problem to compare the performance of estimators $\eta^{(1)}$ in (3.2.8) and $\eta^{(5)}$ in (3.2.15). Their results suggested that estimator $\eta^{(5)}$ led to more efficient p-adaptive algorithms. As shown in Table 4.13, this observation holds true for both marking strategies. However, for this specific problem, other estimators, namely $\eta^{(4)}$ in 3.2.14 and $\eta^{(8)}$ in 3.2.18, demonstrate even better performance.

Classification of estimators

The results in this section indicate that $\eta^{(4)}$ in (3.2.14) yields the most efficient error estimation, when incorporated into a p-adaptive refinement procedure with the Maximum marking strategy.

4.3 Numerical Results in 2D

In the second set of numerical experiments, we implement h-adaptive algorithms for two-dimensional problems in FreeFEM++ [21], once again using Gaussian quadrature. The algorithm applied is loosely based on Example 5.1.9 in the software documentation, but also includes a process to refine the mesh according to a marking strategy based on the Maximum principle. The mesh is refined using the built-in function *adaptmesh* as well as patch-based refinement factors.

For element-based indicators, at each iteration,

- The primal and dual problems are solved;
- The local error indicators η_K are calculated on each element K;
- Each element is assigned a refinement factor - elements satisfying the Maximum criterion are assigned the value 2 (marked for refinement), while the other elements are assigned the value 1;
- The element-based refinement factors are projected into the $\mathbb{P}_1$ space using an L^2 projection with mass lumping, in order to have patch-based refinement factors;
- The mesh size at each patch (average edge length) is divided by the projected refinement factors in order to compute the new isotropic mesh.

For patch-based indicators, at each iteration,

- The primal and dual problems are solved;
- The local error indicators η_i are calculated on each patch ω_i;
- Each patch is assigned a refinement factor - patches satisfying the Maximum criterion are assigned the value 2 (marked for refinement), while the others are assigned the value 1;
- The mesh size at each patch (average edge length) is divided by the assigned refinement factors in order to compute the new isotropic mesh.

Remark 4.3.1. To incorporate coarsening into this algorithm, one could assign a refinement factor inferior to 1 to elements or patches that meet a coarsening criterion. The algorithm used for the numerical tests below, however, does not include coarsening.

The numerical results that follow will therefore only vary according to the chosen local error indicator. In this section, we limit our tests to estimators η^{1-7} and thus consider the Lagrangian $\mathbb{P}_2$ FE basis functions instead of the hierarchical $\mathbb{P}_2$ FE basis functions.

4.3.1 Poisson Problem with Homogeneous Dirichlet Boundary Conditions

The first two-dimensional problem considered is a specific case of the Poisson problem with homogeneous Dirichlet boundary conditions. Setting $f = 1$, the primal and dual problems become

$$\begin{cases} -\Delta u = 1 \text{ in } \Omega \\ u = 0 \text{ on } \partial\Omega, \end{cases}$$

and

$$\begin{cases} -\Delta z = \mathbb{1}_{\Omega^*} \text{ in } \Omega \\ z = 0 \text{ on } \partial\Omega. \end{cases}$$

Figure 4.7 shows the domain Ω and subdomain Ω^*, and Figure 4.8 displays the primal and dual solutions for $\Omega^* = (0.5, 0.7) \times (0.5, 0.7)$.

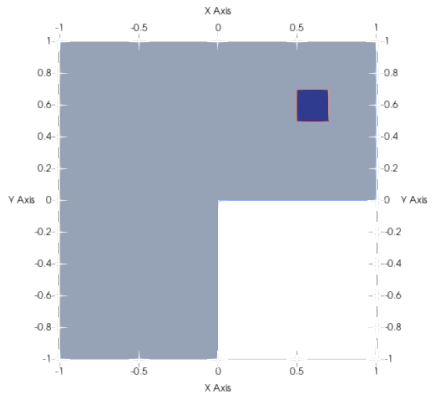


Figure 4.7: $\Omega = (-1,1) \times (-1,1) \setminus [0,1) \times (-1,0]$ and $\Omega^* = (0.5, 0.7) \times (0.5, 0.7)$

(a) Primal solution

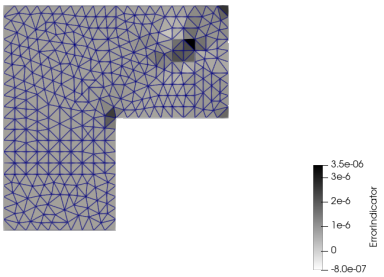
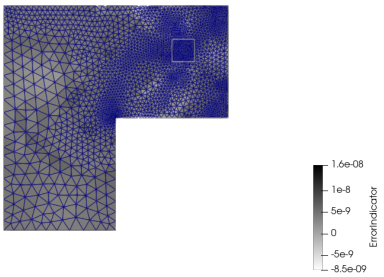
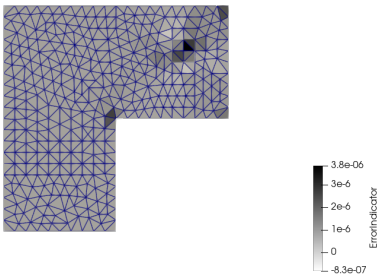
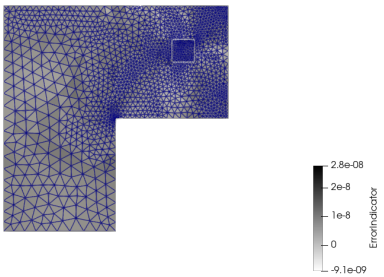
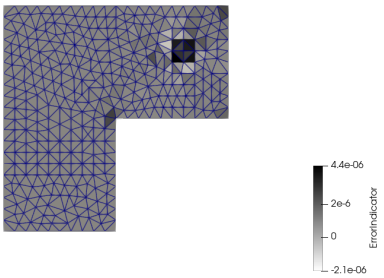
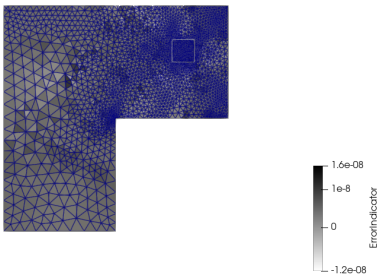
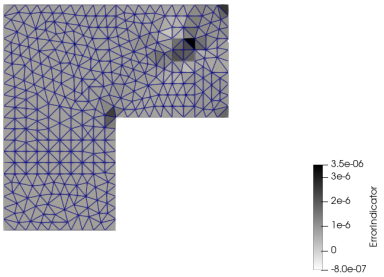
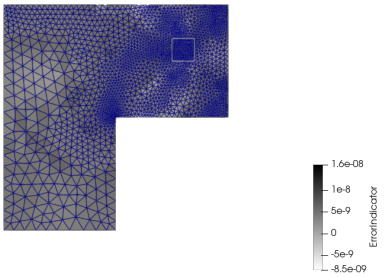
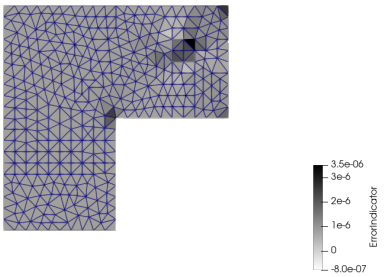
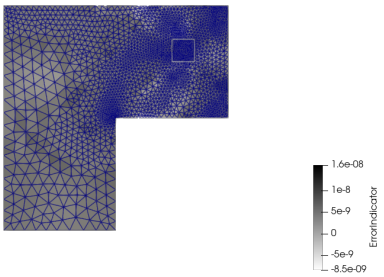
(b) Dual solution

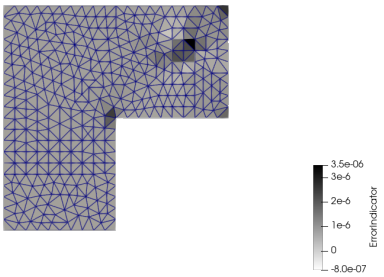
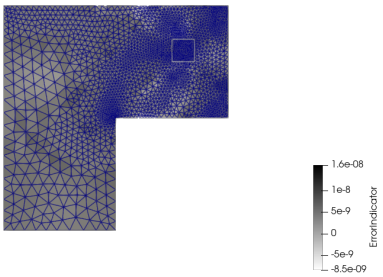
Figure 4.8: Primal and dual solutions for Poisson problem (homogeneous case) on $\Omega = (-1,1) \times (-1,1) \setminus [0,1) \times (-1,0]$, for $\Omega^* = (0.5, 0.7) \times (0.5, 0.7)$

Using h-adaptation, we run the algorithm with a set tolerance of

$$\left| \frac{\eta}{Q(\tilde{u})} \right| < \frac{5 \times 10^{-5}}{\int_{\Omega^*} 1 dx} = 0.00125$$

and a set maximum of $10,000$ degrees of freedom. We compare the distribution of the error estimators on a coarse uniform mesh with the distribution obtained by running the algorithm with the Maximum marking strategy, with $\theta = 0.2$.

Estimator	Uniform mesh	Maximum
$\eta^{(1)}$ in (3.3.1)		
$\eta^{(2)}$ in (3.3.2)		
$\eta^{(3)}$ in (3.3.3)		
$\eta^{(4)}$ in (3.3.5)		
$\eta^{(5)}$ in (3.3.7)		

Estimator	Uniform mesh	Maximum
$\eta^{(6)}$ in (3.3.8)		

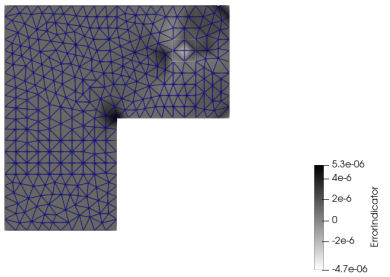
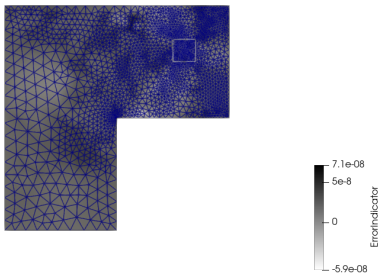
Estimator	Uniform mesh	Maximum
$\eta^{(7)}$ in (3.3.9)		

Table 4.14: Local error estimators before and after running the adaptive algorithm for $\Omega^* = (0.5, 0.7) \times (0.5, 0.7)$ (homogeneous case)

In Table 4.14, the grayscale maps in each plot show the indicator value on each element, and the edges of the mesh are colored blue. We can see that the pre-adaptation distribution of each estimator captures the error in both solutions, and post-algorithm, the error is more evenly distributed across the domain. We now compare the final number degrees of freedom, number of iterations and global errors.

Estimator	$Q(\tilde{u} - u_h)$	N_h	No. of iterations
$\eta^{(1)}$ in (3.3.1)	2.35917×10^{-6}	2,249	9
$\eta^{(2)}$ in (3.3.2)	3.54336×10^{-6}	1,596	10
$\eta^{(3)}$ in (3.3.3)	2.90004×10^{-6}	2,379	15
$\eta^{(4)}$ in (3.3.5)	2.35917×10^{-6}	2,249	9
$\eta^{(5)}$ in (3.3.7)	2.35917×10^{-6}	2,249	9
$\eta^{(6)}$ in (3.3.8)	2.35917×10^{-6}	2,249	9
$\eta^{(7)}$ in (3.3.9)	4.17523×10^{-6}	2,076	5

Table 4.15: Global error values, number of degrees of freedom and number of iterations obtained with each estimator, using the Maximum marking strategy for $\Omega^* = (0.5, 0.7) \times (0.5, 0.7)$ (homogeneous case)

From the values displayed in Table 4.15, we see estimator $\eta^{(2)}$ in (3.3.2) requires the fewest degrees of freedom to reach the prescribed tolerance. The global error is minimized with estimators $\eta^{(1)}$ in (3.3.1), $\eta^{(4)}$ in (3.3.5), $\eta^{(5)}$ in (3.3.7) and $\eta^{(6)}$ in (3.3.8) as the local refinement indicator.

Examining the last column, we observe that all estimators require a comparable number of iterations to reach the prescribed tolerance, with the exception of estimators $\eta^{(7)}$ in (3.3.9) and η^3 in (3.3.3) which require fewer and more iterations, respectively.

Note that applying an algorithm with uniform refinement, the maximal number of degrees of freedom is surpassed before the error tolerance is reached. In this case, we have a final value of $Q(\tilde{u}-u_h) = 3.92732\times10^{-6}$ and 21,441 degrees of freedom.

Remark 4.3.2. From the results in Tables 4.14 and 4.15, we see that estimators $\eta^{(1)}$ in (3.3.1), $\eta^{(4)}$ in (3.3.5), $\eta^{(5)}$ in (3.3.7) and $\eta^{(6)}$ have the same local contributions. This is because we chose $\mathcal{A}(\cdot,\cdot) = B(\cdot,\cdot)$. For a different choice of the alternative bilinear form $\mathcal{A}$, we have different local contributions for these four estimators.

4.3.2 Poisson Problem with Nonhomogeneous Dirichlet Boundary Conditions

The second two-dimensional problem considered is a specific case of the Poisson problem with nonhomogeneous Dirichlet boundary conditions. We consider the problem with the exact solution

$$u(x,y) = (x^2+y^2)^{\frac{1}{3}} \times \sin\left(\frac{2}{3}\arctan\left(\frac{y}{x}\right)\right) - \frac{1}{4}(x^2+y^2),$$

which corresponds to taking

$$f = 1$$

and

$$g(x,y) = (x^2+y^2)^{\frac{1}{3}} \times \sin\left(\frac{2}{3}\arctan\left(\frac{y}{x}\right)\right) - \frac{1}{4}(x^2+y^2).$$

The primal and dual problems become

$$\begin{cases} -\Delta u = 1 \text{ in } \Omega \\ u = g \text{ on } \partial\Omega, \end{cases}$$

and

$$\begin{cases} -\Delta z = \mathbb{1}_{\Omega^*} \text{ in } \Omega \\ z = 0 \text{ on } \partial\Omega. \end{cases}$$

Figure 4.9 shows the primal and solution the dual solution for $\Omega^* = (0.5, 0.7) \times (0.5, 0.7)$.

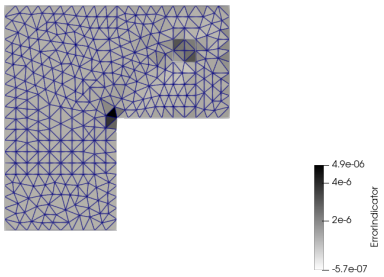
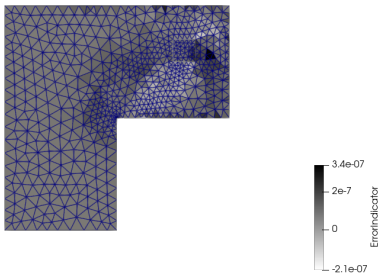
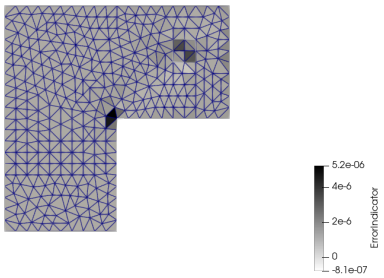
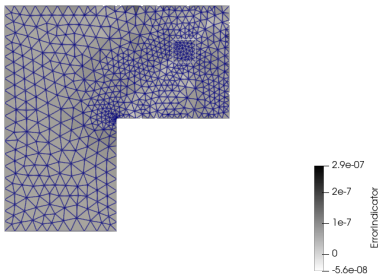
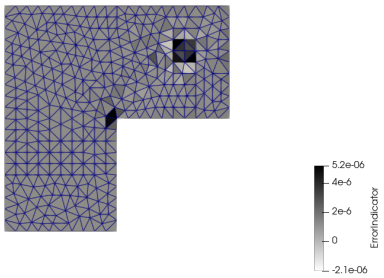
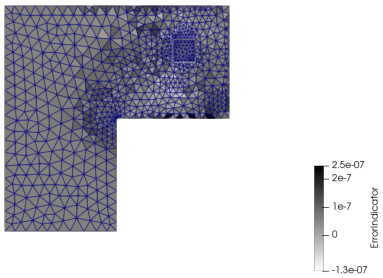
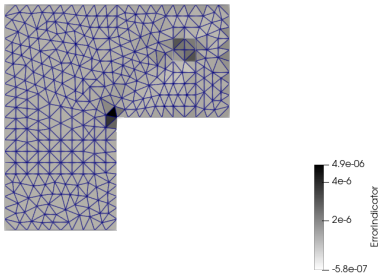
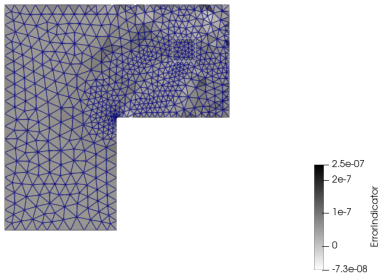
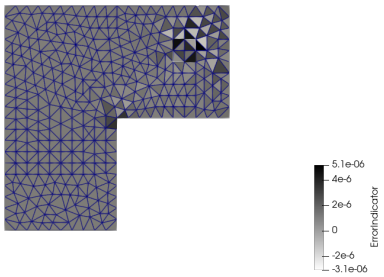
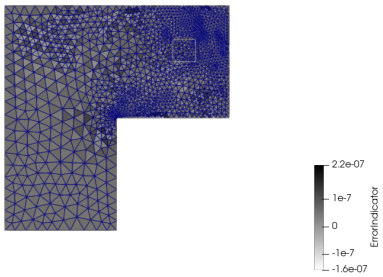
(a) Primal solution (b) Dual solution

Figure 4.9: Primal and dual solutions for Poisson problem (nonhomogeneous case) on $\Omega = (-1,1)\times(-1,1)\setminus[0,1)\times(-1,0]$, for $\Omega^* = (0.5,0.7)\times(0.5,0.7)$

Using h-adaptation, we run the algorithm with a set tolerance of

$$\left|\frac{\eta}{Q(\tilde{u})}\right| < \frac{5\times10^{-5}}{\int_{\Omega^*} 1dx} = 0.00125$$

and a set maximum of $10,000$ degrees of freedom. We compare the distribution of the error estimators on a coarse uniform mesh with the distribution obtained by running the algorithm with the Maximum marking strategy, with $\theta = 0.2$.

Estimator	Uniform mesh	Maximum
$\eta^{(1)}$ in (3.3.1)		
$\eta^{(2)}$ in (3.3.2)		
$\eta^{(3)}$ in (3.3.3)		
$\eta^{(4)}$ in (3.3.5)		
$\eta^{(5)}$ in (3.3.7)		

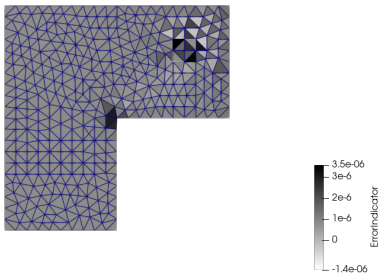
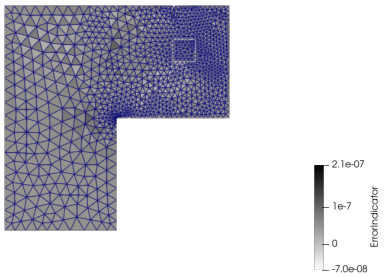
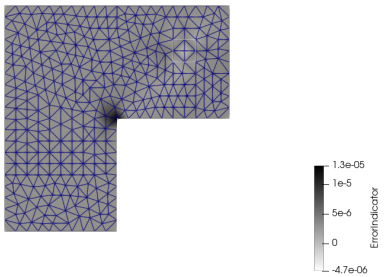
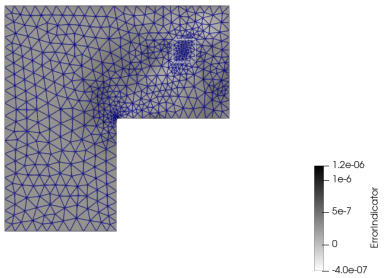
Estimator	Uniform mesh	Maximum
$\eta^{(6)}$ in (3.3.8)		
$\eta^{(7)}$ in (3.3.9)		

Table 4.16: Local error estimators before and after running the adaptive algorithm for $\Omega^* = (0.5, 0.7) \times (0.5, 0.7)$ (nonhomogeneous case)

From the plots shown in Table 4.16, we can see that the pre-adaptation distribution of each estimator once again captures the error in both solutions. However, the error is more concentrated in the primal for estimators $\eta^{(1)}$ in (3.3.29), $\eta^{(2)}$ in (3.3.31) and $\eta^{(4)}$ in (3.3.34), whereas for estimators $\eta^{(3)}$ in (3.3.33) and $\eta^{(5)}$ in (3.3.35), it is more concentrated in the dual. As in the 1D case, the pre-adaptation distribution of $\eta^{(6)}$ in (3.3.36) and $\eta^{(7)}$ in (3.3.37) is relatively evenly localized in both the primal and dual solutions. Post-algorithm, for all estimators, the error in the QoI is more evenly distributed across the domain.

Remark 4.3.3. In the one-dimensional problems treated above, the estimator $\eta^{(4)}$ was mostly localized in the dual, whereas the estimator $\eta^{(5)}$ was mostly localized in the primal. Here, we have the opposite. Therefore, the localization of these two estimators is problem-dependent.

We now compare the final global errors, number of degrees of freedom and number of iterations.

Estimator	$Q(\tilde{u} - u_h)$	N_h	No. of iterations
$\eta^{(1)}$ in (3.3.29)	1.31102×10^{-5}	675	5
$\eta^{(2)}$ in (3.3.31)	9.30895×10^{-6}	716	7
$\eta^{(3)}$ in (3.3.33)	1.32932×10^{-5}	781	5
$\eta^{(4)}$ in (3.3.34)	1.06829×10^{-5}	711	6
$\eta^{(5)}$ in (3.3.35)	1.05932×10^{-5}	1,636	5
$\eta^{(6)}$ in (3.3.36)	8.61683×10^{-6}	1,079	6
$\eta^{(7)}$ in (3.3.37)	1.71478×10^{-5}	620	3

Table 4.17: Global error values, number of degrees of freedom and number of iterations obtained with each estimator, using the Maximum marking strategy for $\Omega^* = (0.5, 0.7) \times (0.5, 0.7)$ (nonhomogeneous case)

In this case, estimator $\eta^{(7)}$ in (3.3.37) requires the fewest degrees of freedom to reach the prescribed tolerance.

The global error is minimized with estimator $\eta^{(6)}$ in (3.3.36) as the local refinement indicator.

As in the previous case, examining the last column, we observe that all estimators require a comparable number of iterations to reach the prescribed tolerance, with the exception of estimator η^7 in (3.3.37), which requires fewer iterations.

Note that applying an algorithm with uniform refinement, the maximal number of degrees of freedom is surpassed before the error tolerance is reached. In this case, we obtain a final value of $Q(\tilde{u} - u_h) = 5.55109 \times 10^{-6}$ and $21,441$ degrees of freedom.

Remark 4.3.4. In this case, as we have access to the exact solution u, we are able to compute the effectivity index for each estimator. However, as previously mentioned, since all estimators have the same global value, the effectivity index will not vary from one estimator to another when considering the same mesh. With the initial mesh, we obtain

$$\frac{\eta^{(1)}}{Q(u - u_{h,p})} = 0.924622,$$

so the estimators provide a relatively accurate approximation of the error $Q(e_u)$. This value also suggests that $Q(u - \tilde{u})$ is negligible, since $Q(u - u_{h,p}) = Q(u - \tilde{u}) + Q(\tilde{u} - u_{h,p})$ and $Q(\tilde{u} - u_{h,p})$ is equal to the global value of $\eta^{(1)}$.

Exact estimators

We now present the results obtained from repeating the numerical tests with the estimators defined in Remark 3.3.5. We obtain the following final degrees of freedom, iterations and global errors.

Estimator	$Q(\tilde{u} - u_h)$	N_h	No. of iterations
$\eta^{(1)}$ in (3.3.41)	1.21347×10^{-5}	$1,181$	5
$\eta^{(2)}$ in (3.3.42)	1.13553×10^{-5}	$1,281$	9
$\eta^{(4)}$ in (3.3.43)	1.21347×10^{-5}	$1,181$	5
$\eta^{(5)}$ in (3.3.44)	1.21347×10^{-5}	$1,181$	5
$\eta^{(6)} = \frac{1}{2}(\eta^{(4)} + \eta^{(5)})$	1.21347×10^{-5}	$1,181$	5
$\eta^{(7)}$ in (3.3.45)	9.06857×10^{-6}	$1,756$	5

Table 4.18: Global error values, number of degrees of freedom and number of iterations obtained with each exact estimator, using the Maximum marking strategy for $\Omega^* = (0.5, 0.7) \times (0.5, 0.7)$ (nonhomogeneous case)

From the final number of degrees of freedom in Table 4.18, we can see that excluding $\eta^{(5)}$ in (3.3.44), the exact representations of $Q(\tilde{u} - u_h)$ are much less efficient than the approximations listed in Table 4.17. This supports our argument in the remark which states that it is more optimal to include the local contributions of

$$\int_{\partial\Omega} (\nabla \tilde{z} \cdot n)(\tilde{u}_g - u_{h,g}) ds$$

in the error representations for the nonhomogeneous Dirichlet case.

Classification of estimators

As an attempt to classify the nine estimators, we sum up the degrees of freedom required to reach the tolerance for each problem.

Estimator	Homogeneous	Nonhomogeneous	Sum
$\eta^{(1)}$ in (3.3.1) or (3.3.29)	$2,249$	675	$2,924$
$\eta^{(2)}$ in (3.3.2) or (3.3.31)	$1,596$	716	$2,312$
$\eta^{(3)}$ in (3.3.3) or (3.3.33)	$2,379$	781	$3,160$
$\eta^{(4)}$ in (3.3.5) or (3.3.34)	$2,249$	711	$2,960$
$\eta^{(5)}$ in (3.3.7) or (3.3.35)	$2,249$	$1,636$	$3,885$
$\eta^{(6)}$ in (3.3.8) or (3.3.36)	$2,249$	$1,079$	$3,328$
$\eta^{(7)}$ in (3.3.9) or (3.3.37)	$2,076$	620	$2,696$

Table 4.19: Sum of degrees of freedom required with each estimator for both instances of Dirichlet boundary conditions

The results in table 4.19 suggest that, among the tested approaches for the Poisson problem, the mostly primal-localized estimator $\eta^{(2)}$ in (3.3.2), (3.3.31) yields the most efficient error estimation, when incorporated into a h-adaptive refinement procedure with the Maximum marking strategy. However, when the number of iterations required to reach the prescribed tolerance is also considered, alongside the final number of degrees of freedom, $\eta^{(7)}$ in (3.3.9), (3.3.37) appears more efficient overall.

Chapter 5

Conclusion

This thesis studied the efficiency and accuracy of several globally equivalent error representations for goal-oriented error estimation. It began by defining an abstract goal-oriented error estimation framework and deriving five error representations, using fundamental results such as duality techniques and Galerkin orthogonalities. The five error representations were defined in the context of a general 1D diffusion-convection-reaction problem with homogeneous Dirichlet boundary conditions, and a general 2D Poisson problem on an L-shaped domain, with both homogeneous and nonhomogeneous Dirichlet boundary conditions. Four additional error representations were also proposed by combining existing ones, incorporating partitions of unity and treating hierarchical basis functions separately. A total of nine estimators were then tested within goal-oriented adaptive algorithms across multiple problem settings.

First, an h-adaptive algorithm for one-dimensional problems was implemented in Octave considering a 1D boundary layer problem with three different quantities of interest. Each estimator was incorporated into the algorithm with two different marking strategies: the Maximum and Dörfler criteria. This process helped determine which error (primal or dual) is better captured by each error estimator. It also demonstrated how the solution in which the error estimator is localized has a better numerical approximation. Comparing the final error and degrees of freedom for each error representation and marking strategy, it was found that the algorithm performed the most efficiently when using the Dörfler marking strategy and error representations localized in the primal problem. In particular, the dual-weighted primal residual $R_{u_{h,p}}(e_z)$ ranked first in terms of efficiency.

Next, a p-adaptive algorithm for one-dimensional problems was implemented in Octave and applied to the 1D Helmoltz problem. The same nine estimators were incorporated into this algorithm with the same two marking strategies. In contrast to the h-adaptive case, the algorithm performed more efficiently with the Maximum marking strategy and estimators localized in the dual problem. In particular, the alternative bilinear form, evaluated at the Riesz representant of the primal residual and the dual error, $\mathcal{A}(\varphi^u, e_z)$, performed the most efficiently in this context.

Finally, an h-adaptive algorithm for two-dimensional problems was implemented in FreeFEM++ considering a 2D Poisson problem on an L-shaped domain, with both homogeneous and nonhomogeneous Dirichlet boundary conditions. The first seven estimators were incorporated into this algorithm with the Maximum marking strategy. In the homogeneous case, the performance of the algorithm was seemingly identical for most estimators, with the exception of the residual-based indicators. The representation that provided the most efficient algorithm was the dual-weighted primal residual $R_{u_{h,p}}(e_z)$, but the smallest post-adaptation error was achieved with the bilinear forms $B(e_u, e_z)$, $\mathcal{A}(\varphi^u, e_z)$ and $\mathcal{A}(e_u, \varphi^z)$. In the nonhomogeneous case, the dual-weighted primal residual with partitions of unity provided the most efficient adaptive algorithm, but it was the average of $\mathcal{A}(\varphi^u, e_z)$ and $\mathcal{A}(e_u, \varphi^z)$ that provided the smallest post-adaptation error.

These numerical results highlight that the optimal error representation depends heavily on the problem type, quantity of interest, marking strategy and adaptation method. Although for problems solved with h-adaptation, the dual-weighted primal residual ranked first in terms of efficiency, for the Helmholtz problem, solved with a p-adaptive algorithm, it ranked near the bottom. However, it is worth noting that among

the nine estimators, the patch-based indicators that separated the $\mathbb{P}_1$ basis functions from higher-degree basis functions consistently ranked among the most efficient in one-dimensional problems. The primal-localized patch-based indicator performed well in h-adaptive refinement, and its dual-localized counterpart was highly effective in p-adaptive refinement. Due to time constraints, these error representations were not fully implemented in the 2D problem. As future work, extending these estimators to higher dimensions could establish them as a universal choice for goal-oriented error estimation, with the potential to significantly improve the efficiency of adaptive strategies across a wide range of applications. Another potentially valuable avenue would be the development of an algorithm that automatically selects the most suitable estimator for a given problem, thereby reducing the burden of manual selection for goal-oriented error estimation.

Appendix A

Code

This appendix provides samples of code in Octave for 1D problems, and in FreeFEM++ for 2D problems.

A.1 Octave Code for 1D Problems

A.1.1 h-Adaptive Algorithm

```
addpath('utilities')
addpath('utilities/pb_solver')
addpath('utilities/error_est')
addpath('utilities/adaptation')

% adapt is a structure variable containing for each iteration n:
% adapt.mesh{n}: current partition of the domain
% adapt.Ndof{n}: number of dof, i.e. length(mesh)
% adapt.uh{n}: approximation for the primal
% adapt.ph{n}: approximation for the dual
% adapt.Err{n}: error |Q(eu)|, eu=u-uh
% adapt.Est{n}: estimation of the error Q(eu) localized on each element/
   node (-> abs(sum(adapt.Est{n})) is an estimation of |Q(eu)|)
% adapt.Err_rel{n}: relative error |Q(eu)/Q(u)| % rmk: can use |sum(Est)/Q
   (uh)| if u unknown
% adapt.Err_H10{n}: H10-norm of the error eu, i.e. ||grad(u-uh)||_L2

% remark: n=1 corresponds to the results for the initial configuration,
% i.e. the 0th iteration

% for the relative error and the error: use either the exact solution
% (known for this problem) or the finer approximation u_tilde (called here
% uh_p2)

clear adapt;

%% problem definition (cf main.m for more details...)

x0=0; x1=1; % interval

epsilon=0.001; % diffusion
```

```
b=1; % convection
c=0; % reaction

f=@(x)1+0*x; % RHS
x_pt=0.1;

u_exact=@(x)x-(exp(-1/epsilon)-exp((x-1)/epsilon))./(exp(-1/epsilon)-1);
u_exact_x=@(x)1+(1/epsilon)*exp((x-1)/epsilon)./(exp(-1/epsilon)-1); %
    derivative of exact primal solution
p_exact=@(x)( (1-exp(-x/epsilon))/(1-exp(-1/epsilon)) )*(1-exp(-(1-x_pt)/
    epsilon)) - (x>x_pt).*(1-exp(-(x-x_pt)/epsilon));

p1=1;
p2=2;

Nel=2; % number of element for the initial mesh
x=linspace(x0,x1,Nel+1); % initial mesh

Tol=1e-10; % tolerance for the stopping criterion

BC_type=['D','D'];
BC_val_u=[0,0];
BC_val_p=[0,0];

%% adaptation

% Choose between the following cases:
% cas1: B(eu,ep)
% cas2: R_{uh}(ep)
% cas3: R_{ph}(eu)
% cas4: A(phi_u,ep)
% cas5: A(eu,phi_p)
% cas6: 0.5*(A(phi_u,ep)+A(eu,phi_p))
% cas2_PU: R(ep sum_i phi_i)
% cas8: R_{uh}(sum_i ep_i varphi_i) <- patch based
% cas9: R_{ph}(sum_i eu_i varphi_i) <- patch based

method='cas1';

% Choose to use the exact solution (cas 'exact') or the "reference"
% solution (cas 'ref')
criter_ex='exact'; % 'ref'

stop=0; n=1;
while ~stop

    if n>1 % no refinement for the initialization
    switch method
            case {'cas2PU','cas8','cas9'} % refine the mesh, cases 2PU, 8,
                9 (node estimator)
                x=adapt_h_nodes(x,Est); % adapt_h_nodes_dorfler(x,Est);
            otherwise % refine the mesh, cases 1,2,3,4,5,6 (element
               estimator)
                x=adapt_h_elements(x,Est); % adapt_h_nodes_dorfler(x,Est);
```

```
    end
end

% compute the inital primal and dual solutions (for p=p1,p2)
uh_p1=solve_diff_conv_reac(x,epsilon,b,c,f,p1);
ph_p1=solve_diff_conv_reac_dirac(x,epsilon,-b,c,x_pt,p1);
uh_p2=solve_diff_conv_reac(x,epsilon,b,c,f,p2);
ph_p2=solve_diff_conv_reac_dirac(x,epsilon,-b,c,x_pt,p2);

Nel=length(x)-1;

% construct vec_p1p2, the vector of indices (in a vector of Vh_p2) of
   the basis functions that belongs to both Vh_p1 and Vh_p2
vec_p1p2=common_basis_fct(p1,p2,Nel);

uh_p1_p2=0*uh_p2;
ph_p1_p2=0*ph_p2;
uh_p1_p2(vec_p1p2)=uh_p1;
ph_p1_p2(vec_p1p2)=ph_p1;

uh_xpt=eval_uh(x,uh_p1,x_pt,p1); % evaluation of uh_p1 at x=x_pt;

[errH10,~]=compute_error_wrt_exact(x,uh_p1,p1,u_exact,u_exact_x); %
   can also specify # Gauss pts used on each subinterval...

eu=uh_p2-uh_p1_p2;
ep=ph_p2-ph_p1_p2;

% computation of Ei, depending on the chosen method

switch method
    case 'cas1' % E1: sum_i B_i(eu,ep)
        Est=estimator_version1(x,epsilon,b,c,eu,ep,p1,p2); % local on
           each element
    case 'cas2' % E2: R(ep)
        Est=estimator_version2(x,epsilon,b,c,f,uh_p1_p2,ep,p1,p2); %
           local on each element
    case 'cas2PU' % E7: R(ep) with PU
        Est=estimator_version2_pu(x,epsilon,b,c,f,uh_p1_p2,ep,p1,p2);
           % local on each node
    case 'cas3' % E3: R(eu)
        Est=estimator_version2_Rp_dirac(x,epsilon,-b,c,x_pt,ph_p1_p2,
           eu,p1,p2); % local on each element
    case 'cas4' % E4: A(phi,ep)
        [Est,~]=estimator_version3(x,epsilon,b,c,f,uh_p1_p2,ep,p1,p2);
            % local on each element
    case 'cas5' % E5: % A(eu,phi)
        [Est,~]=estimator_version4(x,epsilon,-b,c,x_pt,eu,ph_p1_p2,p1,
           p2); % local on each element
    case 'cas6'
        Est1=estimator_version3(x,epsilon,b,c,f,uh_p1_p2,ep,p1,p2);
        Est2=estimator_version4(x,epsilon,-b,c,x_pt,eu,ph_p1_p2,p1,p2)
           ;
        Est=0.5*(Est1+Est2); % local on each element
```

```
        case 'cas8'
            Est=estimate_Ruh_patch(x,epsilon,b,c,f,uh_p1_p2,ep,p1,p2,
               BC_type,BC_val_u); % local on each node
        case 'cas9' % E9: R(eu) patch-based
            Est=estimate_Rph_patch(x,epsilon,-b,c,x_pt,ph_p1_p2,eu,p1,p2);
                % local on each node
        otherwise
            disp('Unknown method.')
    end

    switch criter_ex
       case 'exact'
           E=abs(u_exact(x_pt)-uh_xpt);
           Err_rel=E/abs(u_exact(x_pt));
           crit_stop=abs(sum(Est)/u_exact(x_pt));
       case 'ref'
           u2_xpt=eval_uh(x,uh_p2,x_pt,p2); % evaluation of uh_p2 at x=
              x_pt;
           E=abs(u2_xpt-uh_xpt);
           Err_rel=E/abs(u2_xpt);
           crit_stop=abs(sum(Est)/u2_xpt);
    end

    % store the results
    adapt.mesh{n}=x;
    adapt.Ndof{n}=length(uh_p1); % length(x) ok if p1=1;
    adapt.uh{n}=uh_p1;
    adapt.ph{n}=ph_p1;
    adapt.Err{n}=E;
    adapt.Est{n}=Est;
    adapt.Err_rel{n}=Err_rel;
    adapt.Err_H10{n}=errH10;
    adapt.sum{n}=sum(Est);

    % add a stopping criterion here...
    if crit_stop<=Tol % if n>=31
        stop=1;
    end

    n=n+1;

end

% Q(u-u_h)
Q = u_exact(x_pt)-uh_xpt
% number of nodes
n=length(x)

% plot
plot_res=1;
if plot_res

    % M=length(adapt.Ndof); % M is the number of solve, i.e. M-1 is the
       number of refinement iterations
```

```
    M=length(adapt.Ndof);
    ndof=zeros(1,M); e=zeros(1,M); eq=zeros(1,M); %eH10=zeros(1,M);
    for i=1:M
        ndof(i)=adapt.Ndof{i};
        e(i)=adapt.sum{i};
        eq(i)=adapt.Err_rel{i};
        %eH10(i)=adapt.Err_H10{i};
    end


    % exact solutions

    % figure
    % plot(x, u_exact(x), 'b');

    % figure
    % plot(x, p_exact(x), 'r');


    % estimator graph

    figure

    % local on each element
    hold on;
    for i=1:length(x)-1
        plot(x(i:i+1),[Est(i) Est(i)], 'b');
    end
    for i = 2:length(x)-1
        plot([x(i) x(i)], [Est(i-1) Est(i)], 'b');
    end
    hold off;

    % local on each node
    % plot(x, Est);


    % numerical vs exact solutions

    figure
    ex_x = linspace(x0,x1,1000);
    plot(x, zeros(size(x),1), 'k*', x, uh_p1, '-^r;primal;', x, ph_p1, '-
        vb;dual;', ex_x, u_exact(ex_x), '--r;primal exact;', ex_x, p_exact(
        ex_x), '-.b;dual exact;');
    legend('location', 'southwest');
    ylim([-2 2]);
    legend('nodes', 'primal', 'dual', 'exact primal', 'exact dual');

end
```

A.1.2 p-Adaptive Algorithm

```
clear adapt;

% Choose between the following cases:
% cas1: B(eu,ep)
% cas2: R_{uh}(ep)
% cas3: R_{ph}(eu)
% cas4: A(phi_u,ep)
% cas5: A(eu,phi_p)
% cas6: 0.5*(A(phi_u,ep)+A(eu,phi_p))
% cas7: R_{uh}(ep sum_i phi_i)
% cas8: R_{uh}(sum_i ep_i varphi_i) <- patch based
% cas9: R_{ph}(sum_i eu_i varphi_i) <- patch based

method='cas1';

% Choose to use the exact solution (cas 'exact') or the "reference"
   solution (cas 'ref')
criter_ex='exact'; % 'ref';

x0=0; x1=1;

a=1; % diffusion
b=0; % convection;
k=20*2*pi; % 128 % wavenumber
c=-k^2; % reaction
dxuR=0.5; % -0.5; % Neumann BC at x=1

Nel=ceil(2.5*k/(2*pi));
x=linspace(x0,x1,Nel+1);

% BC conditions
BC_type=['D','N'];
BC_val_u=[0,dxuR];
BC_val_p=[0,0];
BC_val_riesz=[0,0];

% RHS primal and dual
f=@(x)1+0*x;
q=@(x)1*(x<=0.2)+0*x; % q=@(x)1+0*x;

p1=1;
p_incr=2; % for the ref solution, we take p1+p_incr

% exact solution (for the case u'(1)=0.5 and f=1)
u_exact=@(x)(1/(k^2))*cos(k*x)+((0.5+(1/k)*sin(k))/(k*cos(k)))*sin(k*x)
   -1/(k^2);
val=(sin(k)-sin(0.8*k))/(k*cos(k));
p_exact=@(x)((cos(k*x)-1)/k^2+val*sin(k*x)/k).*(x<0.2)+((cos(k*x)-cos(k*(x
   -0.2)))/k^2+val*sin(k*x)/k).*(x>=0.2);

Tol=1e-3; % tolerance for the stopping criterion
%max_it=25; % maximum number of enrichement
pmax=16; % maximum value we can take for p
```

```
%% adaptation
p1=p1*ones(1,length(x)-1);
stop=0; n=1;
while ~stop

    if n>1 % no refinement for the initialization
        switch method % enrichement for the case 7,8,9, i.e. 2PU and
           patches (nodal estimator)
            case {'cas7','cas8','cas9'}
                p1=adapt_p_nodes(p1,Est); %adapt_p_nodes_dorfler(p1,Est);
            otherwise % enrichement for the cases 1,2,3,4,5,6 (element
               estimator)
                p1=adapt_p_elements(p1,Est); %adapt_p_elements_dorfler(p1,
                   Est);
        end
    end

    p2=p1+p_incr;

    % compute the inital primal and dual solutions (for p=p1,p2)
    uh_p1=solve_diff_conv_reac_BC(x,a,b,c,f,p1,BC_type,BC_val_u);
    ph_p1=solve_diff_conv_reac_BC(x,a,-b,c,q,p1,BC_type,BC_val_p);
    uh_p2=solve_diff_conv_reac_BC(x,a,b,c,f,p2,BC_type,BC_val_u);
    ph_p2=solve_diff_conv_reac_BC(x,a,-b,c,q,p2,BC_type,BC_val_p);

    Nel=length(x)-1;
    vec_p1p2=common_basis_fct(p1,p2,Nel);

    uh_p1_p2=0*uh_p2;
    ph_p1_p2=0*ph_p2;
    uh_p1_p2(vec_p1p2)=uh_p1;
    ph_p1_p2(vec_p1p2)=ph_p1;

    eu=uh_p2-uh_p1_p2;
    ep=ph_p2-ph_p1_p2;

    Q=assemble_RHS(x,p1,q);
    Quh=Q'*uh_p1;
    Qu=(1/(k^3))*sin(0.2*k)+((0.5+(1/k)*sin(k))/(k^2*cos(k)))*(1-cos(0.2*k
       ))-0.2/(k^2);

    % computation of Ei, depending on the chosen method
    switch method
        case 'cas1'
            Est=estimator_version1(x,a,b,c,eu,ep,p1,p2); % local on each
               element
        case 'cas2' % residual for uh
            Est=estimate_Ruh(x,a,b,c,f,uh_p1_p2,ep,p1,p2,BC_type,BC_val_u)
               ; % local on each element
        case 'cas3' % residual for ph
            Est=estimate_Ruh(x,a,-b,c,q,ph_p1_p2,eu,p1,p2,BC_type,BC_val_p
               );
        case 'cas4'
```

```
            Est=estimate_A_Ruh(x,a,b,c,f,uh_p1_p2,ep,p1,p2,BC_type,
                BC_val_u,BC_val_riesz); % local on each element
        case 'cas5'
            Est=estimate_A_Ruh(x,a,-b,c,q,ph_p1_p2,eu,p1,p2,BC_type,
                BC_val_p,BC_val_riesz); % local on each element
        case 'cas6'
            Est1=estimate_A_Ruh(x,a,b,c,f,uh_p1_p2,ep,p1,p2,BC_type,
                BC_val_u,BC_val_riesz);
            Est2=estimate_A_Ruh(x,a,-b,c,q,ph_p1_p2,eu,p1,p2,BC_type,
                BC_val_p,BC_val_riesz); % local on each element
            Est=0.5*(Est1+Est2);
        case 'cas7'
            Est=estimate_Ruh_PU(x,a,b,c,f,uh_p1_p2,ep,p1,p2,BC_type,
                BC_val_u);
        case 'cas8'
            Est=estimate_Ruh_patch(x,a,b,c,f,uh_p1_p2,ep,p1,p2,BC_type,
                BC_val_u);
        case 'cas9'
            Est=estimate_Ruh_patch(x,a,-b,c,q,ph_p1_p2,eu,p1,p2,BC_type,
                BC_val_p);
        case 'cas10'
            Est=estimate_Ruh_PU(x,a,-b,c,q,ph_p1_p2,eu,p1,p2,BC_type,
                BC_val_p);
        otherwise
            disp('Unknown method.')
    end

    switch criter_ex
       case 'exact' % u_exact and Qu (exact value, i.e. Q(uexact)) must be
            given above
           E=abs(Qu-Quh);
           Err_rel=E/abs(Qu);
           crit_stop=abs(sum(Est)/Qu);
       case 'ref'
           Q=assemble_RHS(x,p2,q);
           Qe=Q'*eu;
           Qu=Q'*uh_p2;
           E=abs(Qe);
           Err_rel=E/abs(Qu);
           crit_stop=abs(sum(Est)/Qu);
    end

    % store the results
    adapt.p{n}=p1;
    adapt.Ndof{n}=length(uh_p1);
    adapt.uh{n}=uh_p1;
    adapt.ph{n}=ph_p1;
    adapt.Err{n}=E;
    adapt.Err_rel{n}=Err_rel;
    adapt.Est_loc{n}=Est;
    adapt.Eta{n}=abs(sum(Est));
    adapt.UB{n}=sum(abs(Est));
    adapt.Quh{n}=Quh;
    adapt.Qu{n}=Qu;
```

```
    % add a stopping criterion here...
    if crit_stop<=Tol || (max(p1)>=pmax) % n>max_it ||
        stop=1;
    end

    n=n+1;
end

Qeu = Qu-Quh % Q(u-u_h)
maxp=max(p1) % number of intervals
ddl=sum(p1)+1

% plot
plot_res=1;

if plot_res

    %M=length(adapt.Ndof); % M is the number of solve, i.e. M-1 is the
       number of refinement iterations
    %ndof=zeros(1,M); eq=zeros(1,M);
    %for i=1:M
     %   ndof(i)=adapt.Ndof{i};
      %  eq(i)=adapt.Err_rel{i};
    %end

    % numerical vs exact solutions

    ex_x = linspace(x0,x1,1000);

    plot_uh = eval_uh(x, uh_p1, ex_x, p1);
    plot_ph = eval_uh(x, ph_p1, ex_x, p1);

    %plot(x, zeros(size(x),1), 'k*', ex_x, plot_uh, '-r;primal;');
    %hold on;
    %plot(ex_x, u_exact(ex_x), '--r;primal exact;');
    %hold off;

    %figure
    %plot(x, zeros(size(x),1), 'k*', ex_x, plot_ph, '-b;dual;');
    %hold on;
    %plot(ex_x, p_exact(ex_x), '-.b;dual exact;');
    %hold off;

    % polynomial degree on each element

    for i=1:length(x)-1
        hold on;
        plot(x(i:i+1),[p1(i) p1(i)], 'r', 'LineWidth', 2);
        hold off;
    end
    xlim([0 1])
    ylim([0 16])
```

```
end
```

A.2 FreeFEM++ Code for 2D Problems

A.2.1 h-Adaptive Algorithm

```
macro MeshSizecomputation (Th, Vh, h)
{
        real[int] count(Th.nv);

        /*mesh size (lenEdge = integral(e) 1 ds)*/
        varf vmeshsizen (u, v) = intalledges(Th, qfnbpE=1)(v);

        /*number of edges per vertex*/
        varf vedgecount (u, v) = intalledges(Th, qfnbpE=1)(v/lenEdge);

        /*mesh size*/
        count = vedgecount(0, Vh);
        h[] = 0.;
        h[] = vmeshsizen(0, Vh);

        cout << "count␣min␣=␣" << count.min << "␣max␣=␣" << count.max << endl; h[] = h[]./
            count;
        cout << "--␣bound␣meshsize␣=␣" << h[].min << "␣" << h[].max << endl;
} //

macro ReMeshIndicator (Th, Ph, Vh, Sh, vindicator, deg) {

        Vh h=0;

        /*evaluate the mesh size*/
        MeshSizecomputation(Th, Vh, h);

        Ph etak;
        etak[] = vindicator(0, Ph);
        etak = abs(etak);

        real maxetak = etak[].max;
        real thetaval = 0.2;
        real maxthreshold = thetaval*maxetak;

        Ph fn;
        /*Marking based Maximum criterion*/
        for (int i = 0; i < etak[].n; i++) {
                if (etak[][i] < maxthreshold ) {
                                fn[][i] = 1;
                } else {
                        fn[][i] = 2;
                }
        }

        Vh nodeFn, sigma;

        varf veta(unused, v) = int2d(Th)(fn*v);
        varf vun(unused, v) = int2d(Th)(1*v);

        /*L2 projection with mass lumping for element-based indicator*/
        if (deg == 0) {
                nodeFn[] = veta(0, Vh);
                sigma[] = vun(0, Vh);
                nodeFn[] = nodeFn[]./ sigma[];
        } else {
                nodeFn = fn;
        }
```

```
        /*refined if fn > 1 and does not change if fn = 1 */
        h = h / nodeFn;

        /*adapt the mesh*/
        Th = adaptmesh(Th, h, IsMetric=1, splitpbedge=1, nbvx=20000);
        Sh = trunc(Th,((x>0.5) & (x<0.7) & (y>0.5) & (y< 0.7)));

} //

func f = 1;
string method = "cas1";

func real theta(real x, real y) {
    real value = fmod(atan2(y, x) + 2*pi, 2*pi); // Compute the angle modulo 2*pi
    return value;
}

func exsol = (x^2+y^2)^(1.0/3.0)*sin((2.0/3.0)*theta(x,y))-0.25*(x^2+y^2);
func exsolX = (2.0/3.0)*x*sin((2.0/3.0)*theta(x,y))/((x^2+y^2)^(2.0/3.0)) - (2.0/3.0)*y*cos
    ((2.0/3.0)*theta(x,y))/((x^2+y^2)^(2.0/3.0))-0.5*x;
func exsolY = (2.0/3.0)*y*sin((2.0/3.0)*theta(x,y))/((x^2+y^2)^(2.0/3.0)) + (2.0/3.0)*x*cos
    ((2.0/3.0)*theta(x,y))/((x^2+y^2)^(2.0/3.0))-0.5*y;
func exsolXX = (2.0/9.0)*(y^2-x^2)*sin((2.0/3.0)*theta(x,y))/((x^2+y^2)^(5.0/3.0)) + 4*x*y*
    cos((2.0/3.0)*theta(x,y))/((x^2+y^2)^(5.0/3.0))-0.5;
func exsolYY = (2.0/9.0)*(x^2-y^2)*sin((2.0/3.0)*theta(x,y))/((x^2+y^2)^(5.0/3.0)) - 4*x*y*
    cos((2.0/3.0)*theta(x,y))/((x^2+y^2)^(5.0/3.0))-0.5;

// Mesh of L-shaped domain
border ba(t=0, 1){x=-1+t; y=-1; label=1;}
border bb(t=0, 1){x=0; y=-1+t; label=2;}
border bc(t=0, 1){x=t; y=0; label=3;}
border bd(t=0, 1){x=1; y=t; label=4;}
border be(t=0, 2){x=1-t; y=1; label=5;}
border bf(t=0, 2){x=-1; y=1-t; label=6;}

// (0.5,0.7)x(0.5,0.7)
border sa(t=0, 0.2){x=0.7; y=0.5+t; label=7;}
border sb(t=0, 0.2){x=0.7-t; y=0.7; label=8;}
border sc(t=0, 0.2){x=0.5; y=0.7-t; label=9;}
border sd(t=0, 0.2){x=0.5+t; y=0.5; label=10;}

mesh Th = buildmesh(ba(10) + bb(10) + bc(10) + bd(10) + be(20) + bf(20)+ sa(2) + sb(2) + sc
    (2) + sd(2));
mesh Sh = trunc(Th,((x>0.5) & (x<0.7) & (y>0.5) & (y< 0.7))); // subdomain for QoI

// Fespace
fespace Vh(Th, P1); // solution
Vh u, v, p, u0, ug;
Vh h = hTriangle;
Vh uex = exsol;
Vh thetax = theta(x,y);

// Finer space
// load "Element_P3";
fespace Vh2(Th, P2); // solution
Vh2 ut, vt, pt, phiut, phipt, ed, u0t, ugt;
Vh2 h2 = hTriangle;
Vh2 uext = exsol;

// P0 space
fespace Nh(Th, P0);

// Primal problems
problem Poisson (u, v, solver=LU)
        = int2d(Th, qforder=5)(
                dx(u)*dx(v)
                + dy(u)*dy(v)
```

```
        )
        + int2d(Th, qforder=5)(
                - f*v
        )
        + on(1,2,3,4,5,6,u=exsol);

problem PoissonTrace (ug, v, solver=LU)
        = int2d(Th, qforder=5)(
                dx(ug)*dx(v)
                + dy(ug)*dy(v)
        )
        + on(1,2,3,4,5,6,ug=exsol);

problem hPoisson (u0, v, solver=LU)
        = int2d(Th, qforder=5)(
                dx(u0)*dx(v)
                + dy(u0)*dy(v)
        )
        + int2d(Th, qforder=5)(
                - f*v
        )
        + int2d(Th, qforder=5)(
                dx(ug)*dx(v)
                + dy(ug)*dy(v)
        )
        + on(1,2,3,4,5,6,u0=0);
        ;

// Primal problems using P2 FE
problem PoissonP2 (ut, vt, solver=LU)
        = int2d(Th, qforder=5)(
                dx(ut)*dx(vt)
                + dy(ut)*dy(vt)
        )
        + int2d(Th, qforder=5)(
                - f*vt
        )
        + on(1,2,3,4,5,6,ut=exsol);
        ;

problem PoissonTraceP2 (ugt, vt, solver=LU)
        = int2d(Th, qforder=5)(
                dx(ugt)*dx(vt)
                + dy(ugt)*dy(vt)
        )
        + on(1,2,3,4,5,6,ugt=exsol);


problem hPoissonP2 (u0t, vt, solver=LU)
        = int2d(Th, qforder=5)(
                dx(u0t)*dx(vt)
                + dy(u0t)*dy(vt)
        )
        + int2d(Th, qforder=5)(
                - f*vt
        )
        - int2d(Th, qforder=5)(
                dx(ugt)*dx(vt)
                + dy(ugt)*dy(vt)
        )
        + on(1,2,3,4,5,6,u0t=0);
        ;

// Dual problem
problem QoI (p, v, solver=LU)
        = int2d(Th, qforder=5)(
                dx(p)*dx(v)
                + dy(p)*dy(v)
```

```
        )
        + int2d(Sh, qforder=5)(
                - v
        )
        + on(1,2,3,4,5,6,p=0);
        ;

// Dual problem using P2 FE
problem QoIex (pt, vt, solver=LU)
        = int2d(Th, qforder=5)(
                dx(pt)*dx(vt)
                + dy(pt)*dy(vt)
        )
        + int2d(Sh, qforder=5)(
                - vt
        )
        + on(1,2,3,4,5,6,pt=0);
        ;

// Riesz representant for primal residual
problem primalResRep (phiut, vt, solver=LU)
        = int2d(Th, qforder=5)(
                dx(phiut)*dx(vt)
                + dy(phiut)*dy(vt)
        )
        - int2d(Th, qforder=5)(
                f*vt
                - dx(u)*dx(vt)
                - dy(u)*dy(vt)
        )
        + on(1,2,3,4,5,6, phiut=0);

// Riesz representant for dual residual
problem dualResRep (phipt, vt, solver=LU)
        = int2d(Th, qforder=5)(
                dx(vt)*dx(phipt)
                + dy(vt)*dy(phipt)
        )
        + int2d(Th, qforder=5)(
                dx(vt)*dx(p)
                + dy(vt)*dy(p)
        )
        - int2d(Sh, qforder=5)(
                vt
        )
        + on(1,2,3,4,5,6, phipt=0);

varf indicator1 (unused, chiK)
        = int2d(Th)(
                chiK*(
                        ((dx(ut)-dx(u))*(dx(pt)-dx(p)))
                        + ((dy(ut)-dy(u))*(dy(pt)-dy(p)))
                        )
        )
        - int1d(Th,1,2,3,4,5,6)(
                chiK*(N.x*dx(pt) + N.y*dy(pt))*(ugt-ug)
        );

varf indicator2 (unused, chiK)
        = intalledges(Th)(
                (1.0/nTonEdge)*chiK*jump(N.x*dx(u) + N.y*dy(u))*(pt-p)
        )
        + int2d(Th)(
                chiK*(f + dxx(u) + dyy(u))*(pt-p)
        )
        - int1d(Th,1,2,3,4,5,6)(
                chiK*(N.x*dx(pt) + N.y*dy(pt))*(ugt-ug)
        );
```

```
varf indicator3 (unused, chiK)
        = intalledges(Th)(
                (1.0/nTonEdge)*chiK*jump(N.x*dx(p)+N.y*dy(p))*(ut-u)
        )
        + int2d(Th)(
                chiK*(dxx(p) + dyy(p))*(ut-u)
        )
        + int2d(Sh)(
                chiK*(ut-u)
        );

varf indicator4 (unused, chiK)
        = int2d(Th)(
                chiK*(dx(phiut)*(dx(pt)-dx(p))+dy(phiut)*(dy(pt)-dy(p)))
        )
    - int1d(Th,1,2,3,4,5,6)(
                chiK*(N.x*dx(pt) + N.y*dy(pt))*(ugt-ug)
        );

varf indicator5 (unused, chiK)
        = int2d(Th)(
                chiK*((dx(u0t)-dx(u0))*dx(phipt)+(dy(u0t)-dy(u0))*dy(phipt))
        )
        + int2d(Th)(
                chiK*(
                        ((dx(ugt)-dx(ug))*dx(pt))
                        + ((dy(ugt)-dy(ug))*dy(pt))
                        )
        )
        - int1d(Th,1,2,3,4,5,6)(
                chiK*(N.x*dx(pt) + N.y*dy(pt))*(ugt-ug)
        );

varf indicator6 (unused, chiK)
        = int2d(Th)(
                0.5*chiK*(dx(phiut)*(dx(pt)-dx(p))+dy(phiut)*(dy(pt)-dy(p)))
        )
        + int2d(Th)(
                0.5*chiK*((dx(u0t)-dx(u0))*dx(phipt)+(dy(u0t)-dy(u0))*dy(phipt))
        )
        + int2d(Th)(
                0.5*chiK*(
                        ((dx(ugt)-dx(ug))*dx(pt))
                        + ((dy(ugt)-dy(ug))*dy(pt))
                        )
        )
        - int1d(Th,1,2,3,4,5,6)(
                chiK*(N.x*dx(pt) + N.y*dy(pt))*(ugt-ug)
        );

varf indicator7 (unused, chiK)
        = intalledges(Th)(
                chiK*jump(N.x*dx(u) + N.y*dy(u))*(pt-p)
        )
        + int2d(Th)(
                chiK*(f + dxx(u) + dyy(u))*(pt-p)
        )
        - int1d(Th,1,2,3,4,5,6)(
                chiK*(N.x*dx(pt) + N.y*dy(pt))*(ut-u)
        );

// Mesh adaptation loop
real stopcrit = 1;
int iteration = 0;
real areaSubdomain = int2d(Sh)(1.0);
real tolerance = (5e-5)/areaSubdomain;
// cout << "Tolerance: " << tolerance << endl;
```

```
real[int] errorVector;

while (stopcrit > tolerance && iteration < 50 && Vh.ndof < 10000) {

        // Solve
        Poisson;
        PoissonP2;
        PoissonTrace;
        PoissonTraceP2;
        hPoisson;
        hPoissonP2;

        // plot(Th, wait=true, value=true, cmm="adapted mesh");
        // plot(Sh, wait=true, cmm="adapted subdomain");
        // plot(Th, u, wait=f, value=true, fill=1, dim=3, cmm="Primal solution");

        // Solve
        QoI;
        QoIex;

        // Riesz Representants
        primalResRep;
        dualResRep;

        // Error indicator vectors
        Nh rho;
        Vh rhoPatch;
        real Qeu = int2d(Sh)(exsol-u);
        real Qeuapprox = int2d(Sh)(ut-u);

        load "iovtk"
        string filenamePre = "indicator_" + method + "pre_adaptation.vtu";
        string comment = method + "␣(pre-adaptation)";

        if (method == "cas1") {
                rho[] = indicator1(0, Nh);
                if (iteration == 0) {
                        plot(rho, wait=f, value=true, fill=1, cmm=comment);
                        int[int] Order = [0];
                        savevtk(filenamePre, Th, rho, dataname = "ErrorIndicator", order=
                            Order);
                }
                ReMeshIndicator(Th, Nh, Vh, Sh, indicator1, 0);
        } else if (method == "cas2") {
                rho[] = indicator2(0, Nh);
                if (iteration == 0) {
                        plot(rho, wait=f, value=true, fill=1, cmm=comment);
                        int[int] Order = [0];
                        savevtk(filenamePre, Th, rho, dataname = "ErrorIndicator", order=
                            Order);
                }
                ReMeshIndicator(Th, Nh, Vh, Sh, indicator2, 0);
        } else if (method == "cas3") {
                rho[] = indicator3(0, Nh);
                if (iteration == 0) {
                        plot(rho, wait=f, value=true, fill=1, cmm=comment);
                        int[int] Order = [0];
                        savevtk(filenamePre, Th, rho, dataname = "ErrorIndicator", order=
                            Order);
                }
                ReMeshIndicator(Th, Nh, Vh, Sh, indicator3, 0);
        } else if (method == "cas4") {
                rho[] = indicator4(0, Nh);
                if (iteration == 0) {
                        plot(rho, wait=f, value=true, fill=1, cmm=comment);
                        int[int] Order = [0];
                        savevtk(filenamePre, Th, rho, dataname = "ErrorIndicator", order=
```

```
                    Order);
        }
        ReMeshIndicator(Th, Nh, Vh, Sh, indicator4, 0);
} else if (method == "cas5") {
        rho[] = indicator5(0, Nh);
        if (iteration == 0) {
                plot(rho, wait=f, value=true, fill=1, cmm=comment);
                int[int] Order = [0];
                savevtk(filenamePre, Th, rho, dataname = "ErrorIndicator", order=
                    Order);
        }
        ReMeshIndicator(Th, Nh, Vh, Sh, indicator5, 0);
} else if (method == "cas6") {
        rho[] = indicator6(0, Nh);
        if (iteration == 0) {
                plot(rho, wait=f, value=true, fill=1, cmm=comment);
                int[int] Order = [0];
                savevtk(filenamePre, Th, rho, dataname = "ErrorIndicator", order=
                    Order);
        }
        ReMeshIndicator(Th, Nh, Vh, Sh, indicator6, 0);
}
else if (method == "cas7") {
        rhoPatch[] = indicator7(0, Vh);
        if (iteration == 0) {
                plot(rhoPatch, wait=f, value=true, fill=1, cmm=comment);
                int[int] Order = [1];
                savevtk(filenamePre, Th, rhoPatch, dataname = "ErrorIndicator",
                    order=Order);
        }
        ReMeshIndicator(Th, Vh, Vh, Sh, indicator7, 1);
}

real indicatorsum = 0;
real absindicatorsum = 0;

if (method == "cas7" || method == "cas8" || method == "cas9") {
                indicatorsum = rhoPatch[].sum;
                for (int i = 0; i < rhoPatch[].n; i++) {
                        absindicatorsum = abs(indicatorsum);
                }
} else {
    indicatorsum = rho[].sum;
    for (int i = 0; i < rho[].n; i++) {
        absindicatorsum = abs(indicatorsum);
    }
}

cout << "Iteration␣" << iteration << ":␣" << endl;
cout << "Number␣of␣DOFs␣=␣" << Vh.ndof << endl;
cout << "Number␣of␣DOFs␣(P2)␣=␣" << Vh2.ndof << endl;
cout << "indicator␣sum␣=␣" << indicatorsum << endl;
cout << "Qeuapprox␣=␣" << Qeuapprox << endl;

errorVector.resize(errorVector.n + 1);  // Resize the vector to accommodate new
    error
errorVector[iteration] = absindicatorsum;

real Qu = int2d(Sh)(exsol);
// cout << "Qu = " << Qu << endl;

real subintegral = int2d(Sh)(1);
real wholedomain = int2d(Th)(1);

iteration++;
stopcrit = abs(indicatorsum/Qu);
cout << "stopcrit␣=␣" << stopcrit << endl;
```

```
        string filename = "indicator_" + method + ".vtu";
        string filenameSol = "indicator_" + method + "_solution.vtu";

        if (stopcrit < tolerance || iteration == 100 || Vh.ndof >= 10000) {
                if (method == "cas7") {
                        plot(rhoPatch, wait=f, value=true, fill=1, cmm=method);
                        int[int] Order = [1];
                        savevtk(filename, Th, rhoPatch, dataname = "ErrorIndicator", order=
                            Order);
                } else {
                        plot(rho, wait=f, value=true, fill=1, cmm=method);
                        int[int] Order = [0];
                        int[int] Order2 = [1];
                        savevtk(filename, Th, rho, dataname = "ErrorIndicator", order=Order)
                            ;
                }

        }

}


plot(Th, uex, wait=f, value=true, fill=1, dim=3, cmm="Exact␣primal␣solution");
plot(Th, pt, wait=f, value=true, fill=1, dim=3, cmm="Dual␣solution");
plot(Th, wait=f, cmm="Mesh␣(post-adaptation)");

ofstream plotFile("error_plot.dat");
for (int i = 0; i < errorVector.n; i++) {
    plotFile << i << "␣" << errorVector[i] << endl;  // Write iteration and error to file
}
```

Bibliography

[1] S. Adjerid, M. Aiffa, and J.E. Flaherty. Hierarchical finite element bases for triangular and tetrahedral elements. *Computer Methods in Applied Mechanics and Engineering*, 190(22):2925–2941, 2001.

[2] M. Ainsworth and J.T. Oden. A unified approach to a posteriori error estimation using element residual methods. *Numerische Mathematik*, 65:23–50, 1993.

[3] M. Ainsworth and J.T. Oden. *A Posteriori Error Estimation in Finite Element Analysis*. John Wiley, 2000.

[4] I. Babuska and T. Strouboulis. *The Finite Element Method and Its Reliability*. Clarendon Press, 2001.

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

[6] W. Bangerth and R. Rannacher. *Adaptive Finite Element Methods for Differential Equations*. Birkhäuser Basel, 2003.

[7] R.E. Bank and A. Weiser. Some a posteriori error estimators for elliptic partial differential equations. *Mathematics of Computation*, 44(170):283–301, 1985.

[8] R.E. Bank and J. Xu. Asymptotically exact a posteriori error estimators. I. Grids with superconvergence. *SIAM Journal on Numerical Analysis*, 41(6):2294–2312, 2004.

[9] R. Becker and R. Rannacher. A feed-back approach to error control in finite element methods: basic analysis and examples. *Journal of Numerical Mathematics*, 4:237–264, 1996.

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

[11] L. Chamoin and F Legoll. An introductory review on a posteriori error estimation in finite element computations. *SIAM review*, 65(4):963–1028, 2023.

[12] V. Darrigrand, D. Pardo, and I. Muga. Goal-oriented adaptivity using unconventional error representations for the 1D Helmholtz equation. *Computers & Mathematics with Applications*, 69(9):964–979, 2015.

[13] V. Darrigrand, A. Rodriguez-Rozas, I. Muga, D. Pardo, A. Romkes, and S. Prudhomme. Goal-oriented adaptivity using unconventional errorrepresentations for the multidimensional Helmholtz equation. *International Journal for Numerical Methods in Engineering*, 113(1):22–42, 2018.

[14] V. Darrigrand, A. Rodriguez-Rozas, D. Pardo, and I. Muga. Goal-oriented p-adaptivity using unconventional error representations for a 1D steady state convection-diffusion problem. *Procedia Computer Science*, 108(1):848–856, 2017.

[15] L. Dedè, S. Micheletti, and S. Perotto. Anisotropic error control for environmental applications. *Applied Numerical Mathematics*, page 1320–39, 2008.

[16] L. Demkowicz, J.T. Oden, and T. Strouboulis. Adaptive finite elements for flow problems with moving boundaries. Part 1: Variational principles and a posteriori error estimates. *Computer Methods in Applied Mechanics and Engineering*, 46(2):217–251, 1984.

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

[18] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. Introduction to adaptive methods for differential equations. *Acta Numerica*, 4:105–158, 1995.

[19] A. Ern and J.-L. Guermond. *Finite Elements I.* Springer Cham, 2021.

[20] A. Ern and J.-L. Guermond. *Finite Elements II.* Springer Cham, 2021.

[21] F. Hecht. New development in FreeFem++. *J. Numer. Math.*, 20(3-4):251–265, 2012.

[22] D. Hong, J Wang, and R. Gardner. Chapter 5 - Vector spaces, Hilbert spaces, and the L2 space. In *Real Analysis with an Introduction to Wavelets and Applications*, chapter 5, pages 115–153. Academic Press, 2005.

[23] D.W. Kelly, J.P. Gago, O.C. Zienkiewicz, and I. Babuška. A posteriori error analysis and adaptive processes in the finite element method: Part I — error analysis. *International Journal for Numerical Methods in Engineering*, 19(11):1593–1619, 1983.

[24] C. Liu and G. Hu. An MP-DWR method for h-adaptive finite element methods. *Numerical Algorithms*, 94:1309–1329, 2023.

[25] J. T. Oden and S. Prudhomme. Goal-oriented error estimation and adaptivity for the finite element method. *Computers & Mathematics with Applications*, 41(5):735–756, 2001.

[26] J. Ovall. Asymptotically exact functional error estimators based on superconvergent gradient recovery. *Numerische Mathematik*, 102(3):543–558, 2006.

[27] S. Prudhomme and J.T. Oden. On goal-oriented error estimation for elliptic problems: Application to the control of pointwise errors. *Computer Methods in Applied Mechanics and Engineering*, 176(1):313–331, 1999.

[28] A. Quarteroni. *Numerical Models for Differential Problems.* Springer Milano, 2014.

[29] A. Quarteroni and Valli A. *Numerical Approximation of Partial Differential Equations.* Springer, 1994.

[30] T. Strouboulis and K. Haque. Recent experiences with error estimation and adaptivity, Part I: Review of error estimators for scalar elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 97(3):399–436, 1992.

[31] Z. Zhang and J.Z. Zhu. Analysis of the superconvergent patch recovery technique and a posteriori error estimator in the finite element method (I). *Computer Methods in Applied Mechanics and Engineering*, 123(1):173–187, 1995.

[32] O.C. Zienkiewicz and J.Z. Zhu. A simple error estimator and adaptive procedure for practical engineering analysis. *International Journal for Numerical Methods in Engineering*, 24(2):337–357, 1987.

[33] O.C. Zienkiewicz and J.Z. Zhu. The superconvergent patch recovery and a posteriori error estimates. Part 1: The recovery technique. *International Journal for Numerical Methods in Engineering*, 33(7):1331–1364, 1992.

[34] O.C. Zienkiewicz and J.Z. Zhu. The superconvergent patch recovery and a posteriori error estimates. Part 2: Error estimates and adaptivity. *International Journal for Numerical Methods in Engineering*, 33(7):1365–1382, 1992.