Imperial College London
Department of Earth Science & Engineering

Mesh Adaptation and Adjoint Methods for Finite Element Coastal Ocean Modelling

Joseph Gregory Wallwork

Supervised by
Prof Matthew D. Piggott
Dr David A. Ham
Dr Hilary Weller

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computational Geoscience

April 2021

Statement of Originality

I hereby declare that the contents of this thesis comprise of my own work. References are provided wherever external resources have been used.

Joseph G. Wallwork,

23rd April 2021.

Copyright Declaration

Abstract

Adjoint methods have become extremely useful tools for computational geoscience. In this thesis, the two main implementations – the continuous and discrete adjoint methods – are compared within the same code framework, *Firedrake*, in terms of accuracy, computational cost and usability. The discrete adjoint method is deemed best suited for use within gradient-based optimisation routines and is used to invert timeseries data for a tsunami source, with different automatic differentiation tools applied to the source and propagation models.

The second focus is on mesh adaptation techniques suitable for representing time-dependent, multi-scale coastal ocean dynamics. Historically, h- and r-adaptation communities have developed methods independently. However, the metric-based framework is independent of how the discretisation is modified, meaning that it allows for h-, r- and hybrid adaptation. It has the advantage of allowing control of anisotropy, which is demonstrated to be useful for advection-dominated problems. A monitor-based approach – specific to r-adaptation and built upon optimal transport theory – is also considered and is shown to be capable of reducing point-wise error in a sediment transport test case, at comparable overall cost to uniform meshing.

Much time and effort has been spent on mesh adaptation research in the past four decades. However, it is arguably under-utilised in practical applications. This is largely because making effective choices of error estimator, optimal mesh model and adaptation method generally requires both experience and an in-depth understanding of the problem at hand. The two strands of research, on adaptation and adjoint methods, come together most clearly in *goal-oriented mesh adaptation*, which goes some way to reduce user experience requirements. Meshes are constructed based on accurately approximating a diagnostic quantity of interest, with the adjoint method used to evaluate associated error estimators. The culmination of this thesis applies goal-oriented mesh adaptation to four geoscience problems of increasing complexity.

Acknowledgements

Firstly, I would like to express my gratitude to my supervisors for their help and support during my time as a PhD student. My main supervisor, Matthew Piggott, has done an excellent job at guiding me towards interesting research topics, advocating for my work, and always finding time for discussions. I look forward to continuing to collaborate with him in the future. Beginning with an Introduction to Python Programming workshop for the Mathematics of Planet Earth Centre for Doctoral Training (MPE CDT) in 2016, David Ham has continuously provided assistance and recommendations on coding best practices, Firedrake usage and the finite element method. During my PhD, I enjoyed many trips to the University of Reading to meet with Hilary Weller and her research group and would like to thank her for the opportunity to do so, as well as for always showing interest in my work. Whilst not officially one of my supervisors, Nicolas Barral has been a constant source of valuable advice, ideas and technical assistance, for which I am very grateful.

I would also like to say a big thank you to current and former members of both Imperial College's Applied Modelling and Computation Group (AMCG) and MPE CDT for the many interesting discussions, coffee breaks and 'MPE Wednesdays'. In particular, I would like to thank Lucas Mackie, Golo Wimmer, Mariana Clare and Navjot Kukreja.

Finally, I would like to thank my family and friends for their continued support during my undertaking of this degree and for putting up with me over the last few months of writing up. Special thanks to my wife (Alex), parents (Cathy and Richard) and sister (Olivia).

Funding

MPE CDT is funded by the Engineering and Physical Sciences Research Council (EPSRC) [grant number EP/L016613/1]. Many thanks to EPSRC for their generous support and to MPE CDT for paying for me to attend lots of conferences.

Computer Resources

The numerical experiments conducted in this thesis made use of a number of computer resources at Imperial. These include workstations provided by AMCG and MPE CDT, as well as the 'Lovelace' machine run by the HPC service. Thanks to Andy Thomas for providing access to Lovelace and for technical assistance in its usage. Many thanks to Gerard Gorman for providing access to a Microsoft Azure cluster, which was used for development work which went into this thesis.

Contents

List of Tables

List of Figures

Notation

Throughout this thesis, $\Omega \subset \mathbb{R}^n$ stands for a real 'spatial' domain of dimension $n \in \{2, 3\}$. The boundary of Ω is denoted $\partial\Omega$ and is assumed piecewise smooth. Differentials on the interior and boundary of Ω are denoted $\mathrm{d}x$ and $\mathrm{d}s$, respectively. Differentials on immersed surfaces are denoted $\mathrm{d}S$. For time-dependent problems, a 'temporal' domain $(0, T]$ is used, where $T > 0$ is the end time. Differentials in time are denoted $\mathrm{d}t$.

For any subset of the spatial domain, the notation

$$\text{(N1)} \qquad \langle u, v \rangle_A := \int_A u \cdot v \, \mathrm{d}x, \qquad \|u\|_A := \langle u, u \rangle_A^{\,}, \qquad A \subseteq \Omega,$$

is used for the usual L^2 inner product and corresponding norm. In the case $A = \Omega$, the subscript A is dropped, unless instructive. The same notation is used to denote inner products and norms on boundaries of the domain and subsets thereof:

$$\text{(N2)} \qquad \langle u, v \rangle_B := \int_B u \cdot v \, \mathrm{d}s, \quad B \subseteq \partial\Omega, \qquad \langle u, v \rangle_{\partial C} := \int_{\partial C} u \cdot v \, \mathrm{d}S, \quad C \subset \Omega.$$

The notation $\| \cdot \|_2$ is reserved for the usual ℓ^2 norm acting on vectors and vector-valued functions.

In definitions (N1) and (N2), the functions u and v could be scalar, vector or tensor. In general, however, scalars α and scalar-valued functions $u : \Omega \times (0, T] \to \mathbb{R}$ are written using unemphasised lower case, while vectors $\mathbf{v}$ and vector-valued functions $\boldsymbol{\phi} : \Omega \times (0, T] \to \mathbb{R}^n$ are written using boldface lower case. In particular, the time coordinate is denoted t and Cartesian spatial coordinates are denoted $\mathbf{x} = (x, y)$ in 2D and $\mathbf{x} = (x, y, z)$ in 3D. The corresponding canonical unit vectors are denoted $\widehat{\mathbf{x}}$, $\widehat{\mathbf{y}}$ and $\widehat{\mathbf{z}}$. For some geophysical applications, longitude-latitude coordinates $\boldsymbol{\lambda} = (\lambda, \phi)$ are more appropriate. Matrices $\underline{\boldsymbol{\Sigma}}$ and matrix-valued functions $\underline{\boldsymbol{M}} : \Omega \times (0, T] \to \mathbb{R}^{n \times n}$ are written using underlined boldface upper case. Sets are written in unemphasised upper case.

If $\underline{\boldsymbol{S}}$ and $\underline{\boldsymbol{T}}$ are two $n \times n$ matrix (functions), then we denote their Frobenius inner product by $\underline{\boldsymbol{S}} : \underline{\boldsymbol{T}} := \sum_{i=1}^n \sum_{j=1}^n S_{ij} T_{ij}$.

We write $u = u(\mathbf{x}, t)$ to mean that u is a function of both space and time and $u = u(\mathbf{x})$ or $u = u(t)$ to imply that u is only a function of space or time, respectively. Similarly for vector and matrix-valued functions.

The indicator function associated with a subset $A \subseteq \Omega \times (0,T]$ is given by

$$\text{(N3)} \qquad \mathbb{1}_A : \Omega \times (0,T] \to \{0,1\}, \qquad \mathbb{1}_A(\mathbf{x},t) = \begin{cases} 1 & (\mathbf{x},t) \in A \\ 0 & (\mathbf{x},t) \notin A \end{cases}$$

All of the above notation are also used in the steady case, where the time dependency is dropped and subsets take the form $A \subseteq \Omega$.

Finally, the ball of radius $\epsilon > 0$ centred at $\mathbf{y} \in \Omega$ is denoted

$$\text{(N4)} \qquad B_\epsilon(\mathbf{y}) = \{\mathbf{x} \in \Omega \mid \|\mathbf{x} - \mathbf{y}\|_2 \leq \epsilon\}.$$

CHAPTER 1

Preface

1.1. Introduction and Objectives

This PhD thesis is focused on two advanced techniques for coastal ocean modelling using the finite element method: *mesh adaptation* and *adjoint methods*. Subsections 1.1.1 and 1.1.2 give brief overviews of each, setting out objectives to be achieved in these subject areas. Subsection 1.1.3 follows by outlining an area of research where the two techniques come together: *goal-oriented mesh adaptation*. The main body of work of this thesis is in the development of effective, low-cost goal-oriented mesh adaptation strategies and the error estimators required for the construction thereof. Objectives for this topic are laid out in Subsection 1.1.3.

All code development and numerical experiments in this thesis use the finite element library *Firedrake* [**Rathgeber et al., 2016**].

1.1.1. Mesh Adaptation. In the finite element method (FEM), the discretisation of a partial differential equation (PDE) is encapsulated by a discrete representation of the domain (i.e. the mesh), as well as finite dimensional function spaces defined upon it (i.e. finite element spaces). *Adaptation* refers to the act of modifying the initial discretisation in order to obtain one which is more desirable, in some sense. For example, the adapted discretisation may be better suited to reducing some error measure, such as the error attributed when projecting a smooth field into the finite element space. Adaptation is a nonlinear optimisation process in which discretisation and solution evolve together. In order to achieve the aim of arriving at an improved discretisation, three ingredients are required: (a) an error estimate or heuristic (i.e. 'analysis'); (b) a strategy for determining *where* the discretisation should be modified, based on the analysis step (i.e. 'assessment'); and (c) a strategy for modifying the discretisation (i.e. 'adaptation').

Analysis Step. Error analysis for adaptive FEM began in the late 1970s with the pioneering work of [**Babuška and Rheinboldt, 1978a**]. Many a posteriori error estimation frameworks have since been developed (see [**Ainsworth and Oden, 2011**] for a comprehensive review), as well as a number of a priori error estimates (see [**Micheletti et al., 2003**, **Loseille et al., 2010**], for example). In both contexts, *goal-oriented* (also known as *output-based*) error estimators have been developed which focus on the error incurred in evaluating a diagnostic quantity of interest. Such error estimators are introduced more thoroughly in Subsection 1.1.3.

For many error estimators, it is possible to arrive at a formulation which is attractive from a heuristic viewpoint. That is, error estimators involving gradients (see [**Zienkiewicz and Zhu, 1992b**, **Micheletti et al., 2010**], for example) or Hessians (see [**Frey and Alauzet, 2005**, **Loseille and Alauzet, 2011a**], for example) often have the property

that resulting meshes focus resolution so that flow features, such as shockwaves or advected quantities are better resolved.

Evaluating derivative fields for the purposes of error estimation typically involves a recovery step, such as applying an L^2 projection, or the method of [**Zienkiewicz and Zhu, 1992b**]. It is also possible to specify cheap, simple heuristics, such as flux jumps or boundary forcings, as well as velocity fields which act to move the mesh directly.

Assessment Step. In finite element modelling, it is natural to decompose a (global) error estimator into a sum of contributions over mesh elements or patches of elements. Doing so conveys a sense of the regions of the domain where the contribution to approximation error is higher, as well as those which contribute little.

A common strategy for assessing where the mesh should be modified uses these decomposed 'error indicators' to *tag* elements or patches thereof for coarsening or refinement, whenever a pre-specified tolerance is met. What is meant by 'coarsening' or 'refinement' is to be determined by the adaptation method in the following step. However, the former should be interpreted as the claim that the tagged element or element patch has more degrees of freedom than is required to achieve a sufficiently accurate approximation. Similarly, the latter suggests that accuracy would be improved if there were more degrees of freedom in the element or element patch. The element tagging approach fits most naturally with hierarchical methods, such as quad-tree/oct-tree [**Yerry and Shephard, 1983**].

Tagging methods make a direct assessment of the mesh suitability, element by element. There also exist methods which frame error estimators and heuristics using a continuous field defined upon the mesh. One such concept is the *monitor function*, which was introduced by [**White, 1979**]. The monitor function can be interpreted as a density field which prescribes how mesh elements should be distributed over the domain. The concept of *equidistribution* dictates that the integral of the monitor function should be equal in every mesh element, meaning that elements are prescribed to be small where the monitor function is large and vice versa. Mesh adaptation based on monitor functions is considered in detail in Chapter 6.

Another continuous formulation is the *metric-based* framework introduced by [**George et al., 1991**] (and described in detail in Chapter 5). Instead of using a refinement criterion based on a scalar field, this approach uses a Riemannian metric space, defined by a symmetric positive-definite matrix at each point in the domain. The metric provides a continuous analogue for the (discrete) mesh [**Loseille and Alauzet, 2011a**]. The concept of an 'optimal mesh' is one whose elements are all isotropic when viewed in the Riemannian metric space. A crucial advantage of the metric-based formulation is that it allows for control of the shape and orientation of elements, as well as size. That is to say that it can be used to generate anisotropic meshes, which can be useful for advection-dominated geoscience applications [**Piggott et al., 2009**].

Adaptation Step. The three most commonly deployed adaptation techniques are *h-adaptation*, *r-adaptation* and *p-adaptation*. The first two methods act to modify the mesh, whereas the third modifies the degree used in a polynomial function space.

h- and *r*-adaptation methods are distinguished by the fact that the former allows deformations of the mesh topology, whereas the latter does not. That is, *r*-adaptation allows

for the redistribution of mesh entities and the distortion of their shapes, but does not allow for degrees of freedom to be inserted or removed or to change connectivity.

h-adaptation – sometimes referred to using the term *adaptive mesh refinement* – is the most commonly deployed adaptation strategy, with a wide variety of implementations, covering both structured and unstructured meshes and both finite volume and finite element methods. It is especially useful for transient problems which require more degrees of freedom at some time levels than others, since mesh entities can be inserted and removed as the simulation progresses.

Like h-adaptation, r-adaptation – sometimes called *mesh movement* – is actually a family of mesh adaptation approaches, including those which move the mesh based on the solution of a PDE (see [**Huang et al., 1994**] and [**Budd and Williams, 2009**], for example), advect the mesh in a velocity field (see [**Donea et al., 2017**]) and those which re-interpret the mesh using a fictitious structure of stiff beams (such as [**Farhat et al., 1998**]). Because r-adaptation leaves the topology intact, it has desirable properties, such as typically being straightforward to parallelise – something which can be rather involved for h-adaptation methods.

p-adaptation was introduced in [**Babuška et al., 1981**] and has favourable properties when applied to certain PDEs. For example, for a polynomial discretisation of a Helmholtz problem, the approximation error takes the form $\mathcal{O}(h^p)$, where h is a measure of element size and p is the polynomial degree. In this context, h-refinement of the element will induce a *polynomial* reduction in error, but an increase of the polynomial degree, p, yields an *exponential* reduction in error. Whilst this is a very desirable property which applies to a range of PDEs, p-adaptation can be computationally expensive, especially if the polynomial degree is allowed to increase significantly. In addition, the implementation of p-adaptation requires the facility for non-conforming function spaces, which is not currently present in the Firedrake framework. For the reasons above, this thesis does not consider p-adaptation and restricts attention to *mesh* adaptation methods.

As well as using the above methods individually, it is possible to combine them to yield *hybrid* adaptation methods. The literature contains examples of hp-adaptation methods, beginning with the work of [**Babuška and Suri, 1994**]. There also exist a number of hr-adaptation methods, including the anisotropic mesh adaptation toolkit *Pragmatic* [**Barral et al., 2016**] used to generate results in Chapters 5 and 7. In the case of Pragmatic, h-adaptive operations are applied most extensively and r-adaptation is applied to improve element quality via localised smoothing.

Multi-Scale Mesh Adaptation. Coastal ocean engineering problems are frequently multi-scale in nature. This is because the length scales of the spatial domain, tidal processes, bathymetric features and engineered constructs all exist on different orders of magnitude. In the case of tidal turbine modelling, for example, the diameters of turbine blades are on the order of metres or tens of metres, whereas the spatial domain can be as large as hundreds of kilometres. This motivates the use of an accordingly multi-scale discretisation, for purposes of both accuracy and computational efficiency.

There exist software packages, such as *qmesh* [**Avdis et al., 2018**], which enable the generation of static multi-scale meshes for coastal applications from topographic data sets. However, in time-dependent problems, small or medium scale features such as sediment, pollutants or tsunami waves can traverse the domain. In order to accurately capture such features, it is beneficial to have a mesh which is able to move or deform with time,

motivating the repeated application of multi-scale mesh adaptation methods. Due to its capacity to change the number of degrees of freedom, h-adaptation is well suited to this task. r-adaptation is also attractive, given that it can be used to track features in a Lagrangian type manner. However, used on their own, r-adaptation methods do not have the capability to yield meshes of different overall size at different time levels. Further, in some cases, it is also beneficial to have a discretisation which is multi-scale in time. This is enabled in h-adaptation methods by application of the L^p normalised metric-based mesh adaptation formulation [**Alauzet and Olivier, 2010**].

Objectives

- Implement and investigate both h- and r-adaptation methods and discuss their relative advantages and disadvantages.
- Investigate how to effectively tune parameters involved in such methods.
- Implement mesh adaptation techniques which are multi-scale in both space and time in Firedrake and demonstrate their use in coastal ocean applications.

1.1.2. Adjoint Methods. In many coastal modelling and engineering applications, there exists a diagnostic *quantity of interest (QoI)* which is held to be of greater importance than solutions of the prognostic equations themselves. QoIs may be distinguished by whether they seek to quantify:

(I): an error measure of simulation outputs against data;
(II): some measurable aspect of a physical phenomenon.

Note the similarity with the 'analysis' step for mesh adaptation, which focuses on either an error estimate or a heuristic.

Given a prognostic equation and a QoI which may be computed from its solution, it is possible to derive an associated *adjoint equation*. Solving the adjoint equation allows to assess the impact of a perturbation in the state variables on the QoI. As such, it is extremely useful for sensitivity analysis. Sensitivity analysis is just one use-case, however; adjoint methods are now an invaluable tool for computational modelling.

Data Assimilation. Solutions of the adjoint equation can be used to compute derivatives of the QoI with respect to control parameters which appear explicitly in the problem statement. Further, such derivatives may be computed almost independently of the number of controls. This makes adjoint methods extremely attractive in the context of iterative optimisation methods which use a descent direction based on gradient information.

In this frame of reference, QoIs of type I are typically deployed with the aim of model improvement. They include error measures against analytical solutions used for model calibration (see Subsection 3.6.1), error measures against synthetic data (see Section 4.4) and error measures against real data (see Section 4.5). Type I QoIs may be efficiently minimised using gradient-based optimisation methods which compute the gradient by implicitly solving adjoint equations. The main type I application addressed in this thesis is tsunami source inversion (see Chapter 4). This involves a QoI which quantifies the mismatch error between computed and observed timeseries data and uses gradient-based optimisation methods to invert for parameters defining the initial surface disturbance due to an earthquake-tsunami.

Optimisation methods driven by type I QoIs are commonly referred to using the term *data assimilation*, because they seek to make model improvements based on observations.

Optimal Design. QoIs of type II do not rely on the existence of data and instead seek to quantify measurable features of the solution field. A classic example from outside of geoscience is the drag on an aeroplane wing (see [**Nadarajah and Jameson, 2000**] and [**Carnarius et al., 2011**], for example). Some geoscientific examples include the salinity at the inlet pipe of a desalination plant (see Section 7.7) and the inundation of an important coastal region in the approach of a tsunami (see Section 7.8).

As with the examples above, optimisation methods can be applied to type II QoIs. In engineering applications, instead of seeking control parameters which minimise some error measure, control parameters are sought which yield a design which is in some sense 'optimal'. For example, the configuration of a tidal turbine array optimally tuned in order to maximise its power output or profit [**Funke et al., 2014**]. Similarly, we may seek to minimise drag by tuning design parameters related to an aeroplane wing, or minimise salinity at a desalination plant inlet pipe by modifying plant layout.

Sensitivity Analysis and Uncertainty Quantification. Optimisation is just one application area for adjoint methods (albeit one which enjoys a rich literature). As mentioned above, the adjoint equation assesses the impact of a perturbation in the state variables on the QoI. This can be extremely useful for calibration studies for coastal ocean models. Coupled with an appropriate error model, it is possible to go further and perform uncertainty quantification, thereby analysing the propagation of *uncertainties* in the parameters (see [**Warder, 2020**], for example). Uncertainty quantification is important for assessing the limits of predictability of a particular model.

Continuous vs. Discrete. There are two major approaches when it comes to forming and solving adjoint problems, as outlined in the following.

The approach known most commonly as the *continuous adjoint* method involves the derivation of the adjoint equation by differentiation of the 'forward model' (i.e. prognostic equation). This is typically a manual calculation, following which discretisation is made and the adjoint equation is solved (see [**Gunzburger, 2002**] for details). One advantage of such an approach is that the modeller is free to choose a discretisation for the adjoint problem which is unrelated to that used for the forward problem. See Chapter 3 for continuous adjoint derivations associated with tracer transport and shallow water problems.

The *discrete adjoint* method, on the other hand, obtains a generally different discrete set of equations by adjoining the discretised forward model. The discrete adjoint method has a long history in the engineering literature and is difficult to trace back to a single publication. A major advantage of this method is that it enables the automation of the process of deriving and solving an adjoint problem, avoiding tedious and error-prone hand-derivations. As a result, a wide variety of *automatic differentiation (AD)* codes have been developed (such as [**Bischof et al., 1992**], [**Hascoet and Pascual, 2013**] and [**Hückelheim et al., 2019**]), which can be used to drive discrete adjoint methods. The discrete adjoint method may be deployed in Firedrake using *dolfin-adjoint* [**Farrell et al., 2013**].

Objectives

- Review the literature on continuous and discrete adjoint methods and perform experiments to make comparisons in terms of accuracy, efficiency and overall effectiveness.
- Experiment with computing derivatives by composing different AD tools.
- Apply an adjoint-driven gradient-based optimisation method to the problem of tsunami source inversion.

1.1.3. Goal-Oriented Error Estimation and Mesh Adaptation. Goal-oriented error estimation presents yet another use for adjoint methods in coastal ocean modelling. The 'goal' relates to the diagnostic QoI. That is, the error which we seek to estimate is that which is incurred when the QoI is evaluated on a particular finite element space, i.e. the 'QoI error'.

Error Estimation. The a posteriori *dual-weighted residual (DWR)* error result, derived in [**Becker and Rannacher, 1996**] and later applied to linear and nonlinear problems in [**Becker and Rannacher, 2001**], plays an important role in goal-oriented error estimation. It provides a relation between the QoI error and weak residuals of the forward equation evaluated at the error in the adjoint solution, subject to a remainder term. The weak residual (without the remainder term) presents an error estimator for QoI evaluation. Further, a higher order result provides an estimator involving weak residuals of the adjoint equation and errors in the forward solution. In the case of a linear PDE, the characterisations of the QoI error by these first and second order DWR estimators are *exact* for linear and quadratic QoIs, respectively [**Becker and Rannacher, 2001**].

An alternative, *a priori* approach to goal-oriented error analysis is taken in [**Loseille et al., 2010**], wherein the QoI is shown to be related to discretisation errors in the prognostic equation itself, weighted against the adjoint solution. As with the DWR, this result involves an aspect of error incurred in solving the forward problem, along with information on sensitivities from the adjoint problem. That is, the error analysis seeks to investigate: (a) where in the domain errors are being incurred; and (b) the extent to which these errors significantly impacts upon the accuracy of the QoI calculation. Localisation of error estimators is typically achieved by considering contributions from each mesh element, as discussed in Subsection 1.1.1.

Mesh Adaptation. Decomposed into contributions from each mesh element, goal-oriented error estimates can be used to drive mesh adaptation algorithms, with the aim of obtaining discretisations which minimise QoI error. This is the place where the two modelling techniques studied in this thesis – adjoint methods and mesh adaptation – come together most clearly. For a given QoI, the associated adjoint equations may be solved in order to generate goal-oriented error estimates, which may, in turn, be used to generate adapted meshes which allow accurate approximation of the QoI. In the context of a type I QoI, this means establishing a better fit between data and simulation results. In the context of a type II QoI, it means more accurately quantifying some aspect of a physical phenomenon, which could lead to a more accurate hazard assessment, for example.

Whilst developed in the context of h-adaptation (see [**Becker and Rannacher, 1996**] and [**Becker and Rannacher, 2001**]), goal-oriented mesh adaptation has since been implemented in r-adaptive [**Bauer et al., 2014**], p-adaptive [**Ekelschot et al., 2017**] and hybrid [**Dolejší and Roskovec, 2017**] frameworks.

Anisotropy. Early implementations used element tagging quad-tree type methods to interpret goal-oriented error estimators and drive the mesh adaptation process. In later work, the metric-based framework has been used to encapsulate goal-oriented error estimates. As discussed in Subsection 1.1.1, one of the great advantages of this approach is the fact that allows for control of element anisotropy. When applied to advection-dominated problems such as those describing pollutant dispersal, isotropic meshes often contain wasted resolution in streamline directions. This inefficiency can be accounted for by constructing goal-oriented metrics which inherit anisotropy from the solution field and/or its adjoint.

A number of anisotropic goal-oriented metrics have been introduced in the literature, beginning with the work of [**Formaggia et al., 2004**]. Other notable contributions which are considered at length in this thesis include [**Power et al., 2006**], which arises from an a posteriori error estimate, and [**Loseille et al., 2010**], which arises from an a priori estimate. Extension to the time-dependent case is rather complex, but has been achieved in [**Belme et al., 2012**].

Geoscientific Applications. An interesting philosophical aspect of goal-oriented modelling is the difference in the relationship between modeller and stakeholder, as compared with traditional numerical modelling.

Consider, for example, a tsunami propagation model. The modeller uses data to calibrate a model, which is then used to simulate a past or hypothetical tsunami and in particular its propagation towards the coast. In a traditional numerical modelling framework, the output of this simulation may later be presented in different ways in order to satisfy different stakeholders. For example, flood defence engineers might be concerned with the inundation of one particular coastal settlement and the insurance sector with the effect on an oil rig out at sea, whereas policy makers might be more interested in national scale coastal damage. In order to satisfy the needs of all stakeholders, the accuracy of a traditional numerical model should be high across the entire domain, which typically comes with a requirement for significant computational resources.

In the goal-oriented framework, on the other hand, the concerns of the stakeholder are taken into consideration *before* the model is formulated. That is, the model itself is defined by the QoI. Whilst this approach has the disadvantage that satisfying multiple stakeholders typically means doing multiple goal-oriented model runs, it has the advantage of allowing simulations which use far fewer computational resources than the traditional approach and/or yield a more accurate approximation of the QoI. The achievement of such an aim is, of course, dependent on making appropriate choices for the QoI, discretisation and other model parameters.

In this thesis, goal-oriented mesh adaptation is applied in three application areas within coastal ocean modelling: tidal turbine modelling (see Subsection 7.5.4), desalination plant outfall modelling (see Section 7.7) and tsunami hazard assessment (see Section 7.8). To the best of the author's knowledge, the associated works are the first published to do so.

Objectives

- Derive goal-oriented error estimators for both tracer transport and shallow water problems.
- Account for anisotropy in the formulation of Riemannian metrics used for goal-oriented mesh adaptation.

- Apply goal-oriented mesh adaptation techniques to the coastal ocean modelling problems described above, in both steady-state and time-dependent cases.

1.2. Publications

1.2.1. Publications During PhD Studies. This subsection gives details of all publications and preprints for which the corresponding algorithmic development, code implementation and mathematical and experimental results appear in this thesis, either explicitly or in modified form. In either case, the works listed below are appropriately referenced in the text and disclaimers are provided.

Published. J. G. Wallwork, N. Barral, D. A. Ham and M. D. Piggott,
Anisotropic Goal-Oriented Mesh Adaptation in Firedrake,
In: 28th International Meshing Roundtable, Zenodo, pp.83-100 (2020),
DOI: `10.5281/zenodo.3653373`, URL: `https://zenodo.org/record/3653373`.

Novelties:
- First implementation of goal-oriented mesh adaptation in the Firedrake finite element library.
- First comparison of an isotropic goal-oriented metric and (simplifications of) the anisotropic metrics introduced in [**Power et al., 2006**] and [**Loseille et al., 2010**] in the same solver framework.
- Demonstration of improved QoI error convergence using goal-oriented metrics for an established advection-diffusion test case with an analytical solution.

Contributions: *As first author, I derived a goal-oriented error estimator for the advection-diffusion equation, wrote the first draft of the paper and conducted all numerical experiments. N. Barral assisted with plotting 3D meshes, reviewed drafts and edited Section 2. D. A. Ham and M. D. Piggott also reviewed drafts of the paper. M. D. Piggott was main project supervisor.*

Published. J. G. Wallwork, N. Barral, S. C. Kramer, D. A. Ham and M. D. Piggott,
Goal-Oriented Error Estimation and Mesh Adaptation for Shallow Water Modelling,
Springer Nature Applied Sciences, volume 2, pp.1053–1063 (2020),
DOI: `10.1007/s42452-020-2745-9`, URL: `https://rdcu.be/b35wZ`.

Novelties:
- Derivation of a goal-oriented error estimate for the shallow water equations solved using the $\mathbb{P}1_{DG} - \mathbb{P}2$ finite element pair.
- Demonstration of improved QoI error convergence using goal-oriented metrics for an idealised steady-state tidal farm test case.
- First implementation of mesh adaptation methods within the 2D hydrodynamic component of Thetis.

Contributions: *As first author, I derived a goal-oriented error estimator for the shallow water equations, wrote the first draft of the paper and conducted all numerical experiments. S. C. Kramer conceptualised the steady-state tidal array problem and made a first implementation, later modified by N. Barral. N. Barral, S. C. Kramer, D. A. Ham and M.*

D. Piggott reviewed drafts of the paper. M. D. Piggott was main project supervisor.

Under review. M. C. A. Clare, J. G. Wallwork, S. C. Kramer, H. Weller, C. J. Cotter and M. D. Piggott, *Multi-Scale Hydro-Morphodynamic Modelling using Mesh Movement Methods*, EarthArXiv preprint (2020), DOI: 10.31223/X54G6R, URL: https://doi.org/10.31223/X54G6R. Submitted to International Journal on Geomathematics, May 2021.

Novelties:

- First application of mesh movement (r-adaptation) methods to a full hydro-morphodynamic model with both bedload and suspended sediment transport.
- First implementation of mesh adaptation methods within Thetis' coupled 2D hydro-morphodynamic model.

Contributions: *First author, M. C. A. Clare, developed the hydro-morphodynamic model (first presented in* [**Clare et al., 2021**]*), conducted all numerical experiments and wrote the majority of the first draft, except for Section 2. As second author, I wrote the mesh movement code and the first draft of Section 2, which provides details on the mesh movement strategy. The code was developed as a time-dependent extension of the steady-state mesh movement code written by A. T. T. McRae in* [**McRae et al., 2018**]*. S. C. Kramer provided technical assistance with the experiments. S. C. Kramer, H. Weller, C. J. Cotter and M. D. Piggott reviewed drafts of the paper. M. D. Piggott was main project supervisor.*

Under Review. J. G. Wallwork, N. Barral, D. A. Ham and M. D. Piggott, *Goal-Oriented Error Estimation and Mesh Adaptation for Tracer Transport Modelling*, EarthArXiv preprint (2021), DOI: 10.31223/X56021, URL: https://doi.org/10.31223/X56021. Submitted to Computer Aided Design, January 2021.

Novelties:

- Updated implementations of the goal-oriented metrics in [**Wallwork et al., 2020a**] and [**Wallwork et al., 2020b**] which overcome the simplifications therein. For example, stabilisation and boundary terms are no longer neglected.
- Calibration of a point source approximation against an analytical solution using gradient-based optimisation.
- First application of goal-oriented mesh adaptation to an idealised tidally varying desalination plant outfall scenario.

Contributions: *As first author, I extended the goal-oriented error estimation and mesh adaptation framework introduced in* [**Wallwork et al., 2020a**] *to the time-dependent case, wrote the first draft of the paper and conducted all numerical experiments. N. Barral, D. A. Ham and M. D. Piggott reviewed drafts of the paper. M. D. Piggott was main project supervisor.*

Under Review. J. G. Wallwork, L. Mackie, S. C. Kramer, N. Barral, A. Angeloudis and M. D. Piggott, *Goal-Oriented Metric-Based Mesh Adaptive Tidal Farm Modelling*, EarthArXiv preprint (2021), DOI: 10.31223/X5D61K, URL: https://doi.org/10.31223/X5D61K. Submitted to the Proceedings of the IX International Conference on Computational Methods in Marine Engineering, May 2021.

Novelties:

- Applied isotropic metric-based goal-oriented mesh adaptation to the tidal array test case first considered in [**Divett et al., 2013**], providing the first application of goal-oriented mesh adaptation to an idealised tidally varying tidal farm problem.

Contributions: *As first author, I extended the goal-oriented tidal farm example in* [**Wallwork et al., 2020b**] *to the time-dependent, tidally varying case, wrote the first draft of the paper and conducted all numerical experiments. L. Mackie, S. C. Kramer, N. Barral, A. Angeloudis and M. D. Piggott all advised on the experimental setup and reviewed drafts of the paper. M. D. Piggott was main project supervisor.*

1.2.2. Works in Preparation Based on PhD Research. In addition to the five completed papers listed above, another is currently in preparation.

In Preparation. A paper is in preparation, co-authored by S. Warder and M. D. Piggott on the use of adjoint methods in gradient-based source tsunami source inversion. It will present the first composition of two different AD tools for the purpose of invert both source and propagation components of a tsunami model. It will also include parameter redundancy tests for the Okada source model and a quantitative comparison of continuous and discrete adjoint methods in the context of tsunami source inversion. As first author, I am responsible for all numerical experiments and am in the process of writing the first draft of the paper. S. Warder assisted with the experimental setup. M. D. Piggott is project supervisor.

1.2.3. Publications During Interruption of Studies. The work underpinning the following paper was done during an internship at Argonne National Laboratory, whilst the author was taking an interruption of studies. Its contents do not appear in this thesis, but its subject matter is relevant; it is included here for completeness.

Preprint. J. G. Wallwork, P. D. Hovland, H. Zhang, and O. Marin, *Computing Derivatives for PETSc Adjoint Solvers using Algorithmic Differentiation*, arXiv preprint (2019), DOI: arXiv:1909.02836 [cs.MS], URL: https://arxiv.org/abs/1909.02836.

Novelties:

- First application of the AD tool ADOL-C to compute Jacobians and transpose Jacobians for PETSc nonlinear solvers and their adjoints.
- Performance comparison of dense, sparse and matrix-free approaches for automated Jacobian computation.

Contributions: *As first author, I applied the AD tool, wrote the first draft of the paper and conducted all numerical experiments. H. Zhang provided technical assistance and review drafts of the paper. P. D. Hovland and O. Marin also reviewed drafts of the paper. P. D. Hovland was main project supervisor.*

1.3. Communications

This section lists selected conferences attended by the author during PhD studies, the type of presentation given and any associated awards. Two tutorials were also given by the author during this period.

Selected conference talks:

2018: *Advanced Simulation of Coupled Earthquake and Tsunami Events (ASCETE) workshop* – Bayrischzell, Germany.
2018: *PDE Software Frameworks (PDESoft)* – Bergen, Norway.
2018: *Firedrake Annual User Meeting* – London, U.K.
2019: *Adaptive Modelling and Simulation (ADMOS)* – El Campello, Spain.
2019: *Firedrake Annual User Meeting* – Durham, U.K.
2019: 28^{th} *International Meshing Roundtable (IMR28)* – Buffalo, New York, U.S.A. **(Student travel award.)**
2021: *World Congress on Computational Mechanics* – virtual.

Selected conference poster presentations:

2017: *Climate and Mathematics Network (CliMathNet)* – Reading, U.K. **(Prize for best student poster.)**
2017: *Society of Industrial and Applied Mathematics Conference on Mathematical and Computational Issues in the Geosciences (SIAM GS)* – Erlangen, Germany.
2018: *Society of Industrial and Applied Mathematics U.K. and Ireland Annual Meeting (SIAM UK/IE)* – Southampton, U.K. **(Second prize poster and student travel award.)**
2018: *Portable Extensible Toolkit for Scientific Computing (PETSc) Annual User Meeting* – London, U.K.

Selected tutorials given:

2018: *Introduction to Firedrake* – Argonne National Laboratory, Illinois, U.S.A.
2020: *Introduction to Firedrake* (jointly with three other students) – Mathematics of Planet Earth Centre for Doctoral Training, held virtually.

1.4. Thesis in Brief

The remainder of this thesis is split into two parts, which may be summarised as follows.

Part 1, *Forward and Adjoint*, contains three chapters. *Coastal Ocean Modelling* (Chapter 2) provides background on the use of FEM in this application area. In particular, it details the shallow water, sediment transport and passive tracer components of the 2D coupled Thetis coastal ocean model used throughout this thesis. *Adjoint Methods* (Chapter 3) provides background on both continuous and discrete approaches, providing qualitative comparisons of various aspects thereof. It also includes a quantitative comparison in the context of an established advection-diffusion problem with an analytical solution. Finally, *Tsunami Source Inversion* (Chapter 4) uses adjoint methods to invert a tsunami wave propagation model for its initial condition, both in synthetic test cases and using real gauge data. The experimental results presented in this paper comprise a planned future paper which is currently in preparation (see Subsection 1.2.1). Chapter 4 also includes a quantitative comparison of continuous and discrete adjoint methods in the context of tsunami source inversion for a single control parameter.

Part 2, *Mesh Adaptation*, also contains three chapters. *Metric-Based Mesh Adaptation* (Chapter 5) provides background on one adaptation framework and validates the implementation used in later experiments against standard test cases. Similarly, *Monitor-Based Mesh Adaptation* (Chapter 6) provides background on an alternative framework. In addition to validating against standard test cases, experimental results are presented on the application of monitor-based mesh adaptation to a hydro-morphodynamic test case, as presented in [**Clare et al., 2020**]. Finally, *Goal-Oriented Mesh Adaptation* (Chapter 7) provides theoretical background and brings together the contents of Chapters 2, 3 and 5 to present goal-oriented error estimates for the shallow water and tracer transport models and a range of experimental results. These include those which appear in [**Wallwork et al., 2020a**], [**Wallwork et al., 2020b**] and [**Wallwork et al., 2021**].

Part 1

Forward and Adjoint

CHAPTER 2

Coastal Ocean Modelling

Motivation. The development of advanced discretisation schemes in this thesis is motivated by fluid dynamics in the coastal ocean. Coastal ocean modelling problems pose a challenge for numerical models due to their typically multi-scale nature in both space and time. Moreover, the processes of erosion and deposition can act to substantially alter the three-dimensional domain, by modifications to both the sea bed and the coastal boundaries. Given these challenges, modelling the full three-dimensional dynamics of problems involving long stretches of coastline poses a formidable challenge for the computational geoscientist.

This thesis is primarily focused on development, as opposed to simulation of real events in the coastal ocean (although some case studies are considered). For this purpose, it is useful to work with models which are relatively computationally inexpensive, before progressing to complex problems with multiple important features in large, three-dimensional domains. As such, a two dimensional, depth-averaged approximation to the hydrodynamics is assumed in this thesis. Such an approximation comes with significantly reduced computational cost compared to a full 3D model with the same resolution in the horizontal, yet retains an element of three-dimensionality by solving for free surface elevation, in addition to the horizontal fluid velocity. Tracers advected in the flow are also considered in the depth-averaged context for the majority of this thesis.

Throughout this thesis, the unstructured mesh, discontinuous Galerkin based coastal ocean model Thetis [**Kärnä et al., 2018**] is used for modelling 2D hydrodynamics, as well as the simulation of tracer and sediment transport processes. Some modifications to the Thetis model are made by the author, such as the extension of the tracer model to consider continuous finite elements with SUPG stabilisation.

Chapter Organisation. This review chapter is organised as follows. Section 2.1 introduces the two dimensional approximation to coastal hydrodynamics used in Thetis, including modified equation sets to account for the linear case and drag parametrisations that account for tidal turbines positioned in the flow. Sections 2.2 and 2.3 describe additional equation sets which account for the transport of passive tracers and the sedimentary processes of erosion and deposition, respectively. Aspects of the finite element method which are particularly important for the subject of this thesis are briefly described in Section 2.4, including the continuous Galerkin discretisation used for tracer transport modelling throughout. Section 2.5 describes the integration scheme used for time-dependent problems, whilst Section 2.6 outlines the discontinuous Galerkin method. Two of the mixed finite element discretisations for the hydrodynamics equations used in Thetis are given in detail. Stabilisation methods for both tracer and hydrodynamics models are discussed in Section 2.7, including modifications to account for advection-dominated problems. Finally, Section 2.8 compares interpolation methods used for transferring data between meshes.

2.1. Shallow Water Equations

In this thesis, coastal hydrodynamics are modelled using the nonlinear shallow water equations, which are derived by depth-averaging the Navier-Stokes equations (see pp.48-52 of [**Wesseling, 2009**], for example). Applied to coastal ocean modelling, this equation set describes the transport of horizontal fluid velocity, $\mathbf{u}$ [$\mathrm{m\,s^{-1}}$], and free surface perturbations at the atmospheric boundary. Free surface displacement, η [m], is related to the total water depth, H [m], by the relation

$$H = \eta + b, \tag{2.1}$$

where $b = b(\mathbf{x})$ [m] is the water depth at rest, termed the *bathymetry*.

The shallow water hypothesis involves a number of assumptions on the fluid of interest – here seawater. Firstly, it is assumed that density, ρ [$\mathrm{kg\,m^{-3}}$], does not vary with depth. Secondly, the fluid is assumed to be *hydrostatic*, meaning pressure depends only on density and water depth. Together, these assumptions may be expressed by the following relation for pressure p [Pa]:

$$p = \rho g H + p_a, \tag{2.2}$$

where g [$\mathrm{m\,s^{-2}}$] denotes gravitational acceleration and p_a is the pressure at the atmospheric boundary. In addition, the horizontal length scale is assumed to be far larger than the vertical length scale. This is in accordance with oceanic domains, which are on the order of thousands of kilometres wide, but kilometres deep at most. Finally, the wavelengths we seek to model are assumed to be significantly longer than the water is deep.

Given the above assumptions, the shallow water equations may be expressed in the non-conservative form,

$$\kappa\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} + g\nabla\eta + f\widehat{\mathbf{z}}\times\mathbf{u} + \frac{C_d\|\mathbf{u}\|\mathbf{u}}{H} = \nabla\cdot(\underline{\boldsymbol{\nu}}\nabla\mathbf{u}), \tag{2.3}$$

$$\kappa\frac{\partial \eta}{\partial t} + \nabla\cdot(H\mathbf{u}) = 0, \tag{2.4}$$

where we restrict attention to three external and body forces: the Coriolis effect, bottom friction and viscosity. These forces are parametrised using the Coriolis parameter, f [Hz], (dimensionless) quadratic drag coefficient, C_d, and symmetric positive-definite (SPD) kinematic viscosity tensor, $\underline{\boldsymbol{\nu}}$ [$\mathrm{m^2\,s^{-1}}$]. The parameter $\kappa \in \{0, 1\}$ acts as a switch between the time-independent, steady-state regime ($\kappa = 0$) and the time-dependent, unsteady regime ($\kappa = 1$). It is held constant in practice and is used here to allow a concise presentation of the equation set. Equations (2.3) and (2.4) are referred to as the momentum equation and continuity equation, respectively.

The momentum equation describes the advection of horizontal fluid velocity, which is slowed by viscous and drag forces. In this thesis, the viscosity tensor is always assumed to be an isotropic constant, meaning it can be represented as $\underline{\boldsymbol{\nu}} = \nu\underline{\boldsymbol{I}}$, where $\nu > 0$ is a scalar viscosity and $\underline{\boldsymbol{I}}$ is the identity matrix. The kinematic viscosity of water is on the order of $10^{-6}\,\mathrm{m^2\,s^{-1}}$. As such, it rarely plays an important role in the physics of shallow water models. Indeed, in the tsunami modelling experiments performed in this thesis, it is neglected. However, whilst the viscosity parameter does have a physical basis, it can also be used as a numerical tool to introduce stabilisation. Artificially increasing the viscosity parameter can smooth out turbulent effects, improving the conditioning of the resulting systems of equations, thereby making them easier to solve.

In large scale coastal ocean modelling, it is necessary to account for planetary rotation. This is achieved by allowing the Coriolis parameter to vary with latitude, ϕ. In the general case, it is given by $f = 2\Omega_E \sin\phi$, where Ω_E is the Earth's rotation rate. The 'non-rotational' case is given by $f \equiv 0$. The values of constants used in (2.3)–(2.4) are given in Table 2.1.

Parameter	Value assumed throughout
Gravitational acceleration, g	$9.81\,\mathrm{m\,s^{-2}}$
Planetary rotation rate, Ω_E	$7.291 \times 10^{-5}\,\mathrm{Hz}$

TABLE 2.1. Typical values of constants used in shallow water modelling.

Provided the bathymetry is fixed in time, relation (2.1) implies that the continuity equation describes how total water depth is advected by the fluid velocity. Moving bathymetry is accounted for in Section 2.3.

Equations (2.3)–(2.4) are strongly coupled, in the sense that the dependent variables of each equation appear in the largest term of the evolution equation for the other dependent variable. Numerical schemes deployed to solve the shallow water problem usually solve the constituent equations simultaneously as a monolithic system.

2.1.1. Linearised Shallow Water Equations. Both (2.3) and (2.4) are nonlinear equations, due to the advection and quadratic drag terms, as well as the divergence term in the continuity equation. Nonlinearity plays an important role in many coastal flows, especially where current velocities are high and in regions that are particularly shallow. The nonlinear drag term is also particularly important near to the coast. In the deep ocean, its total water depth divisor implies that it can be ignored in certain applications.

For some applications it is therefore sufficient to consider a linearised form of the shallow water equations. This is often the case in tsunami modelling, for instance, as discussed at the end of this subsection. Doing so has the advantage that computational cost may be reduced, by performing a single linear solve per timestep, rather than multiple linear solves within a nonlinear iteration. For the remainder of this subsection, attention is restricted to the time-dependent case.

Ignoring viscous and drag forces and linearising about the 'lake at rest' state, $(\mathbf{u}, \eta) = \mathbf{0}$, yields the linearised shallow water equations,

$$\frac{\partial \mathbf{u}}{\partial t} + f\widehat{\mathbf{z}} \times \mathbf{u} + g\nabla\eta = 0, \qquad \frac{\partial \eta}{\partial t} + \nabla \cdot (b\mathbf{u}) = 0. \tag{2.5}$$

In the non-rotational case, consider differentiating the continuity equation in time. Substituting for the momentum equation gives rise to a wave equation with wavespeed $c = \sqrt{gb}$:

$$\frac{\partial^2 \eta}{\partial t^2} + \frac{\partial}{\partial t}\nabla\cdot(b\mathbf{u}) = 0 \quad \Longrightarrow \quad \frac{\partial^2 \eta}{\partial t^2} + \nabla\cdot\left(b\frac{\partial \mathbf{u}}{\partial t}\right) = 0 \quad \Longrightarrow \quad \frac{\partial^2 \eta}{\partial t^2} = \nabla\cdot(c^2\nabla\eta). \tag{2.6}$$

A similar wave equation may be obtained for fluid velocity by differentiating the momentum equation in time and the continuity equation in space.

As mentioned above, the linear shallow water equations are often used for tsunami modelling – an application which is considered in the context of source inversion in Chapter

4 and hazard assessment in Section 7.8. For an oceanic domain with a relatively flat bathymetry, tsunami propagation is well captured by the wave equation (2.6). That is, the initial perturbation to the free surface displacement caused by the tsunami can be tracked as it propagates across the ocean. The leading order wave speed $c = \sqrt{gb}$ means that the foremost tsunami waves slow as they approach the coast. This leads to a growth in free surface height in a process termed wave shoaling. As gradients of the free surface grow, the nonlinear terms in (2.3)–(2.4) are no longer negligible and act as counter-balances.

It is desirable to use a tsunami modelling framework which accounts for nonlinearity, bottom friction and inundation modelling, for both source inversion experiments and hazard assessment runs. Indeed, the functionality is available in Thetis [**Kärnä et al., 2018**], which uses the wetting-and-drying scheme introduced in [**Kärnä et al., 2011**]. However, for the purposes of model simplicity and computational efficiency, the linearised equations (2.5) are used for tsunami modelling throughout this thesis.

2.1.2. Shallow Water Tidal Turbine Modelling. In this subsection, we consider tidal turbine modelling using the shallow water equations.

Like wind turbines, tidal stream turbines consist of a vertically oriented axis supporting uniformly separated blades. By depth-averaging, it should be acknowledged that we will not be able to model the three dimensional nature of turbines. As such, their effects are depth-averaged, too.

By design, tidal turbines act to extract energy from the coastal flow. They remove momentum from the system, i.e. manifest as drag-like processes. As such, in order to model the turbines in a depth-averaged framework, we employ a parametrisation-based approach using the quadratic drag term in (2.3). The drag coefficient is decomposed into a background drag coefficient, C_b, and a *turbine drag coefficient*, $C_t = C_t(\mathbf{x})$ [**Funke et al., 2014**]:

$$C_d = C_b + C_t. \tag{2.7}$$

The background drag coefficient is typically assumed constant and applied over the entire domain, although it can be made spatially varying for the purposes of model calibration [**Warder et al., 2020**]. The turbine drag coefficient is spatially varying and is dependent upon the turbine/turbine array configuration and its parametrisation.

A tidal farm, $\mathcal{F}$, is interpreted here as a finite collection of disjoint subsets of Ω; each subset $T \in \mathcal{F}$ corresponds to the footprint of a particular turbine. For a given tidal farm, we assume its constituent turbines to be identical. That is, they have blades sweeping out a cross-sectional areas of identical diameter D and their footprints in the horizontal are also identical, with area $A^{\text{footprint}}$. These are reasonable assumptions, because arrays are typically comprised of turbines of the same design. The cross-sectional area in the vertical swept by the blades is given by $A^{\text{swept}} := \pi(\frac{D}{2})^2$.

The thrust force due to one such turbine is dependent on the horizontal fluid velocity and is given by

$$\mathbf{F}(\mathbf{u}) = \frac{1}{2}\rho_{\text{sea}}\, c_t(\mathbf{u})\, A^{\text{swept}}\, \|\mathbf{u}\|\mathbf{u}, \tag{2.8}$$

where $c_t = c_t(\mathbf{u})$ is the *thrust coefficient* associated with the turbines and ρ_{sea} is the (assumed constant) density of seawater. Thrust may be interpreted as a dimensionless drag,

which depends on properties of the deployed turbines. In general, the thrust coefficient is a function of fluid speed, since it may take account of cut-in speed and the imposition of a maximum, so-called rated, power output from the turbine, for example.

Since we are solving the shallow water equations, the fluid velocity at the turbine is depth-averaged. The fluid velocity used in formula (2.8), however, is technically an upstream velocity which is generating power by rotating a vertically oriented turbine. The naïve approach simply substitutes the depth-averaged velocity into this formula when computing the drag. Whilst this does not have a significant effect on coarse meshes, it is shown in [**Kramer and Piggott, 2016**] that the discrepancy becomes noticeable as the turbine farm region is increasingly well resolved. Correction terms for both rectangular and triangular turbine footprints are proposed in [**Kramer and Piggott, 2016**], the former of which is adopted here:

$$c_t^{\text{corrected}} := 4\,c_t\left(1+\sqrt{1-\frac{A^{\text{swept}}}{HD}c_t}\right)^{-2}. \tag{2.9}$$

Note that this formula requires that $c_t < HD/A^{\text{swept}}$, in order to give a real valued corrected thrust.

With (2.9), the turbine drag may be represented as

$$C_t = \sum_{T\in\mathcal{F}} \frac{1}{2}\,c_t^{\text{corrected}}\,\frac{A^{\text{swept}}}{A^{\text{footprint}}}\,\mathbb{1}_T. \tag{2.10}$$

The area fraction ensures that power is calculated in line with the area swept out by the turbine blades, as opposed to the footprint area. In some cases, it is more appropriate to adopt a smooth approximation, as opposed to the (discontinuous) indicator functions in (2.10). One such example is when the shallow water equations are solved within a turbine array optimisation routine, which seeks to optimise array configuration in order to maximise some quantity of interest, such as power output (see [**Funke et al., 2014**], for example). Turbines can then be treated in terms of a 'turbine density' field, rather than as discrete footprints.

As well as appearing in the shallow water equations as a parametrisation for tidal turbines, the drag coefficient C_t may be used to obtain a proxy for the power output of the tidal farm $\mathcal{F}$. This is obtained by integrating the product of (2.8) and velocity:

$$P(\mathbf{u}) = \int_\Omega \rho_{\text{sea}}\,C_t\|\mathbf{u}\|^3\,\mathrm{d}x. \tag{2.11}$$

In the shallow water framework, it should be noted that (2.11) provides an over-estimation, even with the thrust correction (2.9). This is because the turbine is not active over the whole water depth H, but only a fraction near the bottom [**Kramer and Piggott, 2016**]. We cannot simply multiply by a factor D/H because the water which passes over the turbine has not been accounted for in the model. As such, the power output should also be understood as a depth-averaged approximation. In the steady-state case, the power output (2.11) presents an obvious quantity of interest for construction firms, investors and policy makers. It is used for goal-oriented error estimation and mesh adaptation in Subsection 7.5.4. For time-dependent problems, total energy output and average power output both present suitable QoIs and may be obtained by time integration.

Parameter	Value assumed throughout
Background drag coefficient, C_b	0.0025
Seawater density, ρ_{sea}	$1030.0\,\text{kg}\,\text{m}^{-3}$
Turbine diameter, D	18 m
Turbine thrust coefficient, c_t	0.8

(A) Typical values of constants used in shallow water tidal turbine modelling. The turbine thrust coefficient stated has not yet been corrected using formula (2.9), since it depends on the (problem-specific) water depth.

Typical values for constants used in shallow water tidal turbine modelling are given in Table 2.1. These values should be assumed throughout.

2.2. Tracer Transport

As well as coastal hydrodynamics, we are also interested in modelling tracers transported by the flow. Two tracer advection applications of interest include pollution dispersal and desalination outfall modelling. In the former case, we are concerned with the dispersal of some contaminant from a source, such as an oil spill or an outlet pipe from a chemical plant. In the latter, which is examined in Section 7.7, it is the saline water released from the outlet pipe of a coastally situated desalination plant which is of concern.

Given some tracer, its concentration field c is transported by the fluid velocity according to the advection-diffusion equation,

$$\kappa\frac{\partial c}{\partial t} + \mathbf{u}\cdot\nabla c - \nabla\cdot(\underline{\boldsymbol{D}}\nabla c) = S. \tag{2.12}$$

The transport is also affected by the SPD diffusivity tensor $\underline{\boldsymbol{D}} = \underline{\boldsymbol{D}}(\mathbf{x}, t)$ and an external source term, $S = S(\mathbf{x}, t)$. As with the viscosity tensor, we often restrict attention to time-independent, isotropic diffusion, which may be represented using a scalar diffusion coefficient, $D = D(\mathbf{x})$, as $\underline{\boldsymbol{D}} = D\underline{\boldsymbol{I}}$. As with (2.3)–(2.4), the binary parameter κ provides a switch between the steady-state and time-dependent regimes.

Equation (2.12) can be used to model both active and passive tracers. In a coupled simulation, the fluid velocity for the tracer transport model is enforced through the solution of the shallow water equations. Given that the shallow water fluid velocity is depth-averaged, it should be noted that the tracer concentrated is depth-averaged, too. This should be assumed to be the case, unless otherwise stated. For some test cases in this thesis, we consider the simplified case where the fluid velocity is prescribed either by an analytical expression or by data, rather than being derived from the solution of flow equations.

Suppose $R \subset \Omega$ is a region of the domain which we find to have particular importance. In the pollution dispersal case, this might be a nature reserve or a region with significant human activity. In the desalination outfall case, as well as regions where ecology could be damaged by elevated salinity levels, also of significant importance is the location of the plant's inlet pipe. In either case, a suitable time-independent QoI is given by

$$J(c) = \int_R (c - c_b)\,\mathrm{d}x = \int_\Omega \mathbb{1}_R\,(c - c_b)\,\mathrm{d}x, \tag{2.13}$$

where $c_b \geq 0$ is a constant background tracer concentration, for which we have $c \geq c_b$. The extension to the time-dependent case is obtained by time integration.

2.3. Sediment Transport

One example of an *active* tracer is sediment. Sediment transport is particularly important for coastal ocean modelling, since it describes the processes of erosion and deposition which alter both coasts and sea bed. A schematic for sediment transport is shown in Figure 2.1. Sediment particles which are small enough to be suspended are taken up by erosion, transported within the water and deposited to the sea bed. In addition, particles which are too large to be suspended are moved along the sea bed by *bedload transport*.

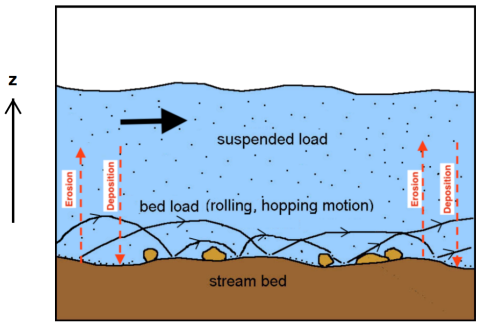


FIGURE 2.1. Diagram depicting sediment transport, as presented in [Clare et al., 2020].

In this thesis, suspended sediment transport is modelled using a non-conservative advection-diffusion equation. Sediment transport is an inherently transient process, so we restrict attention to the time-dependent case. If $s = s(\mathbf{x}, t)$ is the depth-averaged sediment concentration, $\underline{\boldsymbol{D_s}}$ is the diffusion tensor, e_b is the erosion flux and d_b is the deposition flux, then the equation takes the form

$$\frac{\partial s}{\partial t} + \nabla \cdot (F_{\text{corr}} \mathbf{u}\, s) - \nabla \cdot (\underline{\boldsymbol{D_s}} \nabla s) = e_b - d_b. \tag{2.14}$$

The correction factor, F_{corr}, takes account of the fact that depth-averaging a product is not equivalent to the product of two depth-averaged quantities (see [Clare et al., 2021] for details). Aside from the correction term, (2.14) is effectively just (2.12) with modified advection term and a sink.

Suspended sediment affects the bed via erosion and deposition, as conveyed by the source and sink terms on the RHS of (2.14). The deformation of the bed is modelled by an additional scalar equation, called the *Exner equation*, attributed to meteorologist and sedimentologist Felix Maria Exner [Exner, 1920]. The Exner equation,

$$\frac{1 - p'}{m} \frac{\partial b}{\partial t} + \nabla \cdot \mathbf{Q_b} = d_b - e_b, \tag{2.15}$$

is solved for the bathymetry, which is now interpreted as a function of time, as well as space. Here $p' : \Omega \to [0, 1]$ is the bed porosity, $m \geq 0$ is a morphological scale factor and $\mathbf{Q_b} = \mathbf{Q_b}(\mathbf{x})$ represents bedload transport. The porosity of the bedrock acts to lessen bedlevel changes. If $p' = 1$ then the bed is very permeable and there are no bedlevel changes. If $p' = 0$ then the bed is impermeable. As detailed in [Clare et al., 2021], setting $m > 0$ enables the modeller to artificially accelerate bedlevel changes, meaning that simulations can be run over shorter timescales and still exhibit the same deformations to the bed. The default is to model the bedload transport in real time using $m = 1$. The

bedload transport $\mathbf{Q_b}$ is a function of the fluid velocity from the shallow water equations, as well as water depth.

The hydro-morphological model used in [Clare et al., 2020] is comprised of the nonlinear shallow water equations (2.3)–(2.4), sediment transport equation (2.14) and Exner equation (2.15). Note that the momentum equation, continuity equation and Exner equations are solved for velocity, elevation and bathymetry and all three of these variables appear in all three equations, implying a two-way coupling. The sediment equation has a two-way coupling to the Exner equation, since the deposition term, d_b, is dependent on sediment concentration, as well as velocity and depth. By extension, the shallow water equations are also affected by the solution of the sediment equation, even if this variable does not appear explicitly in their formulation. In this thesis, the shallow water equations are solved using a mixed formulation (i.e. as a monolithic system), but all other equations are solved sequentially at each timestep, in the order in which they have been presented.

For further details on the hydro-morphodynamic model, see [Clare et al., 2021].

2.4. Finite Element Method

In this thesis we restrict attention to finite element spatial discretisations for the shallow water and advection-diffusion equation sets. See [Clare et al., 2021] for a description of the FEM discretisation used for hydro-morphodynamic modelling.

One of the fundamental differences between FEM and finite difference and finite volume methods is that, whilst those methods utilise discrete approximations to derivatives, FEM retains the continuity of derivatives and instead discretises solution fields. Assuming solution fields to exist in function spaces comprised of sufficiently simple functions, it is possible to compute their derivatives *exactly* (to machine precision). Thus, when seeking to construct effective finite element methods, the emphasis is on making appropriate choices of function spaces and the underlying meshes.

2.4.1. Linear PDEs. To begin with, consider a time-independent, linear PDE written in the form

$$\mathcal{A}(w) = f, \qquad w \in W, \tag{2.16}$$

along with appropriate boundary conditions, where W is a function space on the domain $\Omega \subset \mathbb{R}^n$ and f is a forcing term which is independent of the prognostic solution, w. For a concrete example, consider a steady-state advection-diffusion equation with a homogeneous Neumann condition on $\partial\Omega$. That is,

$$\mathcal{A}(w) := \mathbf{u} \cdot \nabla w - \nabla \cdot (\underline{\boldsymbol{D}} \nabla w), \tag{2.17}$$

for some velocity field $\mathbf{u} : \Omega \to \mathbb{R}^n$ and diffusivity tensor $\underline{\boldsymbol{D}} : \Omega \to \mathbb{R}^{n \times n}$.

In the following, we refer to W as the *trial space*. In addition, consider another function space $V \subset L^2(\Omega)$, termed the *test space*. In order for the integral formulation

$$\langle \mathcal{A}(w), v \rangle = \langle \mathbf{u} \cdot \nabla w - \nabla \cdot (\underline{\boldsymbol{D}} \nabla w), v \rangle = \langle f, v \rangle, \qquad \forall v \in V \tag{2.18}$$

to be defined for the linear differential operator (2.17), we require that w be twice differentiable and that its first and second derivatives be square integrable, i.e. $W = H^2(\Omega)$. Clearly 'strong' solutions of the PDE – i.e. exact solutions of (2.16) – are also solutions

of the integral formulation (2.18). The converse is also true, by the Fundamental Lemma of the Calculus of Variations. Instead of seeking strong solutions, FEM seeks 'weak' solutions, as detailed in the following.

First, we choose the test space so that first order derivatives are defined and square integrable, i.e. $V = H^1(\Omega) \subseteq L^2(\Omega)$. Integrating the diffusion term in (2.18) by parts gives

$$\langle \mathbf{u} \cdot \nabla w, v \rangle + \langle \underline{\boldsymbol{D}} \nabla w, \nabla v \rangle - \cancel{\langle \underline{\boldsymbol{D}} \nabla w \cdot \widehat{\mathbf{n}}, v \rangle_{\partial\Omega}} = \langle f, v \rangle \,, \qquad \forall v \in V, \tag{2.19}$$

where the surface term drops out due to the Neumann condition. We obtain the weak form by considering a smaller trial space. For the advection-diffusion example, take $W = H^1(\Omega)$, noting that the trial and test spaces now coincide. This is motivated by the fact that the highest order derivatives which appear in (2.19) are first order, rather than second order. Now, whilst strong solutions are still weak solutions, the converse is not true in general.

In order to obtain a numerical solution of (2.19), we consider finite dimensional subspaces $W_h \subset W$ and $V_h \subset V$. There is an element of design involved in making choices of W_h and V_h which are suitable for the problem at hand.[1] For the purposes of this thesis, we assume that each consists of piecewise polynomials defined on some mesh, as described in the following subsections. A finite element approximation, $w_h \in W_h$, to the solution of (2.19) is obtained by solving the equation

$$\langle \mathbf{u} \cdot \nabla w_h, v \rangle + \langle \underline{\boldsymbol{D}} \nabla w_h, \nabla v \rangle = \langle f, v \rangle \,, \qquad \forall v \in V_h. \tag{2.20}$$

Since W_h and V_h are finite dimensional, there exist bases of functions $\mathcal{B}_{W_h} = \{\psi_j\}_{j=1}^N$ and $\mathcal{B}_{V_h} = \{\phi_i\}_{i=1}^M$ such that $W_h = \text{span}(\mathcal{B}_{W_h})$ and $V_h = \text{span}(\mathcal{B}_{V_h})$. Therefore, the weak solution may be represented as a linear combination of members of the basis:

$$w_h = \sum_{i=1}^{N} w_i \psi_i, \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} \in \mathbb{R}^N. \tag{2.21}$$

Since (2.20) is linear, we have that

$$\left\langle \sum_{i=1}^{N} w_i \mathbf{u} \cdot \nabla \psi_i, \phi_j \right\rangle + \left\langle \sum_{i=1}^{N} w_i \underline{\boldsymbol{D}} \nabla \psi_i, \nabla \phi_j \right\rangle = \langle f, \phi_j \rangle \,, \qquad \forall j = 1 : M. \tag{2.22}$$

In our case, each basis function is polynomial on element interiors, meaning its derivatives are also polynomial on element interiors and may therefore be calculated using elementary calculus. In addition, we can integrate polynomials exactly using a quadrature scheme of appropriate degree. Therefore, the integrals in (2.22) can often be computed exactly, depending on the forms of $\mathbf{u}$, $\underline{\boldsymbol{D}}$ and f. This is not true if they involve negative powers, for example. Solving (2.22) for the vector of coefficients $\mathbf{w} \in \mathbb{R}^N$ amounts to solving the linear system

$$\underline{\boldsymbol{A}}\mathbf{w} = \mathbf{f}, \quad \text{where} \quad A_{ij} := \langle \mathbf{u} \cdot \nabla \psi_i, \phi_j \rangle + \langle \underline{\boldsymbol{D}} \nabla \psi_i, \nabla \phi_j \rangle \,, \; f_j := \langle f, \phi_j \rangle \,, \tag{2.23}$$

with $\underline{\boldsymbol{A}} \in \mathbb{R}^{M \times N}$ and $\mathbf{f} \in \mathbb{R}^M$.

[1] Note that spaces W_h and V_h often coincide. When they do not, we call the resulting method a *Petrov-Galerkin* method.

The matrix and vector entries presented in (2.23) contain only integrals over element interiors. In many cases, these objects will also contain integrals over boundary segments and other element facets. Whilst Neumann conditions are typically imposed by replacement of the boundary terms in the integral form, as shown above, Dirichlet conditions may imposed by directly modifying entries of the matrix in the resulting linear system (although this is not appropriate for every choice of function space).

Henceforth, we refer to the dimension of a finite dimensional function space (i.e. the number of constituent basis functions) as the number of *degrees of freedom (DoFs)*, or the *DoF count*.

2.4.2. Meshes. The finite dimensional function space V_h used in a weak form (2.20) is defined upon a *mesh*. A mesh $\mathcal{H}$ provides a decomposition of the spatial domain, Ω, into a disjoint set of non-overlapping *elements*, $K \in \mathcal{H}$.

In this thesis, we restrict attention to meshes comprised of triangular (2D) and tetrahedral (3D) elements, although these are by no means the only options. Since the boundaries of triangular and tetrahedral elements are linear, we assume the piecewise smooth domain boundary to be approximable as piecewise linear. The notation $\partial\Omega$ is used for both the boundary and its piecewise linear approximation, unless otherwise stated. As with the mesh, we interpret the discretised boundary as the set of its constituent facets.

The facet set of element K is denoted ∂K and the corresponding set of outward-pointing normal vectors is denoted $\widehat{\mathbf{n}}_K$. The intersection of two facet sets is either the empty set or a singleton set. Each facet γ may be identified as being either a boundary facet (if $\gamma \in \partial\Omega$) or an inter-element facet (otherwise). Occasionally, an element size measure $h = h(K)$ is mentioned as a subscript, $\mathcal{H}_h$, if instructive. Whilst element size is commonly measured using the circumradius, other measures which account for anisotropy can prove useful, as discussed in Subsection 2.7.3.

Function space dimension is related to the underlying mesh. In general, more mesh elements means more basis functions, a larger linear system (2.23) and therefore a higher computational cost for the solution thereof. At the same time, high mesh resolution is often required in order to obtain an accurate approximation of features in the PDE solution. In the context of tsunami modelling, for example, it is of vital importance that the tsunami wave is well-resolved at any given timestep. Whilst the wave is on the scale of kilometres or tens of kilometres, spatial domains used for tsunami modelling can be on the scale of thousands of kilometres. Whilst uniform meshes comprised of identical elements (up to reflections and rotations) are very simple to design, their use in multi-scale geoscience applications implies an extremely large computational cost. This motivates the search for meshes of spatially varying resolution. One approach is to choose some initial mesh and then transform it using *mesh adaptation* methods. Two paradigms for framing mesh adaptation methods are discussed in detail in Chapters 5 and 6, the first of which allows for mesh resolution which varies in time as well as space, which is particularly useful in the tsunami modelling application mentioned above.

2.4.3. Function Spaces. As well as making a simple choice of element shape, we restrict attention to rather simple function spaces containing functions which are polynomials on each element.

The *discontinuous Galerkin (DG)* function space of order $p \in \mathbb{N} \cup \{0\}$ is denoted

$$\mathbb{P}p_{DG} := \{f \in L^2(\Omega) \mid \forall K \in \mathcal{H}, \exists q \in \mathcal{P}^p(K) : f|_K = q\}, \tag{2.24}$$

where $\mathcal{P}^p(K)$ is the set of polynomials of order at most p defined on element K. As suggested by the name, continuity is not imposed across inter-element facets in DG function spaces. Members of *continuous Galerkin (CG)* function spaces, however, are assumed to have $\mathcal{C}^0$ continuity, which implies

$$\mathbb{P}p := \mathbb{P}p_{DG} \cap H^1(\Omega). \tag{2.25}$$

Note that the case $p = 0$ does not make sense for (2.25) and so we follow the literature in using the notation $\mathbb{P}0 := \mathbb{P}0_{DG}$.

For an order p CG *vector* function space of dimension $n \in \mathbb{N}$, we write $(\mathbb{P}p)^n$, dropping the power if the meaning is clear. For a $n \times n$ CG *tensor* function space of order p, we write $(\mathbb{P}p)^{n\times n}$. The same applies for vector and tensor DG spaces.

It should be pointed out that, because finite element derivatives are computed *on element interiors*, differentiated fields are generally discontinuous across inter-element boundaries, even if the source field is continuous. Moreover, the order of the function space dictates how many finite element derivatives may be taken before the result is uniformly zero. For example, the second derivative of a $\mathbb{P}1$ field is always uniformly zero.

2.4.4. CG Discretisation for Tracer Transport Modelling. The weak form of a steady-state advection-diffusion problem was already derived in Subsection 2.4.1. For the *continuous Galerkin (CG)* approach of order $p \in \mathbb{N}$, we choose the finite dimensional subspaces $W_h = V_h = \mathbb{P}p \subset H^1(\Omega)$.

2.4.5. Coordinate Mapping. In order to evaluate the above integrals of a weak form on an element $K \in \mathcal{H}$ using quadrature, a coordinate transformation $\mathbf{T}_K : \widehat{K} \to K$ is applied so that they may be reinterpreted on a (fixed) *reference element*, $\widehat{K}$. Doing so ensures that the quadrature nodes exist at fixed locations, for ease of computation. For the triangular and tetrahedral meshes considered in this thesis, this transformation is an affine map,

$$\mathbf{T}_K(\boldsymbol{\xi}) = \underline{\boldsymbol{J}}_K \boldsymbol{\xi} + \boldsymbol{\gamma}_K, \tag{2.26}$$

where $\boldsymbol{\gamma}_K \in \mathbb{R}^n$ is a constant and $\underline{\boldsymbol{J}}_K \in \mathbb{R}^{n\times n}$ is the *cell Jacobian*. In the case of the integral of a function f, the relevant computation is

$$\int_\Omega f \,\mathrm{d}x = \sum_{K\in\mathcal{H}} \int_K f \,\mathrm{d}x = \sum_{K\in\mathcal{H}} \int_{\widehat{K}} f(\mathbf{T}_K(\boldsymbol{\xi})) \,|\det \underline{\boldsymbol{J}}_K| \,\mathrm{d}\xi. \tag{2.27}$$

Similar formulae may be derived for boundary and facet integrals.

2.4.6. Céa's Lemma. A fundamental result in error analysis for finite element problems is *Céa's lemma* [Ciarlet, 1978]. It is stated here because is is used repeatedly in later chapters. Given the solution, $w \in W$, of an elliptic PDE and its finite element representation, $w_h \in W_h$, Céa's lemma states that

$$\|w - w_h\| \leq C\|w - \Pi_h w\|, \tag{2.28}$$

where $C > 0$ is a constant and $\Pi_h : W \to W_h$ projects into W_h.

A consequence of this result is that the approximation error, $w - w_h$, may be controlled by controlling the interpolation error, $w - \Pi_h w$. Whilst Céa's lemma was established for elliptic problems, it has also been used to obtain effective error estimators for non-elliptic problems on numerous occasions [**Frey and Alauzet, 2005**].

2.4.7. Riesz Representation. Another fundamental result in finite element analysis is given in the following. It related to the *dual space*, V^*, of a Hilbert space V – the space of all functionals $V \to \mathbb{R}$ which are linear and continuous.

Theorem 1 (Riesz representation theorem)**.** Let V be a Hilbert space with associated inner product $\langle \cdot, \cdot \rangle_V$. For any $J \in V^*$, there exists a unique $\mathcal{R}_V(J) \in V$ such that

$$J(v) = \langle \mathcal{R}_V(J), v \rangle_V , \qquad \forall v \in V. \tag{2.29}$$

Proof. See pp.55–56 of [**Brenner and Scott, 2007**], for example. □

We refer to $\mathcal{R}_V(J)$ as the *Riesz representation* of J in V. This result is extremely important for FEM, revealing the link between V and its dual space.

2.4.8. Derivatives in Infinite Dimensions. Before moving on to treat nonlinear problems using FEM, we first need to extend the notion of differentiability from Euclidean space to the infinite-dimensional case. The following definitions are taken from pp.23–24 of [**Schwedes et al., 2017**].

We say that a functional $F : U \subseteq V \to \mathbb{R}$ acting on an open subset $U \neq \emptyset$ of a Hilbert space is *directionally differentiable* if

$$\mathrm{d}F_u(u; h) := \lim_{\epsilon \to 0^+} \frac{F(u + \epsilon h) - F(u)}{\epsilon} \tag{2.30}$$

exists $\forall u, h \in U$. Moreover, F is *Gâteaux differentiable* if the directional derivative $\mathrm{d}F_u$ is continuous and linear in h. In this case, (2.30) is termed the *Gâteaux derivative* and $\mathrm{d}F_u(u; \cdot) \in V^*$, $\forall u \in U$.

If F is Gâteaux differentiable and

$$\lim_{\|h\|_V \to 0} \frac{\|F(u + h) - F(u) - dF(u; h)\|}{\|h\|_V} = 0, \qquad \forall u \in U \tag{2.31}$$

then it is said to be *Fréchet differentiable* and (2.30) is the *Fréchet derivative* of F.

Suppose now that $F : U \times U \to \mathbb{R}$ and u depends on a parameter, m, in some control space C, i.e. $u : C \to U$. The infinite dimensional extension of the partial derivative of F w.r.t. m is given by

$$\partial F_m(u(m), m; h) := \lim_{\epsilon \to 0^+} \frac{F(u(m), m + \epsilon h) - F(u(m), m)}{\epsilon} \tag{2.32}$$

and the total derivative becomes

$$\mathrm{d}F_m(u(m), m; h) := \lim_{\epsilon \to 0^+} \frac{F(u(m + \epsilon h), m + \epsilon h) - F(u(m), m)}{\epsilon}. \tag{2.33}$$

These definitions satisfy the chain rule

$$\mathrm{d}F_m(u(m), m; \cdot) = \partial F_u(u(m), m; \mathrm{d}u_m(m; \cdot)) + \partial F_m(u(m), m; \cdot). \tag{2.34}$$

2.4.9. Nonlinear PDEs. Consider now a nonlinear PDE,

$$\Psi(w) = 0, \qquad w \in W, \tag{2.35}$$

with solution w in a function space W defined on a domain $\Omega \subset \mathbb{R}^n$ and Ψ being a nonlinear differential operator. As in Subsection 2.4.1, consider multiplying by a test function from a test space V, integrating over Ω and then by parts in order to arrive at a weak form

$$a(w; v) = L(v), \qquad \forall v \in V. \tag{2.36}$$

Here $a : W \times V \to \mathbb{R}$ is linear in v but not w and $L : V \to \mathbb{R}$ does not depend on w. We then have the residual $F(w; v) := L(v) - a(w; v)$ for the nonlinear variational problem.

The standard method for solving (2.36) using FEM is to apply a Newton method. The first step is to linearise the problem. Since we are working with finite element spaces, this requires the Gâteaux derivative at w, in 'direction' $\widetilde{w} \in W$:

$$dF_w(w; v, \widetilde{w}) = \lim_{\epsilon \to 0} \frac{F(w + \epsilon\widetilde{w}; v) - F(w; v)}{\epsilon}, \qquad w, \widetilde{w} \in W, \quad v \in V. \tag{2.37}$$

Note that the derivative is evaluated at the current value of w and can be likened to the infinite-dimensional extension of the directional derivative. Using this, we can obtain the first order truncated Taylor expansion,

$$F(w + \widetilde{w}; v) \approx F(w; v) + dF_w(w; v, \widetilde{w}), \qquad \forall v \in V, \tag{2.38}$$

for $w, \widetilde{w} \in W$. Newton's method takes an initial guess $w_0 \in W$ and forms iterates $w_{k+1} := w_k + \alpha_k p_k \in W$ by solving linear systems for the 'search direction' $p_k \in W$:

$$dF_w(w_k; v, p_k) = -F(w_k; v), \qquad \forall v \in V. \tag{2.39}$$

In practice, (2.39) is solved using finite dimensional subspaces, as in Subsection 2.4.1.

It is common practice to select the 'step length' parameter $\alpha_k > 0$ in an iterative fashion using a *line search* method, so that it meets sufficient descent conditions [**Armijo, 1966**, **Wolfe, 1969**, **Wolfe, 1971**]. For further details on Newton methods with line search, see [**Avriel, 2003**].

2.5. Time Integration

Discretisation aspects of this thesis are focused on meshing, i.e. selection of the mesh underlying a spatial discretisation which makes it appropriate for the numerical solution of a particular problem. As such, temporal discretisation is not considered in detail and only relatively simple methods are used.

The (systems of) prognostic equations we consider typically take the form

$$\frac{\partial w}{\partial t} = \Psi(w), \qquad w(\mathbf{x}, 0) = w_0, \qquad w \in V, \tag{2.40}$$

along with some boundary conditions, where $\Psi(\cdot)$ is a differential operator acting on function space V which includes only spatial derivatives. In particular, we do not consider equations which contain mixed derivatives of both time and space.

It is possible to discretise the time derivative in (2.40) using FEM, as well as the spatial derivatives, i.e *space-time FEM*. Such a formulation unifies error contributions from the spatial and temporal discretisations and simplifies the derivation of error estimates. However, it comes at the cost of having either a very large system to solve, or a rather involved implementation. Instead, we follow the literature by using the most common discretisation approach: the *method of lines*. In this approach, the temporal and spatial discretisations are treated separately.

Typically, the PDE is first discretised in space, with time derivatives remaining in continuous form. This gives the 'semi-discrete' system

$$\left\langle \frac{\partial w_h}{\partial t}, v \right\rangle = F(w_h; v), \quad \forall v \in \widetilde{V_h} \subset V, \qquad w(\mathbf{x}, 0) = w_0, \tag{2.41}$$

where $F(\cdot;\cdot)$ is a variational formulation of $\Psi(\cdot)$ for spatial derivatives. The test space $\widetilde{V_h} = V_h \times L^2$ and trial space $\widetilde{W_h} = W_h \times \mathcal{C}^1$ are finite-dimensional in space, but infinite-dimensional in time. This equation may be interpreted as a system of ODEs. A wide variety of time integration schemes from the literature may be applied to solve such a system. They begin by discretising the time period using a sequence of time levels $\{t^{(k)}\}_{k=0}^{N}$, so that

$$(0, T] = \cup_{k=1}^{N} (t^{(k-1)}, t^{(k)}], \qquad 0 = t^{(0)} < t^{(1)} < \cdots < t^{(N-1)} < t^{(N)} = T. \tag{2.42}$$

Initial data is assumed at time level $t^{(0)}$. Given the finite element solution at time $t^{(k)}$, the time integration scheme approximates the solution at time $t^{(k+1)}$.

In this thesis, time-dependent prognostic equations are always solved with (some variant of) the Crank-Nicolson method [**Crank and Nicolson, 1947**]. Crank-Nicolson is an implicit method, which means that the solution $w_h^{(k+1)} \in W_h$ at the updated time level depends on information from the updated time level, as well as the solution $w_h^{(k)} \in W_h$ from the previous level. As such, its implementation requires the solution of linear systems at each time level. Applied to FEM, the timestepping scheme may be stated as

$$\left\langle \frac{w_h^{(k+1)} - w_h^{(k)}}{\Delta t}, v \right\rangle = \theta F(w_h^{(k+1)}; v) + (1-\theta) F(w_h^{(k)}; v), \quad \forall v \in V_h, \qquad w_h^{(0)} = w_0, \tag{2.43}$$

where $\theta \in [0, 1]$ is a user-specified parameter, usually set as $\theta = \frac{1}{2}$. Note that the choice $\theta = 1$ recovers the implicit Euler method. In the special case $\theta = 0$, the timestepping scheme ceases to be implicit and recovers the explicit Euler method.

2.6. Discontinuous Galerkin Method

DG finite element methods present an increasingly popular discretisation choice. They were first used to model neutron transport in [**Reed and Hill, 1973**]. For DG methods, continuity constraints are not strongly imposed on elemental boundaries and the contributions from each element are localised. Instead, elemental contributions are coupled through flux terms which take the form of integrals over inter-element facets.

Unlike continuous discretisations, DG methods are known to be able to accurately capture shocks. In addition, DG methods allow the construction of fully conservative discretisations, making them ideally posed for application to systems of hyperbolic conservation laws such as those found in coastal ocean modelling [**Kärnä et al., 2018**]. One major advantage of DG methods which has led to their increased popularity in recent years is the fact that the mass matrix associated with a DG space is block-diagonal, meaning it can be inverted efficiently.

Due to the discontinuities across inter-element facets, DG discretisations give rise to approximate *Riemann problems* at every such interface. Such problems were first solved for finite volume discretisations using Godunov's scheme [**Godunov, 1959**]. That work provided a basis for later research extending to higher order approximations. The extension to FEM follows from the fact that a $\mathbb{P}1_{DG}$ discretisation may be reposed as a finite volume method. In the discretisation presented here, Lax-Friedrichs stabilisation is applied in order to avoid the computational cost associated with solving such problems.

2.6.1. Notation. In order to formulate the fluxes used in DG methods mathematically, we introduce some notation. Let Γ_{int} be the set of all mesh edges which are not on the domain boundary, i.e. 'interior' edges. For an edge $\gamma \in \Gamma_{\text{int}}$, its sides are labelled arbitrarily with $+$ and $-$. Each has associated normal vectors $\widehat{\mathbf{n}}^{\pm}$. The *average* and *flux jump* of some function v across the edge are denoted

$$\{\!\{v\}\!\} := \frac{1}{2}(v^+ + v^-) \qquad \text{and} \qquad [\![v]\!] := v^+ - v^-, \tag{2.44}$$

respectively, which we should note as being dependent on the $\pm$ labelling. To extend (2.44) to *all* mesh edges, we define their values on boundary edges to just be the function as defined on the interior element.

For vector-valued functions $\mathbf{v}$, write

$$[\![\mathbf{v}\cdot\widehat{\mathbf{n}}]\!] := \mathbf{v}^+\cdot\widehat{\mathbf{n}}^+ - \mathbf{v}^-\cdot\widehat{\mathbf{n}}^- \qquad \text{and} \qquad [\![\mathbf{v}\widehat{\mathbf{n}}^T]\!] := \mathbf{v}^+(\widehat{\mathbf{n}}^+)^T - \mathbf{v}^-(\widehat{\mathbf{n}}^-)^T. \tag{2.45}$$

2.6.2. Shallow Water Equations.

Integration by Parts. Let $\mathbf{q} = (\mathbf{u}, \eta) \in W$ denote the solution tuple of the steady-state shallow water equations (2.3)–(2.4) (with $\kappa = 0$), where $W = H^2(\Omega) \times H^1(\Omega)$. Multiplying by test functions from $V = H^1(\Omega) \times H^1(\Omega)$ and integrating over Ω, we arrive at the integral form, $\forall (\boldsymbol{\phi}, \zeta) \in V$:

$$\begin{aligned} &\underbrace{\int_\Omega \boldsymbol{\phi}\cdot(\mathbf{u}\cdot\nabla\mathbf{u})\,\mathrm{d}x}_{\text{advection}} + \underbrace{\int_\Omega g\boldsymbol{\phi}\cdot\nabla\eta\,\mathrm{d}x}_{\text{pressure gradient}} + \underbrace{\int_\Omega f\boldsymbol{\phi}\cdot(\widehat{\mathbf{z}}\times\mathbf{u})\,\mathrm{d}x}_{\text{Coriolis}} \\ &+ \underbrace{\int_\Omega \boldsymbol{\phi}\cdot\frac{C_d\|\mathbf{u}\|\mathbf{u}}{H}\,\mathrm{d}x}_{\text{drag}} - \underbrace{\int_\Omega \boldsymbol{\phi}\cdot(\nabla\cdot(\underline{\boldsymbol{\nu}}\nabla\mathbf{u}))\,\mathrm{d}x}_{\text{viscosity}} + \underbrace{\int_\Omega \zeta\nabla\cdot(H\mathbf{u})\,\mathrm{d}x}_{\text{continuity}} = 0. \end{aligned} \tag{2.46}$$

We treat each term in order, restricted to some subset $K \subseteq \Omega$.

Integrating the advection term by parts on K yields

$$\int_K \boldsymbol{\phi}\cdot(\mathbf{u}\cdot\nabla\mathbf{u})\,\mathrm{d}x = -\int_K (\nabla\cdot(\mathbf{u}\boldsymbol{\phi}^T))\mathbf{u}\,\mathrm{d}x + \int_{\partial K}(\mathbf{u}\cdot\boldsymbol{\phi})(\mathbf{u}\cdot\widehat{\mathbf{n}})\,\mathrm{d}S. \tag{2.47}$$

We choose not to integrate the pressure gradient term by parts. Neither of the Coriolis or drag terms are modified, since they do not contain derivatives. For the viscosity term, we have

$$\int_K \boldsymbol{\phi} \cdot (\nabla \cdot (\underline{\boldsymbol{\nu}} \nabla \mathbf{u}))\, \mathrm{d}x = \int_K \nabla \boldsymbol{\phi} : \underline{\boldsymbol{\nu}} \nabla \mathbf{u}\, \mathrm{d}x - \int_{\partial K} \boldsymbol{\phi} \widehat{\mathbf{n}}^T : \underline{\boldsymbol{\nu}} \nabla \mathbf{u}\, \mathrm{d}S. \tag{2.48}$$

Finally, the continuity term gives

$$\int_K \zeta \nabla \cdot (H\mathbf{u})\, \mathrm{d}x = -\int_K \nabla \zeta \cdot (H\mathbf{u})\, \mathrm{d}x + \int_{\partial K} \zeta H \mathbf{u} \cdot \widehat{\mathbf{n}}\, \mathrm{d}S. \tag{2.49}$$

Now that we have integrated by parts, we restrict attention to finite dimensional spaces $W_h \subset W$ and $V_h \subset V$ and consider $K \subseteq \Omega$ to be an arbitrary mesh element $K \in \mathcal{H}$.

2.6.2.1. *Boundary Conditions.* In this thesis, there are three types of boundary condition used for the shallow water model: Dirichlet conditions for the velocity, Dirichlet conditions for the elevation and free-slip conditions. Denote the corresponding boundary segments Γ_u, Γ_η and Γ_{freeslip}. With DG methods, it is common practice to impose Dirichlet conditions in the weak sense. An approximate Riemann solver type formulation is adopted, following [**Kärnä et al., 2013**]. This is based on the solution of a Riemann problem for the linearised shallow water equations (2.5) with no rotation. The Riemann solutions for velocity and elevation are given by

$$\begin{aligned} \mathbf{u}^{\text{rie}} \cdot \widehat{\mathbf{n}} :=& \frac{1}{2}(\mathbf{u}_h + \mathbf{u}^{\text{ext}}) \cdot \widehat{\mathbf{n}} + \sqrt{\frac{g}{H}}(\eta_h - \eta^{\text{ext}}), \\ \eta^{\text{rie}} :=& \frac{1}{2}(\eta_h + \eta^{\text{ext}}) + \sqrt{\frac{H}{g}}(\mathbf{u}_h - \mathbf{u}^{\text{ext}}) \cdot \widehat{\mathbf{n}}, \end{aligned} \tag{2.50}$$

where $\mathbf{q}_h = (\mathbf{u}_h, \eta_h) \in W_h$ is the finite element approximation to $(\mathbf{u}, \eta) \in W$. The velocity $\mathbf{u}^{\text{ext}}$ is imposed on Γ_u and the elevation η^{ext} is imposed on Γ_η. On $\partial\Omega \backslash \Gamma_\eta$, η^{ext} is set to η_h. Similarly, $\mathbf{u}^{\text{ext}}$ is set to $\mathbf{u}_h$ on $\partial\Omega \backslash (\Gamma_u \cup \Gamma_{\text{freeslip}})$.

2.6.2.2. *Mixed DG-CG Pair.* The mixed continuous-discontinuous $\mathbb{P}1_{DG} - \mathbb{P}2$ finite element pair was first introduced for shallow water modelling in [**Cotter et al., 2009**]. That is, the velocity is approximated with piecewise linear and discontinuous elements and the elevation is approximated with piecewise quadratic and continuous elements. It is demonstrated in [**Cotter et al., 2009**] that this finite element pair does not have spurious modes and has 'excellent geostrophic balance properties' when applied to the linear shallow water equations.

We treat each term, summing the contributions from (2.47)–(2.49) over all mesh elements.

The advection term (2.47) includes an integral on ∂K, which contributes boundary terms for edges on $\partial\Omega$ and flux terms elsewhere. The boundary term is used to enforce boundary conditions for the velocity:

$$\begin{aligned} \rho_{\text{adv}}(\mathbf{q}_h, \boldsymbol{\xi}) :=& -\int_\Omega (\nabla \cdot (\mathbf{u}_h \boldsymbol{\phi}^T)) \mathbf{u}_h\, \mathrm{d}x + \int_{\Gamma_{\text{int}}} [\![\boldsymbol{\phi} (\mathbf{u}_h \cdot \widehat{\mathbf{n}})]\!] \cdot \{\!\{ \mathbf{u}_h \}\!\}\, \mathrm{d}S \\ &+ \int_{\partial\Omega} (\boldsymbol{\phi} \cdot \frac{1}{2}(\mathbf{u}_h + \mathbf{u}^{\text{ext}})) \mathbf{u}^{\text{rie}} \cdot \widehat{\mathbf{n}}\, \mathrm{d}s, \end{aligned} \tag{2.51}$$

where $\boldsymbol{\xi} = (\boldsymbol{\phi}, \zeta) \in V_h$. Note that a design choice has been made in interpreting the flux term using the jump and average shown.

Whilst the pressure gradient term is not integrated by parts, a term is included to enforce boundary conditions for the free surface elevation:

$$\rho_{\text{pg}}(\mathbf{q}_h, \boldsymbol{\xi}) := \int_\Omega g\boldsymbol{\phi} \cdot \nabla\eta_h \,\mathrm{d}x + \int_{\partial\Omega} g(\eta^{\text{rie}} - \eta_h)\boldsymbol{\phi} \cdot \widehat{\mathbf{n}} \,\mathrm{d}s. \tag{2.52}$$

Since the Coriolis and drag terms do not involve integration by parts, set $\rho_{\text{cor}}(\mathbf{q}_h, \boldsymbol{\xi})$ and $\rho_{\text{drg}}(\mathbf{q}_h, \boldsymbol{\xi})$ equal to the corresponding terms in (2.46), but with $\mathbf{u}$ replaced with $\mathbf{u}_h$.

The integration by parts of viscosity term in (2.48) introduces both a flux and a boundary term:

$$\begin{aligned}\rho_{\text{vis}}(\mathbf{q}_h, \boldsymbol{\xi}) := &\int_\Omega \nabla\boldsymbol{\phi} : \underline{\boldsymbol{\nu}}\nabla\mathbf{u}_h \,\mathrm{d}x - \int_{\Gamma_{\text{int}}} [\![\boldsymbol{\phi}\widehat{\mathbf{n}}^T]\!] : \{\!\{\underline{\boldsymbol{\nu}}\nabla\mathbf{u}_h\}\!\} \,\mathrm{d}S \\ &- \int_{\partial\Omega} \boldsymbol{\phi}\widehat{\mathbf{n}}^T : \underline{\boldsymbol{\nu}}\nabla\mathbf{u}_h \,\mathrm{d}s.\end{aligned} \tag{2.53}$$

Again, a design choice has been made when deriving the flux term.

Since the elevation space is continuous, the integration by parts of the continuity term in (2.49) does not accumulate any flux terms, only a boundary term:

$$\rho_{\text{cty}}(\mathbf{q}_h, \boldsymbol{\xi}) := -\int_\Omega \nabla\zeta \cdot ((\eta_h + b)\mathbf{u}_h) \,\mathrm{d}x + \int_{\partial\Omega} \zeta(\eta^{\text{rie}} + b)\mathbf{u}^{\text{rie}} \cdot \widehat{\mathbf{n}} \,\mathrm{d}s. \tag{2.54}$$

Gathering together the terms defined above, we arrive at a weak formulation of the steady-state nonlinear shallow water equations in V_h, given by:

$$\begin{aligned}\rho(\mathbf{q}_h, \boldsymbol{\xi}) = 0, \quad \forall\boldsymbol{\xi} \in V_h, \qquad \rho(\mathbf{q}_h, \boldsymbol{\xi}) :=& \rho_{\text{adv}}(\mathbf{q}_h, \boldsymbol{\xi}) + \rho_{\text{pg}}(\mathbf{q}_h, \boldsymbol{\xi}) + \rho_{\text{cor}}(\mathbf{q}_h, \boldsymbol{\xi}) \\ &+ \rho_{\text{vis}}(\mathbf{q}_h, \boldsymbol{\xi}) + \rho_{\text{drg}}(\mathbf{q}_h, \boldsymbol{\xi}) + \rho_{\text{cty}}(\mathbf{q}_h, \boldsymbol{\xi}).\end{aligned} \tag{2.55}$$

Solution Strategy. For steady-state problems, a shallow water solution is sought by applying a Newton method to (2.55), as described in Subsection 2.4.9. The resulting linear systems could be solved using a direct method such as preconditioning with a full LU decomposition, for example.

In the time-dependent case, the nonlinear system is linearised about the initial condition in the first timestep and about the current approximate solution in all timesteps thereafter. The time-dependent system may be decomposed as a 2×2 block system at each time level, where the first block corresponds to the velocity space and the second to the elevation. The resulting linear systems may be solved using GMRES with a multiplicative 'fieldsplit' preconditioner, i.e. block Gauss-Seidel. For details on this preconditioning strategy, see p.294 of [**Wesseling, 2009**] on 'distributive iterations'.

2.6.2.3. *Equal Order DG-DG Pair.* Another discretisation used for shallow water modelling in Thetis is the equal order $\mathbb{P}1_{DG} - \mathbb{P}1_{DG}$ pair. This is very similar to the above, except that there is an additional flux contribution,

$$\rho_{\text{cty}}^{\text{DG-DG}}(\mathbf{q}_h, \boldsymbol{\xi}) := \rho_{\text{cty}}(\mathbf{q}_h, \boldsymbol{\xi}) + \int_{\Gamma_{\text{int}}} [\![\zeta\widehat{\mathbf{n}}]\!]\{\!\{H\}\!\}\mathbf{u}^\star \,\mathrm{d}S, \tag{2.56}$$

for the continuity term and the pressure gradient term takes the form

$$\rho_{\text{pg}}^{\text{DG-DG}}(\mathbf{q}_h, \boldsymbol{\xi}) := -\int_\Omega g\eta_h\nabla\cdot\boldsymbol{\phi} \,\mathrm{d}x + \int_{\partial\Omega} g\eta^{\text{rie}}\boldsymbol{\phi} \cdot \widehat{\mathbf{n}} \,\mathrm{d}s + \int_{\Gamma_{\text{int}}} g\eta^\star\boldsymbol{\phi} \cdot \widehat{\mathbf{n}} \,\mathrm{d}S, \tag{2.57}$$

where

$$\mathbf{u}^\star := \{\!\{\mathbf{u}_h\}\!\} + \sqrt{\frac{g}{\{\!\{H\}\!\}}}[\![\eta_h \widehat{\mathbf{n}}]\!], \qquad \eta^\star := \{\!\{\eta_h\}\!\} + \sqrt{\frac{\{\!\{H\}\!\}}{g}}[\![\mathbf{u}_h \cdot \widehat{\mathbf{n}}]\!] \tag{2.58}$$

are Riemann solutions for the inter-element interfaces.

2.7. Stabilisation

2.7.1. Continuous Galerkin. Whether the advection-diffusion equation (2.12) is dominated by advective or diffusive processes can be determined by the magnitude of the Péclet number,

$$\mathrm{Pe} = \frac{\text{advective transport rate}}{\text{mass diffusion rate}} = \frac{LU}{D}. \tag{2.59}$$

Here L is a characteristic length scale, U is the local fluid velocity and D is the diffusion coefficient. If $\mathrm{Pe} \gg 1$ then advection dominates and if $\mathrm{Pe} \ll 1$ then diffusion dominates. We are primarily interested with the former, advection-dominated, case.

Recall the CG weak formulation of the tracer transport problem and consider the case of lowest polynomial degree, $p = 1$. It is illustrated in [**Donea and Huerta, 2003**] in the 1D case that this discretisation choice introduces truncation error which has the effect of *negative diffusion*. This causes instability for advection-dominated problems, leading to spurious oscillations in the tracer concentration solution, with the potential consequence of severely reduced approximation accuracy. In the context of $\mathbb{P}1$ discretisations for advection-dominated tracer transport problems, it is common practice to apply *stabilisation* schemes, which seek to counteract the negative diffusion.

Streamline Upwind. It is shown in [**Donea and Huerta, 2003**] that a 1D upwind derivative approximation has the effect of inducing *positive* diffusion. For diffusion-dominated problems, upwinding tends to lead to an overly diffusive scheme. However, it can be a useful artefact when solving advection-dominated problems. *Streamline upwind (SU)* is a simple stabilisation approach which attempts to negate under- and overshoots by adding artificial diffusion via upwinding. This is achieved by modifying the test function in the advection term of the weak formulation:

$$\left\langle \underline{\phi + \tau \mathbf{u}\cdot\nabla\phi}, \mathbf{u}\cdot\nabla c_h \right\rangle + \langle \nabla\phi, \underline{\boldsymbol{D}}\nabla c_h\rangle - \langle \phi, \underline{\boldsymbol{D}}\nabla c_h \cdot \widehat{\mathbf{n}}\rangle_{\partial\Omega\setminus\partial\Omega_N} = \langle \phi, S\rangle\,, \quad \forall \phi \in V_h, \tag{2.60}$$

where the stabilisation parameter $\tau > 0$ is to be specified.

Whilst SU stabilisation is able to effectively stabilise an advection-diffusion equation, it is known to be overly diffusive, tending to smooth out discontinuous features. In addition, exact solutions are generally inconsistent for this type of stabilisation. That is, an exact solution of (2.12) is not a weak solution in general. As such, we cannot expect weak solutions to converge to the exact solution on a sequence of increasingly refined meshes.

Streamline Upwind Petrov-Galerkin. The *SUPG* approach is similar to SU, except that it ensures exact solutions of the advection-diffusion equation (2.12) are also weak solutions. The approach can be interpreted as using a test space, V_h, which differs from the trial space, $W_h \ni c_h$. It reads as in the unstabilised case, but with $V_h := \{\psi + \tau\mathbf{u}\cdot\nabla\psi \mid \psi \in W_h\}$. Alternatively, SUPG stabilisation may be interpreted as the unstabilised formulation with

an additional term,

$$\langle \phi, S - \mathbf{u} \cdot \nabla c_h + \nabla \cdot (\underline{\boldsymbol{D}} \nabla c_h \rangle \,. \tag{2.61}$$

On the numerical linear algebra level, SUPG works because it restores diagonal dominance (and hence positive-definiteness), which is lost in the Galerkin approximation at high Péclet numbers [**DeBlois, 1996**].

2.7.2. Discontinuous Galerkin. DG methods can be interpreted as having some in-built stabilisation. If the flux terms have been chosen appropriately, then they can go some way act to balance truncation errors due to the discretisation. However, it is often beneficial to apply additional stabilisation for particular terms.

Lax-Friedrichs. As with the SU stabilisation approach for CG methods, *Lax-Friedrichs* stabilisation is applied to the advection term in a DG discretisation. The use of Lax-Friedrichs schemes in an advection term avoids having to solve Riemann problems on inter-element facets. As a trade-off, it introduces artificial viscosity/diffusion and therefore has a stabilising effect.

Applied to the advection term (2.51) of the shallow water equations, Lax-Friedrichs stabilisation amounts to the modification

$$\rho_{\text{adv}}^{\text{LF}}(\mathbf{q}_h, \boldsymbol{\xi}) := \rho_{\text{adv}}(\mathbf{q}_h, \boldsymbol{\xi}) + \int_{\Gamma_{\text{int}}} \beta [\![\boldsymbol{\phi}]\!] \cdot [\![\mathbf{u}_h]\!] \,\mathrm{d}S + \int_{\Gamma_{\text{freeslip}}} \beta \boldsymbol{\phi} \cdot (\mathbf{u}_h - \mathbf{u}^{\text{ext}}) \,\mathrm{d}s, \tag{2.62}$$

where the Lax-Friedrichs parameter is given by $\beta = \frac{1}{2}|\{\!\{\mathbf{u}_h\}\!\} \cdot \widehat{\mathbf{n}}^+|$. Note that on the boundary the average collapses to the interior value. The positive sign of the normal vector is chosen arbitrarily, but the modulus sign means that it does not actually matter.

Symmetric Interior Penalty Galerkin. Stabilisation may be applied to diffusion or viscosity terms, as well as advection terms. *Interior penalty Galerkin* methods are a DG schemes for elliptic operators which enforce continuity and boundary conditions via penalties. Following the integration by parts in (2.53), additional integrals are added in the viscosity term. The purpose is to penalise the viscosity term sufficiently so that the associated bilinear form is coercive. In return for the penalty, desirable stability and convergence properties are attained. The interior penalty method used for the discretisation of the shallow water equations presented here involves a symmetrisation term and is therefore labelled *symmetric interior penalty Galerkin (SIPG).*

Under the SIPG method described in [**Epshteyn and Rivière, 2007**], the viscosity term (2.53) of the shallow water equations becomes

$$\begin{aligned}
\rho_{\text{vis}}^{\text{SIPG}}(\mathbf{q}_h, \boldsymbol{\xi}) &:= \rho_{\text{vis}}(\mathbf{q}_h, \boldsymbol{\phi}) \\
&- \int_{\Gamma_{\text{int}}} \{\!\{\nabla \boldsymbol{\phi}\}\!\} : \{\!\{\underline{\boldsymbol{\nu}}\}\!\} [\![\mathbf{u}_h \widehat{\mathbf{n}}^T]\!] \,\mathrm{d}S - \int_{\partial\Omega} \nabla \boldsymbol{\phi} : (\underline{\boldsymbol{\nu}}(\mathbf{u}_h - \mathbf{u}^{\text{ext}})) \widehat{\mathbf{n}}^T \,\mathrm{d}s \\
&+ \int_{\Gamma_{\text{int}}} \sigma [\![\boldsymbol{\phi} \widehat{\mathbf{n}}^T]\!] : \{\!\{\underline{\boldsymbol{\nu}}\}\!\} [\![\mathbf{u}_h \widehat{\mathbf{n}}^T]\!] \,\mathrm{d}S + \int_{\partial\Omega} \sigma (\boldsymbol{\phi} \widehat{\mathbf{n}}^T) \cdot (\underline{\boldsymbol{\nu}}(\mathbf{u}_h - \mathbf{u}^{\text{ext}}) \widehat{\mathbf{n}}^T) \,\mathrm{d}s.
\end{aligned} \tag{2.63}$$

The parameter σ is a scalar field, which takes the value $\sigma|_K = \alpha_K / h_K$ on element K, where α_K is the so-called *SIPG parameter*. An effective choice of SIPG parameter for anisotropic meshes is specified in Subsection 2.7.3. Whilst presented for the viscosity term of the shallow water equations here, the SIPG formulation is also valid for DG discretisations of the diffusion term in the advection-diffusion equation.

See [**Epshteyn and Rivière, 2007**] for further details on the SIPG method used.

2.7.3. Stabilisation in the Presence of Anisotropy. In Subsections 2.7.1–2.7.2, four stabilisation methods have been introduced, but values for the associated stabilisation parameters have not been suggested for SU, SUPG or SIPG. In the context of anisotropic mesh adaptation, it is important to take account of the anisotropic nature of the underlying meshes when choosing these parameters.

First, however, the term *anisotropy* should be more clearly defined. Consider a triangular element, K, with angles and sides labelled as in Figure 2.2. The circumscribed circle (red) and inscribed circle (blue) for such a triangle are also shown. These circles have radii termed the *circumradius* and *inradius*, respectively.

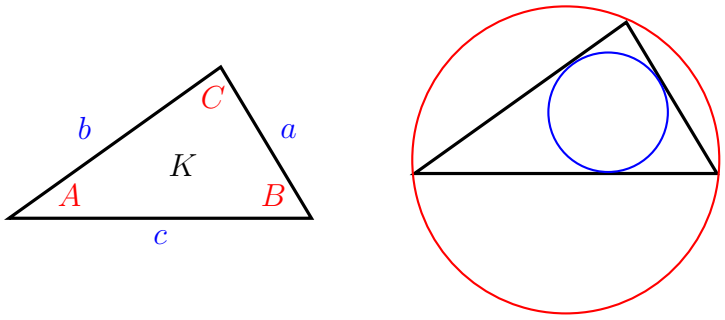


FIGURE 2.2. A general triangle and its circumcircle and incircle.

The *aspect ratio* of a triangle is defined to be the ratio of its circumradius to twice its inradius. It can be expressed using the formula [**Johnson, 1929**]

$$\mathrm{AR}(K) := \frac{abc}{(a+b-c)(b+c-a)(c+a-b)}. \tag{2.64}$$

For an equilateral triangle, the aspect ratio is unity. This is 'true isotropy'. Triangles are still held to be isotropic if their aspect ratio is close to unity. For example, the isosceles right-angled triangular elements in a uniform mesh have $\mathrm{AR}(K) = (1+\sqrt{2})/2 \approx 1.2071$. *Anisotropic* triangles are defined as having $\mathrm{AR}(K) \gg 1$. In this thesis, we say that K is *isotropic* if $\mathrm{AR}(K) < 2$, *moderately anisotropic* if $2 \leq \mathrm{AR}(K) < 20$ and *anisotropic* otherwise.

In the diagram, A is the smallest angle. If this angle is decreased, the circumradius increases and the inradius decreases, meaning that the aspect ratio grows. As such, a small minimum angle is another indicator of anisotropy.

SU and SUPG. The Péclet number defined in (2.59) is related to the advection-diffusion equation itself. It is to be distinguished from the *mesh Péclet number*,

$$\mathrm{Pe}_h(K) := \frac{h_K U}{2D}, \qquad K \in \mathcal{H}, \tag{2.65}$$

which is associated with the discretisation thereof. A standard choice of stabilisation parameter for SU and SUPG is dependent on the mesh Péclet number, given by [**Brooks and Hughes, 1982**]

$$\tau(K) := \min\left(\frac{h_K}{2U}, \frac{\mathrm{Pe}_h(K)}{3}\right). \tag{2.66}$$

The element size measure, h_K, is typically chosen as the circumradius, i.e. the radius of the circumcircle. Whilst suitable for isotropic meshes, this choice is unsuitable for highly anisotropic meshes, because it introduces unnecessary artificial diffusion in cross-stream directions.

An alternative stabilisation parameter for SUPG methods defined on anisotropic meshes is presented in [Micheletti et al., 2003]. Recall the affine map $\mathbf{T}_K : \widehat{K} \to K$ from the reference element to an arbitrary mesh element and its gradient, the cell Jacobian $\underline{\boldsymbol{J}}_K$. Anisotropy is encoded into the cell Jacobian, since it describes how the (isotropic) reference element is stretched when it is transformed into element K. By affinity, $\mathbf{T}_K$ is invertible, meaning $\underline{\boldsymbol{J}}_K$ is, too. As such, $\underline{\boldsymbol{J}}_K$ admits a polar decomposition,

$$\underline{\boldsymbol{J}}_K = \underline{\boldsymbol{B}}_K \underline{\boldsymbol{Z}}_K = \underline{\boldsymbol{V}}_K^T \underline{\boldsymbol{\Lambda}}_K \underline{\boldsymbol{V}}_K \underline{\boldsymbol{Z}}_K \tag{2.67}$$

where $\underline{\boldsymbol{B}}_K$ is SPD and $\underline{\boldsymbol{Z}}_K$ is orthonormal. The orthonormal matrix describes how the reference element is rotated and reflected, whilst the SPD matrix describes how its shape is distorted. Further, since $\underline{\boldsymbol{B}}_K$ is symmetric, it admits an orthogonal eigendecomposition, with eigenvector and eigenvalue matrices $\underline{\boldsymbol{V}}_K$ and $\underline{\boldsymbol{\Lambda}}_K$. The eigenvalues describe how the element is distorted in the directions prescribed by its eigenvectors. More precisely, the eigenvectors and eigenvalues relate to the ellipsoid obtained by transforming the unit n-sphere under $\mathbf{T}_K$. Figure 2.3 illustrates the 2D case, assuming that the reference element is an equilateral triangle with edge length $\sqrt{3}$, centred at the origin.

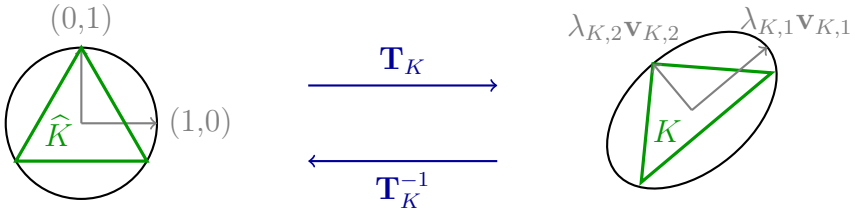
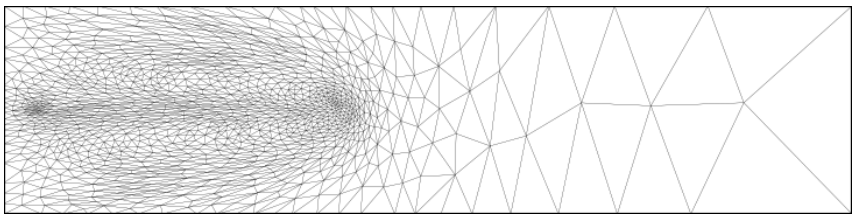


FIGURE 2.3. Affine transformation from a triangular reference element to an arbitrary triangular element.

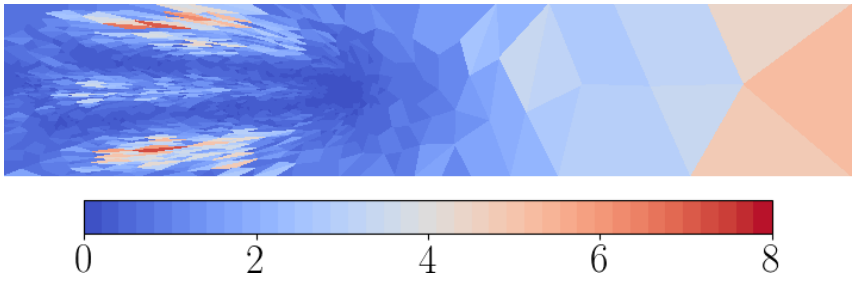
The approach of [Micheletti et al., 2003] is to measure cell size by taking the minimum eigenvalue in $\underline{\boldsymbol{\Lambda}}_K$. By consequence, long, thin elements are interpreted as having a small element size.

Subfigures 2.4b and 2.4c present stabilisation parameter fields using formula (2.66) for the two cell size measures described above. The problem setup in question is the 'Point Discharge with Diffusion' test case from [Riadh et al., 2014], which has constant velocity and diffusivity. This means that the stabilisation parameter is simply a function of the cell size measure. The test case is described in detail in Subsection 3.6.1. Consider solving the tracer transport problem on the moderately anisotropic mesh shown in Subfigure 2.4a using the different stabilisation methods. The resulting linear system with GMRES and SOR as a preconditioner (the default for Thetis' tracer transport model).

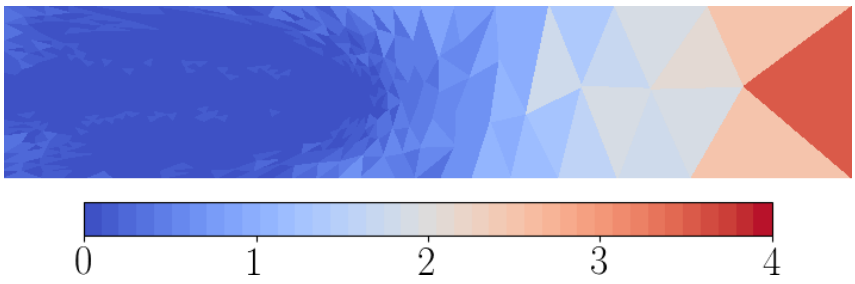
The solution field is losslessly transferred into a four-times uniformly refined space in order to compute relative L^2 errors against the analytical solution, which are displayed in Table 2.3. The linear solver fails to converge without stabilisation. For the mesh presented here, use of the stabilisation parameter designed for anisotropic meshes yields smaller errors than for the standard isotropic version for both SU and SUPG. Moreover, for a given choice of stabilisation parameter, SUPG yields a slightly smaller error than



(A) Moderately anisotropic mesh (max. aspect ratio 12.0)



(B) Isotropic SU/SUPG parameter



(C) Anisotropic SU/SUPG parameter

FIGURE 2.4. Stabilisation parameters for the 'Point Discharge with Diffusion' test case from [**Riadh et al., 2014**] in the presence of mesh anisotropy.

SU. Consequently, we solve tracer transport problems using SUPG with the anisotropic stabilisation parameter henceforth.

Stabilisation	None	SU (iso.)	SU (aniso.)	SUPG (iso.)	SUPG (aniso.)
Relative L^2 error	-	7.318%	6.6353%	7.2822%	6.6289%

TABLE 2.3. Relative L^2 errors (to four decimal places) for the 'Point Discharge with Diffusion' test case under different stabilisation methods on a moderately anisotropic mesh. The abbreviations 'iso.' and 'aniso.' refer to the isotropic and anisotropic stabilisation parameters, respectively.

Comparison of the plots in Subfigures 2.4b–2.4c confirms that strongly anisotropic elements often have large element diameters, but relatively small cell sizes as determined by the minimum eigenvalue.

SIPG. As with SU and SUPG, care should be taken when choosing the SIPG parameter, α_K, in the context of anisotropic elements. The authors of [**Epshteyn and Rivière, 2007**] propose that this parameter be set to

$$\alpha_K = \frac{1}{2}\, p\,(p+1)\frac{\max_K(\nu)}{\min_K(\nu)}\cot\theta_K, \qquad K \in \mathcal{H}, \tag{2.68}$$

on a triangular element K with smallest angle θ_K, where p is the polynomial degree of the velocity space and the max/min term conveys the maximum viscosity ratio (within the element). Formula (2.68) can be applied element-wise, giving rise to a $\mathbb{P}0$ field. However, for the results of [**Wallwork et al., 2020b**] (presented in Subsection 7.5.4), we found the minimum angle over the whole mesh to be sufficient for moderately anisotropic meshes.

2.8. Interpolation

Given that one of the main focuses of this thesis is on mesh adaptation, we require methods for transferring solution fields between meshes. Three approaches are considered, which are useful in different situations.

2.8.1. Hierarchical Case. First, consider the case where we have a mesh $\mathcal{H}$ and a mesh $\mathcal{H}^+$ generated by uniform refinement. That is, for each element $K \in \mathcal{H}$, there exists a unique patch of elements $\chi^+(K) \subset \mathcal{H}^+$ whose union is again element K. Define finite element spaces of the same family and degree, V and V^+, on the two meshes.

Transferring $f \in V$ into V^+ can often be achieved losslessly using a *prolongation* operator, depending on the function space. Since it is true for the CG and DG spaces used in this thesis, we can assume this to be the case. For such spaces, the prolonged field, $f^+ \in V^+$, takes the same values at the nodes and interpolates the values inbetween. Suppose we have $v \in V$ and seek to evaluate the general integral

$$\int (f - f^+)v = \int fv - \int f^+v. \tag{2.69}$$

Integrals of this type need to be evaluated in goal-oriented error estimation, for example. Both f and v live in V, so the first integral on the RHS of (2.69) is trivial to evaluate. Since $V \subset V^+$, we may express v using basis functions $\{\phi_i^+\}_{i=1}^m$ of the enriched space as $v = \sum_i \alpha_i^+ \phi_i^+$, for some coefficients $\{\alpha_i^+\}_{i=1}^m \subset \mathbb{R}$. Thus, the second integral on the RHS of (2.69) may be re-expressed as

$$\int f^+v = \sum_{i=1}^m \alpha_i^+ \int f^+\phi_i^+, \tag{2.70}$$

where $m > n$. Observe that (2.70) is exactly the prolongation operator applied to v.

If v is replaced with the basis functions $\{\phi_j\}_{i=1}^n$ of V in (2.70), we obtain

$$\int f^+\phi_j = \sum_{i=1}^m \alpha_{ij}^+ \int f^+\phi_i^+, \tag{2.71}$$

where $\underline{\boldsymbol{A}} = (\alpha_{ij})$ is a 'tall and skinny' $n \times m$ matrix.

The uniform refinement method used in this thesis is iso-$\mathbb{P}2$ refinement, which amounts to inserting vertices wherever there would be a degree of freedom in a quadratic element. On triangles and tetrahedra, this refinement method has $m = 4n$ and $m = 8n$, respectively.

Transferring information from V^+ to V, on the other hand, cannot be done losslessly. That is, information will generally be lost during this process. One option is *injection*. For CG spaces, this retains the values at nodes shared by both V and V^+ and simply ignores the other values. Another option is to apply one of the many interpolation schemes

designed with no assumption of a mesh hierarchy. Two such schemes are described in the following subsections.

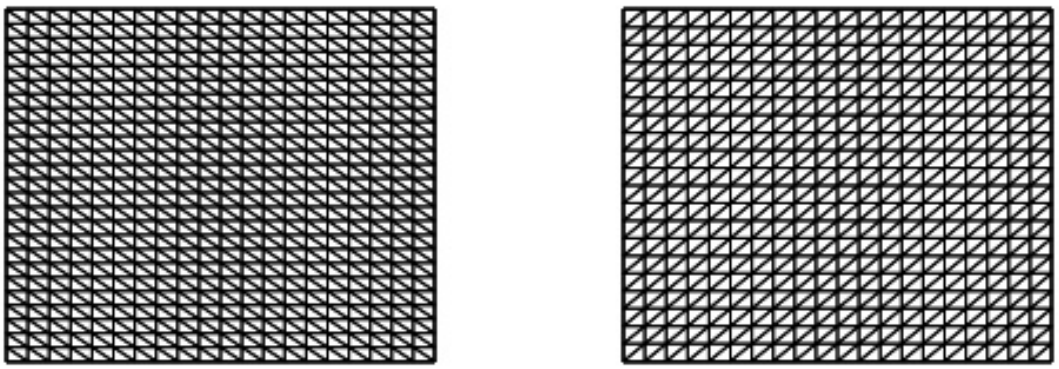
2.8.2. Linear Interpolation. Now consider two triangular and tetrahedral meshes $\mathcal{H}_A$ and $\mathcal{H}_B$ of the same spatial domain Ω, along with function spaces V_A and V_B of the same family and degree defined upon them. Suppose V_A and V_B have bases $\{\phi_i^A\}_{i=1}^m$ and $\{\phi_i^B\}_{i=1}^n$. For simplicity, assume V_A and V_B are $\mathbb{P}1$ spaces, meaning that their degrees of freedom are values at vertices, $\{\mathbf{x}_i^A\}_{i=1}^m$ and $\{\mathbf{x}_i^B\}_{i=1}^n$.

Linear interpolation of a source function $f_A \in V_A$ may be expressed as the action of a $n \times m$ matrix $\underline{\boldsymbol{\Lambda_{AB}}}$ on its vector representation, $\mathbf{f_A}$, where

$$(\underline{\boldsymbol{\Lambda_{AB}}})_{ij} = \phi_j^A(\mathbf{x}_i^B). \tag{2.72}$$

That is, the j^{th} column is obtained by evaluating the j^{th} basis function of V_A at each of the nodes of V_B. This interpolation scheme can also be applied in a matrix-free manner, by simply evaluating f_A at each node of V_B.

For testing purposes, suppose $\mathcal{H}_A$ and $\mathcal{H}_B$ are both uniform triangular meshes of the unit square. Suppose the former is laid out 20×25 with diagonals from top left to bottom right and the latter 20×20 with diagonals from bottom left to top right. The meshes are shown in Figure 2.5.

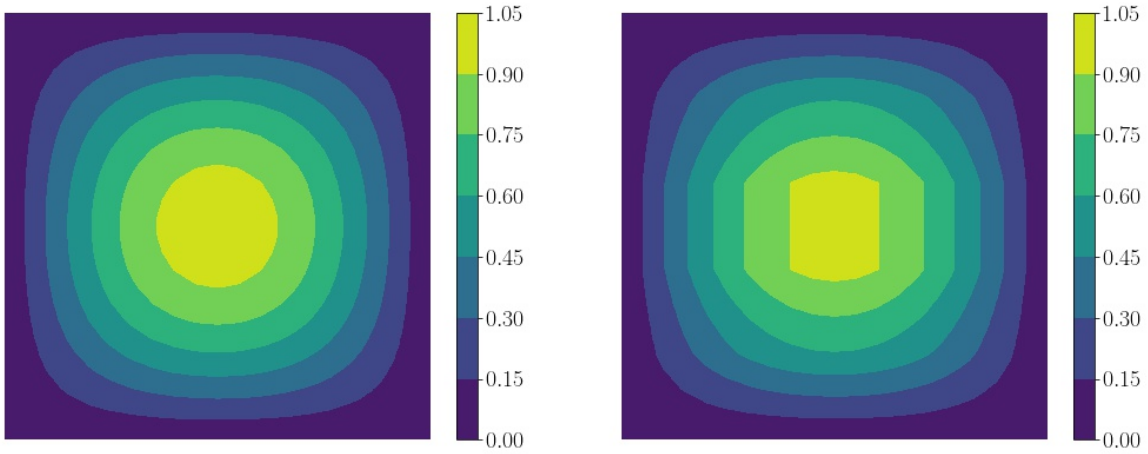


(A) Source mesh (B) Target mesh

FIGURE 2.5. Meshes $\mathcal{H}_A$ and $\mathcal{H}_B$ used in the 'ping pong test'.

Consider projecting a $\mathbb{P}1$ representation of the source function, $f_A(x, y) = \sin(\pi x)\sin(\pi y)$, plotted on mesh $\mathcal{H}_A$ in Subfigure 2.6a.

We perform a 'ping pong test', which amounts to interpolating f_A from $\mathcal{H}_A$ to $\mathcal{H}_B$ and then back again 100 times. The resulting field is shown in Subfigure 2.6b. Observe that the shape is slightly distorted. However, new extrema are not introduced. That this is a general property of linear interpolation follows immediately from the matrix-free interpretation.



(A) Source function (B) After 100 interpolations

FIGURE 2.6. Source function defined in a $\mathbb{P}1$ space on $\mathcal{H}_A$ and its image after 100 linear interpolations to $\mathcal{H}_B$ and back.

An advantage of performing a ping pong test is that the source and result are defined on the same mesh, meaning we are able to compute L^p errors. In this case, the relative L^2 error turns out to be 3.46%.

2.8.3. Conservative Interpolation. In some cases, it is advantageous to have a interpolation operator $\Pi_{AB} : V_A \to V_B$ which is *conservative*. That is,

$$\int_\Omega f_A \,\mathrm{d}x = \int_\Omega \Pi_{AB}(f_A) \,\mathrm{d}x, \qquad f_A \in V_A. \tag{2.73}$$

This is done by solving the optimisation problem,

$$\int_\Omega \Pi_{AB}(f_A) := \operatorname*{argmin}_{v \in V_B} \|f_A - v\|_{L^2(\Omega)}. \tag{2.74}$$

One approach to obtaining a conservative interpolation scheme is through the *supermesh* construct. A *supermesh* of $\mathcal{H}_A$ and $\mathcal{H}_B$ is any mesh $\mathcal{H}_C$ of the same domain whose vertex set contains the vertex sets of $\mathcal{H}_A$ and $\mathcal{H}_B$ and for which the volume of the intersection of any $K_C \in \mathcal{H}_C$ with any $K \in \mathcal{H} \in \{\mathcal{H}_A, \mathcal{H}_B\}$ is either zero or again the volume of K_C [**Farrell et al., 2009**]. That is to say, all elements of the supermesh are completely contained within individual elements of the 'donor' meshes. In the hierarchical case described in Subsection 2.8.1, the refined mesh $\mathcal{H}^+$ is itself a supermesh of the base mesh.

We are able to prolong functions from either V_A or V_B onto an equivalent function space defined on $\mathcal{H}_C$ losslessy for most finite elements. The same cannot be said for the converse, of course. The supermesh construct provides an intermediary step between the two meshes, through which an interpolation matrix may be constructed. In [**Farrell et al., 2009**], it is shown that supermesh projection amounts to solving for the vector $\mathbf{f_B}$ in the linear system

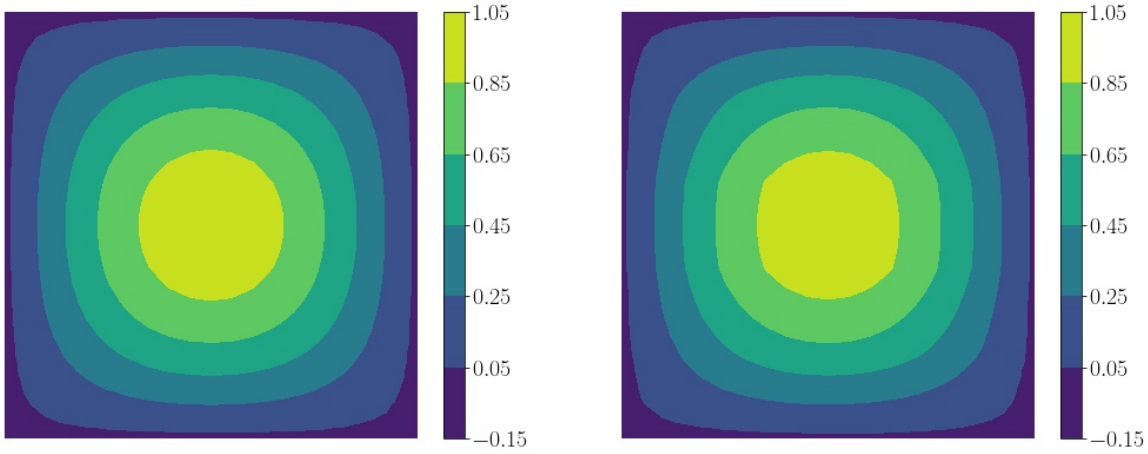
$$\underline{\boldsymbol{M_B}}\,\mathbf{f_B} = \underline{\boldsymbol{M_{BA}}}\,\mathbf{f_A}, \quad \text{where} \quad (\underline{\boldsymbol{M_B}})_{ij} = \int_\Omega \phi_i^B \phi_j^B \,\mathrm{d}x, \quad (\underline{\boldsymbol{M_{BA}}})_{ij} = \int_\Omega \phi_i^B \phi_j^A \,\mathrm{d}x. \tag{2.75}$$

Here $\underline{\boldsymbol{M_B}}$ is the mass matrix for V_B and $\underline{\boldsymbol{M_{BA}}}$ is the *mixed mass matrix* for V_B and V_A.

Let us perform the same ping pong test from Subsection 2.8.2 for the supermesh projector. The source is plotted on V_A again in Subfigure 2.7a and its image after 100 projections to and from V_B is shown in Subfigure 2.7b. Note that the scales are slightly different than previously, to account for new extrema. In this case, the relative L^2 error is 0.86%. The mass error,

$$\epsilon^{\text{mass}} := \left| \frac{\int_\Omega f_A \,\mathrm{d}x - \int_\Omega \Pi_{AB}(f_A) \,\mathrm{d}x}{\int_\Omega f_A \,\mathrm{d}x} \right|, \tag{2.76}$$

is zero, by construction. However, it is 3.18% for linear interpolation.



(A) Source function (B) After 100 interpolations

FIGURE 2.7. Source function defined in a $\mathbb{P}1$ space on $\mathcal{H}_A$ and its image after 100 conservative projections to $\mathcal{H}_B$ and back.

For this test case, supermesh projection yields both lower mass error and L^2 error than linear interpolation. However, a caveat is that the source function used here is rather smooth. Conservative projection is a diffusive process, which has the effect of smoothing out the source field. As such, it cannot be expected to achieve L^2 errors as small as those stated above when applied to noisy or discontinuous data. Another caveat is that new extrema are introduced, where they are not under linear interpolation. For linear finite elements, boundedness of the interpolation operator can be achieved by lumping the mass matrix [**Farrell et al., 2009**]. Lumping introduces additional numerical diffusion. For cases in which this is undesirable, a method which ensures that the interpolation operator is bounded, conservative and minimally diffusive is presented in [**Farrell et al., 2009**].

In practical applications, the interpolation scheme used to transfer data related to a particular field should be chosen with care. If the field is a strictly positive quantity, such as tracer concentration, a bounded interpolation operator is recommended, because it maintains positivity. For quantities such as mass which should be conserved across entire simulations, a conservative interpolation scheme is recommended. Throughout this thesis, the supermesh projection operator is used for transfers between arbitrary (i.e. non-hierarchical) meshes, unless otherwise stated. The software used to do this is the *libsupermesh* package [**Farrell and Maddison, 2011**, **Maddison et al., 2016**].

CHAPTER 3

Adjoint Methods

Motivation. For the purposes of this chapter, suppose that we have some PDE and an associated scalar diagnostic quantity of interest (QoI). Most commonly in this work the PDE will be the shallow water equations or an advection-diffusion equation, with the QoI an integral functional of the PDE solution. Associated with any PDE/QoI pair is an *adjoint equation*, which is itself a linear PDE, along with a set of auxiliary conditions. Together, the adjoint equation and its conditions comprise the *adjoint problem*. Solutions of the adjoint problem (i.e. *adjoint solutions*) may be interpreted as conveying how sensitivities of the QoI w.r.t. the prognostic solution propagate across the domain.

As described in Chapter 1, the adjoint method has proved to be an indispensable tool for computational modelling and optimisation. The literature contains a multitude of uses, including sensitivity analysis, data assimilation and uncertainty quantification, to name just a few.

Adjoint methods are particularly attractive for optimisation problems, because the computational cost associated with computing gradients from the solution of an adjoint equation is effectively independent of the number of control parameters. As such, complex engineering design problems can be solved efficiently using gradient-based optimisation routines which represent the QoI gradient using adjoint methods. An example of such an engineering problem in the coastal ocean modelling context is tidal turbine array design [**Funke et al., 2014**], wherein the layout of a tidal array is configured in order to optimise some QoI, such as power output. Another application is tsunami source inversion; parameters associated with an assumed source model are modified in order to minimise error in the resulting propagation model, compared with real data [**Pires and Miranda, 2001**].

This chapter focuses on the development of adjoint methods themselves, in terms of the (continuous or discrete) formulation and implementation details, such as whether automatic differentiation or checkpointing schemes are used. Applications of adjoint methods are considered in subsequent chapters. Namely, Chapter 4 inverts gauge data for parameters describing a tsunami source and Chapter 7 applies goal-oriented mesh adaptation to a range of geoscience problems.

Novel Contributions. Whilst mostly focused on reviewing adjoint methods and implementation details thereof, this chapter does contain a comparison of continuous and discrete methods applied to an advection-diffusion problem. Although the literature contains many qualitative comparisons of these approaches, there are relatively few examples of direct, quantitative comparisons within the same solver framework. Comparisons are made in terms of the accuracy of computed gradients, computational efficiency and overall effectiveness. The conclusions of this investigation inform implementation choices made in Chapters 4 and 7.

Chapter Organisation. This chapter is organised as follows. Mathematical conventions for the adjoint and tangent linear models associated with a given PDE are introduced in Section 3.1. Section 3.2 describes the gradient-based approach to solving PDE-constrained optimisation problems, which relies on the possession of accurate gradient information. Sections 3.3 and 3.4 describe two different ways to obtain the adjoint problem and solutions thereof – the continuous and discrete adjoint methods. In either case, solution data may be used to compute the gradient information required by the optimisation method. Section 3.5 provides some implementation details specific to the discrete adjoint method. The continuous and discrete methods are compared, both qualitatively and quantitatively, in Section 3.6. Finally, Section 3.7 provides some context on the memory vs. recomputation trade-off associated with memory intensive applications involving adjoint solves.

3.1. Mathematical Conventions

Throughout this chapter, suppose we have a PDE, whose solution w lives in a function space W, which may be expressed in *residual form*,

$$\Psi(w) = 0, \qquad w \in W, \tag{3.1}$$

where $\Psi : W \to W^*$. Here W could be scalar, vector, tensor or a mixed space combining several solution variables and the PDE could be linear or nonlinear. In the context of adjoint modelling, (3.1) is referred to as the *forward equation* and w is termed the *forward solution*. Assume W is defined upon a spatial domain $\Omega \subset \mathbb{R}^d$ with piecewise smooth boundary $\partial\Omega$ and possibly also a time interval $(0, T]$ (in the time-dependent case).

Quantity of Interest. In order to define the adjoint equation, we need to define a QoI,

$$J : W \to \mathbb{R}. \tag{3.2}$$

A number of practical examples are given in Subsection 1.1.2. Many (but not all) of the QoIs considered in this thesis may be expressed as an inner product of the form (2.4.7) with $W = L^2(\Omega)$.

As has been mentioned, one of the major problem classes where the adjoint method is useful is PDE-constrained optimisation. In a problem of this type, the QoI is minimised/maximised, subject to the constraint that the forward problem (3.1) is satisfied. It is to be assumed that the forward problem definition depends on some control parameters, m, which live in a control space C, meaning that both residual and QoI are functions of m, as well as w. Mathematically, this problem may be stated as

$$\min_{w \in W,\, m \in C} J(w, m) \quad \text{subject to} \quad \Psi(w, m) = 0. \tag{3.3}$$

Maximisation problems may be formulated as such by taking the negative QoI.

Under the assumption that $m \in C$ determines a well-posed forward problem with a unique solution, the QoI may be interpreted as a function of m alone. Hence, we introduce the *reduced functional*,

$$\widehat{J} : C \to \mathbb{R}, \qquad \widehat{J}(m) := J(w, m)|_{\Psi(w,m)=0}, \tag{3.4}$$

which implicitly assumes a solution of (3.1) due to control parameters m.

Gradient Computation. One well-established method for solving problems of the form (3.3) is *gradient-based optimisation*. The utilisation of such methods relies on having a representation or approximation of the functional gradient, $\mathrm{d}J_m$.

Recall the definitions of derivatives in infinite dimensions in Subsection 2.4.8 and the chain rule (2.34). Applied to J, we use the shorthand

$$\mathrm{d}J_m = \partial J_u \, \mathrm{d}w_m + \partial J_m. \tag{3.5}$$

The partial derivatives may be computed directly from the QoI expression. However, $\mathrm{d}w_m$ is not directly computable, since w is determined by solving the forward problem.

One approach to obtain the QoI gradient is to compute $\mathrm{d}w_m$ using the *tangent linear model (TLM)*. This system of equations is given by differentiating the forward model w.r.t. the control parameters:

$$\partial \Psi_w \, \mathrm{d}w_m + \partial \Psi_m = 0 \quad \Longrightarrow \quad \mathrm{d}w_m = -\left(\partial \Psi_w\right)^{-1} \partial \Psi_m, \tag{3.6}$$

assuming that $\partial \Psi_w$ is non-singular. This result may also be obtained as a consequence of the Implicit Function Theorem. Solutions of (3.6) describe how sensitivities of the solution variables w.r.t. the control propagate across the domain. As such, equation (3.6) is sometimes referred to as the (first order) *sensitivity system*. In the finite-dimensional case, (3.6) gives rise to a system of $N := \dim(C)$ linear equations. The gradient computation requires solving larger systems of equations as the dimension of the control space is increased, making it infeasibly expensive method for optimising many controls.

Substituting the the right hand equation of (3.6) into (3.5) gives

$$\mathrm{d}J_m = \partial J_m - \underbrace{\partial J_w \, \partial \Psi_w^{-1}}_{(w^*)^T} \partial \Psi_m. \tag{3.7}$$

The indicated definition of w^* gives rise to the *adjoint equation*,

$$\partial \Psi_w^T \, w^* = \partial J_w^T, \qquad w^* \in V. \tag{3.8}$$

Accordingly, w^* is termed the *adjoint solution*. Solving the adjoint equation and then applying formula (3.7) provides an alternative approach for computing $\mathrm{d}J_m$. Unlike the tangent linear model, (3.8) is a single equation in the finite-dimensional case, i.e. its size is independent of N. The cost of its solution typically dominates that of evaluating ∂J_m, $\partial \Psi_m$, $\partial \Psi_w$ and ∂J_w. Hence, the overall cost of the adjoint-based gradient computation is effectively independent of N, giving rise to an efficient method for optimising large numbers of controls.

Continuous vs. Discrete. As detailed in Chapter 2, the first step in seeking the numerical solution of a PDE is to discretise it, using FEM, for example. When it comes to the adjoint equation (3.8), there are two fundamentally different approaches, which differ in the point at which the discretisation is applied.

The first approach is most commonly known as *continuous adjoint*. It is sometimes also referred to as *'differentiate then discretise'* [**Gunzburger, 2002**], since it means first deriving the adjoint equation in a continuous form and then later discretising in order to seek an numerical solution. It has also been referred to as *'optimise then discretise'* [**Carnarius et al., 2011**], since the so-called *optimality system* (defined in Section 3.3) is derived before discretisation.

The second approach is most commonly known as *discrete adjoint*. Similarly to the above, this approach may be referred to as *'discretise then differentiate'* or *'discretise then optimise'*, since the PDE is discretised *before* deriving an associated adjoint equation.

The two are compared in Section 3.6, including qualitative observations, quantitative differences and general advantages and disadvantages. They usually give different results, because differentiation and discretisation do not automatically commute. Both are used in this thesis.

3.2. Gradient-Based Optimisation

Consider again the PDE-constrained optimisation problem (3.3), where we seek to minimise a reduced functional $\widehat{J}$ w.r.t. an N-dimensional control, $m \in C$. Evaluation of the reduced functional involves the numerical solution of some PDE which depends upon the control parameters, but we can ignore this fact for the majority of this section.

Iterative optimisation routines start with an initial guess m_0 for the control parameters and create a sequence which hopefully converges to a minimiser, m^*. Consider an iteration

$$m_{k+1} := m_k + \alpha_k p_k, \tag{3.9}$$

where $\alpha_k > 0$ is the step length used at the k^{th} iteration and p_k is the search direction. As suggested by the name, the search direction depends on the gradient, $\mathrm{d}\widehat{J}_m$, in gradient-based methods.

3.2.1. Steepest Descent and Conjugate Gradients. The simplest gradient-based optimisation routine is *steepest descent*, a.k.a. *gradient descent*, for which

$$p_k := -\mathrm{d}\widehat{J}_m(m_k). \tag{3.10}$$

If $p_k \approx 0$ then we deduce that a stationary point has been reached. Whilst very easy to implement, the steepest descent direction is not always the most effective search direction. The *conjugate gradient (CG)* method modifies (3.10) to ensure that steps are orthogonal, in some sense. Under the assumption that $\widehat{J}$ is a quadratic function whose quadratic term has a symmetric positive-definite (SPD) matrix coefficient A, CG takes steps that are orthogonal in the A-norm and ensures that convergence is attained in a maximum of N iterations [**Teukolsky et al., 1992**]. The reduced functional associated with a PDE often cannot be expressed as a quadratic function of the controls. Nevertheless, CG can be modified in order to be applied to PDE-constrained optimisation problems [**Herzog and Kunisch, 2010**].

3.2.2. Newton's Method. An alternative gradient-based approach to those given above is to use *Newton's method*, which has already been introduced in the context of nonlinear systems in Subsection 2.4.9. Applied to optimisation problems, the search direction of Newton's method is given by the solution of the linear system

$$\mathrm{d}^2\widehat{J}_m(m_k)\, p_k = -\mathrm{d}\widehat{J}_m(m_k), \tag{3.11}$$

where $\mathrm{d}^2\widehat{J}_m$ denotes the Hessian w.r.t. the control parameters.

Provided the initial guess is sufficiently close to the optimum, Newton's method is a very powerful method for PDE-constrained optimisation. Applied to finite dimensional spaces

and an N-dimensional control space, the naïve approach is to assemble the $N \times N$ Hessian. This approach requires the solution of all N equations in the TLM. If the forward PDE is linear then this requires approximately N times the computational cost as the forward model itself! This is clearly infeasible for $N \gg 1$. A computationally cheaper approach is to solve (3.11) matrix-free, which avoids the expensive assembly step.

3.2.3. Quasi-Newton Methods. If solving (3.11) matrix-free is impossible or difficult then another alternative method, which is typically computationally cheaper than assembling the full Hessian, is to approximate it. Doing so gives rise to a family of methods known as *quasi-Newton methods*. Beginning with the work of [**Broyden, 1965**], quasi-Newton methods were developed as a family of nonlinear solvers which build up an approximation to the Hessian (or the inverse thereof) in an iterative fashion, via low rank updates.

In this thesis, the *Broyden-Fletcher-Goldfarb-Shanno (BFGS)* method [**Broyden et al., 1973**] is applied using the SciPy implementation [**Virtanen et al., 2020**]. For the version which approximates the Hessian itself, the BFGS iteration builds up a sequence of iterates $\{B_k\}_{k\in\mathbb{N}} \subset \mathbb{R}^{N\times N}$ as

$$B_{k+1} := B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}, \quad \text{where} \quad \begin{cases} s_k := m_{k+1} - m_k, \\ y_k := \mathrm{d}\widehat{J}_m(m_{k+1}) - \mathrm{d}\widehat{J}_m(m_k) \end{cases}. \tag{3.12}$$

Note that iteration preserves symmetry of the Hessian approximation. Typically, the identity matrix is taken as the initial guess, inducing a symmetric Hessian approximation.

For large problems, storing the (inverse) Hessian approximation can require large amounts of memory. As such, the 'limited memory' L-BFGS method retains only the recent history of updates and the corresponding evaluated gradients, giving an implicit representation of the (inverse) Hessian. For constrained optimisation problems, a modified version is used which accounts for bound constraints: L-BFGS-B [**Zhu et al., 1997**], where the 'B' stands for 'bound'. Again, the SciPy implementation is used in this thesis.

3.2.4. Mesh Dependence. BFGS iteration (3.12) implicitly assumes a Euclidean control space, C. If C is a Hilbert space then it may be written [**Schwedes et al., 2017**]

$$\begin{aligned} B_{k+1}(m,n) := B_k(m,n) &- \frac{\langle \mathcal{R}_C(B_k(s_k)), m\rangle_C \langle \mathcal{R}_C(B_k(s_k)), n\rangle_C}{\langle \mathcal{R}_C(B_k(s_k)), s_k\rangle_C} \\ &+ \frac{\langle \mathcal{R}_C(y_k), m\rangle_C \langle \mathcal{R}_C(y_k), m\rangle_C}{\langle \mathcal{R}_C(y_k), s_k\rangle_C}, \qquad m, n \in C, \end{aligned} \tag{3.13}$$

where $\mathcal{R}_C(\widehat{J})$ is the Riesz representation of $\widehat{J}$. Note that B_k can have one or two arguments, since it may either be understood as mapping $C \times C \to \mathbb{R}$ or $C \to C^*$.

The methods described in Subsections 3.2.1–3.2.3 were designed to solve optimisation problems in Euclidean space. This means that many popular implementations, such as that found in SciPy, hard-code the ℓ^2 inner product, as opposed to the inner product associated with the control space. For PDE-constrained optimisation problems discretised using FEM, controls do not necessarily live in Euclidean space. This is the case when the control is a finite element representation of an initial condition, for example. In the case of BFGS, formula (3.13) suggests $\langle \cdot, \cdot \rangle_C$ as a more appropriate choice than ℓ^2.

In [Schwedes et al., 2017], it is demonstrated that the default ℓ^2 inner product can lead to poor convergence properties if C is a finite element space, whereas L^2 or H^1 are more appropriate, depending on the problem. Moreover, it is demonstrated that the default approach leads to mesh dependence of the optimisation routine, which may be avoided by making a careful choice of inner product. Many of the optimisation experiments performed in this thesis have control spaces $C \subseteq \mathbb{R}^N$, whereby (3.12) and (3.13) coincide, whereby the discussion in this subsection is not of concern and the standard SciPy implementations of BFGS and L-BFGS-B are sufficient. This is not true in the tsunami source inversion experiments considered in Chapter 4, since the initial free surface displacement is a function. The mesh-independent framework described above is not used in this thesis, but future work would benefit from its integration.

3.2.5. Taylor Test. Before passing a gradient computed using the continuous or discrete adjoint approach to an optimisation routine, we should first verify that it does indeed accurately represent the gradient of the (discretised) QoI. The standard method for doing this is to apply a *Taylor test*. This test checks that the second order remainders of a truncated Taylor expansion converge quadratically under the proposed gradient.

For a reduced functional $\widehat{J}$, the first order Taylor remainder is given by

$$T_1(h; m) := \left| \widehat{J}(m + h\,\delta m) - \widehat{J}(m) \right|, \tag{3.14}$$

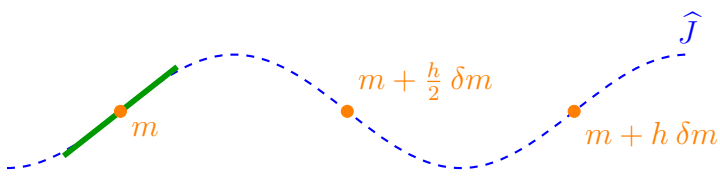
where δm is a search direction and step length $h > 0$. The second order Taylor remainder is given by

$$T_2(h; m) := \left| \widehat{J}(m + h\,\delta m) - \widehat{J}(m) - h\,G \cdot \delta m \right|, \tag{3.15}$$

where G is the proposed representation of $\mathrm{d}\widehat{J}_m$.

Given some $m \in C$, continuity of the reduced functional implies that $\lim_{h\to 0} T_1(h; m) = 0$ at rate $\mathcal{O}(h)$. If G is consistent with the true discrete gradient, then $\lim_{h\to 0} T_2(h; m) = 0$ at rate $\mathcal{O}(h^2)$. The Taylor test verifies that this second order convergence is achieved by halving the value of h and checking that $T_2(h)$ is reduced by a factor of four.

Note that the Taylor test only provides an assessment of gradient consistency at a single point in parameter space. As such, it should be applied repeatedly at multiple points, in order to give greater confidence in the result. In addition, it relies on δm, which determines which directional derivative is taken, as well as the step length, h. If the magnitude of $h\,\delta m$ is too large then the Taylor test can give false negatives. In the one dimensional example given in Figure 3.1a, the Taylor test will claim that the gradient should have converged to zero, whereas the gradient at m is clearly positive. On the other hand, if the step is too small then it is possible to encounter roundoff errors. As such, care should be taken to choose an appropriate, small step size.



(A) Diagram showing how false negatives can arise in a Taylor test. The blue curve represents a reduced functional in a 1D parameter space, the orange dots indicate sampled points and the green line indicates the true gradient.

3.3. Continuous Adjoint

In the following, the continuous adjoint approach is introduced using a Lagrange multiplier approach, with the Lagrangian encapsulating the PDE problem and QoI.

Suppose we have system of time-dependent, scalar PDEs in n dimensions whose solutions reside in a function space W. In addition, suppose we have m_A initial conditions, $\{A_i(w)|_{t=0} = 0\}_{i=1}^{m_A}$, and m_B boundary conditions, $\{B_j(w)|_{\Gamma_j} = 0 \mid \Gamma_j \subset \partial\Omega\}_{j=1}^{m_B}$. Associate with these conditions the Lagrange multipliers $\{\alpha_i\}_{i=1}^{m_A}$ and $\{\beta_j\}_{j=1}^{m_B}$, respectively. With an additional Lagrange multiplier $w^* \in W$, define the Lagrangian

$$\begin{aligned}\mathcal{L}(w, w^*, \alpha_1, \dots, \alpha_{m_A}, \beta_1, \dots, \beta_{m_B}) := J(w) - \langle \Psi(w), u^* \rangle_{\Omega\times(0,T]} \\ - \sum_{i=1}^{m_A} \langle A_i(w), \alpha_i \rangle_{\Omega\times\{0\}} \\ - \sum_{j=1}^{m_B} \langle B_j(w), \beta_j \rangle_{\Gamma_j\times(0,T]} .\end{aligned} \tag{3.16}$$

Observe that the PDE may be recovered from the Lagrangian by setting the first variation w.r.t. w^* to zero and applying the Fundamental Lemma of the Calculus of Variations. That is,

$$\partial\mathcal{L}_{w^*} = 0 \quad \Longrightarrow \quad \Psi(w) = 0. \tag{3.17}$$

Similarly, setting the first variations w.r.t. α_i (resp. β_i) to zero yields the i^{th} initial (resp. boundary) condition. Setting the first variation w.r.t. w to zero and applying the Fundamental Lemma of the Calculus of Variations yields the *continuous adjoint problem*, itself comprised of a PDE with auxiliary conditions. As suggested by the notation, w^* turns out to be the adjoint solution. We denote the adjoint residual by $\Psi^*(\cdot)$, so that

$$\partial\mathcal{L}_w = 0 \quad \Longrightarrow \quad \Psi^*(w^*) = 0. \tag{3.18}$$

For completeness, setting derivatives of the Lagrangian w.r.t. the controls to zero yields the *optimality equation*. The *optimality system* is comprised of this equation, along with (3.17) and (3.18) (see [**Gunzburger, 2002**] for details).

In the following subsections, we derive continuous adjoints for a general advection-diffusion problem and an application-specific shallow water tsunami modelling problem.

3.3.1. Advection-Diffusion Continuous Adjoint. In the notation of Section 2.2 (with $\kappa = 1$), consider the time-dependent advection-diffusion problem

$$\begin{cases} \frac{\partial c}{\partial t} + \mathbf{u}\cdot\nabla c - \nabla\cdot(\underline{\boldsymbol{D}}\nabla c) = S & \text{in } \Omega\times(0,T] \\ c = c_0 & \text{at } t = 0 \\ c = g_D & \text{on } \partial\Omega_D \\ \underline{\boldsymbol{D}}\nabla c\cdot\widehat{\mathbf{n}} = g_N & \text{on } \partial\Omega_N \end{cases}, \tag{3.19}$$

with the boundary decomposed as the disjoint union $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ and $\underline{\boldsymbol{D}}$ satisfying the appropriate compatibility conditions. We allow for an outflow segment $\partial\Omega_{\text{out}} \subset \partial\Omega$, on which natural boundary conditions are to be applied. However, these are equivalent to the Neumann condition in this case, so $\partial\Omega_{\text{out}}$ is absorbed into $\partial\Omega_N$.

For the purposes of illustration, consider a type II QoI which integrates the variation of the tracer concentration against a background value, $c_b > 0$, over a region of interest $R \subset \Omega$ and time period $(T_{\text{start}}, T_{\text{end}}] \subseteq (0, T]$:

$$J(c) := \int_{T_{\text{start}}}^{T_{\text{end}}} \int_R (c - c_b) \, \mathrm{d}x \, \mathrm{d}t = \int_0^T \int_\Omega \mathbb{1}_{R \times [T_{\text{start}}, T_{\text{end}}]} (c - c_b) \, \mathrm{d}x \, \mathrm{d}t. \tag{3.20}$$

In fact, we do not consider the case $T_{\text{end}} < T$, because causality implies that the QoI is independent of dynamics which occur after this time, meaning we may as well redefine $T := T_{\text{end}}$. The QoI is both linear and continuous. As is clear from the second integral in (3.20), it may be represented in the L^2 sense by $\mathcal{R}_{L^2}(J) = \mathbb{1}_{R \times [T_{\text{start}}, T_{\text{end}}]}$, up to a constant.

The strong solution c resides in the function space $W = \{v \in H^2(\Omega) \mid v|_{\partial\Omega_D} = 0\}$. We introduce Lagrange multipliers α, β_D and β_N, corresponding to the initial, Dirichlet and Neumann conditions, as well as the adjoint variable, $c^* \in W$. The Lagrangian is then given by

$$\begin{aligned} \mathcal{L}(c, c^*, \alpha, \beta_D, \beta_N) = &\int_0^T \int_\Omega c^* \left(S - \frac{\partial c}{\partial t} - \mathbf{u} \cdot \nabla c + \nabla \cdot (\underline{\boldsymbol{D}} \nabla c) \right) \mathrm{d}x \, \mathrm{d}t \\ &+ \int_{T_{\text{start}}}^{T_{\text{end}}} \int_R (c - c_b) \, \mathrm{d}x \, \mathrm{d}t + \int_\Omega \alpha (c_0 - c(\mathbf{x}, 0)) \, \mathrm{d}x \\ &+ \int_0^T \int_{\partial\Omega_D} \beta_D (g_D - c) \, \mathrm{d}s \, \mathrm{d}t + \int_0^T \int_{\partial\Omega_N} \beta_N (g_N - \underline{\boldsymbol{D}} \nabla c \cdot \widehat{\mathbf{n}}) \, \mathrm{d}s \, \mathrm{d}t. \end{aligned} \tag{3.21}$$

Setting the first variation w.r.t. c evaluated at any $\widetilde{c} \in V$ equal to zero yields

$$\begin{aligned} 0 = &\int_0^T \int_\Omega \left[c^* \left(-\frac{\partial \widetilde{c}}{\partial t} - \mathbf{u} \cdot \nabla \widetilde{c} + \nabla \cdot (\underline{\boldsymbol{D}} \nabla \widetilde{c}) \right) + \mathbb{1}_{R \times [T_{\text{start}}, T_{\text{end}}]} \widetilde{c} \right] \mathrm{d}x \, \mathrm{d}t \\ &- \int_\Omega \alpha \widetilde{c}(\mathbf{x}, 0) \, \mathrm{d}x - \int_0^T \int_{\partial\Omega_N} \beta_N \underline{\boldsymbol{D}} \nabla \widetilde{c} \cdot \widehat{\mathbf{n}} \, \mathrm{d}s \, \mathrm{d}t, \quad \forall \widetilde{c} \in W, \end{aligned} \tag{3.22}$$

where the Dirichlet term vanishes because $\widetilde{c} \in W$. Applying integration by parts in both space and time yields

$$\begin{aligned} 0 = &\int_0^T \int_\Omega \widetilde{c} \left(\frac{\partial c^*}{\partial t} + \nabla \cdot (\mathbf{u} c^*) + \mathbb{1}_{R \times [T_{\text{start}}, T_{\text{end}}]} \right) \mathrm{d}x \, \mathrm{d}t \\ &- \int_0^T \int_\Omega \nabla \widetilde{c} \cdot \underline{\boldsymbol{D}} \nabla c^* \, \mathrm{d}x \, \mathrm{d}t - \int_0^T \int_\Omega \frac{\partial}{\partial t} (\widetilde{c} \, c^*) \, \mathrm{d}x \, \mathrm{d}t - \int_\Omega \alpha \widetilde{c}(\mathbf{x}, 0) \, \mathrm{d}x \\ &+ \int_0^T \int_{\partial\Omega} (-\widetilde{c} \, c^* \, \mathbf{u} \cdot \widehat{\mathbf{n}} + c^* \underline{\boldsymbol{D}} \nabla \widetilde{c} \cdot \widehat{\mathbf{n}}) \, \mathrm{d}s \, \mathrm{d}t \\ &- \int_0^T \int_{\partial\Omega_N} \beta_N \underline{\boldsymbol{D}} \nabla \widetilde{c} \cdot \widehat{\mathbf{n}} \, \mathrm{d}s \, \mathrm{d}t, \quad \forall \widetilde{c} \in W. \end{aligned} \tag{3.23}$$

Homogeneous Dirichlet conditions are inherited in the adjoint from the forward, since they live in the same function space. Due to this, the initial conditions for the forward

problem and integrating the diffusion term by parts again, we arrive at

$$
\begin{aligned}
0 = &\int_0^T \int_\Omega \widetilde{c} \left(\frac{\partial c^*}{\partial t} + \nabla \cdot (\mathbf{u} c^*) + \nabla \cdot (\underline{\boldsymbol{D}}^T \nabla c^*) + \mathbb{1}_{R \times [T_{\text{start}}, T_{\text{end}}]} \right) \, \mathrm{d}x \, \mathrm{d}t \\
&- \int_\Omega \widetilde{c}(\mathbf{x}, T) \, c^*(\mathbf{x}, T) \, \mathrm{d}x + \int_\Omega \widetilde{c}(\mathbf{x}, 0) \, (c^* - \alpha)|_{t=0} \, \mathrm{d}x \\
&+ \int_0^T \int_{\partial\Omega_N} \widetilde{c} \left(c^* \mathbf{u} \cdot \widehat{\mathbf{n}} + \underline{\boldsymbol{D}}^T \nabla c^* \cdot \widehat{\mathbf{n}} \right) \, \mathrm{d}s \, \mathrm{d}t \\
&+ \int_0^T \int_{\partial\Omega_N} (c^* - \beta_N) \underline{\boldsymbol{D}}^T \nabla \widetilde{c} \cdot \widehat{\mathbf{n}} \, \mathrm{d}s \, \mathrm{d}t.
\end{aligned} \tag{3.24}
$$

Symmetry of $\underline{\boldsymbol{D}}$ implies that the transpose is actually redundant. However, we retain it, for completeness.

Applying the Fundamental Lemma of the Calculus of Variations, (3.24) implies the strong form adjoint equation

$$
\begin{cases}
-\frac{\partial c^*}{\partial t} - \nabla \cdot (\mathbf{u} c^*) - \nabla \cdot (\underline{\boldsymbol{D}}^T \nabla c^*) = \mathbb{1}_{R \times [T_{\text{start}}, T_{\text{end}}]} & \text{in } \Omega \times [0, T) \\
c^* = 0 & \text{at } t = T \\
c^* = 0 & \text{on } \partial\Omega_D \\
\underline{\boldsymbol{D}}^T \nabla c^* \cdot \widehat{\mathbf{n}} + c^* \mathbf{u} \cdot \widehat{\mathbf{n}} = 0 & \text{on } \partial\Omega_N
\end{cases} . \tag{3.25}
$$

Observe that the adjoint equation is a very similar advection-diffusion equation, but with reversed arrow of time, reversed flow velocity and source term given by the Riesz representation. Since information propagates backwards in time for the adjoint equation, we have a *terminal condition* in place of an initial condition. Another difference is that in place of the Neumann condition we have a Robin condition. The Dirichlet condition is inherited on $\partial\Omega_D$.

The adjoint equation may be similarly derived for a QoI which integrates tracer concentration at the end time:

$$
J(c) = \int_R (c(\mathbf{x}, T) - c_b) \, \mathrm{d}x. \tag{3.26}
$$

In this case, the Riesz representation is a pulse at the final time. This means we get an 'initial' value problem which runs backwards in time with this pulse as a starting condition:

$$
\begin{cases}
-\frac{\partial c^*}{\partial t} - \nabla \cdot (\mathbf{u} c^*) - \nabla \cdot (\underline{\boldsymbol{D}}^T \nabla c^*) = 0 & \text{in } \Omega \times [0, T) \\
c^* = \mathbb{1}_R & \text{at } t = T \\
c^* = 0 & \text{on } \partial\Omega_D \\
\underline{\boldsymbol{D}}^T \nabla c^* \cdot \widehat{\mathbf{n}} + c^* \mathbf{u} \cdot \widehat{\mathbf{n}} = 0 & \text{on } \partial\Omega_N
\end{cases} . \tag{3.27}
$$

Let us briefly consider the time-independent case, where the time derivative is dropped from (3.19), the initial condition is ignored and the QoI (3.20) is replaced by

$$
J(c) := \int_R (c - c_b) \, \mathrm{d}x = \int_\Omega \mathbb{1}_R (c - c_b) \, \mathrm{d}x. \tag{3.28}
$$

Again, the Riesz representation, $\mathcal{R}_{L^2}(J) = \mathbb{1}_R$, provides a source term, yielding the system

$$\begin{cases} -\nabla\cdot(\mathbf{u}c^*) - \nabla\cdot(\underline{\boldsymbol{D}}^T\nabla c^*) = \mathbb{1}_R & \text{in } \Omega \\ c^* = 0 & \text{on } \partial\Omega_D \\ \underline{\boldsymbol{D}}^T\nabla c^*\cdot\widehat{\mathbf{n}} + c^*\mathbf{u}\cdot\widehat{\mathbf{n}} = 0 & \text{on } \partial\Omega_N \end{cases} . \tag{3.29}$$

3.3.2. Shallow Water Continuous Adjoint. Next, consider the linearised shallow water system (2.5) – specifically the tsunami source inversion problem, discussed in detail in Chapter 4. Assume the notation from Section 2.1, with zero initial velocity and an initial surface displacement field $\eta_0 \in H^1(\Omega)$ which we seek to invert for. The boundary is split into segments $\partial\Omega_{\text{freeslip}}, \partial\Omega_D \subset \partial\Omega$. In summary, we have the PDE system given by

$$\begin{cases} \frac{\partial\mathbf{u}}{\partial t} + f\widehat{\mathbf{z}}\times\mathbf{u} + g\nabla\eta = \mathbf{0} & \text{in } \Omega\times(0,T] \\ \frac{\partial\eta}{\partial t} + \nabla\cdot(b\mathbf{u}) = 0 & \text{in } \Omega\times(0,T] \\ (\mathbf{u},\eta) = (\mathbf{0},\eta_0) & \text{at } t = 0 \\ \mathbf{u}\cdot\widehat{\mathbf{n}} = 0 & \text{on } \partial\Omega_{\text{freeslip}} \\ \eta = g_D & \text{on } \partial\Omega_D \end{cases} . \tag{3.30}$$

Suppose we have free surface timeseries data from a finite set of gauges $\mathcal{G}$. We have an initial guess for η_0, but would like to optimise it so that the resulting tsunami propagation model best captures the timeseries. Assuming data for all of $(0,T]$, one choice of QoI is the sum of squares

$$J(\mathbf{u},\eta) := \frac{1}{2}\sum_{g\in\mathcal{G}}\int_0^T\int_\Omega \widehat{\mathbb{1}}_g^\epsilon\big(\eta-\eta_g(t)\big)^2\,\mathrm{d}x\,\mathrm{d}t, \tag{3.31}$$

where gauge g has spatial location $\mathbf{x}_g \in \Omega$ and timeseries value $\eta_g(t)$ at time $t \in (0,T]$. Here $\widehat{\mathbb{1}}_g^\epsilon$ is an area-normalised circular indicator function centred at $\mathbf{x}_g$ with small radius $\epsilon > 0$. (For details on the tsunami modelling application, the derivation of (3.31) and the subsequent source inversion experiments, see Chapter 4.) The Riesz representation theorem does not apply because (3.31) is nonlinear.

Establishing a Lagrangian $\mathcal{L}$ for the PDE system (3.30)–(3.31) as before, setting $\partial\mathcal{L}_{\mathbf{q}} = 0$ and integrating by parts, we obtain the continuous adjoint problem,

$$\begin{cases} -\frac{\partial\mathbf{u}^*}{\partial t} + f\widehat{\mathbf{z}}\times\mathbf{u}^* - b\nabla\eta^* = \mathbf{0} & \text{in } \Omega\times[0,T) \\ -\frac{\partial\eta^*}{\partial t} - g\nabla\cdot\mathbf{u}^* = \sum_{g\in\mathcal{G}}\widehat{\mathbb{1}}_g^\epsilon(\eta-\eta_g) & \text{in } \Omega\times[0,T) \\ (\mathbf{u}^*,\eta^*) \equiv \mathbf{0} & \text{at } t = T \\ \mathbf{u}^*\cdot\widehat{\mathbf{n}} \equiv 0 & \text{on } \partial\Omega\backslash\partial\Omega_D \\ \eta^* \equiv 0 & \text{on } \partial\Omega_D \end{cases} , \tag{3.32}$$

after some rearrangement. Note that nonlinearity of the QoI manifests in the RHS of the adjoint continuity equation being dependent upon the forward solution at each time level. This means that numerical solution of (3.32) requires outputs from the numerical solution of (3.30).

The initial surface η_0 is obtained as the output of a source model, $S : C \to H^1(\Omega)$ mapping from a control function space, C. The TLM w.r.t. the control variable, $m \in C$ is given by

$$\begin{cases} \frac{\partial \mathbf{u}'}{\partial t} + f\widehat{\mathbf{z}} \times \mathbf{u}' + g\nabla\eta' = 0 & \text{in } \Omega \times (0,T] \\ \frac{\partial \eta'}{\partial t} + \nabla \cdot (b\mathbf{u}') = 0 & \text{in } \Omega \times (0,T] \\ (\mathbf{u}', \eta') \equiv (\mathbf{0}, \eta_0') & \text{at } t = 0 \\ \mathbf{u}' \cdot \widehat{\mathbf{n}} \equiv 0 & \text{on } \partial\Omega_{\text{freeslip}} \\ \eta' \equiv 0 & \text{on } \partial\Omega_D \end{cases}, \tag{3.33}$$

where $\mathbf{u}' := \partial\mathbf{u}_m$, $\eta' := \partial\eta_m$ and $\eta_0' := \partial(\eta_0)_m$. Differentiating w.r.t. the controls again yields the second order tangent linear model:

$$\begin{cases} \frac{\partial \mathbf{u}''}{\partial t} + f\widehat{\mathbf{z}} \times \mathbf{u}'' + g\nabla\eta'' = 0 & \text{in } \Omega \times (0,T] \\ \frac{\partial \eta''}{\partial t} + \nabla \cdot (b\mathbf{u}'') = 0 & \text{in } \Omega \times (0,T] \\ (\mathbf{u}'', \eta'') \equiv (\mathbf{0}, \eta_0'') & \text{at } t = 0 \\ \mathbf{u}'' \cdot \widehat{\mathbf{n}} \equiv 0 & \text{on } \partial\Omega_{\text{freeslip}} \\ \eta'' \equiv 0 & \text{on } \partial\Omega_D \end{cases}, \tag{3.34}$$

where $\mathbf{u}'' := \partial^2\mathbf{u}_m$, $\eta'' := \partial^2\eta_m$ and $\eta_0'' := \partial^2(\eta_0)_m$. If the source model is linear then $\eta_0'' \equiv 0$, whereby (3.34) has the unique solution $(\mathbf{u}'', \eta'') \equiv \mathbf{0}$. This is due to the linear basis; solutions of the second order TLM are not zero in general.

By the chain rule, we have that

$$\mathrm{d}J_m = \int_0^T \int_\Omega \sum_{g \in \mathcal{G}} \widehat{\mathbb{1}}_g^\epsilon \left(\eta - \eta_g(t)\right) \eta' \,\mathrm{d}x \,\mathrm{d}t. \tag{3.35}$$

Note the appearance of the RHS of the adjoint continuity equation in the integrand. Adding in zero for the RHS of the adjoint momentum equation, we can make a substitution for the LHS of the adjoint equations. Integrating by parts in order to eliminate the adjoint solution variables, we find that almost all of the terms cancel due to the adjoint boundary conditions and the TLM, leaving

$$\mathrm{d}J_m = -\int_\Omega \eta_0'(\mathbf{x}) \eta^*(\mathbf{x}, 0) \,\mathrm{d}x. \tag{3.36}$$

That is, the gradient is given by multiplying the the partial derivative of the source model w.r.t. m and the adjoint free surface at time $t = 0$ and integrating over the spatial domain. This is consistent with formula (3.7).

One method for computing the gradient of the QoI is to solve the forward problem once, solve the adjoint problem once and then assemble (3.36). As has been mentioned, applied to finite-dimensional spaces, this calculation involves the solution of just two PDEs and is completely independent of the number of basis functions – one of the reasons the adjoint method is so powerful.

Similarly as before, the chain rule applied to (3.35) gives

$$\mathrm{d}^2 J_m = \sum_{g \in \mathcal{G}} \int_0^T \int_\Omega \widehat{\mathbb{1}}_g^\epsilon \left[\left(\eta - \eta_g(t)\right) \eta'' + (\eta')^2\right] \,\mathrm{d}x \,\mathrm{d}t. \tag{3.37}$$

As noted, the unique solution of the second order TLM is uniformly zero for a linear source, in which case the Hessian may be computed using the solutions of the first order

TLM:

$$\mathrm{d}^2 J_m = \sum_{g\in\mathcal{G}} \int_0^T \int_\Omega \widehat{\mathbb{1}}_g^\epsilon(\eta')^2 \,\mathrm{d}x\,\mathrm{d}t. \tag{3.38}$$

In the finite dimensional case, with $\dim(C) = N$, the TLM consists N components, meaning computing (3.38) requires the solution of N linear PDEs. It is possible to introduce an *adjoint* tangent linear model to get an alternative formulation, but its use actually requires the solution of *more* PDEs than the version above. In practice, a quasi-Newton method is usually applied to iteratively generate an approximate Hessian. Better still, for gradient-based optimisation applications, it is usually sufficient to evaluate (3.38) in the matrix-free sense.

Whilst the class of advection-diffusion problems considered in Subsection 3.3.1 is fairly general purpose, the shallow water setup in this subsection is rather application-specific, meaning that the derivation is only relevant for this particular problem. Here lies one of the major disadvantages of the continuous adjoint approach: if the PDE, its auxiliary conditions or the QoI are modified then the derivation is no longer valid and must be adjusted, potentially involving significant manual effort. As such, the user of a continuous adjoint approach is required to have at least some high level mathematical knowledge.

3.4. Discrete Adjoint

As discussed previously, the discrete adjoint formulation is obtained from the discretised PDE. All of the PDEs considered in this thesis may be written in the discrete form

$$\underline{\boldsymbol{F}}(\mathbf{w}) = \mathbf{0}, \tag{3.39}$$

where $\mathbf{w}$ is the solution vector and $\underline{\boldsymbol{F}}$ is a vector-valued function. The discrete adjoint system is simply

$$\frac{\partial \underline{\boldsymbol{F}}}{\partial \mathbf{w}}^T \mathbf{w}^* = \frac{\partial J}{\partial \mathbf{w}}^T, \tag{3.40}$$

where $\mathbf{w}^*$ is the adjoint solution vector.

3.4.1. Advection-Diffusion Discrete Adjoint. First, consider the advection-diffusion equation, discretised using Crank-Nicolson with some $\theta \in [0,1]$. We ignore source terms because they do not contribute to the adjoint. In matrix form, we have

$$\left(\tfrac{1}{\Delta t}\underline{\boldsymbol{M}} + \theta\underline{\boldsymbol{T}}_{\text{adv}}^{(k+1)} + \theta\underline{\boldsymbol{T}}_{\text{diff}}^{(k+1)}\right)\mathbf{c}^{(k+1)} = \left(\tfrac{1}{\Delta t}\underline{\boldsymbol{M}} - (1-\theta)\underline{\boldsymbol{T}}_{\text{adv}}^{(k)} - (1-\theta)\underline{\boldsymbol{T}}_{\text{diff}}^{(k)}\right)\mathbf{c}^{(k)}, \tag{3.41}$$

where $\mathbf{c}^{(k)}$ is the solution vector at time level k and $\underline{\boldsymbol{M}}$ is the L^2 mass matrix. Ignoring stabilisation, the advection and diffusion matrices have entries

$$(\underline{\boldsymbol{T}}_{\text{adv}}^{(k)})_{ij} := \int_\Omega \phi_j \mathbf{u}^{(k)} \cdot \nabla\phi_i \,\mathrm{d}x, \qquad (\underline{\boldsymbol{T}}_{\text{diff}}^{(k)})_{ij} := \int_\Omega \nabla\phi_j \cdot \underline{\boldsymbol{D}}^{(k)} \nabla\phi_i \,\mathrm{d}x. \tag{3.42}$$

Dirichlet conditions are applied directly to the solution vector in the CG formulation and are covered later in this subsection. For simplicity of notation, put $\underline{\boldsymbol{T}}_\omega^{(k)} := \omega(\underline{\boldsymbol{T}}_{\text{adv}}^{(k)} + \underline{\boldsymbol{T}}_{\text{diff}}^{(k)})$.

In order to apply formula (3.40), we first need to express (3.41) in the form (3.39). Assuming a vectorised initial condition $\mathbf{c}^{(0)} = \mathbf{g}$, we have the block matrix form

$$\begin{bmatrix} \underline{\boldsymbol{I}} & & & \\ \frac{-1}{\Delta t}\underline{\boldsymbol{M}} + \underline{\boldsymbol{T}}^{(0)}_{\theta-1} & \frac{1}{\Delta t}\underline{\boldsymbol{M}} - \underline{\boldsymbol{T}}^{(1)}_{\theta} & & \\ & \ddots & \ddots & \\ & & \frac{-1}{\Delta t}\underline{\boldsymbol{M}} + \underline{\boldsymbol{T}}^{(N-1)}_{\theta-1} & \frac{1}{\Delta t}\underline{\boldsymbol{M}} - \underline{\boldsymbol{T}}^{(N)}_{\theta} \end{bmatrix} \begin{bmatrix} \mathbf{c}^{(0)} \\ \mathbf{c}^{(1)} \\ \vdots \\ \mathbf{c}^{(N)} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}. \tag{3.43}$$

The equation is linear, so the adjoint is given by

$$\begin{bmatrix} \underline{\boldsymbol{I}} & \frac{-1}{\Delta t}\underline{\boldsymbol{M}}^T + (\underline{\boldsymbol{T}}^{(0)}_{\theta-1})^T & & \\ & \frac{1}{\Delta t}\underline{\boldsymbol{M}}^T - (\underline{\boldsymbol{T}}^{(1)}_{\theta})^T & \frac{-1}{\Delta t}\underline{\boldsymbol{M}}^T + (\underline{\boldsymbol{T}}^{(1)}_{\theta-1})^T & \\ & & \ddots & \ddots \\ & & & \frac{1}{\Delta t}\underline{\boldsymbol{M}}^T - (\underline{\boldsymbol{T}}^{(N)}_{\theta})^T \end{bmatrix} \begin{bmatrix} (\mathbf{c}^*)^{(0)} \\ (\mathbf{c}^*)^{(1)} \\ \vdots \\ (\mathbf{c}^*)^{(N)} \end{bmatrix} = \frac{\partial J^T}{\partial \mathbf{c}}. \tag{3.44}$$

The forward system (3.43) is lower triangular, meaning it can be solved by forward substitution, as is the usual strategy for method of lines type approaches. The adjoint system (3.44), on the other hand, is upper triangular, meaning backward substitution must be used instead. That is, information propagates in the opposite direction, as has already been observed. Interestingly, the discrete adjoint timestepping scheme is the same as backwards-in-time Crank-Nicolson, except that the effect of the implicitness parameter differs. There is no difference in the case $\theta = \frac{1}{2}$. However, in the case $\theta = 1$ (implicit Euler), data defining the PDE is taken from the current timestep, as opposed to the one which is being solved for. Similarly, in the case $\theta = 0$ (explicit Euler), data is taken from the timestep being solved for, rather than the current one.

Now we consider the treatment of boundary conditions. As has already been stated, the adjoint solution has zero Dirichlet conditions imposed on $\partial\Omega_D$, since it lives in the test space. Weakly imposed boundary conditions, such as Neumann conditions, are harder to interpret. In the adjoint, they do not necessarily correspond to prescribed boundary values. Indeed, weakly imposed boundary conditions are effectively forcing terms which act on the boundary, meaning that their discrete adjoints are just different forcings.

3.4.2. Shallow Water Discrete Adjoint. Hand-derivation of the discrete adjoint for tracer transport is fairly straightforward. It becomes more involved for nonlinear PDEs, such as the shallow water equations. The derivation is not included here, for brevity. Again, we find that the arrow of time is reversed and the RHS is taken from derivatives of the QoI integrand. However, the nonlinear advection and quadratic drag terms imply that forming the discrete adjoint is not as simple as transposing the LHS. This implies extra contributions on the upper diagonal of the full space-time linear system.

Whilst tedious and error-prone as a manual calculation, the individual steps involved in generating a discrete adjoint formulation are simple, making automation a desirable alternative. This is made possible by the technologies described in the following section.

3.5. Automatic Differentiation

Automatic differentiation (AD), a.k.a. *algorithmic differentiation*, is the name given to the class of algorithms which take the implementation of some function in computer code

and return code which implements derivatives of that function. In order to motivate AD technologies, we first consider one use case which is relevant to numerical modelling using PDEs: Jacobian evaluation.

3.5.1. Forward Mode. AD has a long history in computational science, starting with several independent 'forward mode' implementations, including [Beda et al., 1959] in the USSR and [Wengert, 1964] in the USA. Suppose we have code which evaluates the discretised residual of nonlinear PDE, F, at the vector representation of a solution vector, w:

$$w \mapsto F(w). \tag{3.45}$$

Then the forward mode of AD produces

$$w, \delta w \mapsto F_d(w; \delta w) =: y, \quad \text{where} \quad y_i = \frac{\partial F(w)_i}{\partial w_j} \delta w_j. \tag{3.46}$$

If the seed vector δw is taken to be the i^{th} canonical unit vector then y corresponds to the i^{th} column of the Jacobian. The (dense) Jacobian may be obtained by repeating this process over all canonical unit vectors, although doing so is expensive, as the cost scales with the vector dimension. Note the similarity to the tangent linear model for PDEs.

If the sparsity pattern of the Jacobian is known then the cost of such an approach can be reduced dramatically by making careful choices of seed vector (see [Gebremedhin et al., 2005]). Often, the Jacobian does not need to be assembled, in which case it is sufficient to apply the forward mode of AD once, 'matrix-free'.

3.5.2. Reverse Mode. For the same residual evaluation (3.45) as above, the 'reverse mode' of AD performs a different calculation. It computes the action of the Jacobian *transpose* on a seed vector, δy, from the *range space*:

$$w, \delta y \mapsto F_b(w; \delta y) =: z, \quad \text{where} \quad z_i = \frac{\partial F(w)_j}{\partial w_i} \delta y_j. \tag{3.47}$$

Note that the reverse mode derivative is evaluated at the original input, w. Again, the Jacobian transpose can be built up by careful selection of seed vectors, or applied matrix-free in a single application. Given that the Jacobian transpose is required by the discrete adjoint approach for the formulation and solution of the adjoint equation, AD enables this matrix to be computed automatically.

3.5.3. Gradient Computation. Forward and reverse mode AD may also be used to compute derivatives of QoIs. The most common alternatives are to differentiate by hand (as in Subsections 3.3.1 and 3.3.2) or to approximate using finite differences (as in Subsection 3.2.5). Hand derivations can be tedious, time-intensive and error-prone. Finite differencing, on the other hand, is typically straightforward to implement, but requires as many function evaluations as there are gradient directions (plus one) and can be inaccurate, unless a suitable step length is chosen. AD provides a variety of alternative Jacobian computation methods, which are of the same level of accuracy as a hand-derivation and typically have a much lower cost than finite differencing.

3.5.4. Implementation. In the 1990s, attempts were made to apply AD to entire CFD codes or subroutines thereof (such as [Giering and Kaminski, 1998]). Such codes were usually written in low-level programming languages such as C and FORTRAN, from which the adjoint model was derived in a line-by-line fashion. The two major implementation approaches for AD are *operator overloading*, which involves overloading elementary operators (such as addition, multiplication and exponentiation) with their adjoint derivatives (see [Griewank et al., 1996], for example) and *source transformation* (see [Bischof et al., 1992, Hascoet and Pascual, 2013], for example). The former is applied at runtime, whereas the latter generates additional code to be compiled. Despite implementation differences, the results should be identical, to machine precision. Whilst applying AD to an entire code requires no knowledge of the code itself and is fully automatic, there are many issues which can arise due to the low-level implementation. These include, but are not limited to, memory allocation, parallel communication, and I/O.

Recent AD tools tend to use a higher level abstraction. That is, codes are not decomposed into elementary operations, but into higher level operations, such as nonlinear solves, interpolations and function assignations. Such attempts use interpreted programming languages such as Python and in many cases their own domain specific language (DSL). Thus, these approaches avoid concerns related to low-level implementation and may automate the discrete adjoint derivation on a high level. Notable contributions include *libadjoint* [Farrell et al., 2013], *FATODE* [Zhang and Sandu, 2014], *pyadjoint* [Mitusch, 2018], *PerforAD* [Hückelheim et al., 2019] and the *TSAdjoint* component of PETSc [Zhang et al., 2019].

As has been mentioned, this thesis uses the Python-based Firedrake finite element package. Firedrake uses a DSL called *Unified Form Language (UFL)* [Alnæs et al., 2014], which provides an intuitive representation of the weak forms used in FEM in a very similar way to how they are written mathematically. Suppose we have the forward system (3.39), encoded in UFL as

```
solve(F == 0, w, solver_parameters={...}),
```

where '`F`' is the weak residual of a time-independent PDE, with finite element solution '`w`'. Differentiating through UFL to obtain the discrete adjoint system (3.40) is straightforward. As stated on p.1058 of [Wallwork et al., 2020b], given a QoI '`J`', a discrete adjoint solution, '`w_star`', may be obtained by running the code

```
dFdw = derivative(F, w, TrialFunction(w.function_space()))
dFdw_transpose = adjoint(dFdw)
dJdw = derivative(J, w, TestFunction(w.function_space()))
solve(dFdw_transpose == dJdw, w_star, solver_parameters={...}).
```

Expressions such as given in the code snippet above are used within pyadjoint [Mitusch, 2018] – the code which generates derivatives for the *dolfin-adjoint* package [Farrell et al., 2013]. Earlier pioneering work on solving discrete adjoint equations using dolfin-adjoint in Firedrake and *FEniCS* [Alnæs et al., 2015] was performed using libadjoint. The libadjoint package also enables fully automated derivation of the adjoint model using UFL. FATODE and PETSc TSAdjoint, on the other hand, differentiate their respective timestepping routines, meaning Jacobian (and hence Jacobian transpose) computation is left to the user. Automated Jacobian computation within PETSc was considered in [Wallwork et al., 2019]. In that work, the operator overloading AD tool *ADOL-C*

[Griewank et al., 1996] was applied on the level of nonlinear solves within PETSc's timestepping routines. However, the implementation was application-specific; wider use of AD technology in PETSc remains ongoing work.

3.6. Comparison of Continuous and Discrete Approaches

This subsection is based upon a calibration experiment for point source parametrisation, presented in [Wallwork et al., 2021]. *All experiments were performed by the author of this thesis.*

Both continuous and discrete adjoint approaches are utilised in this thesis. A number of advantages and disadvantages of the two have already been covered. They are recapitulated in this section, listed alongside some other as yet unmentioned critiques.

Inaccurate Gradients. The gradient-based optimisation routines discussed in Section 3.2 are a major application area for adjoint methods. Accurate approximations to QoI gradients improve the convergence properties of such routines. Whilst the discrete adjoint Jacobian does not yield exact gradients for the *continuous form QoI*, it does provide exact gradients for the *discretised QoI* (to machine precision). Gradients computed using the continuous adjoint approach, on the other hand, are not gradients of any functional at all and merely provide an approximation. There is no guarantee that such gradients accurately approximate those of the discrete QoI.

As observed in Subsection 3.4.1, the consistent adjoint timestepping schemes for implicit and explicit Euler are similar to their forward counterparts, but with subtle differences. Whilst this scheme may be derived without much effort and will give favourable results in terms of CPU time, the hand-derivation is tedious and error-prone for more complex multi-level or multi-stage schemes. The use of AD technology means that users need not worry about such details.

Despite the above critiques, it is argued in [Giles et al., 2005] that computed gradients which inaccurately approximate those of the discrete functional are not necessarily of concern, provided that they accurately approximate those of the *continuous functional*. Ultimately, it is gradients of the continuous functional which we seek to set to zero, rather than the discrete one. The authors argue that, provided that the gradient is driven to zero by the optimisation routine, the result will be as good as that given by the discrete adjoint. The downside is that it is difficult to check for optimality when the continuous problem itself cannot be expressed on a computer.

Implementation Issues. A fundamental disadvantage of continuous adjoint for certain types of problems is its requirement that the optimality system must be differentiable. Reynolds Averaged Navier-Stokes (RANS) applications often include non-differentiable statistical turbulence models, meaning that a continuous adjoint derivation requires special treatment of the corresponding terms. Typically, one assumes that the turbulence model is independent of the control parameters so that it may be neglected in the adjoint derivation [Carnarius et al., 2011].

An implementation disadvantage of continuous adjoint which has already been covered is that it requires hand-derivations and high-level mathematical knowledge. Discrete adjoint codes, on the other hand, can be extremely simple to use. For example, in Section 3.5 a code snippet illustrated how the adjoint of a time-independent weak form PDE can

be generated from the code for the forward solve in just four lines of UFL. Further, adjoining a Firedrake or FEniCS code automatically using dolfin-adjoint can be as simple as importing an extra library and calling a driver function such as `solve_adjoint` or `compute_gradient`. The ease of implementation of a discrete adjoint method is, of course, highly dependent upon the package used and its associated level of abstraction. For example, applying AD directly to a complex low-level code with millions of lines, such as an ocean circulation model, can be extremely laborious, making it more of a manual process than an automatic one!

Discretisation Flexibility. A major advantage of the continuous adjoint approach is its flexibility in the sense that the user is free to choose a different discretisation for the adjoint problem as for the forward problem. That is, we can change the finite element space, solver parameters and even the mesh. A discretisation which is suitable for the forward problem is not necessarily suitable for the adjoint problem. For example, coarse mesh resolution might be deployed in parts of the domain which are unimportant for capturing for the forward dynamics, but which turn out to be important for capturing the adjoint solution. Goal-oriented mesh adaptation takes a different approach to tackling these two issues. It provides a method for automatically generating meshes (which are usually used for both forward and adjoint problems) upon which the forward problem may be solved such that: (a) the QoI is accurately estimated; and (b) mesh resolution is not wasted in regions where the QoI value is insensitive to perturbations in the forward solution. See Chapter 7 for more details.

Whilst a high level abstraction is beneficial in the sense of making the discrete adjoint simple to use, one should be aware of any low level code which might fail to be registered by the AD tool. DG codes often support slope limiters in the forward model. In the case of Thetis, slope limiters are implemented as C kernels and are not registered by the Python-level AD tool (pyadjoint), meaning that they do not contribute to the discrete adjoint solution. For a continuous adjoint approach, on the other hand, the user is free to apply slope limiters. A low-level AD method would differentiate through the slope limiters.

In general, discrete adjoint packages use the same discretisation for the adjoint solve as for the forward solve. Whilst in some cases it is possible to tamper with the tape in order to modify the discretisation, doing so requires in-depth understanding of the AD tool. One particular discretisation feature which can be difficult to account for in discrete adjoint codes is mesh adaptation – a central focus of this thesis. On the contrary, if mesh adaptation is applied to the forward problem then it can be applied to the continuous adjoint problem with the same level of effort.

Numerical Comparisons. There are a number of published comparisons between continuous and discrete adjoint methods for PDEs in the literature, many of which in the context of aerospace engineering. The works of [**Nadarajah and Jameson, 2000**], [**Nadarajah and Jameson, 2001**], [**Nadarajah and Jameson, 2007**] and [**Carnarius et al., 2011**] consider a variety of QoIs (including drag coefficients, lift-to-drag ratios and target pressure distributions), as well as equation sets (including Euler equations, Navier-Stokes and RANS). For the numerical experiments considered, the discrete adjoint approach was found to agree slightly better with high resolution finite difference gradient approximations than the continuous adjoint methods implemented therein. In many cases, however, the authors' advocated the continuous adjoint approach, because of the flexibility afforded in choosing its discretisation.

The following subsection includes a quantitative comparison of continuous and discrete adjoint methods applied to an established tracer transport test case with an analytical solution, in terms of both accuracy and computational cost.

3.6.1. Point Discharge with Diffusion. Consider the *'Point Discharge with Diffusion'* test case from TELEMAC-2D validation document version 7.0 [**Riadh et al., 2014**]. Herein, a ℙ1 SUPG-stabilised Petrov-Galerkin method is used to solve an advection-diffusion problem, with the resulting linear systems are solved using a direct method.

The test case models tracer transport in a rectangular channel domain, $\Omega = [0, 50]\times[0, 10]$, with units of metres, under a uniform fluid velocity, $\mathbf{u} = (u_1, u_2) \equiv (1, 0)\,\mathrm{m\,s^{-1}}$, and constant isotropic diffusivity, $D = 0.1\,\mathrm{m^2\,s^{-1}}$. Tracer concentration enters the domain due to a source term given by a Dirac delta function at $\mathbf{x}_0 = (x_0, y_0) = (1, 5)$ and leaves the domain through the (right-hand) outflow boundary $\partial\Omega_{\text{out}} \subset \partial\Omega_N$. The tracer concentration is set to zero at the (left-hand) inflow boundary $\partial\Omega_D$, whilst zero Neumann conditions are imposed on the channel walls, $\partial\Omega_N$. In the experiments below, the original test case is modified by setting $\mathbf{x}_0 = (2, 5)$, so that the point source is not as close to the inflow.

Analytical Solution. This problem admits an analytical solution. For its derivation, note that the PDE may be written in the form

$$2\mathrm{Pe}\frac{\partial c}{\partial x} - \frac{\partial^2 c}{\partial x^2} - \frac{\partial^2 c}{\partial y^2} = \frac{1}{D}\delta(\mathbf{x} - \mathbf{x}_0), \qquad \mathrm{Pe} := \frac{u_1}{2D}. \tag{3.48}$$

Introducing the change of coordinates $\widetilde{c}(r, \theta) := \exp(-\mathrm{Pe}\ x)c(x, y)$, with polar coordinates centred at $\mathbf{x}_0$, we find that [**Choi et al., 2004**]

$$\nabla^2\widetilde{c} - \mathrm{Pe}^2\widetilde{c} = \frac{1}{D}\delta(\mathbf{x} - \mathbf{x}_0). \tag{3.49}$$

In 2D, the analytical solution of this equation is given in terms of the zeroth order modified Bessel function of the second kind, K_0, which results in [**Riadh et al., 2014**]

$$c_{\text{exact}}(\mathbf{x}) = \frac{1}{2\pi D}\exp\left(\mathrm{Pe}\ x\right) K_0\left(\mathrm{Pe}\|\mathbf{x} - \mathbf{x}_0\|_2\right). \tag{3.50}$$

Note that K_0 blows up at zero, in correspondence with the delta function at the source location. Therefore, we approximate the ℓ^2 norm as

$$d(\mathbf{x}) := \max(\|\mathbf{x} - \mathbf{x}_0\|_2, r). \tag{3.51}$$

This effectively introduces a plateau in the region $B_r(\mathbf{x}_0)$, for some $r > 0$.

Delta functions are difficult to represent in numerical methods and therefore require special treatment. In [**Wallwork et al., 2020a**], the source was parametrised using a circular indicator function over region $B_r(\mathbf{x}_0)$. The radius was chosen by trial-and-error, such that a match between analytical and finite element solutions was achieved. In [**Wallwork et al., 2021**], we instead used a Gaussian parametrisation,

$$S(\mathbf{x}) := q\ \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0\|_2^2}{r^2}\right), \tag{3.52}$$

ensuring smoothness of the source term, S. Here q is a parameter determining the source discharge, which we set to $100\,\mathrm{g\,l^{-1}}$. The parameter r is calibrated using gradient-based optimisation techniques built upon the discrete adjoint method.

Calibration. We seek to minimise the functional

$$J_{\text{calibration}}(c) := \int_{\Omega \setminus B_r(\mathbf{x}_0)} (c - c_{\text{exact}})^2 \, \mathrm{d}x, \tag{3.53}$$

subject to the constraint $r > 0$. That is, $C := (0, \infty) \subset \mathbb{R}$. Due to the plateau enforced by (3.51), we do not attempt to match the analytical solution within the source region. Note that this functional is nonlinear in both c and the control, r.

Figure 3.2 illustrates the progress of the calibration experiment. L-BFGS-B was applied on a uniform mesh with 1,024,000 elements. It starts at an initial guess of 0.1 (bright red) and eventually achieves calibrated radius $r = 5.606535\,\text{cm}$, (shown in bright yellow). Points in blue are sampled points from the parameter space.

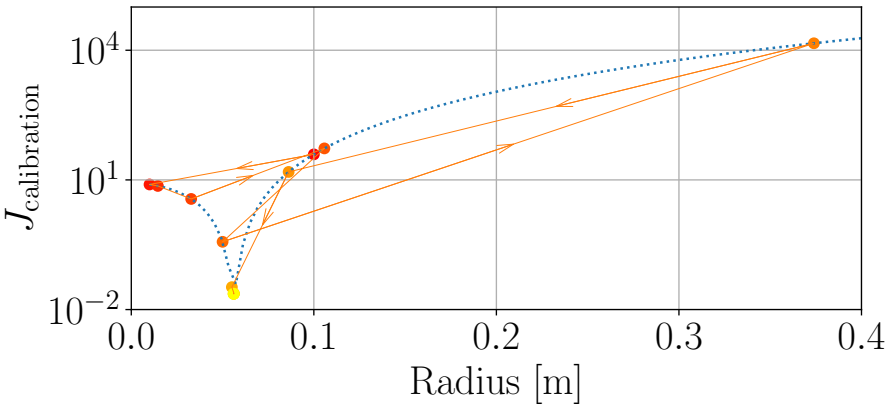


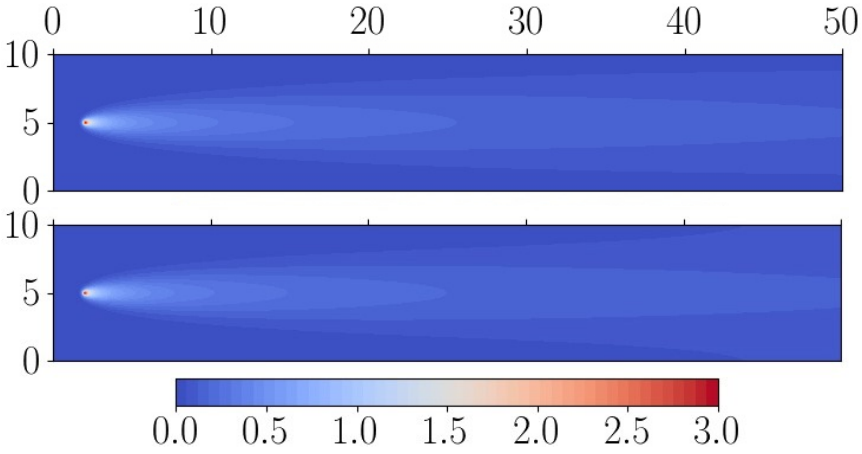
FIGURE 3.2. Optimisation progress for 'Point Discharge with Diffusion' parameter calibration. The initial guess $r_0 = 10\,\text{cm}$ is shown in bright red. Subsequent iterates move along the colour scale until the converged value $r = 5.606535\,\text{cm}$ in yellow. The order in which intermediate iterates proceed is indicated by the orange arrows.

The finite element approximation generated on a 1,024,000 element mesh is shown in Figure 3.3, along with the analytical solution interpolated into the same $\mathbb{P}1$ function space. It may be observed that our $\mathbb{P}1$ Petrov-Galerkin method provides an excellent approximation to the analytical solution on the fine uniform mesh. The only region where Subfigures 3.3a–3.3b differ noticeably is near to the boundaries for $x \in [30, 50]$. The same observation can be made for the approximation shown in [**Riadh et al., 2014**].

Comparison of Adjoint Solution Fields. In order to compare continuous and discrete adjoint methods, consider QoIs of the form

$$J_i(c) := \int_\Omega \mathbb{1}_{R_i} c \, \mathrm{d}x = \int_{R_i} c \, \mathrm{d}x, \qquad R_i := B_{\frac{1}{2}}(\mathbf{x}_i), \tag{3.54}$$

which integrate over two disc 'receiver' regions, $\{R_i = B_{\frac{1}{2}}(\mathbf{x}_i) \subset \Omega\}_{i \in \{1,2\}}$, centred at $\mathbf{x}_1 = (20, 5)$ and $\mathbf{x}_2 = (20, 7.5)$. That is, the former receiver region is directly downstream of the source and the latter is offset by 2.5m to the North. In the context of (2.13), the background tracer concentration is zero.



(A) Analytical forward solution (3.50).

(B) Finite element forward solution.

FIGURE 3.3. Analytical and finite element solutions for the TELEMAC-2D 'Point Discharge with Diffusion' validation experiment [**Riadh et al., 2014**]. Solutions are presented in a $\mathbb{P}1$ space defined on a 1,024,000 element uniform mesh. Figures as displayed in [**Wallwork et al., 2021**].

The continuous adjoint formulation, derived in Subsection 3.3.1, is given by (3.29). As in the forward problem, we use a modified test space, $V_h := \{\psi + \tau_K \nabla\cdot(\mathbf{u}\psi) \mid \psi \in W_h\}$, to define the SUPG stabilised continuous adjoint equation: $\forall \phi \in V_h$,

$$-\langle\phi, \nabla\cdot(\boldsymbol{u}c_h^*)\rangle - \left\langle\nabla\phi, \underline{\boldsymbol{D}}^T\nabla c_h^*\right\rangle + \left\langle\phi, \underline{\boldsymbol{D}}^T\nabla c_h^*\cdot\widehat{\mathbf{n}} + c_h^*\boldsymbol{u}\cdot\widehat{\mathbf{n}}\right\rangle_{\partial\Omega\setminus\partial\Omega_D} = \langle\phi, \mathbb{1}_{R_i}\rangle\,. \tag{3.55}$$

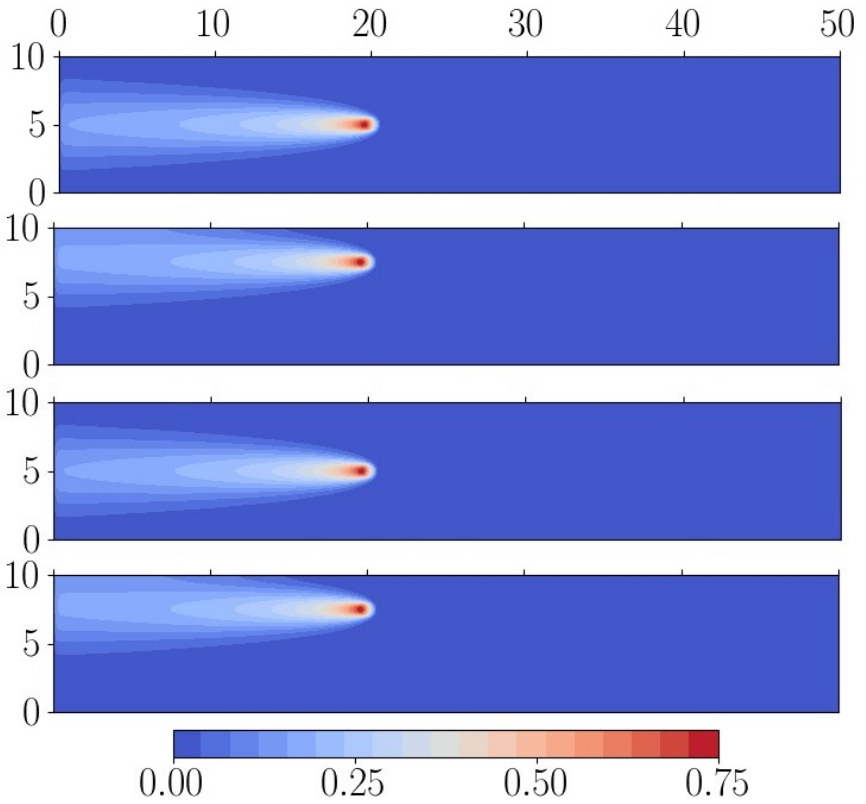
The L^2 Riesz representations, $\{\mathcal{R}_{L^2}(J_i) = \mathbb{1}_{R_i}\}_{i\in\{1,2\}}$, provide source terms for the two (aligned and offset) adjoint equations. Figure 3.4 presents finite element solutions of the SUPG stabilised continuous adjoint problem (3.55), as well as discrete adjoint solutions of the SUPG stabilised forward finite element problem (3.19). In each case, we observe that, whilst the forward tracer concentration propagates downstream, the adjoint tracer concentration propagates *upstream*. The adjoint tracer concentration is near zero downstream of the receiver regions. This is consistent with the fact that the QoIs are independent of the downstream dynamics for advection-dominated problems.

Qualitatively, it is difficult to distinguish any noticeable differences between Subfigures (3.4a) and (3.4c) and between Subfigures (3.4b) and (3.4d). Figure 3.5 presents the absolute difference between the two fields on a logarithmic scale. In terms of L^2 errors against the discrete adjoint solution, the values for the aligned and offset QoIs are 0.1085% and 0.1082%, respectively. On a mesh with 16,384,000 elements, the errors reduce to 0.0037% in both cases. The differences are purely due to the different ways in which stabilisation is handled under the two approaches.

Gradient Accuracy. Whilst the L^2 errors appear to be fairly small, this is not necessarily the case for gradients computed using each method. Suppose we wish to compute gradients w.r.t. the diffusion coefficient, $D \in (0, \infty)$.

Due to the analytical solution (3.50), we can derive the analytical gradient in infinite-dimensional space. The following derivation makes use of the identities [**Teukolsky et al., 1992**]

$$K_\nu'(\omega) = \frac{K_{\nu-1}(\omega) + K_{\nu+1}(\omega)}{-2} \quad\text{and}\quad K_{-\nu}(\omega) = K_\nu(\omega), \qquad \forall\omega \geq 0, \quad \forall\nu \in \mathbb{Z}. \tag{3.56}$$



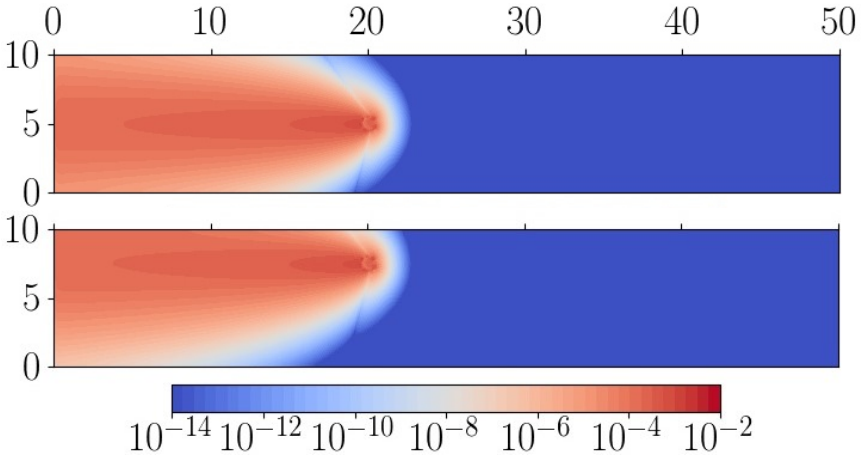
(A) Continuous adjoint in aligned case.

(B) Continuous adjoint in offset case.

(C) Discrete adjoint in aligned case.

(D) Discrete adjoint in offset case.

FIGURE 3.4. Continuous and discrete adjoint solutions for the 'Point Discharge with Diffusion' experiment, corresponding to QoIs of the form (3.54). Each field is presented in $\mathbb{P}1$ space on a 1,024,000 element uniform mesh.



(A) Aligned case

(B) Offset case

FIGURE 3.5. Absolute difference between continuous and discrete adjoint solutions for the 'Point Discharge with Diffusion' experiment, corresponding to QoIs of the form (3.54). Fields are presented in $\mathbb{P}1$ space on a 1,024,000 element uniform mesh.

QoI $J := J_1$ evaluated at the analytical solution c_{exact} takes the form

$$J(c_{\text{exact}}; D) = \frac{1}{2\pi} \int_{R_i} \frac{1}{D} \exp\left(\frac{u_1 x}{2D}\right) K_0\left(\frac{u_1}{2D} \|\mathbf{x} - \mathbf{x}_0\|_2\right) \, \mathrm{d}x. \tag{3.57}$$

By the chain rule (2.34), we have

$$\begin{aligned}\mathrm{d}J_D(c_{\text{exact}};D) = &-\frac{1}{2\pi}\int_{R_i}\left(\frac{1}{D^2}+\frac{u_1 x}{2D^3}\right)\exp\left(\frac{u_1 x}{2D}\right)K_0\left(\frac{u_1}{2D}\|\mathbf{x}-\mathbf{x}_0\|_2\right)\,\mathrm{d}x\\ &+\frac{1}{2\pi}\int_{R_i}\frac{u_1\|\mathbf{x}-\mathbf{x}_0\|_2}{2D^3}\exp\left(\frac{u_1 x}{2D}\right)K_1\left(\frac{u_1}{2D}\|\mathbf{x}-\mathbf{x}_0\|_2\right)\,\mathrm{d}x.\end{aligned} \tag{3.58}$$

For the continuous and discrete adjoint methods, we can approximate the gradient by substituting the appropriate approximation to the adjoint solution into

$$\mathrm{d}J_D(c;D) = -\int_\Omega \nabla c^*(\mathbf{x};D)\cdot\nabla c(\mathbf{x};D)\,\mathrm{d}x. \tag{3.59}$$

The analytical gradient formula (3.58) involves integrals of Bessel functions, meaning we cannot evaluate it exactly. However, applying numerical quadrature on a fine mesh is sufficient for the purposes of this test case. Table 3.1 compiles the approximations due to each approach. The analytical gradient and the gradients due to the continuous and discrete adjoint approaches all converge to three significant figures on the fifth mesh in the hierarchy. On this mesh, both approximations agree with the analytical value to two significant figures, with the continuous adjoint approximation being slightly more accurate. However, the two agree very well on every mesh in the hierarchy and the discrepancies are so small that it is not possible to conclude that one or the other provides a consistently better approximation.

Mesh elements	Mesh vertices	Analytical	Continuous	Discrete
4,000	2,121	-6.3680×10^{-1}	-2.6819×10^{-2}	-2.8118×10^{-2}
16,000	8,241	-7.8079×10^{-1}	-6.2320×10^{-1}	-6.1942×10^{-1}
64,000	32,481	-8.0889×10^{-1}	-8.4707×10^{-1}	-8.3923×10^{-1}
256,000	128,961	-8.0585×10^{-1}	-7.9611×10^{-1}	-7.9493×10^{-1}
1,024,000	513,921	-8.0614×10^{-1}	-7.9616×10^{-1}	-7.9530×10^{-1}

TABLE 3.1. Gradient comparison of continuous and discrete adjoint methods applied to the 'Point Discharge with Diffusion' test case in the aligned configuration, on a hierarchy of uniform meshes.

Comparison of Computational Cost. Next, the continuous and discrete adjoint methods are compared in terms of CPU time. The corresponding values are presented in Table 3.2. Element and vertex counts of the mesh hierarchy are shown in the first two columns. The vertex count is equivalent to the DoF count of the scalar $\mathbb{P}1$ space used. The middle two columns show the total time taken to solve the forward problem and the continuous adjoint problem. The final two columns show the corresponding times for solving the forward problem and discrete adjoint problem. Timings include the problem setup, but not compilation, because the underpinning C kernels have already been compiled.

The reason that the forward solve associated with the discrete adjoint run takes longer than that associated with the continuous adjoint run is because data is annotated to tape. Note that this becomes less significant as mesh resolution is increased. In the discrete adjoint solve, unity is propagated through the reverse mode of AD, which involves just one linear solve. A disadvantage of operator overloading type AD implementations applied to linear problems is that the forward problem needs to be solved in order to annotate the

Mesh		Continuous		Discrete	
Elements	Vertices	Forward [s]	Adjoint [s]	Forward [s]	Adjoint [s]
4,000	2,121	0.3498	0.3114	0.4431	0.3479
16,000	8,241	0.4763	0.4275	0.5733	0.4564
64,000	32,481	1.0066	0.8982	1.1036	0.9393
256,000	128,961	3.3195	2.9707	3.5347	3.0874
1,024,000	513,921	13.6096	12.3890	13.8617	12.6723

TABLE 3.2. Timing comparison of continuous and discrete adjoint methods applied to the 'Point Discharge with Diffusion' test case in the aligned configuration, over a range of uniform meshes. All timings are averaged over ten runs.

tape. In the continuous adjoint case, the adjoint problem can be solved independently. For nonlinear problems, both forward and adjoint need to be solved for both continuous and discrete adjoint implementations.

In summary, if only the adjoint equation is to be solved then the continuous adjoint method takes less than half of the time required for the discrete adjoint method, assuming the equation and QoI to be linear. However, it should be noted that in most cases both forward and adjoint solution data are sought, in which case the additional cost attributed to the discrete adjoint method becomes relatively small as mesh resolution is increased. The difference is similarly small in the nonlinear case.

Impact Upon Adapted Meshes. The comparison between continuous and discrete adjoint made in the above investigation informs the choice of method when returning to the 'Point Discharge with Diffusion' test case in Subsection 7.5.2. In that subsection, a further comparison is made, based on the impact of adjoint method implementation upon the meshes outputted by a goal-oriented mesh adaptation algorithm, as well as the convergence properties thereof.

Uniform Convergence. Having compared the continuous and discrete adjoint methods, we conclude this section with a convergence analysis experiment for the two QoIs. This provides benchmarks for later convergence studies using mesh adaptive methods.

As with the gradient, evaluation of the exact QoI value for J_i requires integrating a modified Bessel function, so it is sufficient to apply numerical quadrature. This is supported by the results displayed in Table 3.3. Convergence to four decimal places is achieved on a mesh with 16,384,000 elements, for each of the four columns.

That the converged QoI values associated with the analytical and finite element solutions differ reveals that there is model error at play, as well as discretisation error. Suppose $c_{\text{converged}}$ is the converged finite element solution on the finest mesh in the hierarchy and c_h is the finite element solution on one of the coarser meshes. Given the analytical solution c, we have the decomposition

$$e = c - c_h = \underbrace{c - c_{\text{converged}}}_{\text{Model error}} + \underbrace{c_{\text{converged}} - c_h}_{\text{Discretisation error}} . \tag{3.60}$$

Discretisation error is ultimately what we care about in mesh convergence studies; it can be calculated using the converged QoI due to the finite element method as 'truth'. Model

Mesh		Aligned		Offset	
Elements	Vertices	Analytical	Finite element	Analytical	Finite element
4,000	2,121	0.12926	0.00614	0.055119	0.002409
16,000	8,241	0.15844	0.12635	0.067488	0.054054
64,000	32,481	0.16414	0.17057	0.069880	0.072804
256,000	128,961	0.16353	0.16145	0.069619	0.068871
1,024,000	513,921	0.16359	0.16150	0.069651	0.068886
4,096,000	2,051,841	0.16339	0.16130	0.069569	0.068803
16,384,000	8,199,681	0.16343	0.16134	0.069586	0.068820

TABLE 3.3. Convergence of QoIs J_1 and J_2 under analytical and finite element solutions on a sequence of uniform meshes.

error is also at play in the experiment comparing gradients due to the continuous and discrete adjoint methods.

3.7. Checkpointing

This section focuses exclusively on issues arising when applying adjoint methods to time-dependent problems.

As has been mentioned, where information propagates forwards in time in the forward model, it propagates backwards in the adjoint model. For nonlinear PDEs and/or QoIs, the adjoint problem depends on the forward solution *at each time level*. As such, integrating the adjoint solution at a particular time level requires forward solution data from at least one time level.

Sequentially integrating the forward problem from time $t = 0$ to $t = T$ and then the adjoint problem in reverse requires capacity to store (at least) the entire forward trajectory in RAM. For problems with very few DoFs and/or timesteps, this can be feasible. For realistic, high resolution applications, however, the solution from a single timestep can be on the order of gigabytes. As such, it quickly becomes infeasible to use the naïve strategy of time integrating both problems over the entire time period $(0, T]$, one after the other.

In the case where both PDE and QoI are linear, the adjoint is independent of the forward and it is possible to solve using the aforementioned 'naïve strategy' whilst only storing the solution fields required to progress at the current time level. However, it can be useful to retain solution data if the goal of the adjoint computation is to assemble goal-oriented error indicators, for example.

The idea of checkpointing schemes is to store solution data from only a subset of time levels. This means that, in order to perform the adjoint integration on some subinterval in time, the forward trajectory should be recomputed over that subinterval. As such, we have a trade-off between recomputation and storage. A *single stage* checkpointing scheme stores all of its checkpoints in RAM, whereas a *multi-stage* checkpointing scheme stores checkpoints in both RAM and also on hard disk. Whilst data storage and retrieval from hard disk in the latter approach are more time consuming than from RAM, doing so can reduce the number of recomputations required [**Carnarius et al., 2011**]. The checkpointing schedule can be chosen manually or automatically. One effective automated

approach is *revolve* [**Griewank and Walther, 2000**]: a single-stage binomial strategy which is provably optimal in terms of minimising the number of forward recomputations.

The only checkpointing scheme used in this thesis is rather simple, decomposing the time interval into subintervals of fixed length, as detailed in Subsection 5.7.2. This scheme is used to facilitate metric-based mesh adaptation for time-dependent problems. It is also used in Subsection 7.6.1 in the goal-oriented case. For further details on checkpointing routines, see
[**Griewank and Walther, 2000**, **Stumm and Walther, 2009**, **Wang et al., 2009**].

CHAPTER 4

Tsunami Source Inversion

Motivation. On the 11th of March 2011, a magnitude 9.1 earthquake beneath the Japan Sea triggered what became known as the Tōhoku tsunami. This event was notable first and foremostly because it was disastrous for Japan, leading to the loss of at least 15,899 lives, 2,529 missing persons and 6,157 injuries [**National Police Agency of Japan, 2020**]. Whilst the earthquake did cause direct damage, the tsunami was the major cause of death [**Mori et al., 2012**]. 121,991 properties were reported to have collapsed due to the earthquake and tsunami, along with hundreds of thousands of other properties and pieces of infrastructure suffering damage. A particularly important piece of infrastructure which was affected by the tsunami was Fukushima Daiichi nuclear power plant, which suffered a meltdown. This led to the mass evacuation of all residents living within a 20 km radius, as well as requiring an extensive clean-up operation. There is a clear motivation to improve early warning systems so that such destruction can be avoided in future earthquake-tsunami scenarios.

Secondly, the Tōhoku tsunami was notable because a very large amount of data was recorded both during and following the event. These data include dozens of timeseries readings from GPS, tide and pressure gauges across the Japan Sea and into the Pacific Ocean, as well as inundation records at thousands of coastal locations. With such a volume of data, there is potential for detailed modelling of the tsunami propagation and its impact upon coastal communities.

Novel Contributions. The main novel contribution of this chapter is the first application of two different AD tools to the source and tsunami propagation models. Like the previous chapter, it also contains a quantitative comparison of continuous and discrete adjoint methods within the same code framework. To the best of the author's knowledge, this is the first such comparison for the shallow water equation system.

Chapter Outline. Firstly, Section 4.1 provides a literature review on tsunami source inversion, with a focus on adjoint-driven approaches. Section 4.2 provides details on the problem specification and data used for the Tōhoku tsunami case study. The adjoint-based source inversion method is described in Section 4.3. The method is first applied to 'synthetic' problems with known solutions in Section 4.4. This involves inversion for a single control parameter, as well as an experiment investigating parameter redundancy in the Okada source model. Finally, real gauge data due to the Tōhoku tsunami are inverted for Okada source parameters in Section 4.5.

4.1. Literature Review

Tsunami Propagation Models. Tsunamis are inherently time-dependent in nature. As such, an important aspect of tsunami propagation modelling is the choice of initial condition, as a result of the earthquake dynamics. A common assumption is that the surface

displacement occurs instantaneously, whereby it may be modelled by imposing an initial condition on the fluid equations (although it is also possible to incorporate time-dependent ruptures using forcing terms). Whilst the assumption of an instantaneous tsunami source is not valid in terms of the physics, it has been shown to have little influence on the far-field solution [**Oishi et al., 2013**]. The justification lies in the fact that earthquake rupture speeds are an order of magnitude faster than those of tsunami propagation speeds [**Kajiura, 1970**]. The assumption of an instantaneous source lacks a representation of the physical processes occurring in the Earth's crust, but it is simpler and computationally cheaper to run a tsunami model as described than a full coupled earthquake-tsunami model. In this work, we seek only to model tsunamigenic processes and use relatively simple algebraic models for the source. For a more rigorous, detailed analysis and simulation of the coupled earthquake-tsunami event, we refer to [**Ulrich et al., 2019**].

A second common assumption is that the sea bed displacement disturbs the entire water column in a uniform way, meaning that it effectively translates to an equal displacement of the ocean surface. This assumption is justified for megathrust tsunamis, such as Tōhoku, because the major component of the displacement is in the vertical [**LeVeque et al., 2011b**].

Along with the initial free surface condition, we follow the literature in assuming zero initial horizontal velocity. See [**Lotto et al., 2017**] for a discussion on the validity of this assumption.

Despite the large amount of data available, initial conditions for the Tōhoku tsunami surface elevation proposed in the literature differ quite dramatically. One early study [**MacInnes et al., 2013**] compared ten sources from the tsunami modelling literature and found them to have not only strikingly different geometries, but also to differ in their ability to capture phenomena such as coastal inundation.

One major reason for the differing tsunami sources has to do with the underlying fluid equations used for the inversion. It is uncommon to use the full 3D Navier-Stokes equations for tsunami modelling, due to the high computational cost, although notable efforts do exist (such as [**Oishi et al., 2013**]). Since large tsunami waves have wavelengths which are far greater than the depths of even the deepest parts of the ocean in which they exist, the depth-averaged shallow water equations usually provide a sufficiently accurate approximation for use in tsunami propagation studies, for a lower computational cost [**Dao and Tkalich, 2007**].

Within the realm of shallow water modelling there are further modelling decisions to be made. Whilst most authors agree that Coriolis and viscous forces do not play important roles in tsunami modelling, the impact of nonlinearity, dispersion and bed friction are more debatable. The linear and nonlinear shallow water equations have been compared in the context of various recent tsunamis in [**Satake, 1995**, **Liu et al., 2009**, **Saito et al., 2014**]. The former equation set is generally sufficient for large tsunamis in the deep ocean, but the assumption of linearity breaks down near to the shore. Similarly, bed friction is much more important in coastal dynamics than in the deep ocean and in particular for inundation studies such as those presented in [**MacInnes et al., 2013**, **Funke et al., 2017**].

Dispersive effects in both linear and nonlinear models have been studied in [**Cho et al., 2007**, **Kirby et al., 2013**, **Saito et al., 2014**, **Ren et al., 2015**]. The Tōhoku tsunami considered in this work was caused by an earthquake whose epicentre was less than 200

kilometres East of Japan. As such, the westward propagating tsunami wave did not travel far before reaching the coast and therefore frequency dispersion did not play a major role. The Eastward propagating wave, on the other hand, crossed the entire Pacific Ocean, making dispersion more significant.

Source Inversion Models. Another important factor which contributes to the differences in source representations for the Tōhoku tsunami is the inversion strategy employed. Early attempts at tsunami inversion used a *'ray tracing'* approach, originally proposed in [**Abe, 1973**]. By solving the linear shallow water equations backwards in time, such an approach is able to use gauge timeseries to determine a spatial region containing the tsunami source with high confidence. However, this approach is not able to deduce anything about the source geometry. To account for this, the author of [**Satake, 1987**] represented the slip on each segment of a fault using a collection of indicator basis functions multiplied by scalar control parameters. Under a linear shallow water regime, with the source depending linearly on the control parameters, the misfit between observed and simulated gauge measurements also depends linearly on the control parameters. Thus, the search for control parameters which minimise the sum of squared misfits can be expressed as a least squares problem. Many authors have used solved least squares problems of this type in order to estimate sources of recent tsunamis, both in terms of slip parameters and also in terms of the initial water surface (see [**Satake, 1987**, **Saito et al., 2011**, **Ren et al., 2018**, **Mulia et al., 2018**], for example).

The first published use of adjoint methods for tsunami source inversion was in [**Pires and Miranda, 2001**]. A continuous adjoint shallow water model was used to compute the gradient of the sum of squared timeseries mismatch errors w.r.t. control parameters. A gradient-based optimisation method was then applied to find the minimum. In addition to representing the source in terms of indicator functions, the authors also considered explicit optimisation for the focal fault parameters used in the Okada model [**Okada, 1985**]. This is not possible in the least squares approach because of the nonlinear relationships between the Okada parameters and the initial free surface. Other source inversions using adjoint methods include [**Blaise et al., 2013**, **Kabanikhin et al., 2014**, **Hossen et al., 2018**, **Zhou et al., 2019**], to name just a few. Typically authors use objective functionals corresponding to misfits with gauge timeseries data, as described above. However, it is also possible to reconstruct the tsunami source from inundation records [**Funke et al., 2017**].

Whilst in most cases continuous adjoint methods are employed, [**Blaise et al., 2013**] uses a discrete adjoint formulation driven by an AD tool. Given a linear combination of pre-defined 'unit sources' (due to [**NOAA, 2021**]), the authors invert the shallow water equations for the coefficients, under constraints to ensure the results are physically acceptable. In the case where the Okada source model is deployed, the use of an AD tool is advantageous because the Okada model is a prime example of where a hand derived continuous adjoint is tedious and error-prone [**Pires and Miranda, 2001**]. In this chapter, we go further than applying a discrete adjoint method to the tsunami propagation model and also apply an AD tool to the source model. To the best of the author's knowledge, this is the first application of different AD tools to both the Okada source model and a tsunami propagation model. However, the work of [**Bifulco et al., 2009**] is similar in that it inverts the Okada model (alone) using computer algebra software and compares its performance against using finite differences and the complex step method. On p.1435 of that paper, the authors remark that

> "It is remarkable that the parameters found by the various schemes are somewhat different even if the corresponding output errors are of the same order, putting into evidence the ill-conditioning of this complex inversion problem."

As such, we opt to invert just two of the nine Okada parameters and perform an experiment which tests for parameter redundancy in these.

4.2. The Tōhoku Tsunami

4.2.1. Problem Specification. All experiments conducted in this chapter are based on the 2011 Tōhoku tsunami case study. Some experiments use real data, whilst others are synthetic. However, all of them use the same meshes, bathymetry data set and gauge locations, which are described in this subsection.

Meshes. Fixed meshes were constructed using *qmesh* [**Avdis et al., 2018**]. Taking GIS data as input, qmesh generates a *gmsh* [**Geuzaine and Remacle, 2009**] geometry file which describes the domain and how mesh resolution should vary across it. This geometry file is then passed to gmsh, which generates a Delaunay tessellation. We construct a non-nested hierarchy of meshes such that the mesh at each level has approximately four times as many elements as its predecessor. By running simulations on each mesh in turn, we are able to perform convergence analyses. Given that the meshes are relatively isotropic in most of the domain, we can liken the transfer from one level of the hierarchy to the level above as akin to a global uniform refinement in both coordinate directions. Element counts of the four meshes used in this study are presented in Table 4.1, where $\mathcal{H}_0$ is the base mesh and $\mathcal{H}_i$ has approximately $(2n)^i = 4^i$ as many elements.

Mesh	$\mathcal{H}_0$	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{H}_3$
Elements	15,441	60,461	240,090	957,978
Vertices	8,117	31,009	121,592	482,072
Max. aspect ratio	8.3530	3.8145	2.6970	2.6140
Boundary length	1096.1 km	1128.1 km	1152.8 km	1169.0 km

TABLE 4.1. Mesh statistics for the Tōhoku tsunami case study.

As shown in the table, mesh $\mathcal{H}_0$ has at least some moderately anisotropic elements, with maximal aspect ratio 8.3530. This is because long, thin elements are needed to tessellate some of the small bays near to the coastal boundary. With a finer mesh, this can be achieved with smaller isotropic elements. Indeed, the maximal aspect ratio becomes smaller as finer meshes are permitted. In addition, note that the boundary length increases going up the mesh hierarchy. This makes sense, due to the fractal nature of coastlines. Whilst this figure should not converge – since the length of a coastline is not well defined – in practice it will converge to the resolution of the bathymetry data set the topographic contours are extracted from. In this case, the ETOPO1 data set [**Amante and Eakins, 2020**] was used. Whilst the ocean boundary is also included in the boundary length stated in Table 4.1, it can be ignored, because it is constant across the refinement levels.

An alternative approach to generating a mesh hierarchy is to apply iso-$\mathbb{P}2$ refinement directly to the base mesh illustrated in Figure 4.1. Whilst this has the desirable nesting

property which enables efficient and accurate interpolation between levels, it means that coastlines on the higher resolution meshes are of the same resolution as the base mesh. Since we do not require interpolation between meshes in the hierarchy, we choose an improved coastal representation over a prolongation/injection mesh transfer structure.

The domain boundary is separated into three disjoint segments, as illustrated in Figure 4.1. In addition to an artificial open ocean boundary, the segment of the coast surrounding Miyagi Prefecture is separated out. This region suffered the most fatalities (58%, followed by 33% in Iwate Prefecture and 9% in Fukushima Prefecture) [**Mori et al., 2012**] and hence accurately modelling the tsunami dynamics there is of interest. By separating the coastal boundary segments, we are able to specify different gradation parameters for each in order to improve the resolution surrounding Miyagi. For each boundary segment, gradation allows for different choices of: (a) mesh resolution; and (b) distances over which resolution scales to that in the mesh interior. As we move up the mesh hierarchy, the gradation distances and the inner and outer gradations are halved.

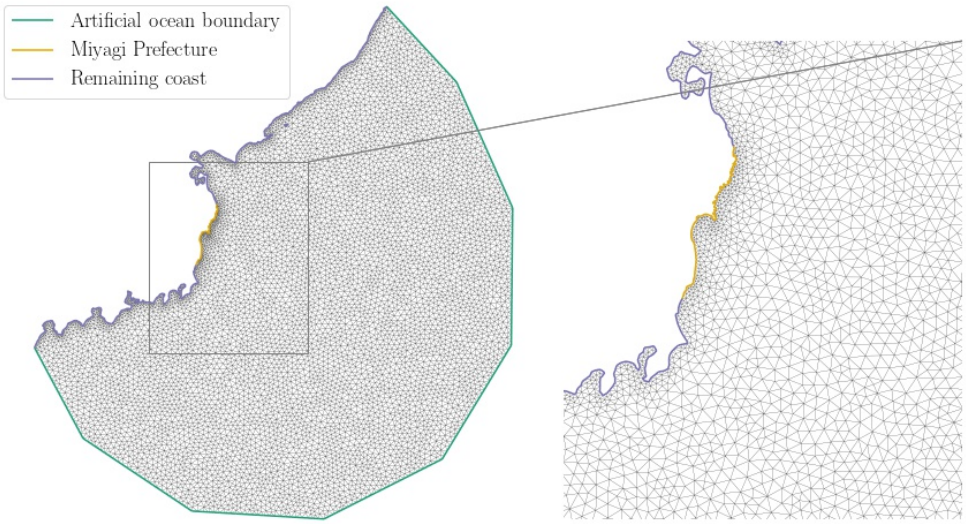


FIGURE 4.1. Base mesh, $\mathcal{H}_0$, with a zoom on Miyagi Prefecture.

GPS and Pressure Gauges. When the Tōhoku tsunami struck in 2011, there were a number of active gauges within the spatial domain, for which timeseries data are available. These include GPS gauges, which measure the variation of the free surface from its rest state, as well as pressure gauges, which take readings at the sea bed. Under hydrostatic assumptions (2.2), we have that

$$p = \rho_{\text{sea}} g H + p_a \quad \Longrightarrow \quad \eta = \frac{p - p_a}{\rho_{\text{sea}} g} - b, \tag{4.1}$$

where p is the observed pressure and p_a is the pressure at the free surface. Using relation (4.1) and appropriate bathymetry data, we may deduce the free surface elevation timeseries at pressure gauges.

The gauge locations are shown mapped on top of the bathymetry field obtained from the ETOPO1 data set in Figure 4.2. Gauges 801, 802, 803, 804, 806 and 807 are GPS gauges, whilst all other gauges are pressure gauges. The GPS gauge timeseries were

obtained from the Japanese Port and Airport Research Institute (PARI). P02 and P06 were temporary gauges controlled by Tōhoku University at the time of the tsunami; their timeseries were obtained through personal correspondence with Tatsuhiko Saito. Gauges KPG1 and KPG2 to the north are controlled by the Japan Agency for Marine Earth Science and Technology (JAMSTEC), as are MPG1 and MPG2 to the south. Finally, gauges 21401, 21413, 21418 and 21419 are controlled by the National Oceanic and Atmospheric Administration (NOAA) as part of the DART system. It is worth noting that the DART gauges are located much further out into the Pacific Ocean than the others considered.

In line with the descriptions above and their locations as shown in Figure 4.2, we categorise the sixteen gauges into five sets, relative to their proximity to the earthquake source. The categories are 'GPS gauges' {801, 802, 803, 804, 806, 807}, 'near-field pressure gauges' {P02, P06}, 'mid-field pressure gauges' {KPG1, KPG2, 21418}, 'far-field pressure gauges' {21401, 21413, 21419} and 'southern pressure gauges' {MPG1, MPG2}.

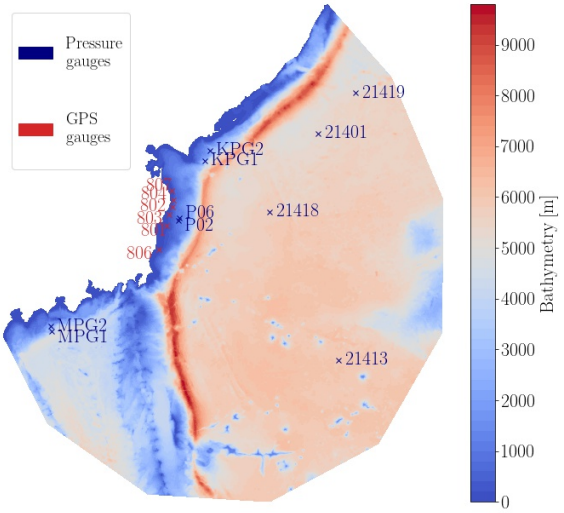


FIGURE 4.2. Bathymetry obtained from the ETOPO1 data set, overlaid with the locations of sixteen GPS and pressure gauges which were active during the 2011 Tōhoku tsunami. Presented in a $\mathbb{P}1$ space defined on mesh $\mathcal{H}_3$.

Categorising the gauges by proximity to the earthquake epicentre means that the arrival and departure times of the tsunami wave are similar within categories. For the near-field pressure gauges, the time period of interest is contained within the first simulated hour, whereas for the far-field gauges it is from approximately 50 minutes post-earthquake onwards, for example. For a full discussion of the gauge timeseries themselves, see Section 4.5.1.

4.3. Adjoint-Based Source Inversion

4.3.1. Quantity of Interest. In this subsection, the QoI (3.31) stated for tsunami source inversion is derived and discussed.

Suppose we have free surface timeseries data from a finite set of gauges $\mathcal{G}$. We have initial guesses for the basis coefficients, but would like to optimise them so that the resulting tsunami propagation model best captures the timeseries. Assuming we have data for all time, we can represent this using the QoI

$$J^{\mathrm{PE}}(\mathbf{u},\eta) := \frac{1}{2}\sum_{g\in\mathcal{G}}\int_0^T \big(\eta(\mathbf{x}_g,t)-\eta_g(t)\big)^2\,\mathrm{d}t = \frac{1}{2}\sum_{g\in\mathcal{G}}\int_0^T\int_\Omega \big(\eta\mathbb{1}_{\{\mathbf{x}_g\}}-\eta_g(t)\big)^2\,\mathrm{d}t, \tag{4.2}$$

where gauge g has spatial location $\mathbf{x}_g \in \Omega$ and timeseries value $\eta_g(t)$ at time $t \in (0, T]$. The superscript 'PE' stands for 'point evaluation'. The second integral in (4.2) reveals that we have a functional involving an indicator function $\mathbb{1}_{\{\mathbf{x}_g\}}$ defined at a single point. As mentioned in Chapter 3, such functions are not well defined in FEM and are difficult to handle in numerical methods in general. Moreover, it is proved in [**Kabanikhin et al., 2014**] that the inversion problem defined by minimising (4.2) constrained by the linear shallow water equations is ill-posed. This is due to lack of regularity in the RHS of the resulting adjoint equation.

Instead, we consider an approximation involving area-normalised circular indicator functions with a small radius, $\epsilon > 0$:

$$J^{\mathrm{AN}}(\mathbf{u}, \eta) := \frac{1}{2} \sum_{g \in \mathcal{G}} \int_0^T \int_\Omega \widehat{\mathbb{1}}_g^\epsilon \big(\eta - \eta_g(t)\big)^2 \,\mathrm{d}x \,\mathrm{d}t, \qquad \widehat{\mathbb{1}}_g^\epsilon := \frac{1}{|B_\epsilon(\mathbf{x}_g)|} \mathbb{1}_{B_\epsilon(\mathbf{x}_g)} \tag{4.3}$$

The superscript 'AN' stands for 'area-normalised'. That J^{AN} converges to J^{PE} under mesh refinement is demonstrated in Subsection 4.4.1.

Accounting for temporal discretisation, the QoI becomes

$$J^{\mathrm{AN},w}(\mathbf{u}, \eta) := \frac{1}{2} \sum_{t \in \mathcal{T}} w_t \sum_{g \in \mathcal{G}} \int_\Omega \widehat{\mathbb{1}}_g^\epsilon \big(\eta - \eta_g(t)\big)^2 \,\mathrm{d}x. \tag{4.4}$$

where $\{w_t\}_{t \in \mathcal{T}}$ are appropriately chosen quadrature weights on a finite subset of time levels, $\mathcal{T} \subset (0, T]$.

Henceforth, the J^{AN} notation is dropped and $J = J^{\mathrm{AN}}$ should be assumed, unless otherwise stated.

For our particular choice of area-normalised QoI (4.3) we have

$$\partial J_{\mathbf{u}} = \mathbf{0}, \qquad \partial J_\eta = \sum_{g \in \mathcal{G}} \widehat{\mathbb{1}}_g^\epsilon (\eta - \eta_g). \tag{4.5}$$

That is, the adjoint shallow water system has a forcing term for the adjoint elevation, but not the adjoint velocity. The form of these derivatives means that, even if we consider the linearised equations, the RHS of the adjoint equation depends on the forward solution. As such, the potential reduction in model complexity afforded by linearising the shallow water equations is negated by the choice of QoI. In particular, this means: (a) the forward equation must always be solved before solving the adjoint equation; and (b) we need to use checkpointing to store the forward solutions for later use.

By area-averaging the contributions to the QoI from each gauge, we are effectively applying a spatial normalisation. We could also divide the QoI by the total length of time for which we seek to match the data, summed over all gauges. However, in practice, we find that this causes problems when solving the adjoint equations. With the time integrated QoI normalised to a $\mathcal{O}(1)$ value, the instantaneous contributions to the RHS of the adjoint equation in (4.5) become very small indeed. In some cases, the forcing term ends up being on the level of machine precision. This can lead to artificial zero gradients – amongst other issues – and therefore we opt not to normalise in time.

Gradient of the QoI. There is one ingredient remaining to express the tsunami source inversion problem in the form (3.3): the source model. We assume that the source model may be represented as a mapping, $S : C \to V_h$, from control space into the discretised

solution space. That is, for a vector of control parameters, $\mathbf{m}$, we define the initial free surface by $\eta(\mathbf{x}, 0) := S(\mathbf{m})$. The choice of source model is discussed in the following subsection.

Given a solution of the adjoint equation, the functional gradient may be obtained using formula (3.7). For the tsunami source inversion setup described, it is a vector with k^{th} component

$$\left(\frac{\mathrm{d}J}{\mathrm{d}\mathbf{m}}\right)_k = -\int_\Omega \eta^*|_{t=0} \left(\frac{\mathrm{d}S}{\mathrm{d}\mathbf{m}}\right)_k \mathrm{d}x. \tag{4.6}$$

4.3.2. Source Model. There are many choices of bases for representing earthquake-tsunami sources. Typically, an array of subfaults is considered, each of whose dislocation is expressed in terms of at least one independent basis function. A linear combination over all such subfaults determines the overall fault dislocation, or the sea level perturbation due to it. As mentioned in the literature review, such arrays could include piecewise constants [**Pires and Miranda, 2001**], radial basis functions [**Saito et al., 2011**], or so-called Okada functions [**Okada, 1985**]. It should be noted that, unless the mesh resolution is high, the discontinuities associated with piecewise constants cannot be well resolved.

For the purpose of comparing continuous and discrete adjoint-based optimisation methods in Subsection 4.4.1, we adopt a radial basis formulation. Thereafter, the Okada basis is used in order to take account of the physical processes in the earthquake fault. Arrays of radial basis functions and piecewise constants were also considered in development work, but are not included here, for brevity. As may be expected, the radial basis representation was found to result in smooth timeseries profiles, whereas piecewise constant arrays led to more 'noise'. Another alternative would be to just invert for η_0, given a prior provided by Okada, say. This possibility will be investigated in future work.

Radial Basis. Following the source inversion of the Tōhoku tsunami due to [**Saito et al., 2011**], the radial basis functions considered take the form of rectangular Gaussians:

$$S^{\text{radial}}(\mathbf{m}) := \sum_{i=1}^{N} m_i \phi_i, \qquad \phi_i(x, y) = \exp\left(\frac{(x - x_i)^2}{L^2/4} + \frac{(y - y_i)^2}{W^2/4}\right), \tag{4.7}$$

where x is the coordinate direction parallel to the fault and y is perpendicular. Each subregion has length L and width W in the component directions and subregion R_i has centre (x_i, y_i). In this case, gradient formula (4.6) corresponds to (3.36) derived in Section 3.3.2.

4.3.2.1. *Okada Basis.* Unlike the radial basis approach, the Okada model aims at representing the dislocation of the Earth's crust due to an earthquake, as opposed to the ocean surface. It assumes that the crust may be described as a semi-infinite homogeneous elastic medium, bounded from above by the sea bed. Under such assumptions, the sea bed deformation due to an earthquake may be computed using a Greens function solution [**Okada, 1985**].

The Okada model is effectively a formula which constructs a dislocation field contribution on each subfault of an earthquake fault in terms of algebraic combinations of analytic functions involving nine control parameters. These are the length and width of the subfault, the latitude, longitude and depth of some point on the subfault (below the sea bed) and the associated *strike*, *slip*, *dip* and *rake*. Typically, subfaults are assumed to be

rectangular patches which exist on the fault plane, similarly to how the radial basis is used above. Assuming such an arrangement, the length, width, latitude, longitude and strike parameters are determined.

The Okada parameters are interpreted geometrically in Figure 4.3. The strike vector $\widehat{\mathbf{s}}$ determines the fault line in the horizontal. Strike is its angle from North. The fault plane is orthogonal to $\widehat{\mathbf{s}}$, with its angle below the horizontal given by the dip. As such, dip is an angle in the range $[0°, 90°]$. The slip vector determines the average displacement of the subfault. It is tangential to the fault plane. Its magnitude is what we refer to as slip and its angle above the horizontal (in the fault plane) is what we refer to as rake. Like strike, rake is an angle in the range $[-180°, 180°]$. For further details on the Okada model, see [**Okada, 1985**].

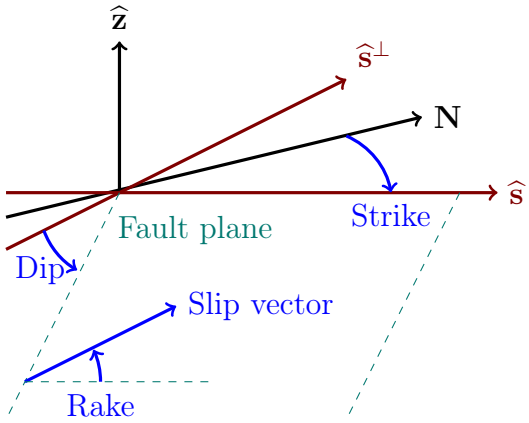


FIGURE 4.3. Diagram illustrating the slip vector, its magnitude and the strike, dip and rake angles of an earthquake fault.

As mentioned previously, it is common to assume that rapid displacement of the Earth's surface directly translates to a shallow water free surface displacement. As such, the deformation of the sea bed due to the Okada model may be translated to a deformation of the ocean surface. The nonlinearity of Okada functions implies that the gradient term in (4.6) is also a nonlinear function of the controls. However, the tsunami source S^{okada} is obtained from Okada function contributions across the array as a linear combination, meaning no extra nonlinearity is introduced. Note that, whilst each Okada function is related to an individual subfault, it is defined across the whole domain. Moreover, the function associated with each subfault is continuous, meaning the overall Okada function obtained by linear combination is continuous, too.

The implementation for the Okada model used in this chapter is given by the geophysics module *GeoClaw* [**LeVeque et al., 2011a**], of the finite volume library *ClawPack* [**Clawpack Development Team, 2020**]. Applied at the vertices of a mesh, it generates a dislocation field, which may be interpreted in $\mathbb{P}1$ space.

4.4. Inversion for a Known Source

Before attempting to invert for the Tōhoku tsunami source using real data, we begin by inverting for a known 'synthetic' source. That is, a source model and 'optimal' control parameters are specified and the model is run in order to generate synthetic timeseries data. The goal is to reconstruct the source using these data alone. An advantage of such experiments is that they remove much of the model error, because 'the truth' arose from the shallow water model we seek the solution with. Model error is not completely

removed for QoI (4.3), however, because it is evaluated in the area-averaged sense. In the quadrature version (4.4) we have a weighted sum of integrals of the form

$$I_g^{(k)} := \int_\Omega \widehat{\mathbb{1}}_g^\epsilon(\eta(\mathbf{x}, t^{(k)}) - \eta_g(t^{(k)}))^2 \,\mathrm{d}x, \qquad g \in \mathcal{G}, \tag{4.8}$$

where the superscript k indicates the time level. A necessary condition for $I_g^{(k)}$ to be zero is that η is constant over $B_\epsilon(\mathbf{x}_g)$. This is unlikely on coarse meshes, where ϵ needs to be taken as a fairly large value. However, on refined meshes with smaller choices of ϵ, it is feasible that $I_g^{(k)}$ could approach zero.

In order to avoid interpolation error between the $\mathbb{P}1$ dislocation field and the elevation space used for the tsunami propagation model, we choose a matching $\mathbb{P}1$ elevation space, so that the dislocation can be copied directly over. A $\mathbb{P}2$ space is used for the velocity, resulting in the Taylor-Hood element pair. Effectively, this is just given by dropping flux terms from the shallow water formulation (2.55) and enforcing the Dirichlet condition on the elevation in the strong sense. No stabilisation methods are applied.

4.4.1. Source Inversion for a Single Radial Basis Coefficient. We begin by inverting the *linear* shallow water equations for a tsunami source defined in a one dimensional basis, $V = \{\phi\}$, with the single (rectangular Gaussian) radial basis function ϕ as illustrated in Figure 4.4. Synthetic data are obtained using an 'optimal' basis coefficient $m^\star = 5$, which we seek to invert for, starting from some initial guess m_0. For the purposes of this experiment, take $m_0 = 10$. The control space is simply $C := \mathbb{R}$. In this experiment, we only invert timeseries at the near-field gauges (P02, P06 and 801–807) over a thirty minute period.

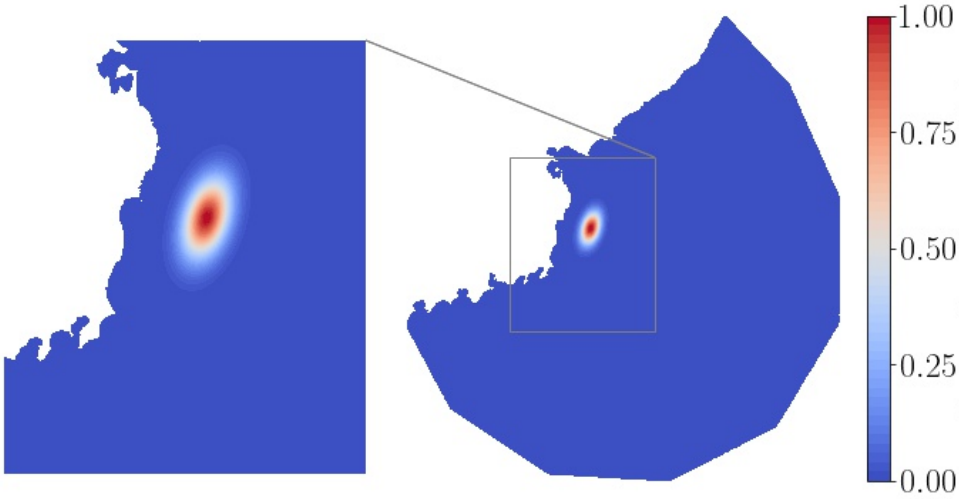


FIGURE 4.4. The rotated rectangular Gaussian basis function used in the numerical experiments presented in Subsection 4.4.1. Presented in a $\mathbb{P}1$ space defined on mesh $\mathcal{H}_3$.

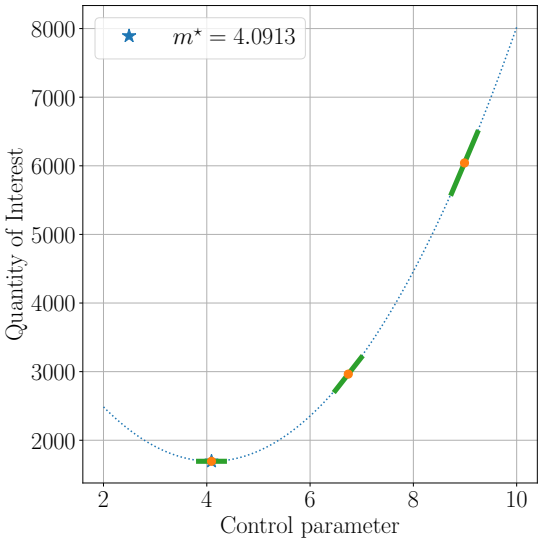
The linearity of the tsunami propagation and source models implies that the shallow water solution tuple is a linear function of the control parameter, at every time level. Therefore, the reduced functional is actually just a quadratic function of one variable. By sampling three points in the parameter space, it is possible for us to fit a quadratic interpolant and deduce the unique minimiser using elementary calculus. In addition, the solution, η', of

the TLM satisfies $\eta = m\,\eta'$, where η is the elevation due to control parameter m. As such, by formulae (3.35) and (3.38), the gradient and Hessian are simply

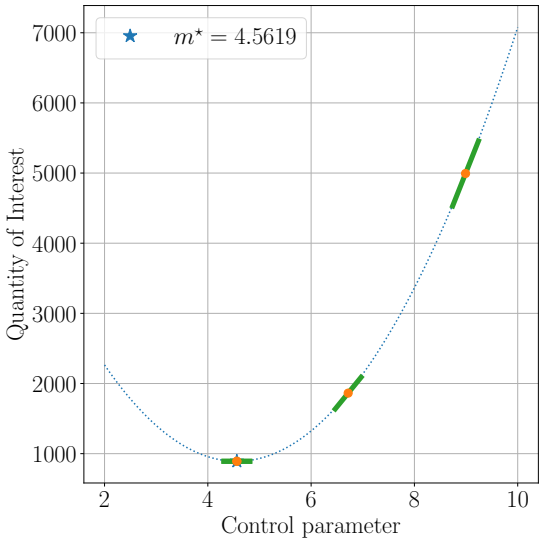
$$\text{(4.9)}\quad \mathrm{d}J_m = \frac{1}{m}\sum_{g\in\mathcal{G}}\int_0^T\int_\Omega \widehat{\mathbb{1}}_g^\epsilon\,(\eta-\eta_g(t))\,\eta\,\mathrm{d}x\,\mathrm{d}t, \qquad \mathrm{d}^2J_m = \frac{1}{m^2}\sum_{g\in\mathcal{G}}\int_0^T\int_\Omega \widehat{\mathbb{1}}_g^\epsilon\eta^2\,\mathrm{d}x\,\mathrm{d}t,$$

the latter of which is effectively the scaled sum over all squared gauge timeseries. Consequently, both functional derivatives may be computed using the forward solution alone. This is not generally the case for nonlinear problems or bases of dimension greater than one.

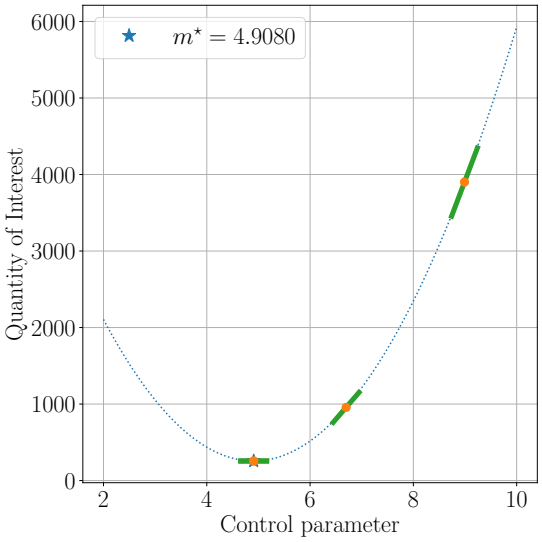
Figure 4.5 illustrates the progress of optimisation strategy based on the continuous adjoint method on $\mathcal{H}_0$, $\mathcal{H}_1$ and $\mathcal{H}_2$, overlaid on top of a plot of the corresponding quadratic control spaces. The discrete adjoint versions are not shown because they are indistinguishable by eye.



(A) Mesh $\mathcal{H}_0$



(B) Mesh $\mathcal{H}_1$



(C) Mesh $\mathcal{H}_2$

FIGURE 4.5. Progress of a continuous adjoint optimisation strategy for the inversion of a single basis coefficient on increasingly refined meshes. Orange circles show the evaluated QoI values during the optimisation routine. Green lines indicate the gradients computed using the adjoint models. Trajectories are superimposed on top of the (known quadratic) parameter space associated with each setup.

Note that the minimum over parameter space is smaller than the true optimum, $m^\star = 5$, in each case. This is because the extent to which the optimum of J^{PE} is approximated by that

of J^{AN} is mesh dependent. The fitted quadratic interpolants reveal the minima to be at 4.0913, 4.5619 and 4.9080 to four decimal places, for meshes $\mathcal{H}_0$, $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively. These values appear to be converging to the minimum of the discrete functional, $m^\star = 5$. Indeed, on mesh $\mathcal{H}_3$, fitting a quadratic through three points in the control space reveals the minima to be at 4.9745. Thus, we confirm that the area-averaged QoI tends to the point evaluation one with increased mesh resolution.

Subfigures 4.5a, 4.5b and 4.5c show gradients computed by the continuous adjoint method. The discrete adjoint versions are visually identical for the unstabilised Taylor-Hood discretisation used here; this would not necessarily be the case for other, stabilised methods. These gradients appear to be tangential to the control space. Moreover, they allow the optimisation method to successfully converge to the minimum of the discrete QoI in a small number of iterations. For closer inspection, Table 4.2 compares the gradients computed by the two different methods at both the initial guess and the (true) optimal control parameter value. Since we know the control space to be quadratic, we can also compare the three gradient approximations against the exact gradient given by differentiating the interpolant. Observe that the gradients due to both the discrete and continuous adjoint methods agree well with those generated by formula (4.9) in every case. The adjoint-free formula (4.9) is shown to be exact in every case, but is not extensible beyond this simple test case. In summary, Table 4.2 validates both the discrete and continuous adjoint approaches applied to this test case.

$m = 10$				
Mesh	Exact	Discrete adjoint	Continuous adjoint	Adjoint-free
$\mathcal{H}_0$	2141.2076	2141.2076	2141.2075	2141.2076
$\mathcal{H}_1$	2275.5510	2275.5510	2275.5510	2275.5510
$\mathcal{H}_2$	2228.0384	2228.0383	2228.0383	2228.0384
$m = 5$				
Mesh	Exact	Discrete adjoint	Continuous adjoint	Adjoint-free
$\mathcal{H}_0$	329.2901	329.2900	329.2901	329.2901
$\mathcal{H}_1$	183.3293	183.3292	183.3293	183.3293
$\mathcal{H}_2$	40.2428	40.2428	40.2428	40.2428

TABLE 4.2. Gradients at the initial ($m = 10$) and (true) optimal ($m = 5$) control parameter values, computed using three different methods. The 'adjoint-free' method refers to formula (4.9); it is specific to this test case.

Table 4.3 summarises the number of gradient evaluations and terminated line searches taken by each adjoint-based method. Overall CPU time is also stated, which does not include time taken to generate the 'data' or trace the forward model. Both methods are able to solve the optimisation problem efficiently. For the mesh resolutions considered here, the use of the continuous adjoint method appears to offer a modest reduction in computational cost. This is likely due to the fact that it instantiates a Firedrake `LinearVariationalSolver` object once and then repeatedly calls its `solve` method, whereas the discrete adjoint method instantiates a new solver at every timestep. The associated overhead diminishes as the resolution increases and the CPU time for the linear solves increase relatively.

In some cases, one method or the other takes an extra iteration near the minimum. This is presumably due to round-off error; whilst the continuous and discrete models should give

Mesh	Discrete adjoint J evaluations	dJ evaluations	Line searches	CPU time [s]
$\mathcal{H}_0$	5	5	4	40.876
$\mathcal{H}_1$	5	5	4	151.390
$\mathcal{H}_2$	4	4	3	739.020
Mesh	Continuous adjoint J evaluations	dJ evaluations	Line searches	CPU time [s]
$\mathcal{H}_0$	4	4	3	10.554
$\mathcal{H}_1$	5	5	4	82.572
$\mathcal{H}_2$	5	5	4	633.840

TABLE 4.3. Iteration counts and CPU time for continuous and discrete adjoint methods applied to a 1D tsunami source inversion problem. 'J evaluations' and (resp. dJ evaluations') refers to how many times the reduced functional (resp. gradient) are evaluated (implying a forward (resp. adjoint) solve) and CPU time is averaged over four runs.

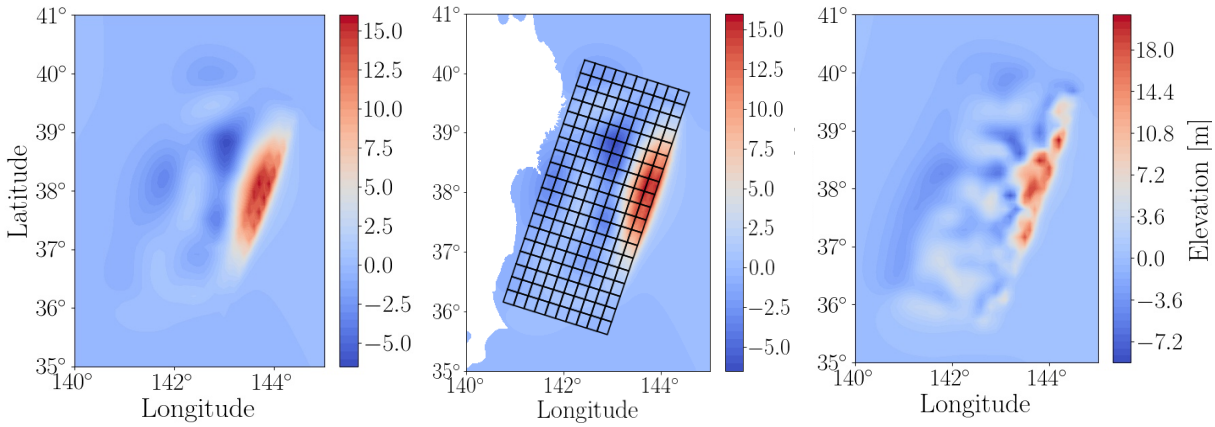
the same result analytically for this test case, the order in which they apply operations can differ, implying numerical differences.

To conclude the comparison, we find that there is not a great deal of difference whether the discrete or continuous adjoint is used for this particular problem. For the remainder of this section, we adopt the discrete adjoint method, which we seek to couple to another automatically differentiated model: the Okada source model.

4.4.2. Checking for Parameter Redundancy in the Okada model. In Subsection 4.4.1, both discrete and continuous adjoint-driven optimisation methods were tested in the context of a single radial basis function. Before moving on to test optimisation in the Okada basis, there is an issue that must be addressed.

For arrays of indicator functions, there is no issue with parameter redundancy, because of the non-overlapping support. Similarly, the exponential decay of the Gaussian basis functions ensures that the radial basis does not have this issue either. The Okada model, on the other hand, does come with the potential for different parameter configurations giving rise to very similar surface fields. The implication for inversion methods is the potential existence of multiple local minima and non-uniqueness of solutions to the global optimisation problem. In order to investigate this, we conduct an additional experiment which is concerned only with inverting the source model.

Consider the elevation profile illustrated in Subfigure 4.6a, which is defined on the 19×10 array of subfaults illustrated in Subfigure 4.6b. This source was generated from the one obtained in [**Shao et al., 2011**]. The 190 Okada functions are evaluated and their sum is interpolated onto a triangular mesh, $\mathcal{H}^{\text{okada}}$, covering the same region. The mesh has 50 elements in both longitudinal and latitudinal directions. In this thesis, we assume the entire fault to rupture instantaneously, whereas the authors of [**Shao et al., 2011**] allow for a (more realistic) time-dependent rupture, with varied rise and end times over the grid of subfaults. For that reason, the initial surface $\eta^\star$ which we generate from their data has an unrealistically high surface displacement. The initial surface $\eta^\star$ is the optimum, by construction. The Okada parameters were retrieved from [**Shao et al., 2020**].



(A) Target dislocation field (B) Subfault array (C) Initial guess

FIGURE 4.6. Target dislocation field for the Okada parameter redundancy experiment, the array of subfaults it is defined upon and an initial guess obtained by perturbing the Okada parameters associated with the target dislocation.

The experiment inverts the mean square error between some dislocation field η and the optimum for the slip and rake Okada parameters. That is, we have a QoI,

$$J(\eta;\mathbf{m}) := \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\eta_{ij}-\eta_{ij}^{\star})^2, \qquad \eta = S^{\text{okada}}(\mathbf{m}), \tag{4.10}$$

where $N = 51$ is the number of vertices of $\mathcal{H}^{\text{okada}}$ in each direction and η_{ij} and $\eta_{ij}^{\star}$ denote values at the $(i,j)^{th}$ vertex. Note that there is no PDE constraint for this optimisation. Instead, we are inverting the Okada model, for which the output is an algebraic combination of analytical functions of the control parameters, $\mathbf{m}$. Note also that the QoI is written in discrete form because we are currently only investigating the Okada model, which acts on vectors. Its output is later interpolated to provide a representation in a finite element space.

We choose the initial guess by perturbing the controls with random Normal noise:

$$\mathbf{m}_0 := \mathbf{m}^{\star} + \mathcal{N}(\mu,\sigma^2), \tag{4.11}$$

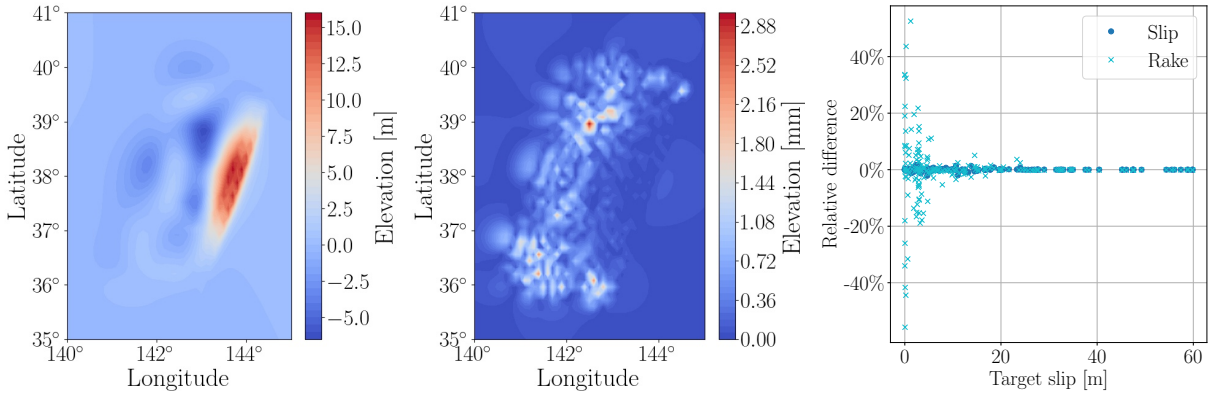
where $\mu = 0$ and σ denotes the standard deviation in the data associated with the relevant control. Since slip must be strictly positive, any perturbed slip values with negative sign are taken in modulus. In this experiment, we follow [**Shao et al., 2011**] in assuming constant strike and dip values of 198° and 10°, respectively, across all subfaults. Therefore, the 'active' control parameters are the slip and rake parameters on each subfault, giving rise to a $19 \times 10 \times 2 = 380$ dimensional control parameter space, $C := [(0,\infty)\times\mathbb{R}]^{190} \subset \mathbb{R}^{380}$. Subfigure 4.6c illustrates the perturbed initial guess.

In this work, derivatives of (4.10) are computed by applying the Python wrapper *PyADOL-C* [**Walter, 2014**] for the C++ operator overloading AD tool *ADOL-C* [**Griewank et al., 1996**] to the Okada model implementation found in GeoClaw. Doing so requires only minor modification of the `Fault` class in order to make annotations to tape.

The optimisation begins with an initial tracing of the annotated version of the Okada model. At each function evaluation step of the optimisation routine thereafter, the tape is replayed, which effectively reruns the model for a new set of input parameters. A flag is

raised to prepare for a reverse mode propagation; this is required due to the nonlinearity of the Okada model. At each gradient evaluation step, unity is propagated through the reverse mode of AD. The required gradient is then given by taking the transpose.

For an absolute tolerance $\texttt{tol} = 10^{-8}$ on the ℓ^∞ norm of the projected gradient, L-BFGS-B converges in 3,254 iterations. Given the initial guess shown in Subfigure 4.6c, Subfigure 4.7a shows the dislocation field after termination of the optimisation routine. It is difficult to distinguish differences between this plot and that shown in Subfigure 4.6a by eye. Thus, the absolute error is shown in Subfigure 4.7b, with units of millimetres.



(A) Optimised source (B) Absolute error (C) Distribution with slip

FIGURE 4.7. Optimised dislocation field, alongside the absolute error against the true optimum shown in Subfigure 4.6a, as well as a scatterplot of relative differences in control parameter values against target slip. Relative differences are computed based on the maximum absolute slip or rake value in the original field, as appropriate.

Whilst both the QoI error and the difference between the dislocation fields are very small, this is not necessarily the case for the control parameters. In order to investigate this, we plot the differences between the slip and rake parameters due to the inversion and the target in Subfigure 4.7c. The x-axis shows the corresponding slip value in the target field on each subfault. Differences are typically on the order of a few degrees. However, there are some parameters which remain 'out of tune', including one case where the rake differs by as much as 50.2°. The scatterplot reveals that the rake parameters are the ones which tend to differ most, with all slip parameters well matched. It may be observed from Subfigure 4.7c that the largest discrepancies occur when the target slip is small. This makes sense because if the slip is zero then the dislocation field will be zero, no matter the rake value. Indeed, slip and rake correspond to the magnitude and direction of the slip vector. In the limit of small magnitude, direction is poorly defined.

In conclusion, it is possible for a gradient-based inversion method to successfully recover dislocation fields by tuning the Okada parameters, but there are potential issues related to local minima and non-uniqueness. However, these issues are not particularly important because they tend to occur when slip values are small, whereby the rake value has little effect on the overall dislocation. As such, we may proceed with the optimisation for both slip and rake.

4.5. Inversion using Real Timeseries Data

The experiments conducted in Section 4.4 commit 'inverse crimes' in the sense that they involve inversions for synthetic solutions which were generated using the same forward model used within the optimisation routine. In this section, inverse crimes are excluded by inverting for real timeseries data collected by the pressure and GPS gauges described in Section 4.2. All experiments conducted in this section use the discrete adjoint method.

4.5.1. Gauge Data Post-Processing. Having obtained gauge data from the sources listed in Section 4.2, the timeseries are post-processed in a number of ways.

Firstly, they are truncated in order to only consider the time periods of interest between T_{start} and T_{end}, where T_{start} is just before the arrival of the tsunami wave. The values specified in this study are documented in Table 4.4. Truncation ensures that the inversion method does not 'overfit' to hydrodynamic features other than the tsunami wave, such as acoustic waves which are registered by some of the gauges. For gauges close to the shore, we cannot expect the shallow water model to be particularly accurate after the arrival of the tsunami wave. This is due the simple free-slip boundary conditions applied at the coasts, as opposed to the use of inundation modelling. Thus, a value $T_{\text{end}} < T$ is imposed in those cases.

Gauge category	Gauges	T_{start}	T_{end}
GPS	801, 802, 803, 804, 806, 807	0 min	60 min
Near field pressure	P02, P06	0 min	60 min
Mid field pressure	KPG1, KPG2, 21418	0 min	60 min
Far field pressure	21401, 21413, 21419	50 min	120 min
Southern pressure	MPG1, MPG2	80 min	120 min

TABLE 4.4. Time intervals of interest, $[T_{\text{start}}, T_{\text{end}}]$, related to five gauge categories used in the Tōhoku tsunami case study.

The second post-processing step is to apply a *de-tiding* algorithm to the timeseries. Doing so removes tidal constituents from the data, making them compatible with a simulation which does not include a tidal forcing. De-tiding is achieved using the Python re-implementation [**Bowman, 2020**] of the MATLAB package, *UTide* [**Codiga, 2020**]. The effect is most noticeable in the timeseries for the DART pressure gauges.

The final post-processing step is to filter out noise by sampling. Different levels of sampling are used for different timeseries, due to the varied measurement frequencies and noise levels in the data. Suppose a timeseries contains m observations in total and we wish to sample it every ℓ observations, where $\ell | m$. On the k^{th} subinterval, we average over the values from observation $k\ell$ to $(k+1)\ell$ and also over the times, yielding a single elevation-time pair. A linear interpolant is then constructed from the data.

Having applied these post-processing steps, we arrive at the timeseries shown by orange crosses in Figure 4.8.

4.5.2. Inversion Strategy. This subsection outlines a gradient computation strategy for the Okada basis.

Consider again a rectangular array of subfaults, which together constitute the earthquake fault. As discussed in Subsection 4.3.2, choosing the array in this way implies that the longitude, latitude, length, width and strike parameters are all fixed. In addition, we choose to fix the dip parameter at 10°, in line with [**Shao et al., 2011**]. Depth is also chosen in line with that work. Slip and rake values remain variable on each subfault; these are the parameters we seek to optimise. Note that the nonlinearity of the Okada source model means that the control space is not quadratic. Hence, we do not claim to have any a priori knowledge about the existence or uniqueness of any minima.

Recall that the discrete adjoint method implementation due to dolfin-adjoint uses UFL, meaning it is not able to differentiate the (pure Python) implementation of the Okada model due to GeoClaw. However, it is perfectly reasonable for us to differentiate the Okada model using a different AD tool. Again, the Python wrapper PyADOL-C is used.

In the operator overloading context, we always begin by tracing the forward model (in this case Okada), thereby recording its constituent operations to tape. One gradient computation method is given by propagating the active control parameters through the forward mode of AD for the source, using the resulting dislocation field to solve the tsunami propagation forward and adjoint problems and then finally assembling the gradient as in (4.6). Instead, we opt for a fully reverse mode approach, where the forward model is solved in its entirety and then unity is back-propagated in order to accumulate the gradient. Doing so implies that the cost of the gradient computation is (approximately) independent of the number of controls.

Let '`eta0`' denote the Firedrake `Function` used to represent the tsunami initial surface, which we take to be piecewise linear. Suppose that the Okada model is defined on a subset of the mesh vertices underlying the `P1` space, as indexed by '`indices`'. The forward propagation of a control vector '`m`' may be implemented (in serial) as

```
eta0 = Function(P1)
eta0.dat.data[indices] = adolc.zos_forward(tape_tag, m, keep=1)
J = rf_tsunami(eta0).
```

The ADOL-C command `zos_forward` stands for 'zero order scalar forward' mode, which amounts to replaying the tape identified by '`tape_tag`'. The flag '`keep=1`' is used to prepare for a reverse mode propagation. The dolfin-adjoint `ReducedFunctional` object, '`rf_tsunami`', enables the user to replay *its* tape with different input parameters and compute derivatives, amongst other functionalities. Note that `eta0` is automatically initialised to zero in the first line.

Reverse mode differentiation of the Okada model has already been presented in Subsection 4.4.2. Differentiating the QoI w.r.t. the Okada parameters amounts to applying the chain rule:

$$\frac{\mathrm{d}J}{\mathrm{d}\mathbf{m}} = \frac{\mathrm{d}J}{\mathrm{d}\eta_0}\frac{\mathrm{d}S^{\text{okada}}}{\mathrm{d}\mathbf{m}}, \qquad \eta_0 := S^{\text{okada}}(\mathbf{m}). \tag{4.12}$$

Written in code, it reads

```
dJdeta0 = rf_tsunami.derivative()
dJdS = dJdeta0.dat.data[indices]
dJdm = adolc.fos_reverse(tape_tag, dJdS).
```

The command `fos_reverse` stands for 'first order scalar reverse' mode. It computes the action of the transpose Jacobian of the source model on a single vector.

4.5.3. Experimental Results. In this subsection, we neglect the Coriolis force and solve the linear shallow water system as a wave equation with wavespeed $c = \sqrt{gb}$ (see (2.6)). Doing so helps to keep both the computational cost and memory requirements associated with the inversion to a minimum. From the continuity equation of the shallow water system, we deduce that $\eta_t|_{t=0} \equiv 0$, because $\mathbf{u}|_{t=0}$ is uniformly zero. The system is solved using explicit timestepping in the form

$$\left\langle \eta^{(k+1)} - 2\eta^{(k)} + \eta^{(k-1)}, \phi \right\rangle = -\Delta t^2 \left\langle c^2 \nabla \eta^{(k)}, \nabla \phi \right\rangle, \qquad \forall \phi \in V. \tag{4.13}$$

The linear system at each timestep is solved using GMRES with SOR as a preconditioner. An iterative approach is preferred over a direct method to avoid the memory costs associated with assembling the associated matrices on high resolution meshes. Note that a Neumann condition for the free surface is used on all boundary segments, in place of a free-slip condition for the velocity. The finite element space, V, is taken to be piecewise linear and continuous, for consistency with the output of the Okada model.

Initial Guess. Finally, we are in the position where two AD tools may be used to invert gauge data due to the Tōhoku tsunami. Slip and rake are optimised, starting with the uniform initial guess documented in Table 4.5.

Length	Width	Strike	Dip	Slip	Rake
25km	20km	198°	10°	1 m	90°

TABLE 4.5. Initial guess Okada parameters used when inverting for the Tōhoku tsunami source.

A uniform initial guess is chosen, given by a 1 m slip acting fully up-dip. The latitude-longitude grid and the associated lengths, widths and depths, as well as the strike and dip values are chosen based on those taken across the 19×10 subfault array presented in [**Shao et al., 2011**]. The resulting initial dislocation field is presented in Figure 4.8.

Generating the free surface perturbation due to the initial guess and then running the propagation model gives rise to the timeseries presented in Figure 4.8. Post-processed gauge data are represented by orange crosses. The solid blue curve depicts the simulated timeseries due to point evaluation, which is what we would ideally like to match the data. Instead, our optimisation method seeks to match the dotted blue curve to the data. For each gauge, $g \in \mathcal{G}$, this corresponds to the area-normalised integral of η over $B_\epsilon(\mathbf{x}_g)$. For this initial guess, the two simulated timeseries are almost identical, because the initial surface is near zero and therefore only causes small amplitude waves.

Source Inversion. In the following, we perform a sequence of inversions for gauge timeseries in different parts of the domain. Doing so verifies that we are able to recover the tsunami dynamics in different parts of the ocean in the Okada basis. Eventually, we invert for all gauge timeseries together.

To begin with, we seek to match the two Southern timeseries, MPG1 and MPG2. That is, the set of gauges, $\mathcal{G}$, used in the area-normalised QoI (4.3) is restricted to just these two. Neither of these is captured by the initial guess. This category is potentially challenging, because the gauges are partly sheltered by the coast and only have a 20cm peak.

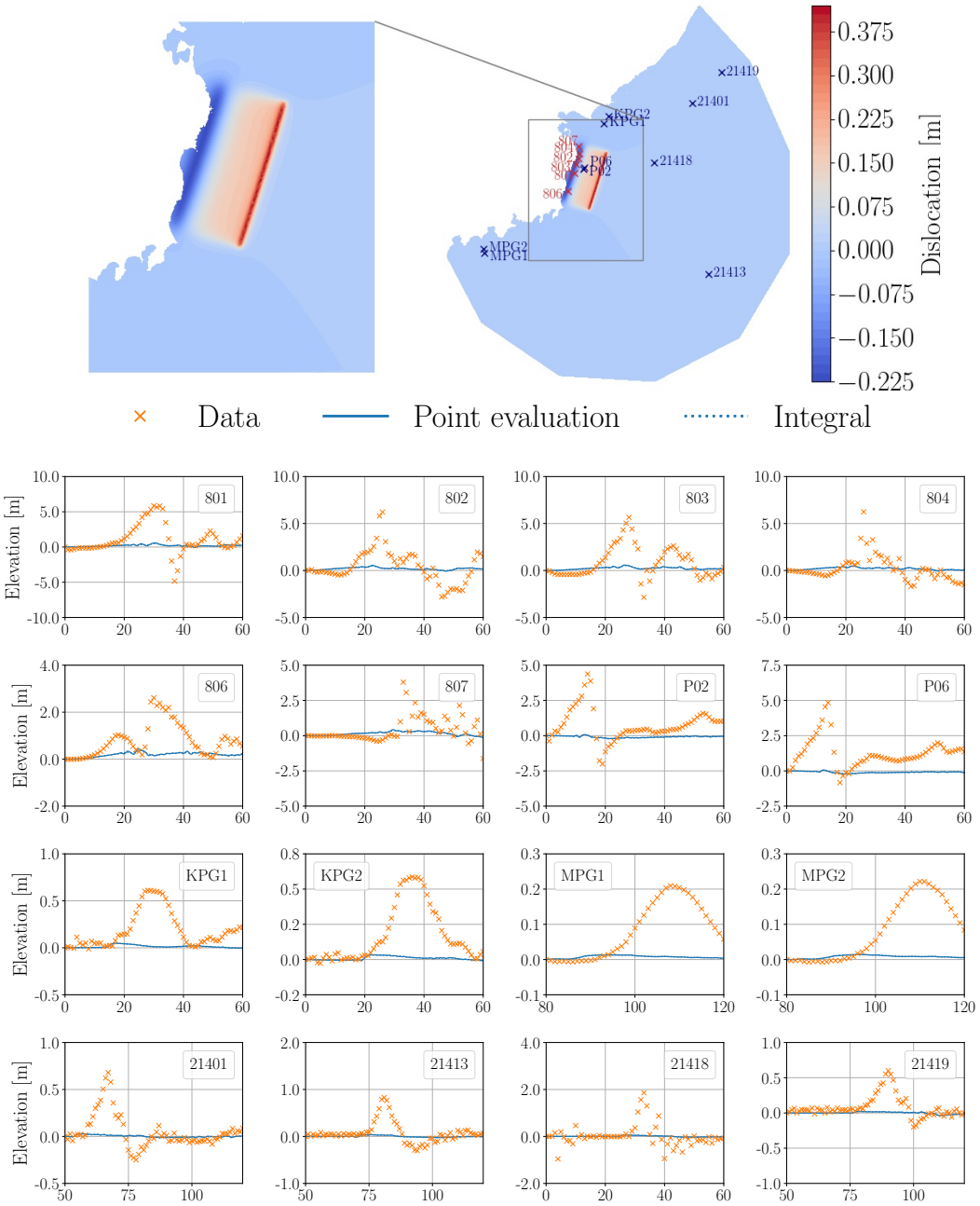


FIGURE 4.8. Dislocation field and simulated timeseries due to an initial guess with Okada parameters as listed in Table 4.5. The dotted blue curves due to the integral QoI are almost unidentifiable because they nearly match the solid blue curves due to the point evaluation version.

Inverting both the source and tsunami propagation models using the method outlined in Subsection 4.5.2, we obtain the dislocation field shown in Figure 4.9. Clearly, a high quality approximation is attained at these gauges. However, at the expense of doing so, the approximation quality is poor for all other gauges. This makes sense, because MPG1 and MPG2 are isolated from the other gauges, meaning that different parts of the tsunami wave are important for them than for the others. Besides, the optimisation routine has no knowledge of the other gauges. The fact that the Southern gauge timeseries are well captured by both point evaluated and area-normalised outputs validates the composed AD tool approach.

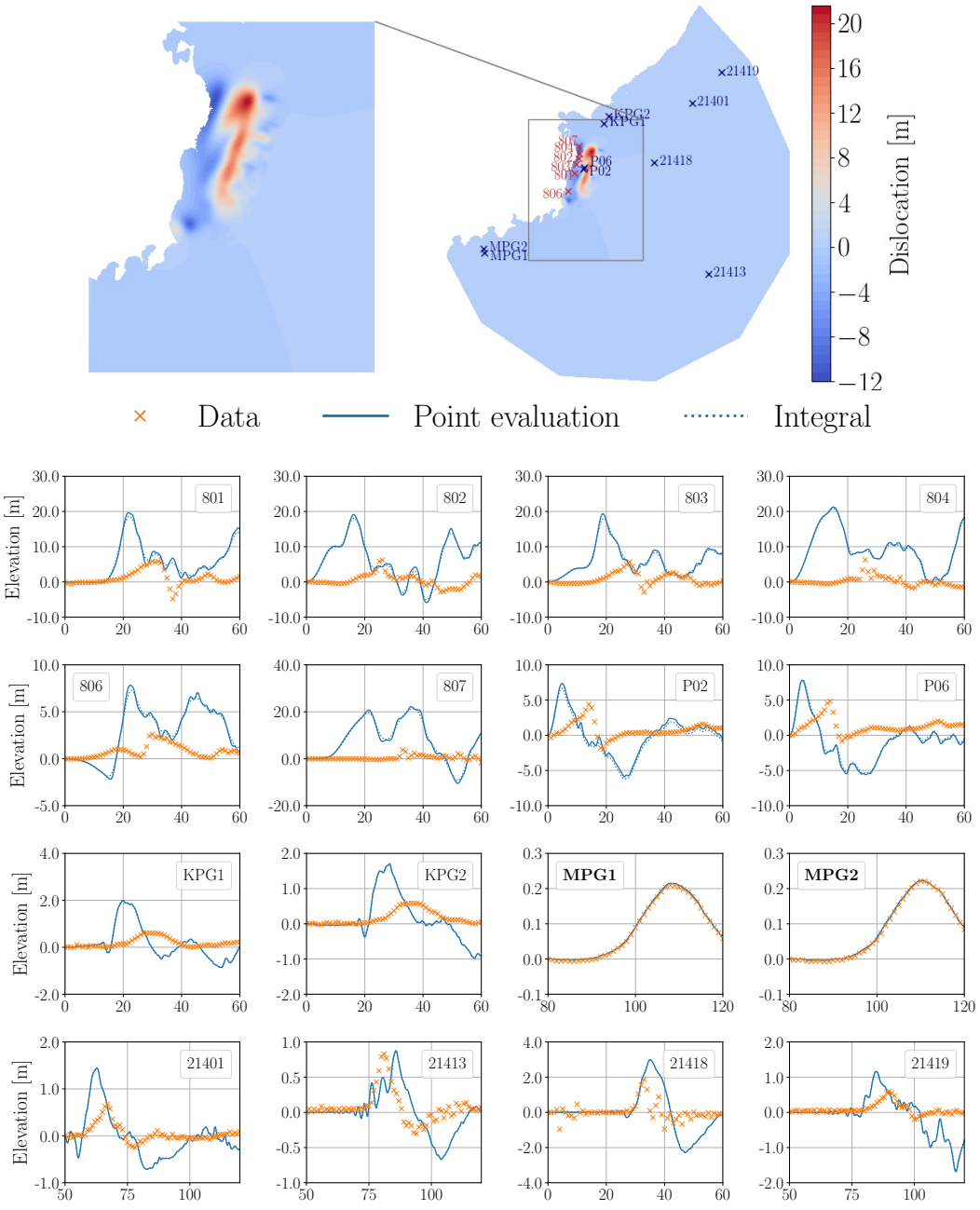


FIGURE 4.9. Dislocation field and simulated timeseries resulting from inversion for Southern gauges alone. The corresponding gauges are shown in bold.

Next, consider restricting $\mathcal{G}$ to the mid-field gauges, KPG1, KPG2 and 21418. Figure 4.10 illustrates effective capture of the major components of these timeseries. There are small scale perturbations in the data which are not resolved. However, these features are not of major concern for the purposes of this inversion experiment.

Similarly, restricting $\mathcal{G}$ to the far-field gauges, 21401, 21413 and 21419, gives rise to the timeseries shown in Figure 4.11. In this case, the initial tsunami wave and its arrival time are well matched. Further, the peaks at each of the gauges are well captured. Whilst the trajectory after eighty minutes is not as well matched, this is not of major concern; we are primarily interested in matching arrival times and peaks. Note that the dislocation

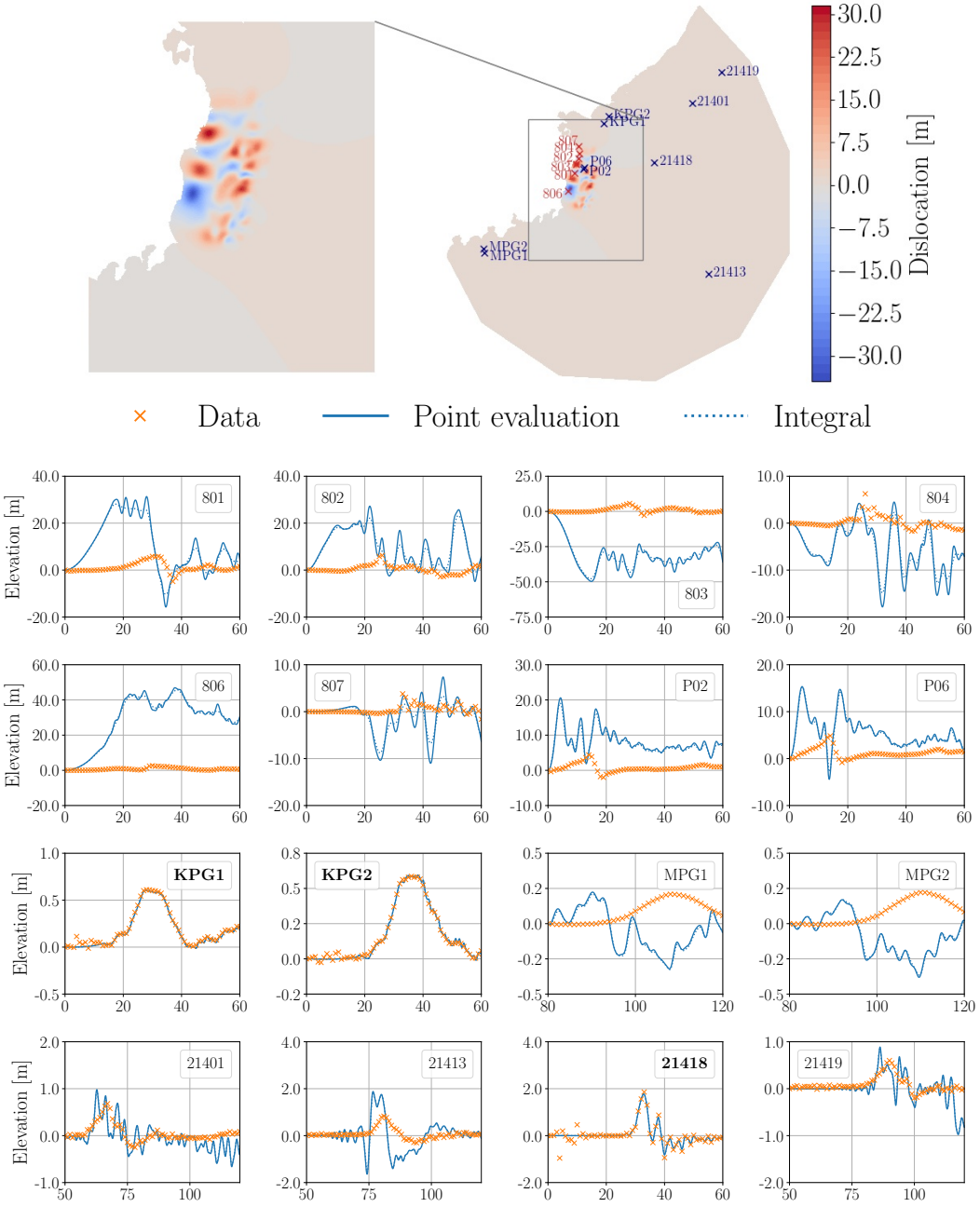


FIGURE 4.10. Dislocation field and simulated timeseries resulting from inversion for mid-field gauges alone. The corresponding gauges are shown in bold.

field due to this inversion is physically unrealistic, with values over 60m in one location near the coast. Indeed, none of the other timeseries are well matched at all.

Finally, we attempt to invert for all sixteen gauge profiles simultaneously. Since the different sets of gauges have amplitudes of varying sizes, ℓ^∞ normalisation is applied for this particular experiment. That is, the error contribution due to each gauge is divided by the square of the maximum free surface displacement reported in the data. If this were not applied, it is likely that the near-field gauges (which have maximum amplitudes around 5m) would be better matched than further field ones, such as MPG1 and MPG2 (which have maximum amplitudes around 20cm).

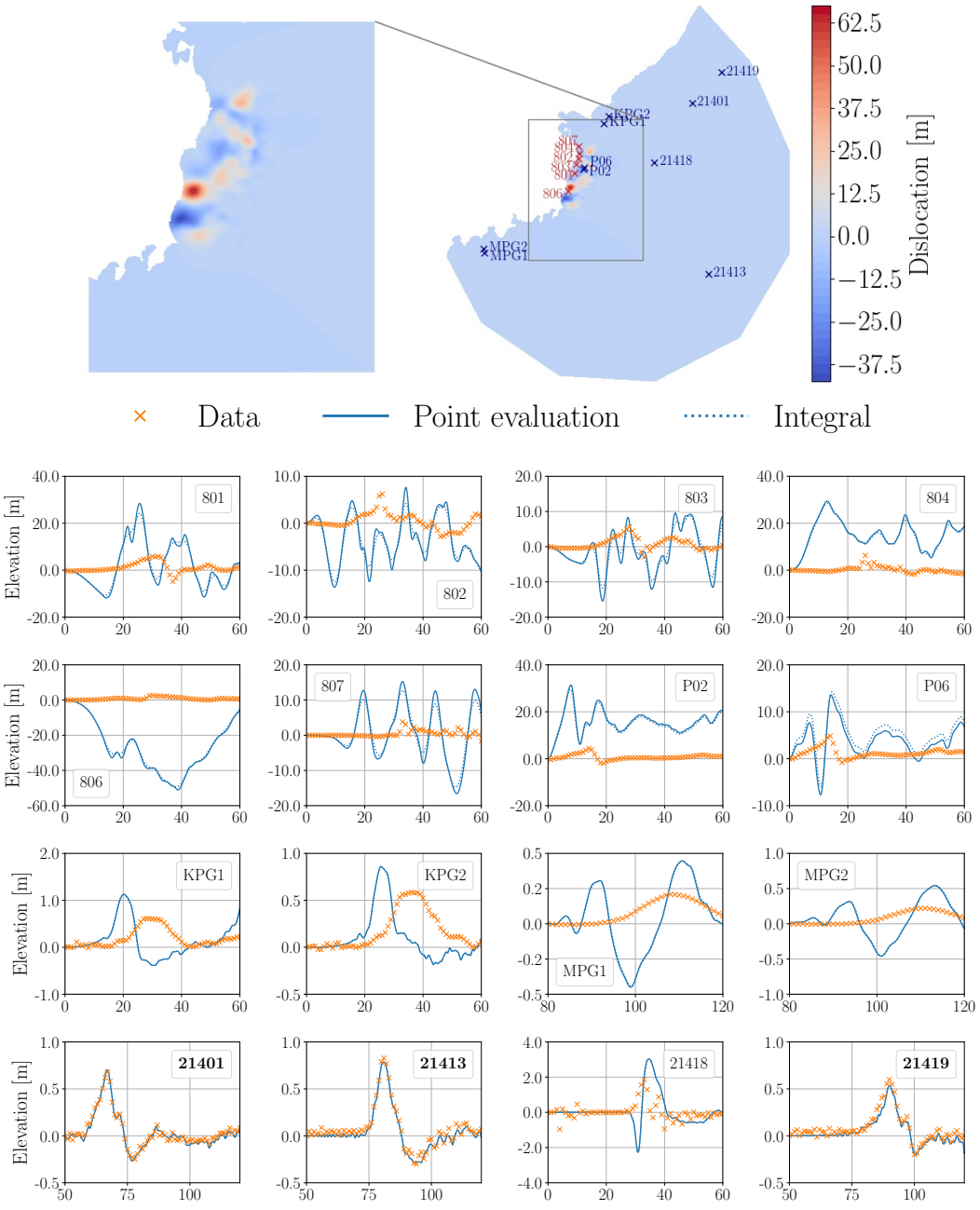


FIGURE 4.11. Dislocation field and simulated timeseries resulting from inversion for far-field gauges alone. The corresponding gauges are shown in bold.

When data due to all sixteen gauges is inverted, the fit of timeseries to data is not as tight as for the individual gauge categories considered above. This is most likely because the Okada model's range is not rich enough with all parameters except slip and rake fixed. This could potentially be addressed by inverting for other Okada parameters, such as depth, strike and dip, too. However, arrival times are fairly well matched for most of the gauges. In the cases of 803, 806, P02, MPG1, MPG2, 21401 and 21419, the peak amplitudes are also met. For the other gauges, the peak is not quite met.

The approximation quality at GPS gauges 802, 803, 804 and 807 is poorer than for the other gauges. There are a number of possible factors which could explain this. The first

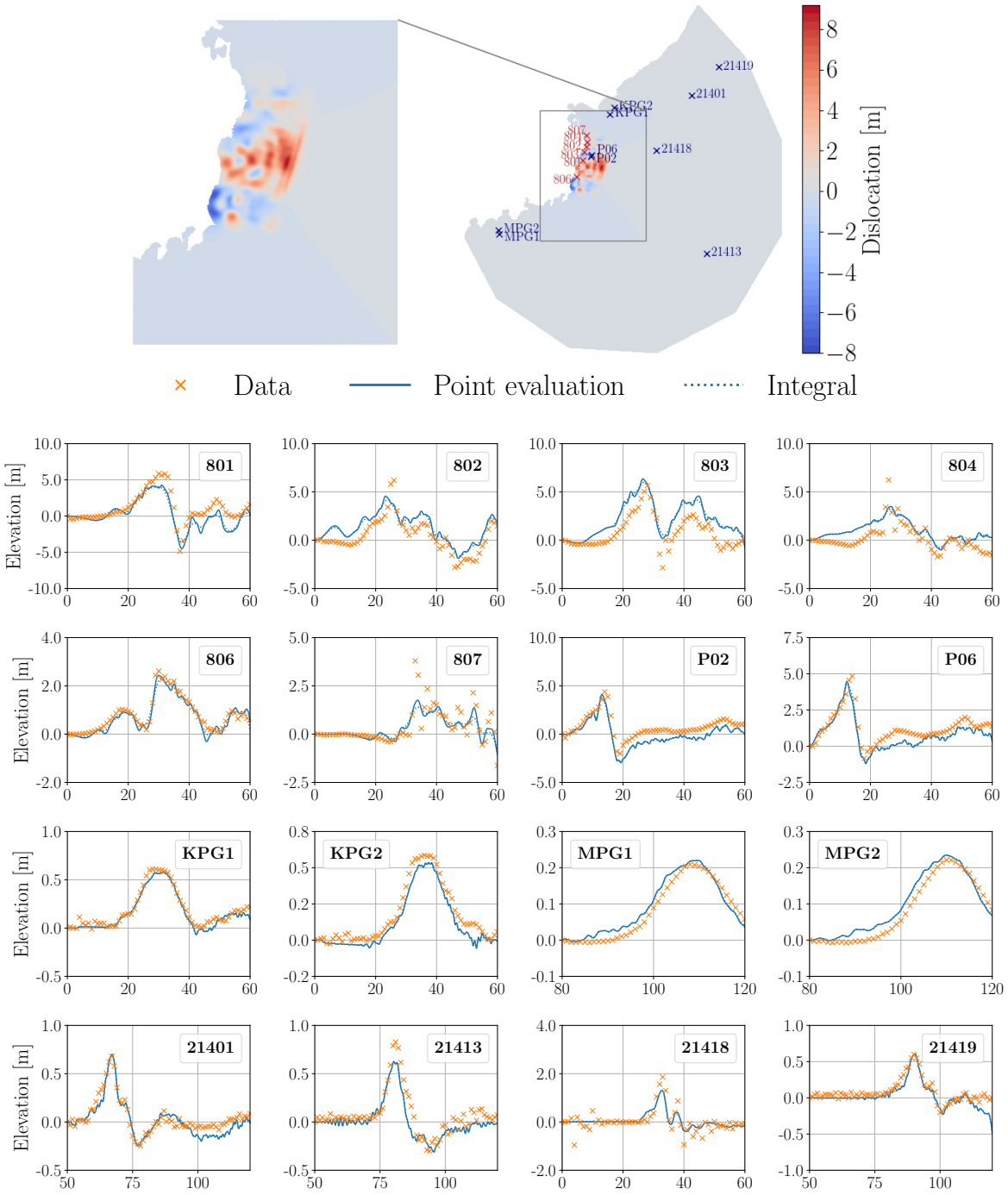


FIGURE 4.12. Dislocation field and simulated timeseries resulting from inversion for all gauges.

is the unrealistic Neumann condition on the free surface elevation imposed on the coastal boundary, which leads to artificial reflections. This is particularly impactful on these coastally situated gauges. Another possibility is that the initial guess is not sufficiently close to a solution which admits timeseries which are well matched at these gauges. After all, the gradient-based optimisation method used is a (quasi-)Newton method and therefore relies on being provided a suitable initial guess.

Whilst there are small scale differences, the dislocation field shown in Figure 4.12 bears some resemblance of the source generated by inverting dispersive linear shallow water equations using a least squares approach in [**Saito et al., 2011**]. That work is notable because it was the first to incorporate GPS and near-field pressure gauge data in an inversion study for the Tōhoku source. The region of highest displacement is similarly sized and located. Both the source in that work and the one presented here have a peak around 8m.

Summary. This subsection presents the first composition of two different AD tools applied to the task of tsunami source inversion. Despite the fact that the tsunami propagation is modelled using a simple wave equation, the discrete adjoint-driven inversion strategy is shown to find source parameters such that the resulting elevation profiles provide accurate matches for the timeseries data they seek to assimilate.

The approximation of the gauge data misfit timeseries using an area-normalised formulation – rather than point evaluation – is validated by the fact that the pairs of curves plotted in blue in Figures 4.8–4.12 are almost indistinguishable. However, it should be noted that this only holds for meshes which are sufficiently refined around the gauges.

The results presented in Figures 4.9–4.12 all assume constant strike and dip angles. It would be interesting to see if a better representation of the GPS gauge data is attainable when strike and dip are inverted for. This is proposed as future work. Before undergoing such a study, the parameter redundancy experiment presented in Subsection 4.4.2 should be extended to cover these variables. Further, given that the large scale geometry of the fault is known, it would make sense to constrain the strike and dip parameters accordingly.

Another aspect that can be deduced from the dislocation plots in Figures 4.9–4.11 (but not Figure 4.12) is that their magnitude is rather large, with maximum values over 60 m reported. Translated into a free surface perturbation, this is a very high displacement – much higher than the 8 m peak in the source generated by [**Saito et al., 2011**]. Whilst 60 m surface displacements are not impossible, they are inconsistent with the literature on this particular tsunami. One way to avoid artificially high magnitudes is to regularise the QoI. For example, an extra term could be added which penalises large initial surfaces. If this is done effectively then the initial surface due to the data assimilation will be less likely to take unrealistically large values. For details on regularisation techniques, we refer to [**Gunzburger, 2002**].

Large displacements are not observed in the final experiment which inverts all sixteen gauges. As such, regularisation is not necessary for that configuration. Indeed, the fact that the free surface heights determined in the full 16 gauge inversion are consistent with those found in the literature acts as further validation for the inversion strategy presented in this chapter.

Part 2

Mesh Adaptation

CHAPTER 5

Metric-Based Mesh Adaptation

Motivation. Classical h-adaptation methods typically make use of a hierarchical structure. Elements (or patches thereof) are tagged for refinement or coarsening based on whether the local value of an error indicator meets pre-specified tolerances. This approach allows for efficient implementations, such as the space-filling curve approach presented in [**Behrens and Zimmermann, 2000**]. Whilst hierarchical h-adaptation may most readily be applied to initially uniform meshes, which are effectively structured, many implementations are compatible with general unstructured meshes.

Hierarchical h-adaptation usually only allows for control of mesh element size, based on a scalar error indicator. In general, the aspect ratio of elements in an adapted mesh is inherited from that of the initial mesh, although it is possible to refine elements in an anisotropic manner. The *alignment* of elements, on the other hand, cannot be easily changed under hierarchical adaptation – alignment is not precisely the same thing as anisotropy. For strongly advection-dominated problems, it can be advantageous to have the facility to control element shape and orientation, as well as size. For example, if a tracer field is advected in a strongly directional velocity field then this can often lead to situations where its variation in cross-stream directions is much greater than the tangential component. In such a situation, alignment of anisotropic mesh elements with the flow allows for the possibility of attaining the same accuracy using fewer overall spatial DoFs. This is especially true if elements in the initial mesh exist on coordinate axes which do not align with the flow. In a hierarchical method, multiple refinements are required in order to remove 'shark tooth' imprinting (see Figure 5.12) of the initial mesh. If element orientation can be controlled, however, the same can be achieved using fewer elements. Further, if size, shape and orientation can all be controlled and a sufficient number of mesh adaptation iterations are applied, it is possible to remove dependence of the adapted mesh upon the initial mesh. To summarise, full anisotropic control of mesh adaptation implies a highly flexible approach.

It is for the reasons above that researchers began investigating anisotropic mesh adaptation in the 1990s. The anisotropic framework considered in this chapter, *metric-based mesh adaptation*, was first introduced in [**George et al., 1991**]. That work was published in the same year as other works on optimal anisotropic triangulation in terms of the linear interpolation of quadratic functions [**D'Azevedo, 1991**, **D'Azevedo and Simpson, 1991**]. Later works provided extensions to account for more general functions and a wide range of norms. A notable contribution is the continuous metric framework, introduced in [**Loseille and Alauzet, 2011a**] and [**Loseille and Alauzet, 2011b**]. An alternative, element-based approach also exists, introduced in [**Formaggia et al., 2004**].

In the continuous metric framework, the idea is to compute necessary geometrical quantities for mesh adaptation using a *Riemannian metric space*. Having done so, the ultimate aim of metric-based mesh adaptation is to generate a mesh whose elements are all 'unit' when viewed in the metric space. That is, their edges all have unit length. Given a unit

mesh, a number of interpolation error estimates may be deduced, such as those appearing in [**Frey and Alauzet, 2005**] and [**Alauzet and Olivier, 2010**]. By Céa's lemma (2.28), control of interpolation error implies control of approximation error, at least for elliptic problems solved using an appropriate, stable discretisation.

Recall from Section 1.1 that there are three steps in the adaptation process: 'analysis', 'assessment' and 'adaptation'. The metric-based framework provides a means of representing an error estimator or heuristic, so that it can be used to drive mesh adaptation, i.e. it is part of the 'assessment' phase. In this thesis, we focus on metrics constructed from error estimates. This is motivated by the concept of goal-oriented mesh adaptation, explored in Chapter 7, which relies on goal-oriented error estimates.

Suppose we have an error estimator related to the numerical solution of a PDE on some mesh. The assessment step involves constructing a metric so that the mesh resulting from the 'adaptation' step either: (a) enables the attainment of a pre-specified interpolation error level (see [**Pain et al., 2001**], for example); or (b) has a target number of vertices (see [**Loseille et al., 2010**], for example). It should be noted that condition (b) is achieved in an approximate sense, via the continuous analogue of vertex count. The number of vertices cannot be controlled directly by a metric – only within the adaptation step – although it can certainly be influenced. Under either approach (a) or (b), we need to establish the error estimator and express it as a metric.

Chapter Overview. This chapter is focused on introducing the metric framework, both in continuous and element-based formulations. A range of tools for making metrics relevant to practical problems are discussed, as well as methods for combining information due to different metrics. These tools are compared using numerical experiments. An investigation is made which demonstrates how the metric-based framework may be effectively used to generate anisotropic meshes using the Pragmatic mesh adaptation toolkit [**Barral et al., 2016**]. Improved convergence properties are presented, compared with uniform meshing, in terms of accuracy vs. computational cost.

Chapter Organisation. The chapter is organised as follows. The Riemannian metric framework is introduced mathematically in Section 5.1 and a simple, isotropic metric construction method is given in Section 5.2. Two methods for constructing anisotropic metrics from Hessian matrices are described in Section 5.3, along with a procedure for recovering the Hessian on the domain boundary. Sections 5.4, 5.5 and 5.6 describe details relating to the post-processing of Riemannian metrics by normalisation, combination and gradation, respectively. Algorithms for building mesh adaptation methods upon the metric-based framework are provided for both steady-state and time-dependent cases in Section 5.7. Finally, Hessian recovery, normalisation and combination strategies are compared using numerical experiments in Section 5.8. A time-dependent simulation using metric-based mesh adaptation is also demonstrated.

5.1. Riemannian Metric Fields

Meshes are inherently discrete objects, comprised of a finite number of mesh entities. However, it can be useful to formulate a continuous analogue of a mesh, particularly if we seek to apply gradient based-optimisation routines to adapt the mesh. In order to do so, we require differentiability of the mesh construction, which is not available in the discrete case. The continuous mesh framework may be understood using *Riemannian metric*

spaces and was developed in [**Loseille and Alauzet, 2011a**, **Loseille and Alauzet, 2011b**]. The underlying mathematics are introduced in the following subsections.

5.1.1. Metric Spaces. A *metric space* $(\mathcal{V}, \underline{\boldsymbol{M}})$ is comprised of a vector space $\mathcal{V} \subset \mathbb{R}^n$, equipped with a symmetric positive definite (SPD) matrix, $\underline{\boldsymbol{M}} \in \mathbb{R}^{n\times n}$. The matrix gives rise to an inner product and a norm,

$$\begin{aligned} \langle\cdot,\cdot\rangle_{\underline{\boldsymbol{M}}} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad \langle \mathbf{v}_1, \mathbf{v}_2\rangle_{\underline{\boldsymbol{M}}} := \mathbf{v}_1^T \underline{\boldsymbol{M}} \mathbf{v}_2, \quad \mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^n, \\ \|\cdot\|_{\underline{\boldsymbol{M}}} : \mathbb{R}^n \to [0,\infty), \quad \|\mathbf{v}\|_{\underline{\boldsymbol{M}}} := \sqrt{\langle \mathbf{v}, \mathbf{v}\rangle_{\underline{\boldsymbol{M}}}}, \quad \mathbf{v} \in \mathbb{R}^n. \end{aligned} \tag{5.1}$$

It also gives rise to distance function $d_{\underline{\boldsymbol{M}}} : \mathcal{V} \times \mathcal{V} \to [0,\infty)$, which may be represented using the norm as $d_{\underline{\boldsymbol{M}}}(\mathbf{v},\mathbf{w}) := \|\overrightarrow{\mathbf{vw}}\|_{\underline{\boldsymbol{M}}}$, where $\overrightarrow{\mathbf{vw}}$ denotes the vector between two points $\mathbf{v}, \mathbf{w} \in \mathcal{V}$. By construction, the distance function is symmetric, satisfies the triangle inequality and takes the value zero if and only if its arguments are equal.

A well-known example is n-dimensional *Euclidean space*, $\mathbb{E}^n := (\mathbb{R}^n, \underline{\boldsymbol{I}}_n)$, which is equipped with the ℓ^2 distance function, $d_2(\mathbf{v},\mathbf{w}) := d_{\underline{\boldsymbol{I}}_n}(\mathbf{v},\mathbf{w}) = \|\overrightarrow{\mathbf{vw}}\|_2$.

5.1.2. Matrix Decompositions and Coordinate Transformations. Due to its symmetry, $\underline{\boldsymbol{M}}$ has an orthogonal eigendecomposition,

$$\underline{\boldsymbol{M}} = \underline{\boldsymbol{V}}\,\underline{\boldsymbol{\Lambda}}\,\underline{\boldsymbol{V}}^T, \qquad \text{where} \qquad \underline{\boldsymbol{V}} = \begin{bmatrix} \mathbf{v}_1 & \dots & \mathbf{v}_n \end{bmatrix} \quad \text{and} \quad \underline{\boldsymbol{\Lambda}} = \operatorname{diag}(\lambda_1, \dots, \lambda_n) \tag{5.2}$$

are the corresponding eigenvector and eigenvalue matrices. Due to its positive-definiteness, the eigenvalues are all strictly positive. Therefore, fractional powers of $\underline{\boldsymbol{M}}$ are well defined and may be computed by taking powers of the eigenvalues. The transformation between Euclidean space and the metric space $(\mathbb{R}^n, d_{\underline{\boldsymbol{M}}})$ is given by multiplying by $\underline{\boldsymbol{M}}^{\frac{1}{2}}$. The inverse mapping is well defined due to positive-definiteness.

Decomposition (5.2) gives rise to a useful geometrical interpretation of the metric: as the image of the unit ball under the mapping from metric space to Euclidean space. The unit ball in a 2D metric space is shown on the RHS in Figure 5.1. Applying the inverse mapping from metric space to $\mathbb{E}^n$, the representation becomes an ellipse (or ellipsoid in 3D), whose major axes are specified by the eigenvectors. Magnitudes in each of those directions are specified by $\left\{ h_i := \lambda_i^{-\frac{1}{2}} \right\}_{i=1}^n$.

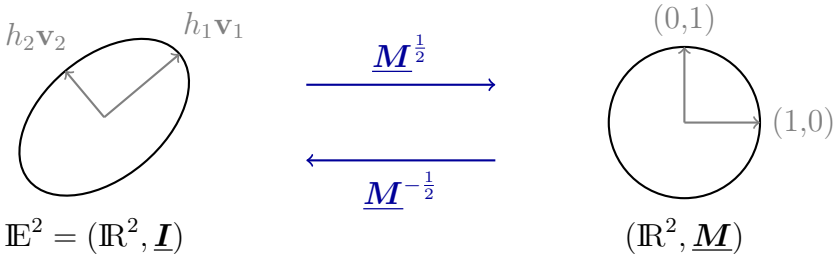


FIGURE 5.1. Ellipse representation of a metric in both Euclidean space and the metric space it defines.

Given an element K with volume $|K|_2$ in Euclidean space, its volume in the metric space $(\mathbb{R}^n, d_{\underline{\boldsymbol{M}}})$ is given by the transformation

$$|K|_{\underline{\boldsymbol{M}}} := \sqrt{\det \underline{\boldsymbol{M}}}\,|K|_2. \tag{5.3}$$

An alternative decomposition of $\underline{\boldsymbol{M}}$ can be derived in terms of the *density* and *anisotropy quotients*,

$$\rho := \prod_{i=1}^{n} \frac{1}{h_i} = \sqrt{\prod_{i=1}^{n} \lambda_i}, \qquad r_i := h_i^n \prod_{j=1}^{n} \frac{1}{h_j}, \qquad \forall i = 1:n. \tag{5.4}$$

Transforming a vector from $\mathbb{E}^n$ to $(\mathbb{R}^n, d_{\underline{\boldsymbol{M}}})$, the density ρ can be interpreted as a measure of the distortion of the vector's *magnitude*. The anisotropy quotients convey a sense of how its orientation is distorted in the direction associated with each eigenvector. When applied to volumes, the anisotropy quotients also convey a sense of shape distortion. Inserted into the eigendecomposition (5.2), definitions (5.4) yield

$$\underline{\boldsymbol{M}} = \rho^{\frac{2}{n}}\, \underline{\boldsymbol{V}}\, \mathrm{diag}\left(r_1^{-\frac{2}{n}}, \dots, r_n^{-\frac{2}{n}}\right) \underline{\boldsymbol{V}}^T. \tag{5.5}$$

Decomposition (5.5) is instructive because it separates out the components of $\underline{\boldsymbol{M}}$ which account for distortion to the size, shape and orientation of volumes.

5.1.3. Riemannian Metric Spaces. Subsection 5.1.1 defines a metric space using a (fixed) SPD matrix $\underline{\boldsymbol{M}} \in \mathbb{R}^{n\times n}$. Consequently, the distance function is the same across all of the domain. Now consider an SPD matrix *function*, $\underline{\boldsymbol{M}} : \Omega \to \mathbb{R}^{n\times n}$, which we denote by $\mathcal{M} = \{\underline{\boldsymbol{M}}(\mathbf{x})\}_{\mathbf{x}\in\Omega}$. Since the function is defined point-wise, we do not have a global concept of distance in the context of Euclidean geometry. Instead, the space defined by $\mathcal{M}$ exists in Riemannian geometry and distances must be calculated using parametric integration. Given the parametrisation $\boldsymbol{\gamma} : [0,1] \to \Omega$ of vector $\overrightarrow{\mathbf{vw}}$, defined by $\boldsymbol{\gamma}(\xi) = \mathbf{v} + \xi\overrightarrow{\mathbf{vw}}$, we have

$$d_{\mathcal{M}}(\mathbf{v},\mathbf{w}) := \int_0^1 \|\boldsymbol{\gamma}'(\xi)\|_{\underline{\boldsymbol{M}}(\boldsymbol{\gamma}(\xi))}\, \mathrm{d}\xi = \int_0^1 \sqrt{\overrightarrow{\mathbf{vw}}^T \underline{\boldsymbol{M}}(\mathbf{v}+\xi\overrightarrow{\mathbf{vw}})\overrightarrow{\mathbf{vw}}}\, \mathrm{d}\xi. \tag{5.6}$$

Thus, we obtain the *Riemannian metric space*, $(\mathbb{R}^n, \mathcal{M})$, with distance function (5.6). The notation $\ell_{\mathcal{M}}(\overrightarrow{\mathbf{vw}}) := d_{\mathcal{M}}(\mathbf{v},\mathbf{w})$ is introduced for the length of a vector.

Calculating volumes in the Riemannian metric space also amounts to integration:

$$|K|_{\mathcal{M}} := \int_K \sqrt{\det \underline{\boldsymbol{M}}(\mathbf{x})}\, \mathrm{d}x, \quad K \subset \Omega. \tag{5.7}$$

Other geometrical quantities such as angle may also be derived (see [**Loseille and Alauzet, 2011a**] for details).

Henceforth, we refer to the function $\mathcal{M} = \{\underline{\boldsymbol{M}}(\mathbf{x})\}_{\mathbf{x}\in\Omega}$ as a *Riemannian metric field*, or simply a *metric*.

5.1.4. Mesh Analogue. As described above, Riemannian metric spaces provide a continuous analogue for the discrete mesh. *Metric complexity* can be viewed as the continuous analogue of the number of mesh vertices. It can be computed using the formula

$$\mathcal{C}(\mathcal{M}) := \int_\Omega \sqrt{\det\left(\underline{\boldsymbol{M}}(\mathbf{x})\right)}\, \mathrm{d}x. \tag{5.8}$$

Another key concept is that of the *unit mesh*. If $\mathcal{H}$ is a unit mesh, then all of its elements should have unit edge lengths when mapped into the Riemannian metric space. That is, we have a uniform mesh of regular triangles or tetrahedra. As remarked in Subsection

2.4.2, it is generally impossible to tessellate a domain using such shapes, so we relax the definition to give a *quasi-unit* mesh:

$$\ell_{\mathcal{M}}(\boldsymbol{\gamma}) \in \left[\frac{1}{\sqrt{2}}, \sqrt{2}\right], \quad \forall \boldsymbol{\gamma} \in \partial K \quad \text{and} \quad Q_{\mathcal{M}}(K) \in [1, \alpha], \qquad \forall K \in \mathcal{H}. \tag{5.9}$$

As in Chapter 2, the notation $K \in \mathcal{H}$ interprets a mesh as a set of elements and the notation $\gamma \in \partial K$ interprets the boundary of element K as a set of edges. An admissible length interval $[\frac{1}{\alpha}, \alpha]$ is chosen to provide a convergence criterion for mesh adaptation schemes built upon the metric-based framework, with the choice $\alpha = \sqrt{2}$ ensuring symmetry about unity [**Loseille and Alauzet, 2011a**]. The second condition is an additional constraint, which ensures that elements do not have zero volume. Moreover, it ensures that a *quality function*, $Q_{\mathcal{M}}$, is bounded from above by some pre-specified $\alpha > 1$. For triangles and tetrahedra, the quality function takes the forms [**Loseille and Alauzet, 2011a**]

$$Q^{2D}_{\mathcal{M}}(K) := \frac{\sqrt{3}}{12} \frac{\sum_{\boldsymbol{\gamma} \in \partial K} \ell_{\mathcal{M}}(\boldsymbol{\gamma})^2}{|K|_{\mathcal{M}}}, \qquad Q^{3D}_{\mathcal{M}}(K) := \frac{\sqrt{3}}{216} \frac{\left(\sum_{\boldsymbol{\gamma} \in \partial K} \ell_{\mathcal{M}}(\boldsymbol{\gamma})^2\right)^{\frac{3}{2}}}{|K|_{\mathcal{M}}}. \tag{5.10}$$

A regular triangle or tetrahedron with unit sides has unit quality using the above formulae.

5.1.5. Element-Based Metric Formulation. The continuous metric formulation described in the previous subsections uses a point-wise characterisation. In the context of (discrete) meshes, it takes the form of piecewise linear and continuous ($\mathbb{P}1$) fields, i.e. the metric is defined at vertices. The authors of [**Formaggia et al., 2004**, **Micheletti et al., 2010**, **Carpio et al., 2013**] (among others) instead advocate an element-based ($\mathbb{P}0$) approach.

Recall the affine map (2.26) between the reference element, $\widehat{K}$, and a mesh element $K \in \mathcal{H}$. As an invertible map, it has a polar decomposition (2.67). Instead of relating to ellipsoid representations at a *point*, the eigenvectors now convey the principal directions of an ellipsoid which the vertices of *element* K all lie on. In particular, they relate to the ellipsoid obtained by transforming the unit ball under the affine map. The eigenvalues are then measures of length in each of the associated semi-axes. Without loss of generality (because the polar decomposition is not unique), assume $\lambda_{K,1} \geq \cdots \geq \lambda_{K,n} > 0$. Recall that Figure 2.3 illustrates the 2D transformation. Comparing with the point-wise ellipse representation in Figure 5.1, note that the magnitudes are defined by the eigenvalues directly, rather than their square root reciprocals.

Element shape may be characterised by so-called *stretching factors*. A general definition for the n-dimensional case is presented in [**Micheletti et al., 2010**]:

$$s_{K,i} := \left(\prod_{j=1,\, j\neq i}^{n} \lambda_{K,j}\right)^{-\frac{2}{n}} \lambda_{K,i}^{2(n-1)/n} = \left(\prod_{j=1}^{n} \lambda_{K,j}\right)^{-\frac{2}{n}} \lambda_{K,i}^2, \qquad i = 1:n. \tag{5.11}$$

In the 2D case, for example, we have $s_{K,1} = \lambda_{K,1}/\lambda_{K,2}$ and $s_{K,2} = \lambda_{K,2}/\lambda_{K,1}$. The stretching factors convey element shape deformation, compared with the isotropic case where $s_{K,i} = 1$, $\forall i = 1:n$. By construction, $s_{K,1} \geq s_{K,2} \geq \cdots \geq s_{K,n} > 0$, meaning that the first stretching factor conveys the greatest deformation.

The element-based framework has a number of interpolation error bounds involving the element-wise Hessian $\underline{\boldsymbol{H}}_K$, the first of which was published in [**Formaggia and Perotto,**

2001]:

$$\|u - \Pi_h u\|_K \leq C \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_{K,i}^2 \lambda_{K,j}^2 \int_K (\mathbf{v}_{K,i}^T \underline{\boldsymbol{H}}_K(u)\, \mathbf{v}_{K,j})^2 \,\mathrm{d}x}, \tag{5.12}$$

for some constant $C = C(K) > 0$. Here Π_h is the linear Lagrange interpolant, meaning that (5.12) only strictly applies to $\mathbb{P}1$ and $\mathbb{P}1_{DG}$ finite element discretisations. Further interpolation errors have been derived, such as interpolation errors on element boundaries [**Formaggia et al., 2004**], estimators related to gradient recovery [**Micheletti et al., 2010**] and goal-oriented error estimates [**Carpio et al., 2013**]. Recall that error estimates related to interpolation errors are motivated by Céa's lemma (2.28).

It is possible to perform metric-based mesh adaptation purely using element-wise metrics, such as those derived from the Hessian metric on an element. However, since the metric-based mesh adaptation toolkit used in this thesis requires Riemannian metrics defined at mesh vertices, $\mathbb{P}0$ metrics derived in the element-wise formulation are projected into $\mathbb{P}1$ space. The element-wise formulation is only ever used in this thesis in order to make use of special goal-oriented metric formulations (see Subsections 7.4.1 and 7.4.2). In all other situations, a discretised version of the continuous formulation is used, i.e. the metric is always computed in a $\mathbb{P}1$ space.

5.2. Isotropic Metric

The simplest way to use scalar error estimators to drive a metric-based mesh adaptation routine is to generate an *isotropic metric*, as described in the following.

For an isotropic metric, the eigenvalues $\{\lambda_i\}_{i=1}^n$ in (5.2) are identical, taking some value, $\lambda > 0$. The same applies to the anisotropy quotients and magnitudes in each principal direction. In particular, the anisotropy quotients collapse to unity, as one might expect. Therefore, the isotropic metric is defined exclusively by the density. Under the assumption of isotropy, (5.5) becomes

$$\mathcal{M} = \rho^{\frac{2}{n}}\, \underline{\boldsymbol{I}_{\boldsymbol{n}}} = \lambda\, \underline{\boldsymbol{I}_{\boldsymbol{n}}}, \tag{5.13}$$

where $\underline{\boldsymbol{I}_{\boldsymbol{n}}}$ is the n-dimensional identity matrix.

Suppose $\mathcal{E}$ is a global error estimator, which may be calculated as a sum over all mesh elements as

$$\mathcal{E} := \sum_{K \in \mathcal{H}} \mathcal{E}_K, \tag{5.14}$$

where $\mathcal{E}_K$ is an *error indicator* associated with element K. Multiplying each error indicator in (5.14) by the corresponding indicator function, $\mathbb{1}_K$, we obtain a piecewise constant field. This error indicator field reveals local contributions to the estimated error $\mathcal{E}$.

A vertex-wise isotropic metric may be obtained by projection of this $\mathbb{P}0$ field into piecewise linear space using a projection $\Pi_1 : \mathbb{P}0 \to \mathbb{P}1$:

$$\mathcal{M}_{\mathcal{E}}^{\text{isotropic}} := \Pi_1 \left(\sum_{K \in \mathcal{H}} \mathcal{E}_K \mathbb{1}_K \right) \underline{\boldsymbol{I}_{\boldsymbol{n}}}. \tag{5.15}$$

One approach is to use an L^2-projection, which amounts to a volume-average over cells adjacent to a vertex. To use a metric such as (5.15) in practice, normalisation should be applied, as described in Section 5.4.

5.3. Hessian-Based Metric

The metric construction described in Section 5.2 allows the control of element size in accordance with some user-specified error estimate. However, as motivated at the beginning of the chapter, we seek a reliable method for constructing *anisotropic* metrics, whereby element shape and orientation are also controlled.

A common way to construct an anisotropic metric is to base it on the Hessian of a field (or fields) relating to the PDE solution. From a heuristic point of view, such an approach is attractive because it means controlling mesh resolution according to curvature. Extra resolution is deployed surrounding sharp gradients, enabling the better capturing thereof. Further, the mesh anisotropy is aligned with the curvature, meaning that resolution is not wasted in orthogonal directions.

Provided the underlying field is sufficiently smooth, a Hessian is already symmetric. Therefore, to obtain a valid metric, it is sufficient to take the Hessian's eigenvalues in modulus. We justify taking the absolute value by noting that it is the magnitude, rather than sign, of errors which is most important.

As well as attractive properties of Hessian metrics from the heuristic point of view, there are also a number of connections to error estimates. Given a sufficiently smooth scalar field u, suppose its Hessian can be approximated as $\underline{\boldsymbol{H}}$. From the pioneering work of [**Ciarlet, 1978**], the following bound may be deduced for elliptic problems:

$$\|u - \Pi_h u\|_K \leq C \|\underline{\boldsymbol{J}}_K\|_2^2 \|\underline{\boldsymbol{H}}\|_K, \qquad K \in \mathcal{H}, \tag{5.16}$$

where $C > 0$ is a constant and the cell Jacobian, $\underline{\boldsymbol{J}}_K$, is as defined in (2.26). This provides a clear relation between interpolation error and the Hessian. Further, the authors of [**Frey and Alauzet, 2005**] extended the result to give an L^∞ bound for interpolation into a piecewise linear space:

$$\|u - \Pi_h u\|_{L^\infty(K)} \leq \zeta \max_{\mathbf{x}\in K} \max_{\boldsymbol{\gamma}\in\partial K} \boldsymbol{\gamma}^T \left|\underline{\boldsymbol{H}}(\mathbf{x})\right| \boldsymbol{\gamma}, \qquad K \in \mathcal{H}, \tag{5.17}$$

where edges $\boldsymbol{\gamma}$ are interpreted as vectors between vertices. Here $\zeta > 0$ is a constant related to the spatial dimension.

Suppose also that we would like to adapt $\mathcal{H}$ in such a way that the representation $\Pi_h u$ of u has L^∞ interpolation error smaller than some $\epsilon > 0$. A metric suitable metric to achieve this aim is given by the formula [**Pain et al., 2001**]

$$\underline{\boldsymbol{M}}_\epsilon^{\text{Hessian}}(\mathbf{x}) := \frac{\zeta}{\epsilon} \left|\underline{\boldsymbol{H}}(\mathbf{x})\right|. \tag{5.18}$$

In general, the constant $\zeta > 0$ is both difficult to compute and unimportant with regards to the adaptation routine. In the work of [**Pain et al., 2001**] it is recommended to simply set $\zeta := 1$.

Further relations between interpolation error and Hessian metrics were later established within the continuous metric formulation. Suppose Π_h interpolates into a finite element

space defined on a mesh which is unit w.r.t. some metric which has complexity $\mathcal{C}_T$. Then, for any $p \in [1, \infty)$, the global L^p interpolation error is bounded from above by the following expression involving the Hessian [**Alauzet and Olivier, 2010**]:

$$\|u - \Pi_h u\|_{L^p(\Omega)} \leq n\mathcal{C}_T \left(\int_\Omega \det(|\underline{\boldsymbol{H}}|)^{\frac{p}{2p+n}} \,\mathrm{d}x \right)^{\frac{2p+n}{np}}. \tag{5.19}$$

The corresponding result in the L^∞ norm is given by taking the limit $p \to \infty$. For the form of the metric corresponding to (5.19), see Subsection 5.4.1.

In order to derive bounds of the form (5.19), the authors of [**Loseille and Alauzet, 2011a**] introduce a *continuous interpolant* $\pi_{\mathcal{M}}$ which interpolates into the metric space. It is analogous to the interpolant Π_h onto the discrete mesh. For a solution field u, the L^p continuous interpolation error may be expressed in terms of its Hessian as

$$\|u - \pi_{\mathcal{M}} u\|_{L^p(\Omega)} = \left(\int_\Omega \mathrm{trace}\left(\underline{\boldsymbol{M}}(\mathbf{x})^{-\frac{1}{2}} \, |\underline{\boldsymbol{H}}(\mathbf{x})| \, \underline{\boldsymbol{M}}(\mathbf{x})^{-\frac{1}{2}} \right)^p \,\mathrm{d}x \right)^{\frac{1}{p}}. \tag{5.20}$$

The most important thing to note is that the continuous interpolation error bounds the discrete interpolation error from above, provided the underlying mesh is unit w.r.t. the associated metric. As such, control of interpolation error in the continuous metric formulation implies control of interpolation error on the discrete mesh.

5.3.1. Hessian Recovery. Suppose we seek to perform Hessian-based mesh adaptation based on a scalar field, u. In practice, this field is represented in a discrete sense as u_h. Under finite element discretisations, direct differentiation may be applied to obtain discontinuous derivative fields. However, derivatives are uniformly zero if the polynomial order is not high enough. Moreover, a continuous representation is often required. Instead, the Hessian may be constructed using a *recovery* technique.

Suppose that $u \in V \subset C^2(\Omega)$ is suitably smooth, with gradient and Hessian denoted by

$$\mathbf{g} = \nabla u, \qquad \underline{\boldsymbol{H}} = \nabla\nabla u. \tag{5.21}$$

We seek to approximate $\underline{\boldsymbol{H}}$ in a finite dimensional $n \times n$ tensor function space, Σ_h.

L^2 projection. One recovery approach is to solve a mixed finite element problem, effectively recovering the gradient in a vector function space W_h as an intermediate step [**Pain et al., 2001**]:

$$\int_\Omega \boldsymbol{\psi} \cdot \mathbf{g}_h \,\mathrm{d}x = -\int_\Omega \mathrm{div}(\boldsymbol{\psi}) u_h \,\mathrm{d}x, + \int_{\partial\Omega} (\boldsymbol{\psi} u_h) \cdot \widehat{\mathbf{n}} \,\mathrm{d}s, \quad \forall \boldsymbol{\psi} \in W_h, \tag{5.22}$$

$$\int_\Omega \underline{\boldsymbol{\Sigma}} : \underline{\boldsymbol{H}}_h \,\mathrm{d}x = -\int_\Omega \mathrm{div}(\underline{\boldsymbol{\Sigma}}) \cdot \mathbf{g}_h \,\mathrm{d}x + \int_{\partial\Omega} (\underline{\boldsymbol{\Sigma}}\, \mathbf{g}_h) \cdot \widehat{\mathbf{n}} \,\mathrm{d}s, \quad \forall \underline{\boldsymbol{\Sigma}} \in \Sigma_h. \tag{5.23}$$

The first equation (5.22) can be solved independently, giving rise to an L^2 projection recovery method for the gradient. That is, it provides a weak formulation of the first equation in (5.21).

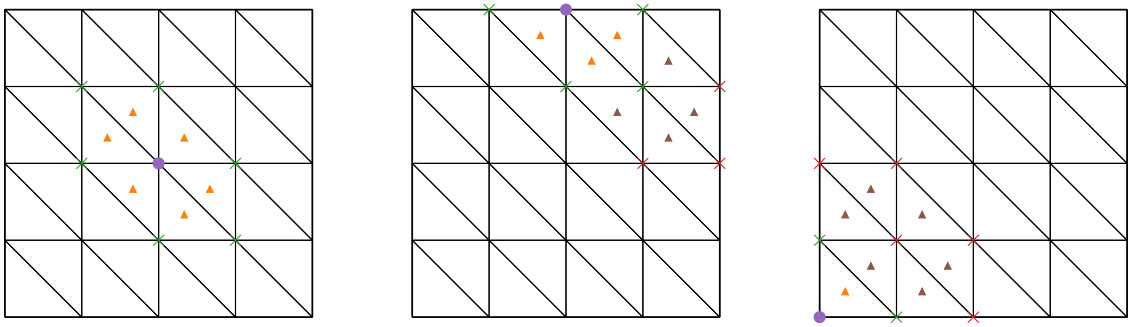
The second equation (5.23) effectively applies this method again to recover a Hessian, providing a weak formulation for the second equation in (5.21). As such, we refer to this approach as *double L^2 projection*.

Note that the formulation of (5.22)–(5.23) does not require the finite element approximation u_h to have any derivatives at all, meaning it can be used to recover gradients and

Hessians from $\mathbb{P}0$ approximations. In fact, u_h could even be replaced by an expression, such as an analytical expression involving the mesh coordinates.

Zienkiewicz-Zhu. An alternative approach was presented in [**Zienkiewicz and Zhu, 1992a**]. The gradient recovery technique described therein is commonly known as *Zienkiewicz-Zhu (Z^2)*, after the authors. The method is essentially heuristic, but has been shown to yield $\mathcal{O}(h^4)$ convergence of ℓ^2 errors at interior vertices and can be modified to account for boundary vertices, albeit with a reduced convergence rate. Hessian recovery involves repeated application of the gradient recovery technique. As such, it suffices to discuss gradient recovery.

Unlike the L^2 projection technique above, which solves a PDE over the whole domain, the Z^2 method uses a local, 'patch-wise' construction. Given a vertex in the domain interior, the 'patch' of surrounding elements looks as in Subfigure 5.2a. Assuming $u_h \in \mathbb{P}1$, the idea of the Z^2 method is to construct a linear gradient based on its finite element gradient, $\nabla u_h \in (\mathbb{P}0)^n$. This is done by sampling the finite element gradient at the centroids of the patch elements.



(A) Interior vertex (B) Boundary vertex (C) Corner vertex

FIGURE 5.2. Examples of patches around vertices in a uniform mesh used for Zienkiewicz-Zhu gradient recovery. The vertex in question is shown by a purple circle and its neighbouring vertices by green crosses. Vertices in the extended patch are shown as red crosses. Centroids of neighbouring elements and elements in the extended patch are shown by orange and brown triangles, respectively.

For each vertex, v, we have a least squares problem. It involves finding coefficients $\mathbf{g}^v$ which minimise

$$F(\mathbf{g}^v) = \sum_{i=1}^{k} (\underline{\boldsymbol{P}}(\mathbf{x}_i^v)\mathbf{g}^v - \nabla u_h(\mathbf{x}_i^v)) \cdot (\underline{\boldsymbol{P}}(\mathbf{x}_i^v)\mathbf{g}^v - \nabla u_h(\mathbf{x}_i^v)), \quad \underline{\boldsymbol{P}}(\mathbf{x}) := \begin{bmatrix} 1 & \mathbf{x} \end{bmatrix}, \tag{5.24}$$

where $\{\mathbf{x}_i^v\}_{i=1}^k$ is the set of centroids associated with elements in the patch. This may be represented as the matrix-matrix linear system

$$\underline{\boldsymbol{A}}^v\, \mathbf{g}^v = \begin{bmatrix} \mathbf{b}_1^v, \dots, \mathbf{b}_N^v \end{bmatrix}, \quad \underline{\boldsymbol{A}}^v := \sum_{i=1}^{k} \underline{\boldsymbol{P}}(\mathbf{x}_i^v)^T \underline{\boldsymbol{P}}(\mathbf{x}_i^v), \quad \mathbf{b}_j^v := \sum_{i=1}^{k} \underline{\boldsymbol{P}}(\mathbf{x}_i^v)^T \frac{\partial u_h}{\partial x_j}(\mathbf{x}_i^v), \tag{5.25}$$

where the derivative should be understood in the finite element sense. If u is scalar then $N = n$ and the recovered gradient at vertex v is given by $\mathbf{g}^v \in \mathbb{R}^N$. If it is vector – as

when recovering a Hessian from a recovered gradient – then $N = n^2$ and $\mathbf{g}^v$ should be reshaped so that it is $n \times n$, as a post-processing step.

For corner vertices, there is usually insufficient data in the patch of immediate neighbours to solve (5.25). Therefore, it is necessary to extend the patch. Doing so for general boundary vertices also improves the approximation accuracy. Subfigures 5.2b and 5.2c illustrate examples of how patches can be extended when their defining vertices lie on the domain boundary.

Both L^2 and Z^2 recovery strategies are demonstrated in Subsection 5.8.1. Whilst the presentation above concerns the recovery of a $\mathbb{P}1$ field from a $\mathbb{P}0$ one, the method can also be extended to higher orders. See [**Zienkiewicz and Zhu, 1987**] and [**Zienkiewicz and Zhu, 1992a**] for further details, as well as [**Zienkiewicz and Zhu, 1992b**] for its use for error estimation and mesh adaptation.

In the element-based framework described in Subsection 5.1.5, it is sufficient to recover a $\mathbb{P}0$ field, meaning that it is possible to explicitly write down the recovery operator. For an approximation to the gradient, this takes the form [**Micheletti et al., 2010**]

$$R_{\Delta_K}[\nabla u_h](\mathbf{x}) := \frac{1}{|\Delta_K|} \sum_{K' \in \Delta_K} |K'| \ \nabla u_h|_{K'}, \qquad \mathbf{x} \in \Delta_K, \tag{5.26}$$

where Δ_K is a patch around *element* K, rather than around a vertex. Here $|\Delta_K|$ is the patch volume, which is simply the sum over the volumes of its constituent elements. That is, we have a volume-average of the discrete gradient over neighbouring elements.

5.3.2. Hessian Recovery on the Boundary. The double L^2 Hessian recovery technique breaks down on the domain boundary. For many applications, this is not of concern. However, in CFD and coastal engineering, it is often desirable to also have a representation for the Hessian on (i.e. tangential to) $\partial\Omega$.

For a 2D mesh with linear elements, the tangential component of the Hessian is given by

$$\underline{\boldsymbol{H}}^s := \widehat{\mathbf{s}}_1^T \underline{\boldsymbol{H}}\, \widehat{\mathbf{s}}_1, \tag{5.27}$$

where $\widehat{\mathbf{s}}_1 = \widehat{\mathbf{n}}^\perp$ is one choice of tangent vector. In this case, the choice of sign does not actually matter. Despite the fact $\underline{\boldsymbol{H}}^s$ is 1×1 dimensional, we retain the tensor notation, for consistency.

In the linear 3D case, boundary segments are 2D, meaning we need two orthonormal tangent vectors, $\{\widehat{\mathbf{s}}_1, \widehat{\mathbf{s}}_2\}$. These may be constructed using Gram-Schmidt orthogonalisation, for example. With these, the boundary Hessian takes the form [**Loseille et al., 2010**]

$$\underline{\boldsymbol{H}}^s := (\widehat{\mathbf{s}}_1, \widehat{\mathbf{s}}_2)^T \underline{\boldsymbol{H}}\, (\widehat{\mathbf{s}}_1, \widehat{\mathbf{s}}_2) = \begin{bmatrix} \widehat{\mathbf{s}}_1^T \underline{\boldsymbol{H}} \widehat{\mathbf{s}}_1 & \widehat{\mathbf{s}}_1^T \underline{\boldsymbol{H}} \widehat{\mathbf{s}}_2 \\ \widehat{\mathbf{s}}_2^T \underline{\boldsymbol{H}} \widehat{\mathbf{s}}_1 & \widehat{\mathbf{s}}_2^T \underline{\boldsymbol{H}} \widehat{\mathbf{s}}_2. \end{bmatrix} \tag{5.28}$$

Here, the choice of tangent vectors has an effect on the resulting Hessian.

Approximating $\underline{\boldsymbol{H}}^s$ amounts to solving an auxiliary Poisson equation on the domain boundary only. Denote the finite element space on the boundary by $\overline{\Sigma_h}$ and the Galerkin approximation of $\underline{\boldsymbol{H}}^s$ by $\underline{\boldsymbol{H}}^s_h \in \overline{\Sigma_h}$. Since the boundary is closed, a number of terms drop

out of the integration by parts, leaving

$$\oint_{\partial\Omega} \underline{\boldsymbol{\Sigma}} : \underline{\boldsymbol{H}}_h^s \, \mathrm{d}s = -\sum_{i=1}^{n}\sum_{j=1}^{n} \oint_{\partial\Omega} \frac{\partial \underline{\boldsymbol{\Sigma}}_{ij}}{\partial \widehat{\mathbf{s}}_j} \frac{\partial u_h}{\partial \widehat{\mathbf{s}}_i} \, \mathrm{d}s, \quad \forall \underline{\boldsymbol{\Sigma}} \in \overline{\Sigma_h}. \tag{5.29}$$

As has been noted, a recovered Hessian – or any other metric – should be normalised before it is applied to a particular problem, as well as before its combination with other metrics. This is especially true when combining interior and boundary terms, which exist in different dimensions. The following section presents normalisation strategies for interior metrics, which extend to the boundary because the dimension is not fixed. However, the combination of interior and boundary is more involved, involving the solution of an algebraic problem which determines the target complexity for each component; for this, we refer to [**Loseille et al., 2010**].

5.4. Metric Normalisation

Metric normalisation has the purpose of accounting for the domain geometry, solution fields and timescales. There are a number of different approaches. One simple approach (used in [**Frey and Alauzet, 2005**]) is to formulate (5.17) as a *relative* error and thereby divide by the modulus of the (scalar) solution field. Given a metric $\mathcal{M} = \{\underline{\boldsymbol{M}}(\mathbf{x})\}_{\mathbf{x}\in\Omega}$ to be normalised, it reads

$$\mathcal{M}_\epsilon := \frac{1}{|u|_\epsilon}\mathcal{M}, \qquad \text{where} \qquad |u|_\epsilon := \max\{|u|, \epsilon\|u\|_{L^\infty}\}, \tag{5.30}$$

for a desired interpolation error level $\epsilon > 0$. This ensures that the metric is not biased towards the most significant flow features [**Pain et al., 2001**] and goes some way to establishing multi-scale metrics.

The following subsection describes a more general normalisation approach based on bounds of the form (5.19). The technique enables the generation of metrics which minimise interpolation error in the L^p norm. It is fully capable of accounting for multi-scale problems whose solution fields vary on several orders of magnitude, given an appropriate choice of p.

5.4.1. L^p Normalisation for Time-Independent Problems. L^p normalisation is framed around the continuous interpolation error (5.20). The idea is to obtain a metric which satisfies the optimisation problem

$$\min_{\mathcal{M}} \|u - \pi_{\mathcal{M}} u\|_{L^p(\Omega)} \quad \text{subject to} \quad \mathcal{C}(\mathcal{M}) = \mathcal{C}_T, \tag{5.31}$$

for some fixed $p \in [1, \infty)$ and desired complexity $\mathcal{C}_T > 0$. The constraint acts to close the otherwise under-determined problem. It is useful for controlling the number of mesh vertices so that resulting simulations are achievable using limited computational resources.

Since the continuous interpolation error bounds the discrete version from above, minimising (5.20) is approximately equivalent to solving

$$\min_{\mathcal{H}_h} \|u - \Pi_h u\|_{L^p(\Omega)} \quad \text{such that } \mathcal{H}_h \text{ has } N \text{ vertices.} \tag{5.32}$$

Given a unit mesh, metric complexity and vertex count are directly proportional (see p.57 of [**Loseille and Alauzet, 2011a**]). The constant of proportionality depends upon domain geometry and the mesh adaptation algorithm used; it is difficult to determine in practice. As remarked above, the constraint can only be satisfied approximately in practice. The extent to which this is achieved is determined by the adaptation algorithm used.

It is proved in [**Loseille and Alauzet, 2011b**] that the unique solution of the continuous optimisation problem (5.31) is given by the metric

$$\mathcal{M}_{L^p} := \mathcal{C}_T^{\frac{2}{n}} \left(\int_\Omega \det\left(|\underline{\boldsymbol{H}}|\right)^{\frac{p}{2p+n}} \,\mathrm{d}x \right)^{-\frac{2}{n}} \det\left(|\underline{\boldsymbol{H}}|\right)^{-\frac{1}{2p+n}} |\underline{\boldsymbol{H}}| \,. \tag{5.33}$$

In some cases, it is more appropriate to limit the interpolation error, rather than the metric complexity, as considered in (5.30). An equivalent optimisation problem to (5.31) may be formulated, with its unique solution presented in [**Alauzet and Olivier, 2010**]. In order to reduce interpolation error, smaller values for the desired error ϵ are requested. This, in turn, means allowing heightened overall DoF count. For the remainder of this thesis, however, we choose to enforce metric complexity. The main reason for this is that it is hard to predict the number of DoFs in the resulting mesh if error level is specified. On the contrary, normalisation under complexity provides a way to strongly influence this. This can be useful in mesh convergence studies, where some error quantity is to be assessed as the mesh resolution is increased.

L^∞ normalisation is obtained by passing to the limit $p \to \infty$ in (5.33):

$$\mathcal{M}_{L^\infty} := \mathcal{C}_T^{\frac{2}{n}} \, \mathcal{C}(|\underline{\boldsymbol{H}}|)^{-\frac{2}{n}} \, |\underline{\boldsymbol{H}}| \,. \tag{5.34}$$

Whilst commonly used, a major disadvantage of the L^∞ normalisation strategy is that it is not able to fully capture discontinuities. In many cases, L^∞ normalisation will mean very high levels of mesh refinement are used surrounding discontinuous features and sharp gradients. Instead, we advocate setting small values such as $p \in [1, 10]$. Doing so circumvents issues stated for L^∞ normalisation and also permits multi-scale mesh adaptation, as demonstrated in Subsection 5.8.2.

5.4.2. L^p Normalisation for Time-Dependent Problems. The optimum L^p metric (5.33) is effectively a normalisation over the spatial domain. For time-dependent problems, the temporal domain should also be accounted for. This enables the DoF count to vary in time, as well as space. That is, it enables the discretisation to have more DoFs *when* they are needed as well as *where*. Similarly, the mesh can have fewer overall DoFs at time levels where they would be unnecessary. Note that the *overall* DoFs count at any given timestep cannot be specified by a purely r-adaptive mesh adaptation routine.

For a time period $(0, T]$ and timestep $\Delta t > 0$, the extension of (5.33) to the time-dependent case is given by [**Alauzet and Olivier, 2010**]

$$\mathcal{M}_{L^p} := \mathcal{C}_T^{\frac{2}{n}} \left(\int_0^T \frac{1}{\Delta t} \int_\Omega \det\left(|\underline{\boldsymbol{H}}|\right)^{\frac{p}{2p+n}} \,\mathrm{d}x \,\mathrm{d}t \right)^{-\frac{2}{n}} \det\left(|\underline{\boldsymbol{H}}|\right)^{-\frac{1}{2p+n}} |\underline{\boldsymbol{H}}| \,, \tag{5.35}$$

where $\mathcal{C}_T > 0$ is now the target *space-time* complexity, obtained by integrating (5.8) in time. Space-time complexity is analogous to the total number of mesh vertices, summed

over all timesteps. Whilst the timestep Δt is not necessarily of fixed length, we assume it to be in this thesis, for simplicity.

5.5. Combining Metrics

In many cases, there may be more than one metric containing information which we regard as important. For instance, if the PDE comprises a mixed system of equations then we may wish to consider Hessians for each component of its solution tuple. This section outlines two common approaches for combining metric information and provides geometrical interpretations thereof.

5.5.1. Metric Intersection. A common means of combining two Riemannian metric fields, $\mathcal{M}_1 = \{\underline{\boldsymbol{M_1}}(\mathbf{x})\}_{\mathbf{x}\in\Omega}$ and $\mathcal{M}_2 = \{\underline{\boldsymbol{M_2}}(\mathbf{x})\}_{\mathbf{x}\in\Omega}$, is using *metric intersection* (a.k.a. *metric superposition*). The idea is to generate the metric with the maximal ellipsoid which fits within the ellipsoids associated with the two component metrics, as illustrated in Figure 5.3. Since the ellipsoid of intersection is smaller than both component ellipsoids, a mesh adaptation algorithm driven by $\mathcal{M}_1 \cap \mathcal{M}_2$ will always yield meshes with more vertices than the number of vertices in meshes generated from either $\mathcal{M}_1$ or $\mathcal{M}_2$.

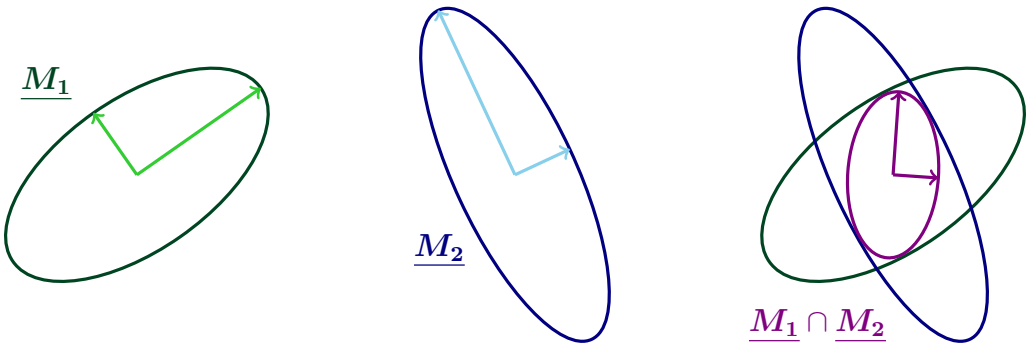


FIGURE 5.3. Geometric interpretation of metric intersection in terms of ellipses.

Computing the intersection of *two* metrics amounts to simultaneously reducing both quadratic forms $\langle\cdot,\cdot\rangle_{\underline{\boldsymbol{M_1}}}$ and $\langle\cdot,\cdot\rangle_{\underline{\boldsymbol{M_2}}}$ in order to find a common basis [**Barral, 2015**]. To achieve this, begin by mapping the two metrics into the space spanned by the eigenvectors of $\underline{\boldsymbol{M_1}}$:

$$\widehat{\underline{\boldsymbol{M_1}}} := \left(\underline{\boldsymbol{M_1}}^{-\frac{1}{2}}\right)^T \underline{\boldsymbol{M_1}}\,\underline{\boldsymbol{M_1}}^{-\frac{1}{2}} = \underline{\boldsymbol{I_n}}, \qquad \widehat{\underline{\boldsymbol{M_2}}} := \left(\underline{\boldsymbol{M_1}}^{-\frac{1}{2}}\right)^T \underline{\boldsymbol{M_2}}\,\underline{\boldsymbol{M_1}}^{-\frac{1}{2}}. \tag{5.36}$$

Under this mapping, the ellipsoid representing $\underline{\boldsymbol{M_1}}$ is the unit ball. Since the fields in (5.36) are again metrics, they have orthogonal eigendecompositions, the former of which is trivial. The two may be expressed as

$$\widehat{\underline{\boldsymbol{M_1}}} = \underline{\boldsymbol{Q}}\,\underline{\boldsymbol{I_n}}\,\underline{\boldsymbol{Q}}^T, \qquad \widehat{\underline{\boldsymbol{M_2}}} = \underline{\boldsymbol{Q}}\,\underline{\boldsymbol{\Lambda}}\,\underline{\boldsymbol{Q}}^T, \qquad \text{where} \qquad \underline{\boldsymbol{\Lambda}} = \operatorname{diag}(\lambda_1,\dots,\lambda_n). \tag{5.37}$$

We construct a new metric by taking the maximal eigenvalue for each component:

$$\widehat{\underline{\boldsymbol{M}}} := \underline{\boldsymbol{Q}}\,\widehat{\underline{\boldsymbol{\Lambda}}}\,\underline{\boldsymbol{Q}}^T, \qquad \text{where} \qquad \widehat{\underline{\boldsymbol{\Lambda}}} := \operatorname{diag}(\max(1,\lambda_1),\dots,\max(1,\lambda_n)). \tag{5.38}$$

The intersected metric is formed by mapping back into Euclidean space

$$\underline{\boldsymbol{M_1}} \cap \underline{\boldsymbol{M_2}} := \left(\underline{\boldsymbol{M_1}}^{\frac{1}{2}}\right)^T \widehat{\underline{\boldsymbol{M}}}\, \underline{\boldsymbol{M_1}}^{\frac{1}{2}}. \tag{5.39}$$

As may be expected from the geometrical interpretation, the intersection order is commutative applied to two metrics, i.e. $\mathcal{M}_1 \cap \mathcal{M}_2 = \mathcal{M}_2 \cap \mathcal{M}_1$ for any two metrics $\mathcal{M}_1$ and $\mathcal{M}_2$. However, the order matters when more than two metrics are intersected, as demonstrated on p.28 of [**Loseille, 2008**]. This is because, for more than two metrics, formula (5.39) no longer coincides with the geometrical interpretation in general and only provides an approximation.

Consider the case of combining two metrics which happen to have the same eigenvector matrix, $\underline{\boldsymbol{V}}$. This implies that the directions of greatest anisotropy of the two metrics are either fully aligned or orthogonal. Following the definition above, we find that

$$\underline{\boldsymbol{M_1}} \cap \underline{\boldsymbol{M_2}} = \underline{\boldsymbol{V}}\, \widetilde{\underline{\boldsymbol{\Lambda}}}\, \underline{\boldsymbol{V}}^T, \quad \text{where} \quad \widetilde{\underline{\boldsymbol{\Lambda}}} = \operatorname{diag}(\max(\lambda_1, \mu_1), \dots, \max(\lambda_n, \mu_n)), \tag{5.40}$$

with $\{\lambda_i\}_{i=1}^n$ and $\{\mu_i\}_{i=1}^n$ being the eigenvectors of $\underline{\boldsymbol{M_1}}$ and $\underline{\boldsymbol{M_2}}$. That is, for metrics with either aligned or orthogonal anisotropies, intersection acts to maximise each eigenvalue, thereby minimising the associated magnitude.

5.5.2. Metric Average. Another means of combining metric information is using a convex combination

$$\mathcal{M} := \alpha\mathcal{M}_1 + (1-\alpha)\mathcal{M}_2, \qquad \alpha \in (0,1). \tag{5.41}$$

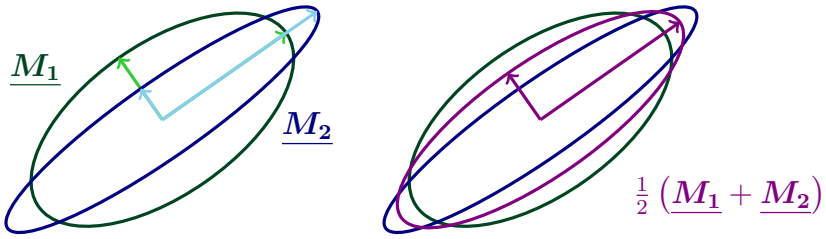
That (5.41) is indeed a metric follows immediately from the definition of positive-definiteness. The extension of (5.41) to $m \in \mathbb{N}$ metrics is trivial, involving $m-1$ independent parameters. These parameters can be chosen to weight the resulting metric towards one particular error estimate or field of interest, or to weight all contributions equally, yielding the *metric average*.

Whilst the metric average is simple to implement, its geometric interpretation is less obvious than that of metric intersection. Consider again the case of combining two metrics which happen to have the same eigenvectors. Then

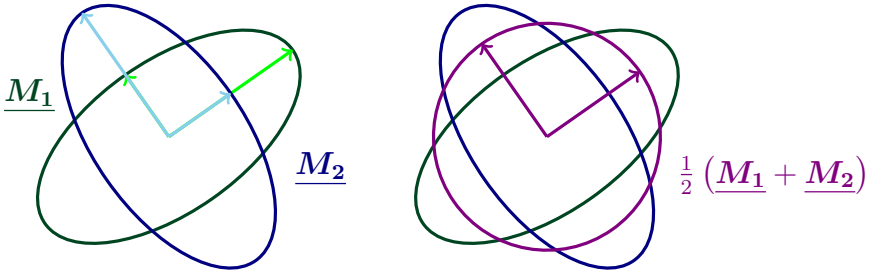
$$\frac{1}{2}\left(\underline{\boldsymbol{M_1}} + \underline{\boldsymbol{M_2}}\right) = \underline{\boldsymbol{V}}\left(\frac{1}{2}\left(\underline{\boldsymbol{\Lambda_1}} + \underline{\boldsymbol{\Lambda_2}}\right)\right)\underline{\boldsymbol{V}}^T. \tag{5.42}$$

That is, averaging the metrics corresponds to averaging the eigenvalues. As shown in Figure 5.4, if the two metrics have aligned anisotropy then their average also shares that anisotropy, as we would hope. If the anisotropies are orthogonal, then they cancel out, to some extent. The anisotropy of the average metric is dictated for the most part by whichever of the component metrics has the strongest anisotropy. The example shown in Subfigure 5.4b was chosen to illustrate the case where the anisotropies cancel one another out entirely, leaving an isotropic metric.

Whilst the resulting anisotropy agrees with that due to metric intersection in the cases described above, the magnitudes do not agree, since the ellipses are not contained within the intersections of the component ellipses. Therefore, the number of vertices in a mesh generated using averaged metrics may in some cases be smaller than in one generated using one of the component metrics. For a counter-example, consider metrics $\underline{\boldsymbol{M_1}} = \underline{\boldsymbol{I_2}}$ and $\underline{\boldsymbol{M_2}} = 3\underline{\boldsymbol{I_2}}$, which have density 1 and 9, respectively. Their average $\frac{1}{2}(\underline{\boldsymbol{M_1}} + \underline{\boldsymbol{M_2}}) = 2\underline{\boldsymbol{I_2}}$



(A) Aligned anisotropy



(B) Orthogonal anisotropy

FIGURE 5.4. Geometric interpretation of metric averaging in the case of aligned eigenvectors in terms of ellipses.

has density 4. Further, the number of vertices in a mesh generated using averaged metrics may not be as high as for one generated using intersected metrics.

Whilst the geometric interpretation for general eigendecompositions is not obvious, experimental results in Subsection 5.8.3 show that it can be used to combine metrics in a meaningful and effective way.

5.6. Metric Gradation

Sharp changes in mesh element size can act as artificial internal boundaries in the domain, which could potentially either reflect or absorb incoming waves in a way which is not present in the physics. (See Figure 12 in [**Alauzet, 2010**] for an example of such an internal boundary.) One way to avoid this numerical phenomenon is to modify the metric using a *metric gradation* routine, which ensures that the sizes it prescribes at neighbouring vertices do not vary by more than a specified threshold. In this thesis, we use the value $\beta = 1.4$ for this threshold. That is, β bounds the ratio of the prescribed sizes from above, meaning they can change by at most 40%.

The metric gradation routine used in this thesis is the one given in [**Alauzet, 2010**]. We refer to that work for details on the implementation.

5.7. Mesh Adaptation Strategy

5.7.1. Time-Independent Metric-Based Mesh Adaptation. The sections covered thus far in this chapter are sufficient to consider mesh adaptation applied to time-independent PDEs. Such problems are usually defined on a single mesh. The aim is to modify this mesh so that it is quasi-unit when viewed in some specified metric space. The metric space usually (but not always) depends on solutions of the PDE, which change

when the mesh is changed. As such, a fixed point iteration approach can be applied and run to convergence.

Algorithm 1 presents such a fixed point iteration loop. It is assumed that $m \in \mathbb{N}$ metrics are to be constructed from error indicators and/or heuristics; these metrics are to be combined using metric intersection or averaging. Convergence of the loop may be determined (for example) by a relative tolerance on an error estimate of interest or by a relative tolerance on the change in mesh element count between iterations. A maximum iteration count is also usually imposed.

Given target metric complexity $\mathcal{C}_T > 0$;
Given initial mesh $\mathcal{H}_0$;
Set $i := 0$;
while *not converged* **do**
  Solve PDE on $\mathcal{H}_i$;
  for *metric component* j **do**
    Extract metric $\mathcal{M}_i^j$ from solution fields or error indicators;
    Normalise $\mathcal{M}_i^j$ based on complexity $\mathcal{C}_T$;
  end
  Combine $\{\mathcal{M}_i^j\}_{j=1}^m$ to yield a single metric, $\mathcal{M}_i$;
  Apply gradation to $\mathcal{M}_i$;
  Adapt mesh $\mathcal{H}_i$ using $\mathcal{M}_i$ to obtain mesh $\mathcal{H}_{i+1}$;
  Increment i;
end

Algorithm 1: Metric-based mesh adaptation routine for time-independent problems.

As motivated at the start of this chapter, taking more iterations means reducing dependence of the adapted mesh on the initial mesh. In practice, it is often the case that three or more fixed point iterations are required in order for an anisotropic metric to properly introduce anisotropy into an initially uniform or isotropic mesh. As such, at least three iterations are performed before the convergence criteria are checked.

5.7.2. Time-Dependent Metric-Based Mesh Adaptation. When applying mesh adaptation to method of lines type discretisation approaches, it is often necessary to adapt the spatial mesh during the time interval. This is true when, for example, tracking shockwaves propagating across the domain. In this subsection, we examine two different approaches.

'On-The-Fly' Approach. The naïve approach is to apply steady-state mesh adaptation during the time-integration process. That is, at each timestep when the mesh is to be adapted, Algorithm 1 is applied, with 'solve PDE on $\mathcal{H}_i$' replaced by 'interpolate all relevant fields onto $\mathcal{H}_i$'. Updated metric information on adapted meshes may be obtained either by interpolating the metric or by construction of a new metric from interpolated data. In either case, interpolation error is introduced.[1] Such an approach is a simple extension of the steady-state implementation and has low memory requirements, because the only information from the previous mesh which needs to be stored is that which is to be outputted.

[1]Note that here interpolation error relates to the mesh-to-mesh data transfer, not an error estimate.

A disadvantage of this approach is that there is no concept of space-time complexity built into the normalisation strategies. In order to obtain meshes with (significantly) different DoF counts, the user must deploy a manual approach, whereby the target complexity is modified at each adaptation step. It is difficult to see how this might be achieved; this is one case where normalisation by target interpolation error might be more useful than normalisation by target complexity.

Another disadvantage is that the metric computed at one timestep only contains information from that timestep. In particular, it does not contain information on solution fields between that timestep and the next one where adaptation is to be applied. As such, the metric quickly ceases to be relevant. If mesh resolution has been concentrated around a flow feature, such as a tsunami wave, front or eddy-like structure, this feature could traverse into a coarsely meshed reason before the next mesh adaptation, thereby suffering numerical diffusion. Consequently, the approximation accuracy is diminished.

One way to avoid this is to adapt the mesh very frequently, ensuring that any important solution features do not have sufficient opportunity to pass into coarse regions. However, doing so comes with a significant computational cost and means that interpolation error can become significant. Alternatively, there are means of making the metric relevant until the next adaptation step. One approach to achieving this in the context of advection-dominated problems is the notion of 'metric advection'.

Metric Advection. Suppose that a 'wind field' $\mathbf{w} : \Omega \to \mathbb{R}^n$ may be determined for the problem at hand. Using $\mathbf{w}$, the regions where higher mesh resolution will be required in future timesteps can be preempted, to some extent. Whilst the choice of wind field is problem-dependent, an effective choice is often obvious for advection-dominated problems. For tracer transport, the obvious choice is the velocity field in which the tracer is advected.

The metric constructed at a particular timestep may be advected in $\mathbf{w}$, allowing a coarse-grained forecast of where mesh resolution will be required in near-future timesteps. In the case of Crank-Nicolson timestepping with implicitness θ, for example, the metric $\mathcal{M}^{(k+1)}$ at timestep $k+1$ may be approximated using the metric $\mathcal{M}^{(k)}$ computed at timestep k using

$$\frac{\mathcal{M}^{(k+1)} - \mathcal{M}^{(k)}}{\Delta t} + \theta \mathbf{w}^{(k+1)} \cdot \nabla \mathcal{M}^{(k+1)} + (1-\theta)\mathbf{w}^{(k)} \cdot \nabla \mathcal{M}^{(k)} = 0. \tag{5.43}$$

This relation may be applied multiple times, whereby $\mathcal{M}^{(k)}$ is replaced by the forecasted metric at that time level. Each of the forecasted metrics may then be accumulated using L^1 normalisation (averaging), intersection or some weighted combination approach, as appropriate. In principle, advecting a metric seeks to reduce the remeshing frequency and yet maintain a sufficiently accurate representation of the fluid dynamics being studied [**Smith, 2016**]. Equation (5.43) may be interpreted as acting on the whole metric tensor or its individual components.

Metric advection has been successfully applied to a number of computational fluids problems, including tsunami propagation [**Smith, 2016**, **Smith et al., 2016**], reservoir modelling [**Mostaghimi et al., 2016**], tectonic subduction [**Garel et al., 2014**] and gravity current resolution in lock-exchange problems [**Hiester et al., 2011**, **Rossa and Coutinho, 2013**, **Parkinson et al., 2014**]. However, there are a number of issues with this approach. Firstly, whilst advecting the metric over more timesteps means fewer mesh adaptation steps, it also means that the metric complexity – and hence DoF count – grows if the metrics are combined using intersection, as is common. As such, it becomes

difficult to control the overall metric complexity. If metric averaging is repeatedly applied instead, the result is a smoothing which means that the resulting meshes do not resolve flow features as strongly, as demonstrated in Subsection 5.8.3. Further, because metric advection is approximate for anything other than certain pure advection problems with prescribed velocity fields, it is likely that it will introduce mesh resolution in regions where it is unnecessary.

Despite the above criticisms, on-the-fly adaptation strategies which use metric advection typically have relatively cheap costs and low memory requirements, especially if an explicit time integration scheme is used. Nonetheless, this advantage is less useful in cases where outer loop processes such as adjoint methods are applied. If the adjoint equation is to be solved using the same discretisation as an adaptive forward model then the meshes must be stored, at the very least. For nonlinear problems or QoIs, forward solution fields must also be stored, meaning that the memory requirements are already significant. Assuming that the adjoint problem is to be solved, the application of a mesh adaptation algorithm which uses *solution data* from future timesteps of previous runs becomes justifiable, as detailed in the following.

Fixed-Point Iteration Approach. In the remainder of this subsection, we consider a 'fixed-point iteration' approach to time-dependent metric based mesh adaptation, whereby the prognostic equation is solved over the whole time period during each iteration. Such an approach for metric-based mesh adaptation in time-dependent problems was first considered in [**Alauzet, 2003**] and further developed in [**Alauzet et al., 2007**]. The extension to the goal-oriented case in Chapter 7 allows not only to use solution fields from future timesteps, but also information on sensitivities, via the adjoint equation. This means that the mesh is adapted not only according to contributions to some error estimator, but also based on relevance to a diagnostic quantity of interest.

First, we introduce some notation. Recall decomposition (2.42) of the time interval $(0,T]$ into a finite number of timesteps. For the purposes of time-dependent mesh adaptation, we make further partitions of the time interval, each of which is a collection of consecutive timesteps. These subintervals – or 'windows' – are denoted

$$[0,T] = \cup_{k=1}^{N} \mathcal{W}^k, \quad \text{where} \quad \mathcal{W}^k := [t^{(\ell(k-1))}, t^{(\ell k)}]. \tag{5.44}$$

The intersection of two distinct subintervals is either the empty set or a singleton set (if the subintervals are adjacent). In addition, we introduce a mesh associated with each subinterval, giving rise to a set, $\{\mathcal{H}^k\}_{k=1}^N$. That is, a fixed mesh is assumed on each subinterval and mesh-to-mesh interpolation is required for the transition. In this framework, $\ell \in \mathbb{N}$ is the number of timesteps per mesh iteration. That is, the number of timesteps spent on each mesh in (5.44). The number of timesteps per mesh iteration is assumed fixed, although this need not be so. With the fixed timestep assumption, (5.44) can be rewritten as

$$\mathcal{W}^k = [\ell(k-1)\Delta t, \ell k \Delta t], \quad k = 1:N. \tag{5.45}$$

The idea of the fixed point iteration is to repeatedly solve the PDE over all subintervals, updating each mesh $\mathcal{H}^k$ using mesh adaptation. As such, a subscript is used to denote the iteration count. A metric is constructed for each $\mathcal{W}^k$, using solution data from the entire subinterval. Thus, information on the future solution trajectory (as approximated on the previous mesh) is used to generate the mesh used over all of $\mathcal{W}^k$. In this way, the fixed-point iteration method incorporates predictions of the future solution trajectory

within each subinterval. It also offers flexibility to choose a single subinterval (whereby predictions of the entire solution trajectory are used) or one subinterval for each timestep (whereby frequent mesh adaptation is deployed). Note that, whilst the PDE is solved on half-open intervals, mesh adaptation is to be performed on closed intervals. This is because, whilst not solved for, the initial value on $\mathcal{W}^k$ influences the construction of $\mathcal{H}^k$.

Riemannian metrics are defined point-wise in space. They are also defined at instances in time. As such, they can be understood at individual time levels, but not inbetween. This is accounted for by *accumulating* metrics in time. There are two standard methods, which make use of the machinery from Section 5.5.

The first is L^1 normalisation in time, which is effectively time integration. Given metrics at consecutive timesteps, this amounts to applying an appropriate quadrature rule. For Crank-Nicolson with the default implicitness $\theta = \frac{1}{2}$, this is the trapezium rule. Applied on a single timestep, we simply have the metric average, scaled by Δt. The other standard approach is to apply metric intersection. Sequences of meshes due to these approaches are compared in Subsection 5.8.4.

Unlike in the on-the-fly method, the spatio-temporal discretisation is known at the start of every fixed point iteration. As such, error analysis may be performed across space and time, instead of at an instant. This is exactly what Algorithm 2 builds upon, so that a global error estimate may be reduced. Again, assume that $m \in \mathbb{N}$ metric components are to be combined. Convergence may be determined (for example) by a relative tolerance on an error estimate of interest or by a relative tolerance on changes to the number of elements in *each mesh*. Again, a minimum iteration count allows for more anisotropic meshes and to avoid dependence on the initial mesh.

Note that Algorithm 2 assumes that metrics are computed and accumulated at every timestep, for simplicity of presentation. If multiple metrics are to be constructed via recovery of derivative fields, it can easily be the case that more CPU time is spent doing this than solving the PDE, for a given timestep. In practice it is usually sufficient to only construct metrics every few timesteps and accumulate as appropriate.

It should be noted that metric advection is not required by the fixed-point iteration method, since it already incorporates predictions on the future solution trajectory within each subinterval.

5.7.3. Software. In Firedrake, meshes are represented using the PETSc DMPlex topology abstraction. This unstructured mesh format is designed to decouple application programming concerns from lower level concerns, such as domain decomposition and I/O, and allows problems to be written independently of spatial dimension [**Lange et al., 2016**].

In this thesis, metric-based mesh adaptation is applied using the *hr*-adaptive toolkit *Pragmatic* [**Barral et al., 2016**]. A Riemannian metric field is defined in high-level Firedrake code as a `Function` in a tensor $\mathbb{P}1$ space. The metric is represented in the underlying PETSc code as a `Vec` (vector data structure), which is interpreted as a simple `double` precision array, for the purposes of Pragmatic. Mesh adaptation is performed and the resulting mesh is propagated back to the Firedrake level via the Pragmatic `Mesh` class and PETSc `DMPlex` struct. Figure 5.5 presents a workflow representation for the traversal of data between abstraction levels.

Given target space-time metric complexity $\mathcal{C}_T > 0$;
Given initial meshes $\{\mathcal{H}_0^k\}_{k=0}^{N-1}$;
Set $i := 0$;
while *not converged* **do**
  for *each subinterval* $\mathcal{W}^k$ **do**
    Interpolate fields onto $\mathcal{H}_i^k$ from formulae/data/previous mesh;
    for *each time level* $t \in \mathcal{W}^k$ **do**
      Integrate PDE forward by Δt on mesh $\mathcal{H}_i^k$;
      for *metric component* j **do**
        Extract metric and accumulate into $\mathcal{M}_i^{k,j}$;
      end
    end
  end
  for *metric component* j **do**
    Normalise $\{\mathcal{M}_i^{k,j}\}_{k=1}^{N}$ in space and time according to $\mathcal{C}_T$;
  end
  for *each mesh* $\mathcal{H}_i^k$ **do**
    Combine $\{\mathcal{M}_i^j\}_{j=1}^{m}$ to yield a single metric, $\mathcal{M}_i^k$;
    Apply gradation to $\mathcal{M}_i^k$;
    Adapt mesh $\mathcal{H}_i^k$ using $\mathcal{M}_i^k$ to obtain mesh $\mathcal{H}_{i+1}^k$;
  end
  Increment i;
end

Algorithm 2: Metric-based mesh adaptation routine for time-dependent problems

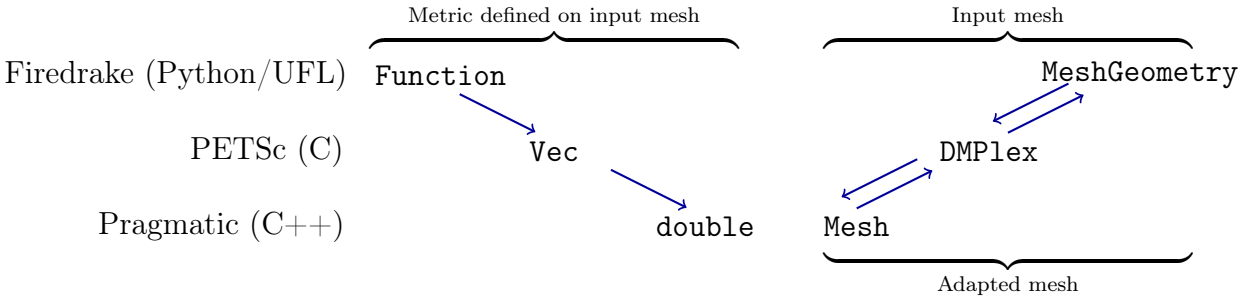


FIGURE 5.5. Workflow for mesh adaptation in Firedrake using PETSc and Pragmatic.

The mesh adaptation step is driven by both h-adaptive and r-adaptive operations. In the 2D case, node insertion, node removal, edge swapping and node movement are used. Modifications are made with the aim of achieving a quasi-unit mesh, as defined in (5.9). Laplacian smoothing is applied in a local sense to further improve the quality function (5.10), if such an improvement is achievable. For further details on the mesh operations in such a hr-adaptation strategy, see [**George et al., 2002**]. For further details on Pragmatic, see [**Barral et al., 2016**].

Note that the metric-based framework is not tied to the adaptation approach described above. For example, a notable r-adaptation strategy which has the capability to use Riemannian metrics is the MMPDE framework [**Huang and Kamenski, 2015**]. However,

since it is not possible to have different numbers of DoFs across different timesteps in such an approach, space-time normalisation may not be as effective as expected.

5.8. Numerical Experiments

In this section we test the metric construction strategies and operations introduced in this chapter. Comparisons are made between recovery techniques, normalisation orders and combination approaches. Application of metric-based mesh adaptation to a time-dependent tracer transport problem is also demonstrated.

5.8.1. Recovery. Two recovery techniques were introduced in Subsection 5.3.1: L^2 projection and the approach of Zienkiewicz and Zhu (Z^2). We perform two experiments to validate these techniques.

Hessian of a Quadratic Function. First, we check that the strategies are able to recover the constant Hessian of a quadratic function. Consider $u : [-1,1]^2 \to \mathbb{R}$ given by $u(x,y) := \frac{1}{2}(x^2+y^2)$, whose Hessian is the identity matrix. On a uniform mesh of $[-1,1]^2$ comprised of 20,000 isosceles right-angled triangles, double L^2 projection is applied to recover the Hessian in $\mathbb{P}1$ space.

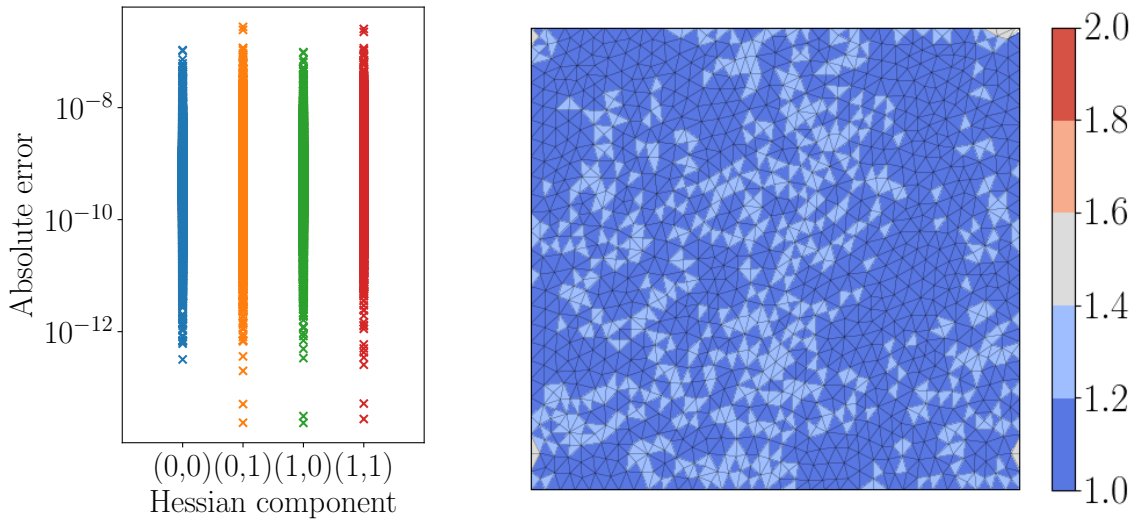
First consider the application of recovery techniques to u as an expression (as opposed to an interpolant thereof). Subfigure 5.6a shows that the vertex-wise errors in each component vary on the order of at most $\mathcal{O}(10^{-7})$, indicating an excellent approximation. Note that an excellent approximation has been achieved on the boundary, even though the boundary Hessian was not recovered. The reason for the high quality approximation is that the Hessian was recovered directly from the analytical expression, which is quadratic and is therefore twice differentiable. Note that off-diagonal entries are not equal, because of numerical errors incurred during the projection. In practice, this is corrected by averaging off-diagonal components when forming a metric from the Hessian.

The mesh resulting from an L^1 normalised metric with target complexity 1,000 is shown in Subfigure 5.6b. Four mesh adaptation steps have been applied. As may be expected, it is an isotropic mesh (with aspect ratio everywhere less than 2).

Now consider the same double L^2 projection experiment, but where u is first interpolated into $\mathbb{P}1$ space, meaning its second finite element derivatives are zero. The resulting vertex-wise errors presented in Subfigure 5.7a are much more significant. The resulting mesh in Subfigure 5.7b is almost isotropic, although it does contain more anisotropy than in Subfigure 5.6b.

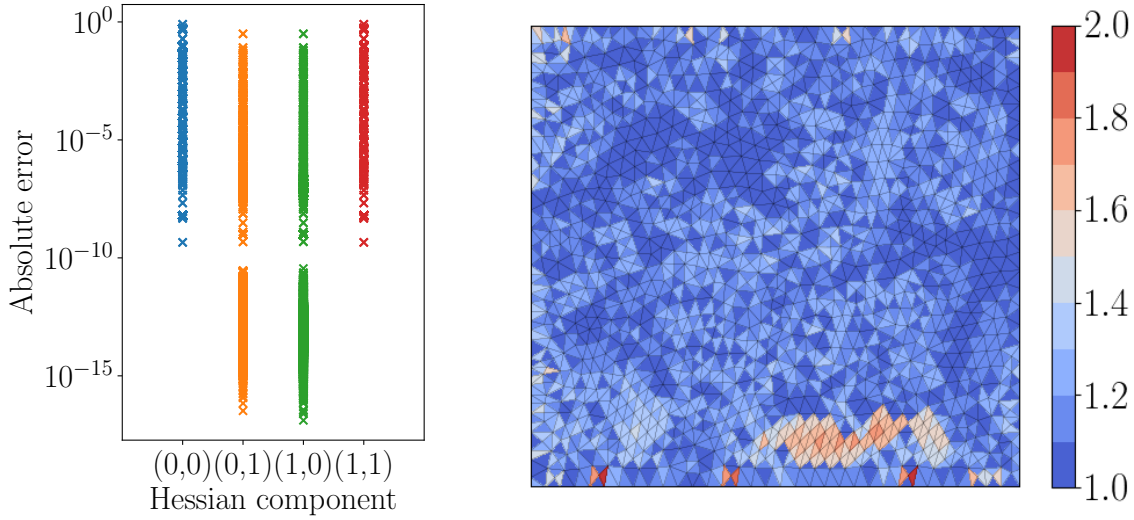
To summarise, the double L^2 projection recovery technique has been shown to be able to successfully recover the Hessian of a quadratic function. In addition, it has shown to be capable of recovering the Hessian from a piecewise linear interpolant, albeit with lower accuracy. Whilst the point-wise accuracy is not always high, meshes resulting from adapting w.r.t. the Hessian are satisfactory for this example.

In Figure 5.8 the Hessian is recovered from the same $\mathbb{P}1$ interpolated field, but using the Z^2 method. Whilst the scatterplot in Subfigure 5.8a indicates a few noticeable point-wise errors (on the boundaries), the majority of the error contributions are much smaller than



(A) Vertex-wise errors (B) Aspect ratios of elements in the adapted mesh

FIGURE 5.6. Vertex-wise errors in each entry of the Hessian of $u(x, y) = \frac{1}{2}(x^2 + y^2)$ as recovered using double L^2 projection, along with an adapted mesh generated using the resulting L^1 normalised Hessian metric.

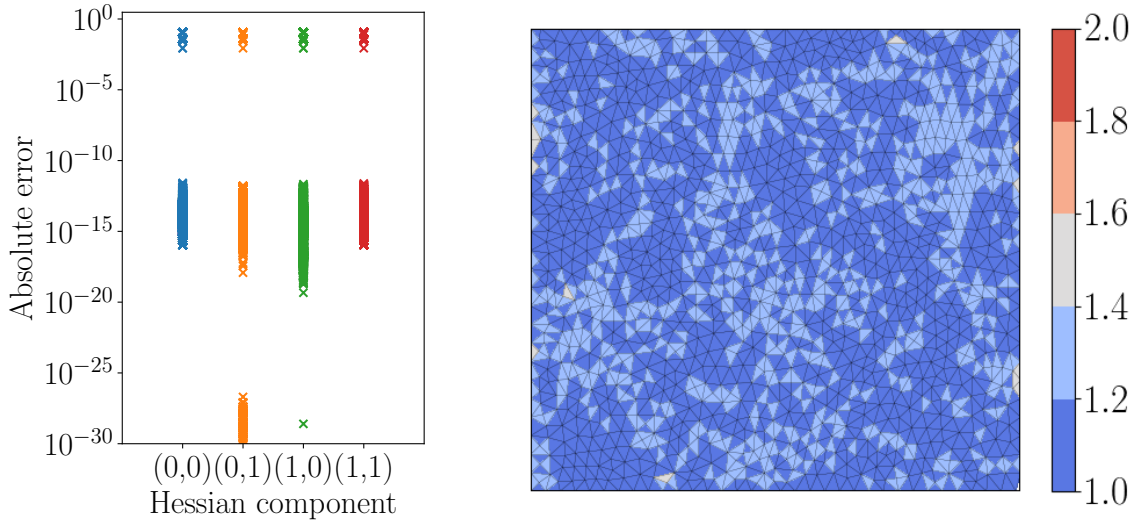


(A) Vertex-wise errors (B) Aspect ratios of elements in the adapted mesh

FIGURE 5.7. Vertex-wise errors in each entry of a ℙ1 approximation of the Hessian of $u(x, y) = \frac{1}{2}(x^2 + y^2)$ as recovered using double L^2 projection, along with an adapted mesh generated using the resulting L^1 normalised Hessian metric.

in Subfigure 5.7a. The mesh is clearly of much better quality than the one presented in Subfigure 5.7b, in terms of isotropy.

Whilst Z^2 recovery refers explicitly to fields which reside in a function space and is not applicable to expressions comprised of multiple such fields, this is not a major concern,



(A) Vertex-wise errors (B) Aspect ratios of elements in the adapted mesh

FIGURE 5.8. Vertex-wise errors in each entry of a $\mathbb{P}1$ approximation of the Hessian of $u(x, y) = \frac{1}{2}(x^2 + y^2)$ as recovered using Zienkiewicz-Zhu, along with an adapted mesh generated using the resulting L^1 normalised Hessian metric.

because typical use cases considered in this work do not apply recovery techniques to expressions. Applied directly to fields, we observe that Z^2 enables the recovery of derivatives with a higher point-wise accuracy than L^2 projection.

Gradient of a Sinusoid. For the second experiment, suppose we seek to recover the gradient of the sinusoidal 'source function' presented in Figure 2.6a from its $\mathbb{P}1$ interpolant. The exact gradient can be computed by hand and ℓ^2 errors may be computed from its values at the vertices. Given a uniform mesh of the unit square, we study the convergence of each recovery technique as global iso-$\mathbb{P}2$ refinements are made.

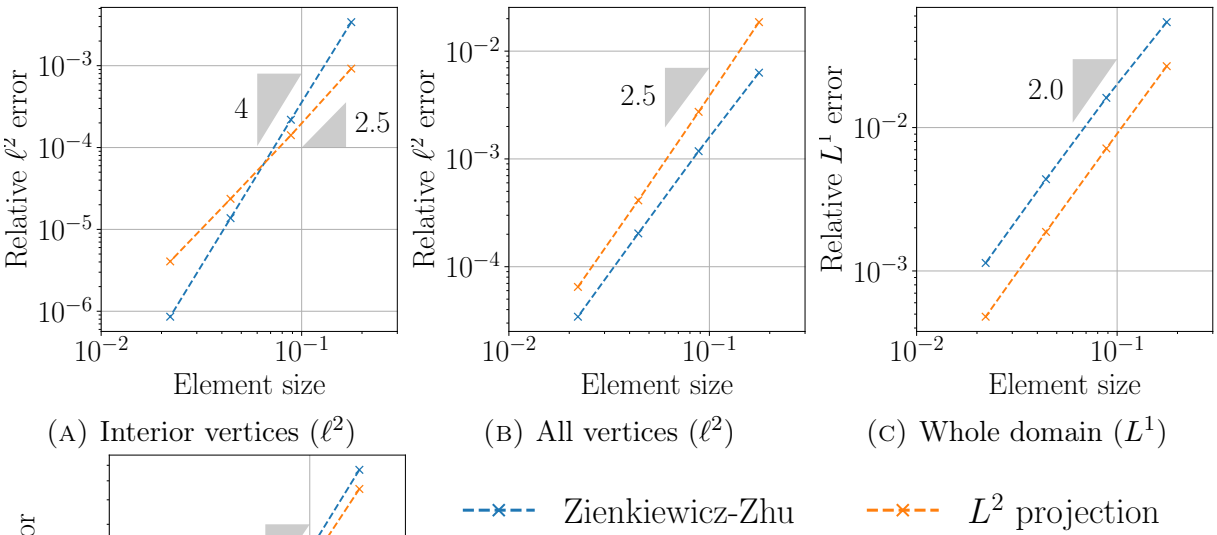
Subfigure 5.9a presents convergence plots for both techniques in the case where boundary nodes are not considered. We observe what the authors of [**Zienkiewicz and Zhu, 1992a**] refer to as $\mathcal{O}(h^4)$ 'ultraconvergence' for Zienkiewicz-Zhu, as expected from that and related works.[2] $\mathcal{O}(h^{2.5})$ superconvergence is observed for L^2 projection.

Subfigure 5.9b shows the case where boundary vertices are included. The Z^2 method is no longer ultraconvergent, although its convergence rate does match that of L^2 projection. Subfigures 5.9c and 5.9d consider L^1 and L^2 errors on the whole domain. Again, the convergence rate of Z^2 is found to match that of L^2 projection.

In summary, Z^2 provides an excellent, ultraconvergent recovery technique regarding vertex-wise errors in the domain interior. If boundary vertices are included or L^1 or L^2 error is deemed to be more important then the method performs comparably to L^2 projection.

In Subsections 7.4.2–7.4.4, a number of goal-oriented metrics are described, each of which involves Hessians and/or gradients of solution variables. As such, their computation

[2]The authors use this term because the convergence is 'two orders higher than normal' – see p.1333 of [**Zienkiewicz and Zhu, 1992a**].



(A) Interior vertices (ℓ^2)

(B) All vertices (ℓ^2)

(C) Whole domain (L^1)

(D) Whole domain (L^2)

FIGURE 5.9. Convergence plots for gradient recovery of a sinusoidal function using L^2 and Z^2 methods. In the first case, boundary vertices are not considered. Note that the 'L^2' in the y-axis label of 5.9d is related to error quantification and should not be confused with the L^2 projection recovery method shown by the orange curves.

requires recovery techniques. The numerical experiments at the culmination of Chapter 7 are all focused around accurately estimating QoIs which take the form of integrals of the forward solution. Where integrated error quantities are concerned, the convergence rate of Z^2 matches that of L^2 projection. As such, either method is a suitable candidate for derivative recovery. The L^2 projection implementation used in this thesis makes use of Firedrake's code generation functionality and is therefore very efficient. The Z^2 implementation used in the above experiments, on the other hand, is written in pure Python and has not been optimised for computational efficiency. For this reason, we opt to use L^2 projection for derivative recovery henceforth, but remark that – with a more efficient implementation – Z^2 would provide an effective alternative.

5.8.2. Comparison of Normalisation Methods. The L^p normalisation approaches introduced in Subsection 5.4.1 allow for any $p \geq 1$, as well as the limit $p \to \infty$. In this subsection, we compare meshes resulting from the cases $p = 1$, $p = 2$, $p = 4$ and $p \to \infty$.

Given the same uniform initial mesh of $[-1, 1]^2$ as before, define the 'sensor' function

$$u : [-1,1]^2 \to \mathbb{R}, \quad u(x,y) := \frac{1}{10}\sin(50x) + \tan^{-1}\left(\frac{1}{10(\sin(5y) - 2x)}\right). \tag{5.46}$$

This function was used to demonstrate multi-scale mesh adaptation in [**Olivier, 2011**]. It contains two scales: a long wavelength, low frequency oscillation in the y-direction and a short wavelength, high frequency oscillation in the x-direction. The long wavelength profile is more prominent, as it is bounded by $\pm\frac{\pi}{2} \approx \pm 1.5708$, whereas the short wavelength profile is bounded by ± 0.1.

Figure 5.10 illustrates meshes which arise with various values of p and a target metric complexity of 10,000. The mesh is adapted four times, with metrics constructed based on the function as represented on each mesh. The lowest order case $p = 1$ is clearly the most multi-scale mesh. It is able to resolve both scales, despite also being the mesh with the fewest elements. With $p = 2$, both scales are still well resolved. The case $p = 4$ could be argued as having a very coarse resolution of the small scale features, but it cannot be said to be multi-scale.

As the norm order is increased, more resolution is dedicated to the more prominent, long amplitude wave. With L^∞ normalisation, 75% more elements are deployed than in the lowest order case, with almost all of them used to resolve the larger scale. L^∞ normalisation is not able to capture the smaller scale whatsoever.

Note that some of these elements are extremely anisotropic, with maximal aspect ratios over one hundred stated in all four cases. In practice, a maximum anisotropy is often enforced, as well as minimum and maximum element sizes. These constraints are enforced in an approximate sense, by thresholding the metric eigenvalues. Doing so seeks to limit the minimum eigenvalue and hence the maximum edge length. This gives rise to a maximum bound on the aspect ratio of an element, without compromising the associated interpolation error [**Piggott et al., 2009**].

It should be noted that if a target interpolation error is imposed for the normalisation, as opposed to a target complexity, then the adaptation routine crashes for L^∞ applied to the problem above, unless bounds on element size and anisotropy are enforced. Admittedly, this is due to an out-of-memory error where only 8GB RAM is available. However, this demonstrates the point that it is harder to predict the mesh complexity under this normalisation strategy than under the target metric complexity approach adopted in this thesis.

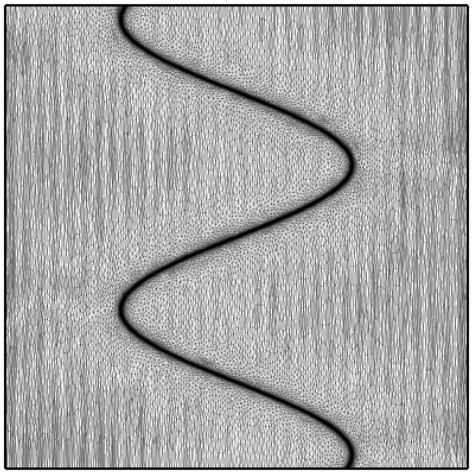
In summary, we find that a lower normalisation order, such as $p = 1$ or $p = 2$ is suitable for multi-scale meshing, whilst anything higher than around $p = 4$ is unsuitable for this purpose. This can be explained by considering the determinant weighting term in (5.33), which has the power $\frac{-1}{2p+n}$. As discussed in [**Loseille and Alauzet, 2011b**], this term is more sensitive for lower orders of p, compared with higher orders where it has little effect. As such, a small value of p is required in order to capture all scales in the solution field.

5.8.3. Intersection vs. Averaging. Consider now two different functions defined on the unit square $[0, 1]^2$:

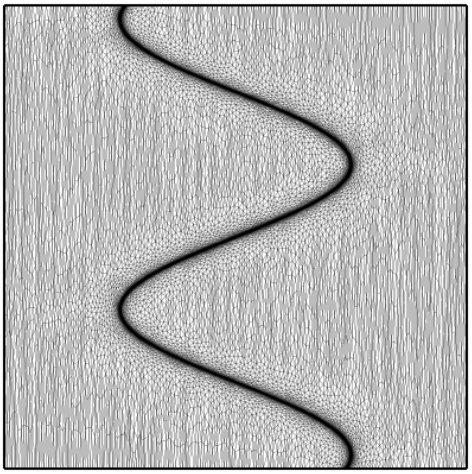
$$u_1(x, y) := \exp\left(-\left|0.5 - x^2 - y^2\right|\right), \quad u_2(x, y) := \exp\left(-\left|0.5 - (1 - x)^2 - y^2\right|\right). \tag{5.47}$$

Under L^1-normalisation, the meshes due to the resulting Hessian metrics are shown in Subfigures 5.11a and 5.11b, respectively. Again, four adaptation steps are applied. The two meshes have comparable element counts and maximum aspect ratio, each with a single arc of high resolution and coarse resolution elsewhere.

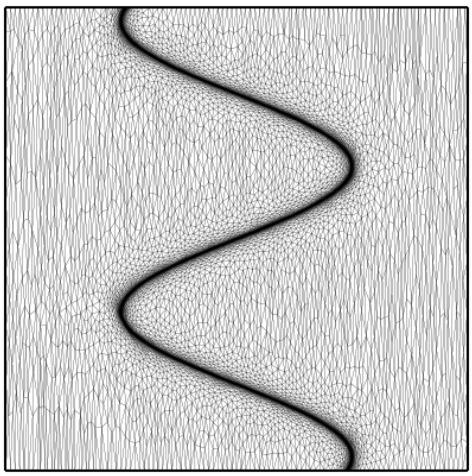
The mesh resulting from adapting w.r.t. the metric average is shown in Subfigure 5.11c. As expected, the mesh resolution is higher surrounding both arcs than elsewhere. However, since the metrics have effectively been averaged against zero in most of the domain, they are not as well resolved as adaptation with either of the constituent metrics. Further, the overall element and DoF counts are lower than both cases.



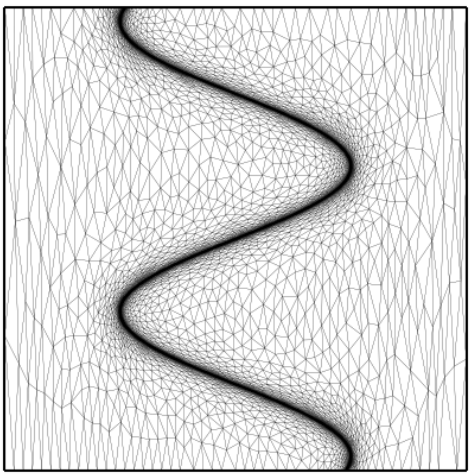
(A) $p = 1$ (45,279 elements, 22,899 vertices, max. AR 103.0)



(B) $p = 2$ (60,015 elements, 30,220 vertices, max. AR 112.2)



(C) $p = 4$ (72,109 elements, 36,202 vertices, max. AR 124.4)

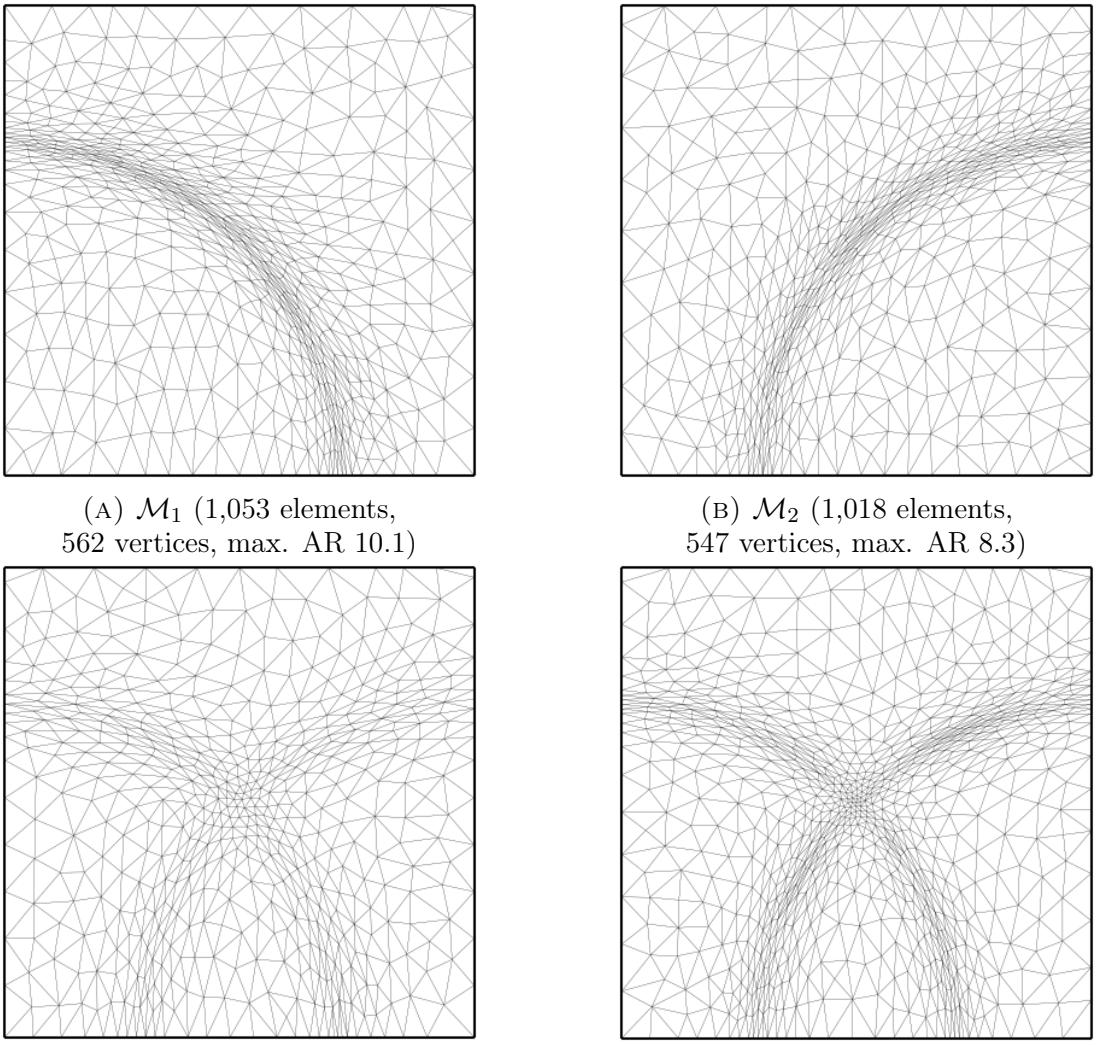


(D) $p \to \infty$ (79,020 elements, 39,600 vertices, max. AR 130.9)

FIGURE 5.10. Meshes adapted to L^p normalised Hessian metrics recovered from scalar-valued function (5.46), which has multiple scales. The cases $p = 1$, $p = 2$, $p = 4$ and $p \to \infty$ are considered. Here 'AR' stands for 'aspect ratio'.

By construction, this is not the case for metric intersection, which yields higher element and DoF counts than both cases, as captioned in Subfigure 5.11d. Using metric intersection, both arcs are captured with at least as much resolution as in the separate cases.

Note that where the arcs meet, both combined metrics yield relatively isotropic elements. Further, the maximum aspect ratio is smaller with combination than without.



(A) $\mathcal{M}_1$ (1,053 elements, 562 vertices, max. AR 10.1)

(B) $\mathcal{M}_2$ (1,018 elements, 547 vertices, max. AR 8.3)

(C) $\frac{1}{2}(\mathcal{M}_1 + \mathcal{M}_2)$ (980 elements, 527 vertices, max. AR 5.6)

(D) $\mathcal{M}_1 \cap \mathcal{M}_2$ (1,490 elements, 791 vertices, max. AR 7.5)

FIGURE 5.11. Meshes adapted to L^1 normalised Hessian metrics of two sensor functions (5.47), as well as the average and intersection thereof.

5.8.4. Application to a Time-Dependent Problem. Consider the advection of a circular tracer bubble in the non-uniform velocity field

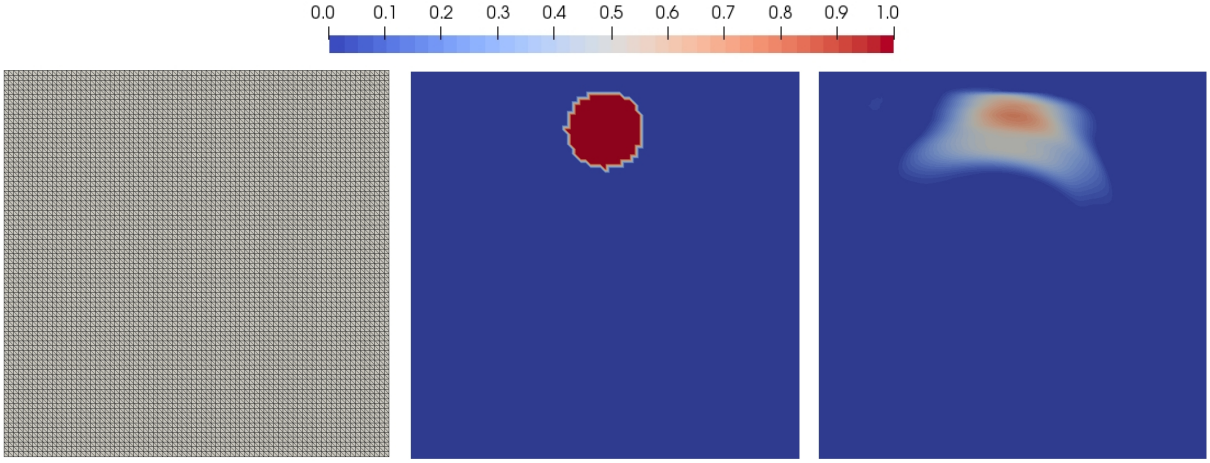
$$\mathbf{u}(x,y,t) := 2\sin(\pi x)\sin(\pi y)\cos\left(\frac{\pi t}{2T}\right)(\sin(\pi x),\sin(\pi y)). \tag{5.48}$$

This velocity field is periodic in time, with period $2T = 6\,\mathrm{s}$. Moreover, $\mathbf{u}(x,y,t+T/2)$ is an odd function in time, meaning that the flow reverses at $T/2$ and the analytical solution at the final time is identical to the initial condition. Tracer concentration is set to zero on all boundaries of the unit square domain. This test case is a two dimensional version of the 3D test case considered in [**Barral et al., 2016**]. Henceforth, we refer to it as the 'Bubble Shear' test case, since the initial tracer bubble undergoes significant shear forces, stretching out into a thin sliver. There is a clear QoI, given by the error at time $t = T$, which we interpret in the L^2 sense:

$$J(c) := \sqrt{\int_\Omega (c(x,y,T) - c_0(x,y))^2 \,\mathrm{d}x}. \tag{5.49}$$

The solution strategy applied in this subsection uses $\mathbb{P}1$ elements with SUPG stabilisation and Crank-Nicolson timestepping with implicitness $\theta = \frac{1}{2}$. The fixed timestep is chosen based on the mesh refinement level. Linear systems arising during the time integration are solved using GMRES with SOR as a preconditioner. Whilst the system at each timestep could be solved in one iteration using LU as a preconditioner, an iterative approach is preferred to avoid assembling the associated matrices on meshes with many DoFs.

Figure 5.12 shows the initial condition and final solution field for a fixed mesh simulation in a $\mathbb{P}1$ space with 6,561 DoFs and timestep $\Delta t = 0.0025$. Clearly, the final solution is far from the analytical solution. The relative L^2 error (as evaluated on the fixed mesh) is 61.20%. The initial condition itself is poorly represented on this mesh, so we cannot expect a high quality simulation. Subfigure 5.12b exhibits the 'shark tooth' mesh imprinting due to uniform meshing referred to in the motivation for this chapter.



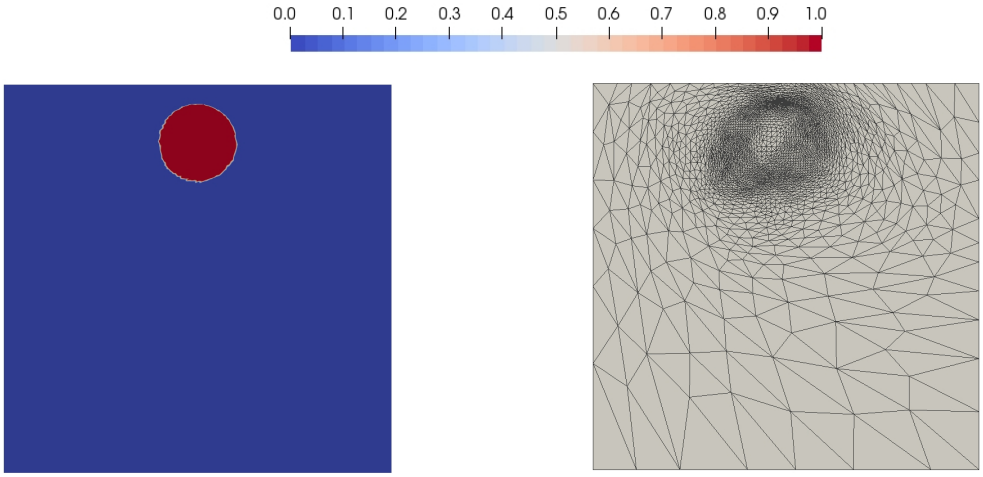
(A) Uniform mesh (B) Initial solution (C) Final solution

FIGURE 5.12. Base mesh used for the 'Bubble Shear' test case, along with initial and final solutions obtained using a $\mathbb{P}1$ space defined upon it. The fixed mesh has 12,800 elements and 6,561 vertices.

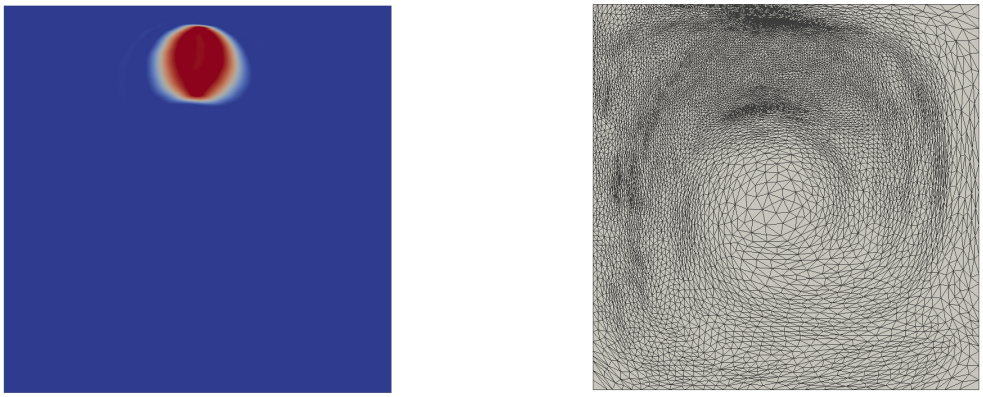
For mesh adaptation based on the space-time normalised Hessian metric, Figure 5.13 illustrates the initial and final tracer concentration, along with corresponding meshes. The time interval is split into 50 uniform subintervals and Hessian metrics are accumulated in time using L^1 normalisation. The number 50 was chosen as a compromise between having sufficiently frequent adaptation to resolve the shearing bubble, versus avoiding high computational cost and interpolation error. L^1 normalisation is also applied in space. Clearly, the discontinuous initial condition is much better represented by these anisotropic elements than by a uniform mesh, despite the fact that continuous finite elements are used. Further, the final solution is a much closer approximation of the analytical solution than was attained on a fixed mesh. The computational cost and accuracy of this implementation are quantified later in this subsection.

Figure 5.14 shows further snapshots at intermediate times. Observe that both the DoF count and maximal anisotropy grows towards the middle of the simulation period, when the tracer cloud is most distorted. As a consequence, the arc of the sheared bubble is captured very well.

By increasing the target space-time complexity, $\mathcal{C}_T$, we are able to perform adaptive runs with more DoFs overall. Subfigure 5.15a shows a plot of relative L^2 error at the final



(A) $t = 0$ (5,138 elements, 2,599 vertices, max. aspect ratio 7.3)

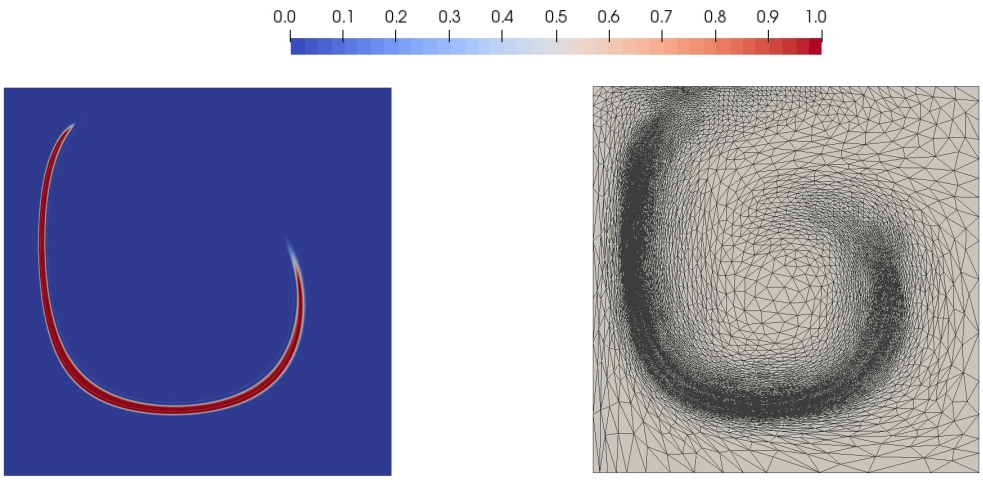


(B) $t = T$ (15,618 elements, 7,882 vertices, max. aspect ratio 19.1)

FIGURE 5.13. Initial and final solutions for an adaptive simulation of the 'Bubble Shear' test case, using a space-time L^1 normalised Hessian-based metric.

time against mean DoF count over the fifty meshes considered. In this experiment, the (constant) timestep is adjusted according to the target space-time complexity. The two different accumulation methods for the Hessian metric – L^1 normalisation (integration) and intersection – are compared against uniform refinement. Clearly, both methods enable greatly improved convergence curves than uniform refinement, in terms of L^2 error vs. mean DoF count. The shape of the two curves due to the Hessian metric are similar, although metric intersection tends to deploy more DoFs, for the same accuracy. It is expected that intersection will use more DoFs in general, but it is interesting that the accuracy is not noticeably improved by doing so.

Subfigures 5.15b and 5.15c show corresponding plots involving overall run times. For uniform refinement, these run times account only for PDE solves. For the adaptive cases, the CPU times associated with mesh adaptation and interpolation are included. It is interesting to observe that, whilst intersection uses more DoFs to attain a similar accuracy, the overall run times are comparable to L^1 normalisation. This illustrates that DoF count is not necessarily a good proxy for computational cost in the context of time-dependent simulations. It should be remarked that a minimum iteration count of three was imposed on the mesh adaptation fixed point iteration in both cases. If this was not the case, it is



(A) $t = T/4$ (13,611 elements, 6,850 vertices, max. aspect ratio 23.1)

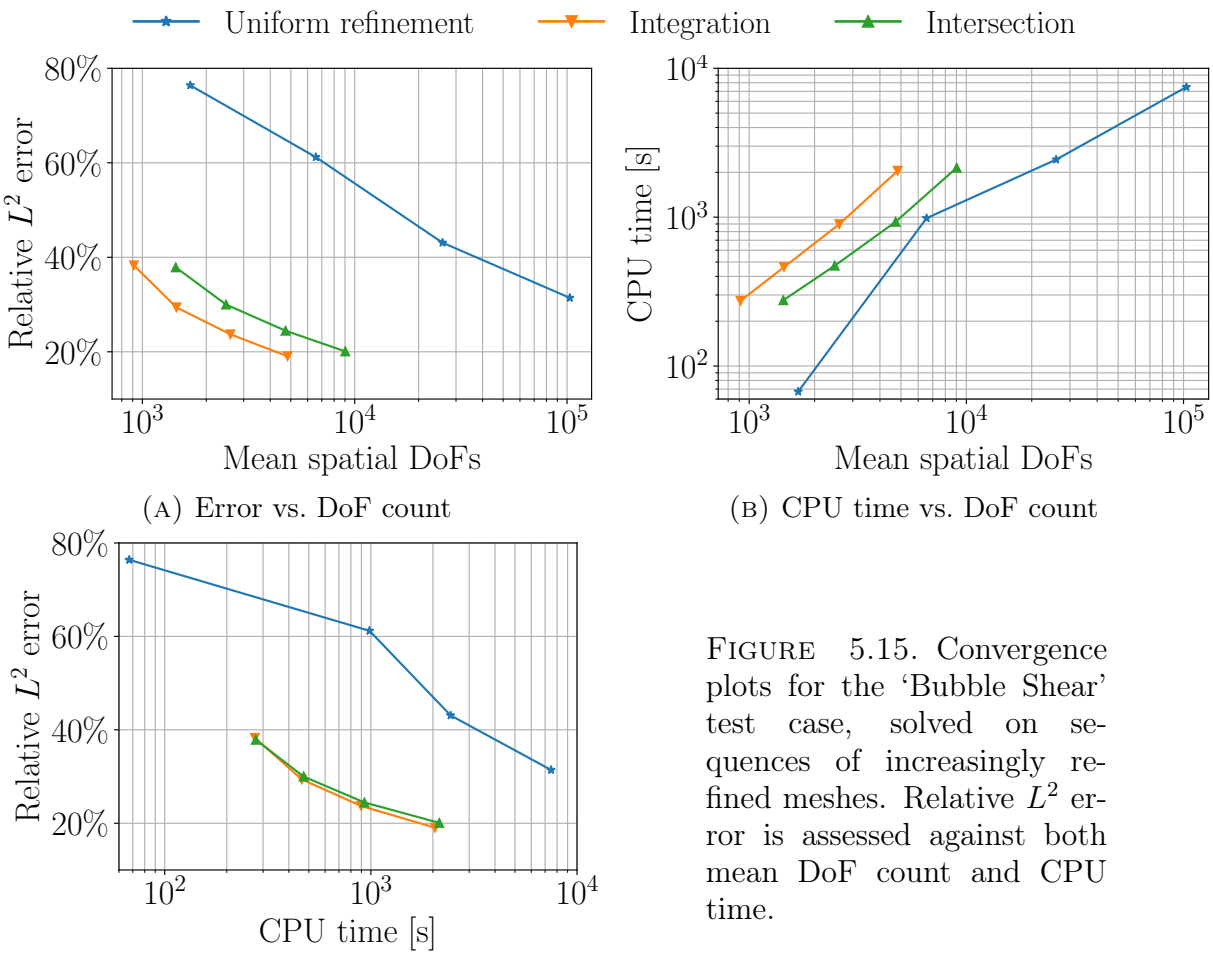
(B) $t = T/2$ (24,689 elements, 12,391 vertices, max. aspect ratio 37.5)

(C) $t = 3T/4$ (16,920 elements, 8,514 vertices, max. aspect ratio 34.5)

FIGURE 5.14. Intermediate solutions for an adaptive simulation of the 'Bubble Shear' test case, using a space-time L^1 normalised Hessian-based metric.

plausible that a simulation with more DoFs overall converges in fewer iterations than one with fewer DoFs, leading to a shorter run time.

Figure 5.16 shows DoF counts for the meshes associated with each of the 50 subintervals. Both accumulation by L^1 normalisation and intersection are considered, as well as a range of target metric complexities, $\mathcal{C}_T$. In both cases, the DoF count peaks around halfway through the simulation, when the tracer bubble is at its most distorted. This is consistent



(A) Error vs. DoF count

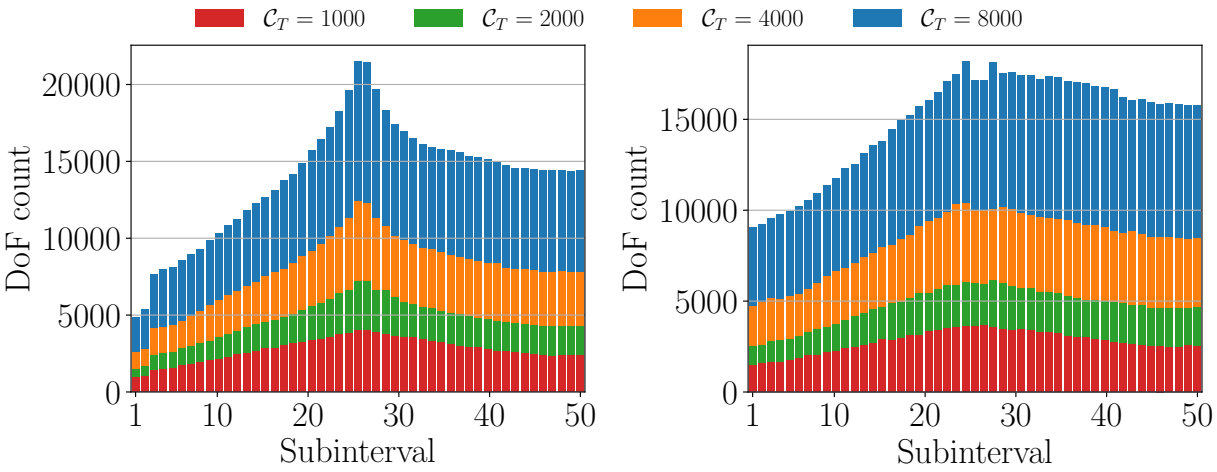
(B) CPU time vs. DoF count

(C) Error vs. CPU time

FIGURE 5.15. Convergence plots for the 'Bubble Shear' test case, solved on sequences of increasingly refined meshes. Relative L^2 error is assessed against both mean DoF count and CPU time.

with the observations in Figure 5.14. It appears that, under L^1 normalisation, DoF count ramps up from a relatively small initial value and peaks sharply at the halfway point. Under the latter approach, the DoF count does not reach as high a peak and is less varying over the second half of the simulation. Given the wider range of DoF counts, the bar charts due to L^1 normalisation makes more use of the fact that h-adaptation methods can have varying numbers of DoFs in time, as well as space. This fits with the spatial L^1 normalisation concept, which is the most multi-scale, out of all spatial L^p normalisation methods.

In summary, the time-dependent metric-based framework has been validated for a 2D tracer transport problem with an analytical solution. For the setup considered, a reduction in L^2 error of 20% or more is attainable for the same overall CPU time, as compared with uniform meshing. Two methods for accumulating metric information in time have been considered, both of which are demonstrated to be effective in attaining the above improvement. Whilst the two methods deploy DoFs differently in both time and space, they are found to require a similar cost in order to attain the same accuracy.



(A) L^1 normalisation (B) Intersection

FIGURE 5.16. DoF counts for the meshes associated with each subinterval for Hessian-based simulations of the 'Bubble Shear' test case, under two accumulation methods.

CHAPTER 6

Monitor-Based Mesh Adaptation

Motivation. As discussed in Chapter 5, the Riemannian metric framework is not specific to any particular mesh adaptation method, enabling the incorporation of anisotropy under h-adaptive, r-adaptive and hybrid methods. This chapter focuses on a similar framework, which is only relevant for r-adaptation: adaptation driven by *monitor functions*.

Like a Riemannian metric, a monitor function is a strictly positive field defined over the spatial domain, which prescribes *where* the mesh should be refined or coarsened. However, monitor functions are scalar valued, rather than tensor valued. They convey the concept of *mesh density*, in the sense that high mesh resolution (i.e. high element count) is prescribed where the monitor function takes a large value. Conversely, low mesh resolution is prescribed where the monitor function takes a small value. What is meant by large or small should be understood in a relative sense. The mechanism by which alterations of the mesh geometry are prescribed is called *equidistribution*. A mesh is said to equidistribute the monitor function if the volume integral thereof is equal across all elements. This notion was first introduced by [**White, 1979**], with the monitor function relating to the arc length of a one dimensional solution field.

It is also possible to extend the monitor-based framework to the tensor case, allowing for control of anisotropy (see [**Huang and Kamenski, 2015**]). However, scalar monitor functions were found to be sufficient for the experiments documented in this chapter.

Restricting to r-adaptation means losing the capability to alter mesh topology. In cases where important flow features traverse the entire domain, this can involve major changes to the mesh geometry, introducing severe element stretching, as shown in subsequent numerical experiments. In extreme cases, elements can become so stretched that they experience a change of orientation, rendering them invalid. This phenomenon, known as *mesh tangling*, can cause major issues for the r-adaptation algorithm and/or PDE solvers built upon the adapted mesh.

Despite the disadvantages of losing h-adaptation capability, pure r-adaptation has a number of desirable numerical and computational properties. The fixed topology means that r-adaptation methods are typically simple to parallelise, since there is no need to modify the parallel decomposition when the mesh is modified. Moreover, r-adaptation can be used with directly addressed meshes. The implementation of r-adaptation can often be straightforwardly achieved in a variational framework, and in particular within the same finite element package as the PDE solver is written. This reduces dependency on external libraries and gives heightened flexibility. For some forms of r-adaptation, such as Lagrangian type methods, the transfer of data between meshes by remapping may be avoided, implying reduced interpolation error.

The main focus of this chapter is on optimally transported mesh movement driven by solutions of a Monge-Ampère type equation. Such an approach has been advocated in a

number of recent papers on mesh movement applied to geophysical flows, including [**Budd and Williams, 2009**, **Budd et al., 2013**, **Weller et al., 2016**, **McRae et al., 2018**]. Optimal transport based methods enable the tracking of features of the solution field held to be of importance, whilst minimising deformations to the mesh geometry. They also come with the advantage of having inbuilt avoidance of mesh tangling. However, the Monge-Ampère equation is a nonlinear PDE which can be computationally expensive to solve to a high tolerance. This means that it is unclear whether the improvement in solution accuracy afforded by this mesh movement method is worth the additional computational expense.

Novel Contributions. The main novel contribution in this chapter is the application of optimal transport based mesh adaptation methods to a sediment transport problem, as presented in [**Clare et al., 2020**]. For a test case describing the erosion and deposition of a bathymetry trench, a number of experiments are performed regarding mesh adaptation parameter choices. First, the effect of solver tolerance for the Monge-Ampère equation is assessed. This is motivated by the idea that, with a relative tolerance as large as $\mathcal{O}(10^{-4})$ – or even $\mathcal{O}(10^{-3})$ – the computational cost may be reduced significantly, making the method a more viable choice. The effect of the mesh movement frequency is also assessed. Adapting the mesh every timestep is usually prohibitively expensive, so it would be motivating to be able to adapt less often and still retain significant improvements in accuracy. Finally, parameters related to the choice of monitor function are considered, to inform future moving mesh sediment transport simulations.

Chapter Organisation. The chapter is organised as follows. Section 6.1 formulates the concept of mesh tangling mathematically and illustrates how it can occur in a Lagrangian r-adaptation method. The use of global remeshing as a de-tangling tool is also demonstrated. In Section 6.2, the concept of mesh equidistribution is introduced and then used to define a mesh adaptation method which optimally transports a user-specified monitor function in Section 6.3. This mesh adaptation routine and the underlying monitor function framework are tested in Section 6.4, including a comparison of monitor combination methods. Numerical experiments in Section 6.5 demonstrate the application of this r-adaptation method to an established sediment transport modelling test case referred to above, with experimental data available due to [**Van Rijn, 1980**]. In these experiments, it is shown that, with appropriate parameter tuning, it is possible to attain a higher accuracy than the fixed mesh case, for a comparable computational cost. The terms *mesh movement* and *r-adaptation* are used interchangeably throughout.

6.1. Mesh Tangling

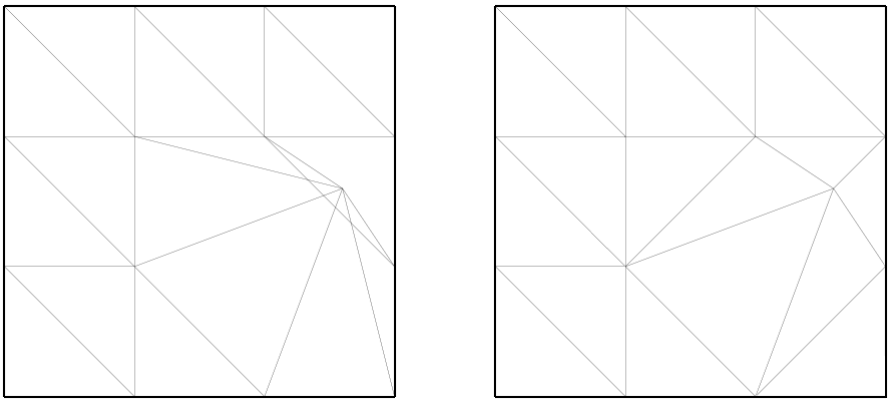
As discussed above, *mesh tangling* is an important consideration for mesh movement algorithms. Adaptation in these cases is driven solely by changes to the mesh coordinates, which may be interpreted as positional changes to the mesh vertices, i.e. a $(\mathbb{P}1)^n$ coordinate field (assuming a linear mesh). These changes, in turn, alter the geometry of mesh elements. If vertices move in such a way that they never cross an edge, mesh orientation is preserved. However, if they do cross, then the orientation of at least one element changes. This is what we call mesh tangling.

For some test cases, such as where mesh vertices are simply advected in a uniform or rotational velocity field, tangling issues do not arise even with simple Lagrangian mesh movement methods. Consider, for example, the classic rotational advection test case

described on pp.649–653 of [LeVeque, 1996]. If the flow is even slightly non-uniform, however, mesh tangling can arise.

Recall the transformation from the reference element $\widehat{K}$ to a mesh element $K \in \mathcal{H}$, as given in (2.26). The cell Jacobian $\underline{\boldsymbol{J}_K}$ in this affine map conveys a sense of how $\widehat{K}$ gets distorted in shape and orientation. The sign of the determinant is negative if an orientation change has occurred. Mesh tangling occurs when $\det(\underline{\boldsymbol{J}_K})$ changes sign following mesh adaptation.

Subfigure 6.1a shows a mesh with an inverted element. One means of remedying this inversion is to apply a *remeshing* step. This involves the regeneration of the entire mesh or a local patch thereof. An example of a local remeshing method relevant to tangled meshes is the application of a de-tangling algorithm, such as [Stees and Shontz, 2018]. Here we use a simple global remeshing method given by passing the vertices of the (tangled) mesh to a mesh generator – in this case the implementation of constrained Delaunay provided by the 2D triangular mesh generator, *triangle* [Rhufat, 2020]. Subfigure 6.1b demonstrates de-tangling by remeshing in this way. An alternative approach is to apply a h-adaptation method in the static sense.



(A) Before remeshing (B) After remeshing

FIGURE 6.1. A mesh with an inverted element and a remeshed version using *triangle*.

6.1.1. Lagrangian Mesh Movement. In order to illustrate mesh tangling during a PDE solve, we consider a very simple mesh movement method.

Aside from the Coriolis term, which represents the effects of a rotating frame of reference, this thesis focuses on the *Eulerian* interpretation of physics. That is, fluid dynamics and tracer transport are considered in a *fixed frame of reference*. In contrast, the *Lagrangian* interpretation considers frames of reference moving with the fluid. In the context of coastal ocean modelling, such an approach is useful for tracking objects – such as buoys – which are transported by currents and tides. The Lagrangian interpretation lends itself well to particle-based, meshless methods, such as smooth particle hydrodynamics [Lucy, 1977, Gingold and Monaghan, 1977].

Suppose we have a time-dependent PDE involving an advective velocity $\mathbf{u}$, which has been discretised on a mesh $\mathcal{H}$ with coordinates $\boldsymbol{\xi}$. Consider the subset of r-adaptation

methods for which the mesh transformation can be expressed in terms of a *mesh velocity* $\mathbf{v}$, through the relation

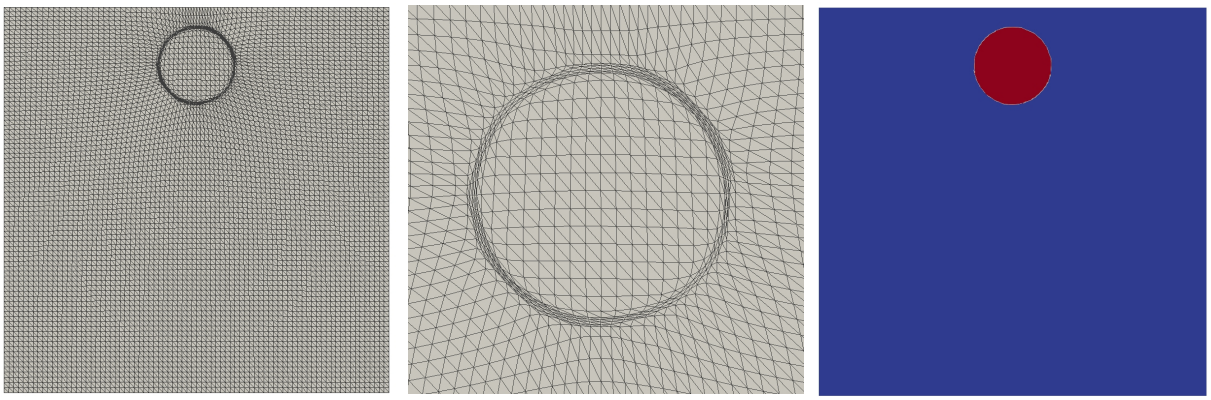
$$\left.\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\right|_{\boldsymbol{\xi}} = \mathbf{v}(\boldsymbol{\xi}, t). \tag{6.1}$$

Lagrangian mesh movement methods are those for which the coordinate field is transported by the fluid velocity, i.e. $\mathbf{v} := \mathbf{u}$. Equation (6.1) may be coupled to the prognostic equation by subtracting $\mathbf{v}$ from the fluid velocity $\mathbf{u}$ multiplying the advection term. Since $\mathbf{v} = \mathbf{u}$ in the Lagrangian case, the advection term simply drops out of the equation. The Eulerian case may be recovered by setting $\mathbf{v} := \mathbf{0}$. Other choices of $\mathbf{v}$ are also possible, giving rise to *Arbitrary Lagrangian-Eulerian (ALE)* type methods (see [**Donea et al., 2017**], for example).

A major advantage of moving the mesh with the flow is that advection errors are avoided in the resulting numerical schemes. However, if no precaution is made for mesh tangling when the mesh velocity is subtracted from the fluid velocity then issues can arise for non-trivial flows, as shown in the following subsection.

6.1.2. Bubble Shear Test Case. Consider again the 'Bubble Shear' test case introduced in Subsection 5.8.4. As illustrated in Subfigure 5.12c, serious numerical diffusion is introduced under the (fixed mesh) Eulerian interpretation, unless a small element size is prescribed. We apply mesh movement in an attempt to avoid this without increasing DoF count. Mesh movement equation (6.1) is time integrated using Crank-Nicolson with implicitness $\theta = \frac{1}{2}$.

In order to accurately capture the initial condition, the methods later introduced in Section 6.3 are used to obtain a mesh with increased mesh resolution surrounding the initial tracer cloud. This ensures that the (discontinuous) jump in tracer concentration is well-resolved, even with a $\mathbb{P}1$ function space. In the following we demonstrate that, despite the fact that we have a pure advection problem, it is not sufficient to apply this monitor-based approach and then advect the mesh in a Lagrangian sense.

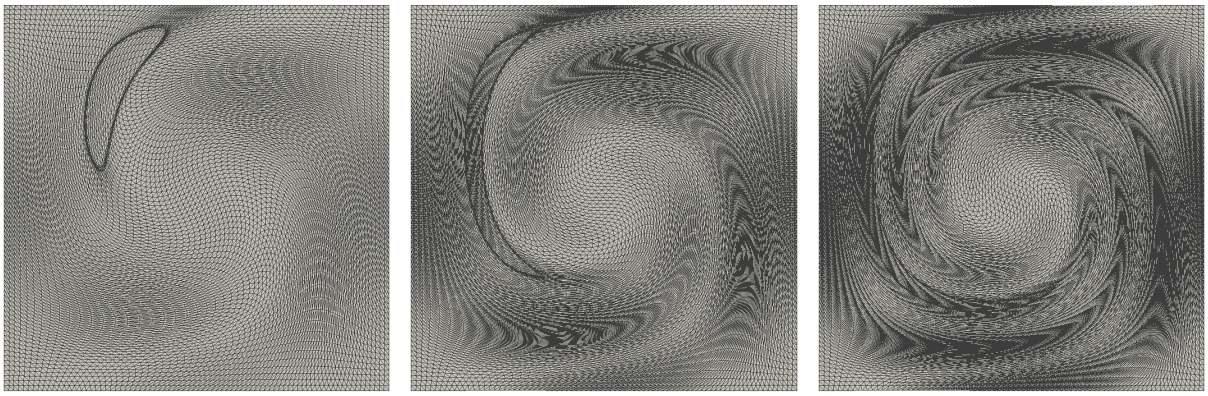


(A) Initial mesh (B) Zoom on refined region (C) Initial tracer concentration

FIGURE 6.2. Initial mesh and tracer concentration for 'Bubble Shear' test case.

Given the initialisation shown in Figure 6.2, the snapshots in Figure 6.3 show advected meshes. Elements are clustered very densely around the boundary of the initial tracer

cloud and their advection leads to inverted elements after 75 timesteps. The solver manages to withstand these invalid geometries until timestep 196, when 478 of the 12,800 elements are inverted. After this, the mesh is completely deformed and both the domain volume and solution norm grow rapidly. This is clearly unrealistic in a problem concerned with the advection of a fixed tracer quantity.



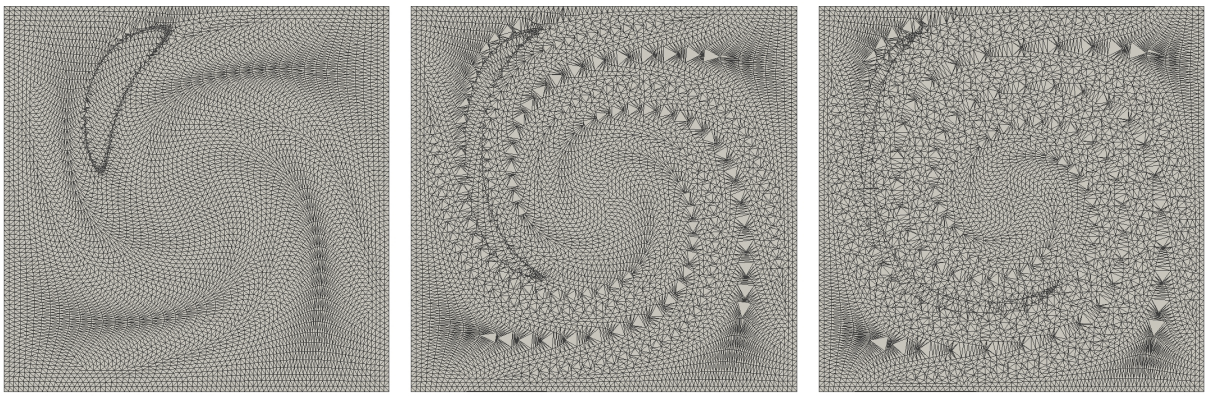
(A) Timestep 65 (B) Timestep 131 (C) Timestep 196

FIGURE 6.3. Snapshots of Lagrangian mesh movement under the velocity field in the 'Bubble Shear' test case, starting from the initial mesh shown in Subfigure 6.2a.

Consider applying the same remeshing as in Figure 6.1 whenever an element becomes tangled. Doing this ensures that the PDE is always solved on a valid mesh. However, it leads to rather unsatisfactory elements which do not accurately represent the solution field, as demonstrated in Figure 6.4. Since the problem is pure advection, there are only two sources of discretisation error: the timestepping routine and the mesh-to-mesh data transfer. The mesh topology is only ever modified by connectivity changes, so one valid choice of mesh-to-mesh data transfer for the $\mathbb{P}1$ fields used here is to just hold the value at each vertex fixed. This is equivalent to linear interpolation at the vertices and is exact in the ℓ^p sense because the vertices (resp. quadrature nodes) of the source and target meshes (resp. function spaces) are identical. If a different approach is used, such as conservative interpolation, this becomes the main source of error (assuming a sufficiently small timestep so that timestepping errors are negligible).

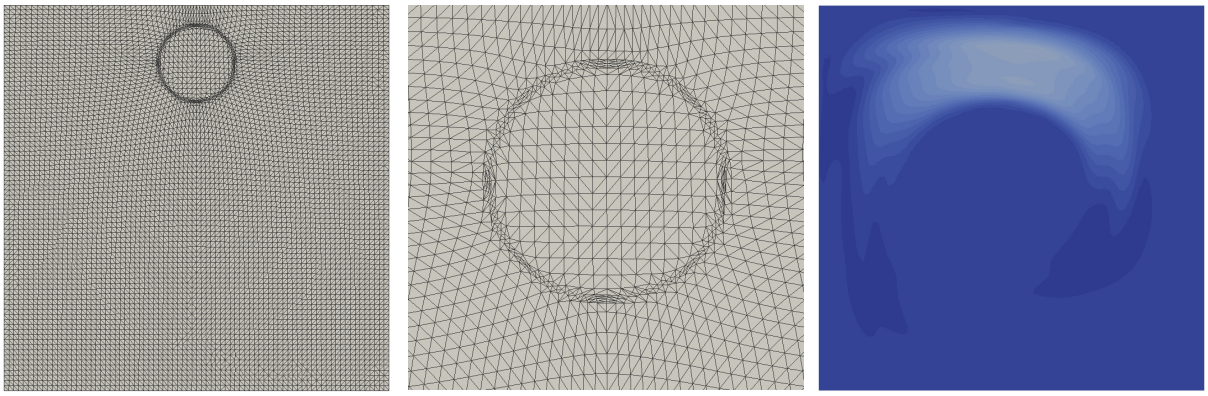
Figure 6.5 shows the mesh and solution field at the final time, in the case where conservative interpolation is used for the mesh-to-mesh data transfer. Since the velocity field is periodic, we expect the mesh vertices at the final time to be positioned as in the initial mesh. That this is the case (up to small time integration errors) is shown by comparing the close-ups in Subfigures 6.2b and 6.5b.

All of the artefacts due to constrained Delaunay unwind when the velocity field reverses. Whilst the vertex positions are almost identical, the remeshing has clearly reduced anisotropy in the curve of increased refinement. This is because *triangle* seeks to minimise aspect ratio of elements in the triangulation. Where before the domain volume grew uncontrollably, it is almost conserved in this case, with error 0.01%. (This small discrepancy is due to errors in the vertex positions introduced by the timestepping scheme.) However, comparing Subfigures 5.12c and 6.5c, we observe that the final tracer solution is of much worse quality than the fixed mesh case. Again, this would not be the case if vertex-wise solution data were simply held fixed during mesh connectivity changes.



(A) Timestep 65 (B) Timestep 131 (C) Timestep 196

FIGURE 6.4. Snapshots of a Lagrangian mesh simulation of the 'Bubble Shear' test case with tangling fixed by remeshing.



(A) Final mesh (B) Zoom on refined region (C) Final tracer concentration

FIGURE 6.5. Final mesh and tracer concentration for a Lagrangian simulation of the 'Bubble Shear' test case with tangling fixed by remeshing.

Alternatively, we could apply a h-adaptive method, instead of remeshing. In the case of Pragmatic, this means applying the steady-state fixed point iteration with a metric constructed from the Hessian of the tracer solution (say). Doing so means we do not benefit from the space-time normalisation techniques available in the metric-based framework. Under h-adaptation, linear interpolation is clearly no longer ℓ^p-exact – ℓ^p-exactness does not even make sense.

Whilst de-tangling tools, static h-adaptation and remeshing provide possible approaches for avoiding tangled meshes, it is entirely possible that a mesh movement algorithm, which has led to tangling, will do so once again once the mesh has been de-tangled, requiring further applications of the de-tangling routine. For mesh movement driven by error indicators, for example, a mesh tangles because the error indicator has told the mesh to move in such a way that it tangles. De-tangling means going against what the indicator dictates; undoing it will likely go against this. Further, it is difficult to predict in advance how often the mesh will tangle, meaning the overall computational cost cannot be easily assessed in advance.

In summary, the demonstration in this section motivates the use of r-adaptation methods which ensure that mesh tangling never occurs, as opposed to dealing with it whenever

it occurs or is about to occur. Monitor-based mesh adaptation driven by solutions of Monge-Ampère type equations, as described in Section 6.3, is an example of one such method.

6.2. Equidistribution

A mesh movement problem may be interpreted as the search for a transformation between a fixed computational domain, Ω_C, and a physical domain, Ω_P:

$$\mathbf{x} : \Omega_C \to \Omega_P. \tag{6.2}$$

More specifically, we seek a discrete representation of (6.2) such that, for a given mesh of the computational domain, $\mathcal{H}_C$, a mesh of the physical domain is determined by the image:

$$\mathcal{H}_P = \mathbf{x}(\mathcal{H}_C). \tag{6.3}$$

In practice, we usually do not obtain the map (6.2) itself; really we are only interested in its image, in a similar way to how a matrix-vector product can be computed using a matrix-free method. Since this chapter is concerned with r-adaptation strategies, the map is defined such that meshes $\mathcal{H}_C$ and $\mathcal{H}_P$ are topologically equivalent.

In general, computational and physical domains admit infinitely many transformations of the form (6.2). To make the mesh movement problem well-posed, we impose constraints on the transformation. One approach is to assert that some user-provided monitor function is *equidistributed* in the adapted space.

A consequence of the Radon-Nikodym Theorem (see [**Capinski and Kopp, 2013**], for example) is that, for any non-singular map (6.2), there exists a unique (monitor) function $m : \Omega_P \to (0, \infty)$ which satisfies

$$\int_R \mathrm{d}\xi = \int_{\mathbf{x}(R)} m \,\mathrm{d}x, \qquad \forall R \subset \Omega_C. \tag{6.4}$$

In [**Budd et al., 2009**], it is shown that (6.4) gives rise to a nonlinear PDE

$$m \det \underline{\boldsymbol{J}} = \theta, \quad \text{where} \quad \theta := \frac{\int_{\Omega_P} m \,\mathrm{d}x}{\int_{\Omega_C} \mathrm{d}\xi} \quad \text{and} \quad \underline{\boldsymbol{J}} := \frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} \tag{6.5}$$

is the Jacobian of the map w.r.t. the computational coordinates, $\boldsymbol{\xi}$. Equation (6.5) is called the *equidistribution relation*.

The normalisation coefficient $\theta > 0$ exists to ensure that the domain volume is conserved. This is important when the domain boundary is allowed to deform, whereby its volume changes. If boundary deformations are not allowed (or constrained in some way) then appropriate boundary conditions must be enforced when solving (6.5).

Observe that the monitor function plays a very important role in (6.5). If it is constant then the Jacobian is determined by boundary deformations alone. In the case of no boundary deformations, this implies a mesh whose elements have uniform volume. If, however, the monitor function is not constant and takes larger values in some region of the domain than elsewhere, then the Jacobian determinant is forced to be smaller in that region than elsewhere. When (6.5) is used to drive a mesh movement algorithm, the result

is that mesh elements have a smaller volume in said region. As such, we may interpret the monitor function as a 'mesh density'.

In the above, we have implicitly assumed that the computational and physical domains are fixed in time. For time-dependent simulations, it is often desirable to have a physical mesh which moves as time progresses. For example, it is common to consider monitor functions which are dependent on solution fields from another PDE which is solved on the physical mesh. All that is required in order to incorporate this is to allow m to vary with time, which implies changes to both the normalisation coefficient and physical domain. Whenever we desire that the mesh be moved, the equidistribution system is solved and the new physical coordinates are computed. The computational domain and mesh remain fixed for all time. See Section 2 of [**Clare et al., 2020**] for an explicit mathematical rendering of the time-dependent case.

For one dimensional problems, the determinant of the Jacobian is simply the first derivative and (6.5) is well-posed (assuming appropriate choices of monitor function and normalisation coefficient). For two or more dimensions, additional constraints must be imposed in order to obtain a well-posed problem. A constraint choice based on optimal transport theory is discussed in the following section, although it is by no means the only option available.

6.3. Optimally Transported Mesh Movement

6.3.1. The Monge-Ampère Equation. The ultimate goal of mesh movement is to obtain the map (6.2) between the computational and physical domains. The idea of optimal transport is for this map to take the minimum amount of work in terms of deformation of the computational space. That is, it seeks to solve the optimisation problem

$$\min_{\mathbf{x}:\Omega_C\to\Omega_P} \|\mathbf{x}(\boldsymbol{\xi},t)-\boldsymbol{\xi}\|_{L^2(\Omega_C)} \quad \text{subject to} \quad (6.5). \tag{6.6}$$

Surprisingly, (6.6) admits a unique solution. In [**Delzanno et al., 2008**] (for example), it is proved that the solution takes the form $\mathbf{x} = \nabla_{\boldsymbol{\xi}}\widetilde{\phi}$, where $\widetilde{\phi} = \widetilde{\phi}(\boldsymbol{\xi},t)$ is a convex potential. Alternatively, the form

$$\mathbf{x}(\boldsymbol{\xi}) = \boldsymbol{\xi} + \nabla\phi_{\boldsymbol{\xi}}(\boldsymbol{\xi}), \tag{6.7}$$

can be used, where $\phi + \frac{1}{2}\boldsymbol{\xi}\cdot\boldsymbol{\xi}$ is convex. An advantage of the (6.7) version is that it extends to manifolds other than Euclidean space [**McRae et al., 2018**].

Substituting the additional constraint (6.7) into equidistribution relation (6.5) gives a nonlinear PDE of *Monge-Ampère* type,

$$m\ \det(\underline{\boldsymbol{I}} + \underline{\boldsymbol{H}}(\phi)) = \theta, \qquad \text{where} \qquad \underline{\boldsymbol{H}}(\phi)_{ij} := \frac{\partial^2\phi}{\partial\xi_i\partial\xi_j} \tag{6.8}$$

denotes an entry of the Hessian of ϕ and $\underline{\boldsymbol{I}}$ is the identity matrix. For a given choice of monitor function and boundary conditions defining boundary deformations, (6.8) may be solved for the scalar potential, ϕ. Substitution into (6.7) then determines the map, which is used to drive the mesh movement.

If (6.8) is solved to a high tolerance, we have grounds to claim that the resulting meshes are *optimally transported* w.r.t. the monitor function. Whilst this is mathematically desirable,

it comes with computational expense. In practice, for the purposes of tracking features in coastal flows, preliminary testing in Section 6.5 indicates that it is sufficient to solve to a relative tolerance of $\mathcal{O}(10^{-4})$ or even $\mathcal{O}(10^{-3})$. Whilst this will likely reduce the computational cost of the mesh movement algorithm, we are no longer able to claim that the mesh is optimally transported.

The theory underpinning the solution of Monge-Ampère type equations such as (6.8) requires a degree of domain convexity (see Theorem 4.3 of [**Budd and Williams, 2009**]). Coastlines are fractal, meaning that any coastal domain is inherently non-convex. This is usually still true when the coastline is approximated as piecewise smooth. As such, a major criticism of the mesh movement method discussed in this section is that it is not designed to handle coastal geometries. However, as remarked in [**Clare et al., 2020**], the inclusion of certain types of wetting-and-drying scheme in the hydrodynamics solver allows the circumnavigation of such issues. That is, the total domain can be chosen to be a convex superset of the oceanic domain of interest. This is possible with the wetting-and-drying scheme developed in [**Kärnä et al., 2011**], which advocates a modified (everywhere positive) water depth, as opposed to classifying cells as 'wet' or 'dry'. This means that the prognostic and mesh movement equations are always solved over the total domain, rather than over potentially non-convex subdomains. It also means that islands within the domain do not pose issues for the mesh movement solver. Whilst wetting-and-drying is not used in the numerical experiments presented in this thesis, the application of Monge-Ampère mesh movement methods to hydro-morphodynamic models with wetting-and-drying was addressed in [**Clare et al., 2020**].

6.3.2. Mesh Tangling. As motivated at the end of Subsection 6.1.2, mesh movement driven by solutions of the Monge-Ampère equation is constructed such that mesh tangling cannot occur. Given the relation in (6.5), it must be the case that $\det \underline{\boldsymbol{J}} > 0$, since both the monitor function and normalisation coefficient are strictly positive. This implies that no two vertices of the original mesh can meet and no two edges of the original mesh can meet. However, it should be noted that this is only true *provided that (6.8) has been solved to a sufficiently high accuracy.*

6.3.3. Finite Element Discretisation. In this subsection, we describe the strategy used for solving (6.8) in this thesis.

We opt to solve this nonlinear elliptic equation using parabolisation, introducing a 'pseudotime' variable, τ, as well as a 'pseudotimestep', $\Delta\tau > 0$. Parabolisations for equations of Monge-Ampère type equations were investigated in detail in [**Budd and Williams, 2009**]. Herein, we adopt the related approach advocated in [**McRae et al., 2018**]:

$$-\frac{\partial}{\partial\tau}\nabla^2\phi = m\ \det(\underline{\boldsymbol{I}} + \underline{\boldsymbol{H}}(\phi)) - \theta. \tag{6.9}$$

The idea is that, if we integrate (6.9) in pseudotime, the solution should approach steady state, whereby the pseudotime derivative term drops out and we recover solutions of (6.8). That this occurs is proved in [**Awanou, 2015**], assuming an appropriate choice of pseudotimestep and that the initial guess is sufficiently close to the solution. Throughout this chapter, a value $\Delta\tau = 0.1$ is found to be sufficient for practical purposes.

For the pseudotime integration, it is sufficient to use explicit Euler. For the spatial discretisation, we follow [**McRae et al., 2018**] in using mixed FEM. Suppose the potential $\psi \in H^2(\Omega)$ and its Hessian $\underline{\boldsymbol{H}} \in L^2(\Omega)$ may be approximated in a finite

dimensional function spaces $\Psi_h \subset H^2(\Omega)$ and $\Sigma_h \subset L^2(\Omega)$. A sequence of iterates $\{(\phi_h^{(k)}, \underline{\boldsymbol{H}}_h^{(k)}) \in (\Psi_h, \Sigma_h)\}_{k\in\mathbb{N}}$ is generated by first applying explicit Euler as

$$\left\langle \nabla\psi, \nabla\phi_h^{(k+1)} \right\rangle = \left\langle \nabla\psi, \nabla\phi_h^{(k)} \right\rangle + \Delta\tau \left\langle \psi, m\ \det(\underline{\boldsymbol{I}} + \underline{\boldsymbol{H}}_h^{(k)}) - \theta \right\rangle, \quad \forall\psi \in \Psi_h, \tag{6.10}$$

and then recovering the Hessian using an L^2 projection:

$$\left\langle \underline{\boldsymbol{\sigma}}, \underline{\boldsymbol{H}}_h^{(k+1)} \right\rangle = -\left\langle \nabla\cdot\underline{\boldsymbol{\sigma}}, \nabla\phi_h^{(k+1)} \right\rangle + \left\langle \underline{\boldsymbol{\sigma}}\cdot\widehat{\mathbf{n}}, \nabla\phi_h^{(k+1)} \right\rangle_{\partial\Omega}, \quad \forall\underline{\boldsymbol{\sigma}} \in \Sigma_h. \tag{6.11}$$

In this thesis, we choose both Ψ_h and Σ_h to be $\mathbb{P}1$ spaces of appropriate dimension.

For steady-state problems, $\underline{\boldsymbol{H}}_k^{(0)}$ in (6.10) is initialised to zero, as well as for the first timestep of a time-dependent problem. This resembles the implicit assumption that the physical and computational meshes initially coincide.

In the time-dependent case, the algorithm is simply applied at whichever timesteps we desire the mesh to be moved. The choice of initial guess is important; if the same initial guess is used for every mesh movement timestep then the map (6.2) will be constructed from scratch every time. With a naïve initial guess that the Hessian is zero, the parabolised system (6.10)–(6.11) takes as many as 50 or 60 iterations to solve for the test case considered at the end of this chapter. This implies a prohibitively high cost if it is to be paid frequently. However, under the assumption that the mesh movement does not induce large changes from one timestep to the next, the value of ϕ generated by one mesh movement step provides an effective initial guess for the next and enables convergence in one or two iterations. This assumption is valid, provided the timestep is sufficiently small and the mesh is moved sufficiently frequently.

6.3.4. Monitor Function Choice. Mesh movement algorithms driven by solutions of Monge-Ampère type equations such as (6.8) are examples of *monitor-based* mesh adaptation, in the same way that the approach described in Chapter 5 is metric-based. The monitor function plays a crucial role and hence it is of the utmost importance that it is chosen appropriately.

Suppose there exists some scalar field of interest u which we would like to accurately represent on a mesh. This could be a solution field from a PDE, for example. Intuitively, it makes sense to deploy mesh resolution where u is changing significantly. As such, one obvious candidate monitor function involves the point-wise ℓ^2 norm of the gradient:

$$m_\alpha^{\text{gradient}} := 1 + \alpha\frac{\sqrt{\nabla u\cdot\nabla u}}{\|\sqrt{\nabla u\cdot\nabla u}\|_{L^\infty(\Omega)}}, \qquad \alpha \geq 0. \tag{6.12}$$

In practice, ∇u is approximated using a recovery technique when computing (6.12). Note that the gradient contribution to the monitor function is normalised in the L^∞ norm. The positive constant – here unity – is not important. What is important is the ratio of the other term to this constant. Under r-adaptation, the mesh size is fixed, meaning an overall scaling factor is redundant. If $\alpha = 0$ then the monitor function is constant, meaning uniform element volume is prescribed. Otherwise, it determines the extent to which high magnitude gradients are to be resolved.

As well as slopes of u, curvature often plays an important role. The monitor function

$$m_\alpha^{\text{Hessian}} := 1 + \alpha\frac{\sqrt{\underline{\boldsymbol{H}}(u) : \underline{\boldsymbol{H}}(u)}}{\|\sqrt{\underline{\boldsymbol{H}}(u) : \underline{\boldsymbol{H}}(u)}\|_{L^\infty(\Omega)}}, \qquad \alpha \geq 0, \tag{6.13}$$

focuses on such information, involving the point-wise Frobenius norm of the Hessian. Accounting for regions where the curvature of u is significant enables the accurate capture of transitions between regions where u is sloped and where it is relatively unchanging. Again, a recovery technique is required to approximate the Hessian.

A natural way to incorporate information on sizes due to a Riemannian metric into a scalar monitor function is to use the (metric) density of the absolute Hessian, $|\underline{\boldsymbol{H}}|$, (as defined in (5.5)) in place of the Frobenius norm. Doing so provides a neat connection between the metric-based and monitor-based frameworks.

The above monitor functions may also be combined. For example, we can take the point-wise maximum. Doing so is analogous to metric intersection:

$$m_1 \cap m_2 := \max\{m_1, m_2\}. \tag{6.14}$$

Similarly, we have an analogue of the metric average. The property that intersected metrics have higher complexity than their constituents carries over, in the sense that $m_1 \cap m_2 \geq m_1$ and $m_1 \cap m_2 \geq m_2$. These inequalities should be understood in the local sense, since global scaling has no effect.

Monitor function smoothness is important for the convergence of the mesh movement algorithm. If the monitor is strongly focused on a sharp interface, for example, an infeasibly large number of mesh elements may be required along the interface in order to satisfy the equidistribution relation. If the discretised Monge-Ampère equation becomes too ill-conditioned then r-adaptation cannot advance. In the experiment considered in Subsection 6.5, we found that applying monitor functions of the form (6.12) or (6.13) led to exactly this. In order to avoid such issues, it can be useful to apply diffusive filters to ensure smoothness of the monitor function, with the caveat that excessive diffusion will presumably mean a less desirable mesh. In subsequent experiments, the latter monitor function is smoothed by interpreting the Frobenius norm in (6.13) in the element-wise sense and then projecting this $\mathbb{P}0$ field into $\mathbb{P}1$ space. Galerkin projection is a diffusive operator (recall Subsection 2.8.3) and this is an example in which numerical diffusion is beneficial, similarly to SU(PG) stabilisation methods. However, there is little control available in such an approach and a simpler, more flexible approach would be to apply Laplacian smoothing, as described in Section 4.2 of [**Clare et al., 2020**].

6.4. Numerical Experiments

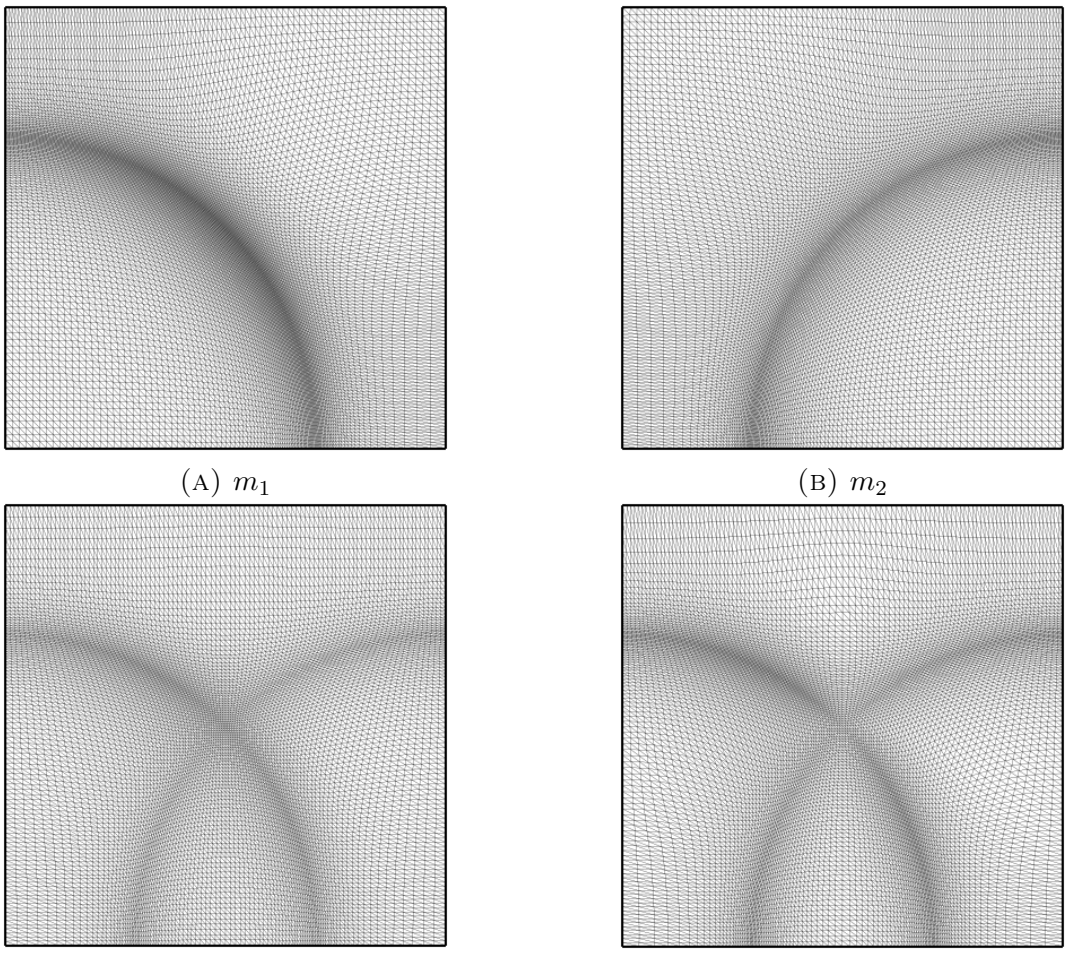
In this section we test the monitor function construction strategies and operations introduced in this chapter, as is done for the metric-based framework in Section 5.8. It is verified that monitor functions based on the Hessian of a quadratic function yield uniform meshes. In addition, comparisons are made between combining information due to two monitor functions by either averaging or intersection.

6.4.1. Recovery. Recall the experiment in Subsection 5.8.1, which demonstrates that a Hessian metric for a quadratic sensor is isotropic. The same can be achieved using a monitor function based on the Hessian Frobenius norm or metric density. Starting with the same uniform mesh as in the previous experiment, the mesh movement algorithms built on these monitor functions converge immediately, since they are unity and this is equidistributed by a uniform mesh.

6.4.2. Intersection vs. Averaging. Recall the comparison of metric intersection and averaging in Subsection 5.8.3. We now make the same comparison for monitor-based methods. Similar to the sensors defined in (5.47), define monitor functions

$$
\begin{aligned}
m_1(x,y) &:= 1 + \alpha \exp\left(-\beta \left|0.5 - x^2 - y^2\right|\right), \\
m_2(x,y) &:= 1 + \alpha \exp\left(-\beta \left|0.5 - (1-x)^2 - y^2\right|\right),
\end{aligned} \tag{6.15}
$$

with $\alpha, \beta > 0$. The α parameter controls the prominence of the arc, whereas β controls its width. The cases of very large α or very small β are ones in which non-smoothness issues can arise. We choose $\alpha = \beta = 10$ in order to have narrow, prominent arcs.



(A) m_1 (B) m_2

(C) $\frac{1}{2}(m_1 + m_2)$ (D) $m_1 \cap m_2$

FIGURE 6.6. Meshes adapted to two different analytically prescribed monitor functions (6.15), as well as the average and intersection thereof.

Solving the Monge-Ampère problem to relative tolerance `tol` $= 10^{-8}$ on a 100×100 uniform square mesh, with each square subdivided into triangles, yields the meshes shown in Subfigures 6.6a and 6.6b. The arc is more prominent in Subfigure 6.6a, because the upper-left to lower-right diagonals align with the curve of the arc.

As in the combination experiment for metric-based methods, the overall resolution of the arcs is higher for intersection than for averaging, as expected. This is particularly evident for the arc due to m_1, for which the element diagonals are aligned. Consequently, the mesh

due to monitor intersection is coarser and more distorted away from the high resolution zones. Averaging, on the other hand, leads to smoothing of the element size transition.

6.5. Migrating Trench Test Case

This section is based upon experimental results presented in [Clare et al., 2020]. *The same test case setup is used. However, the experiments have been rerun with some modifications. These include sampling element count so as to emulate uniform refinement and an additional experiment on the impact of solver tolerance upon accuracy. All modified experiments were conducted by the author of this thesis.*

The numerical experiments conducted in this section use the equal order $\mathbb{P}1_{DG} - \mathbb{P}1_{DG}$ finite element pair for the shallow water equations. We refer to [Clare et al., 2021] for details on the hydro-morphodynamic model and the discretisation thereof.

The 'Migrating Trench' test case was presented in Section 4.1 of [Clare et al., 2020]. Given the spatial domain $\Omega = [0, 16] \times [0, 1.1]\,\mathrm{m}^2$, the erosion of a bathymetry trench with initially sharp transitions in slope is modelled under constant inflow and outflow conditions. Both suspended and bedload transport are simulated for 15 hours, after which time the resulting bathymetry profile is compared against experimental data due to [Van Rijn, 1980].

The initial trench is shown in Figure 6.7. Note the discontinuous gradients at the transitions between flat and sloped sections.

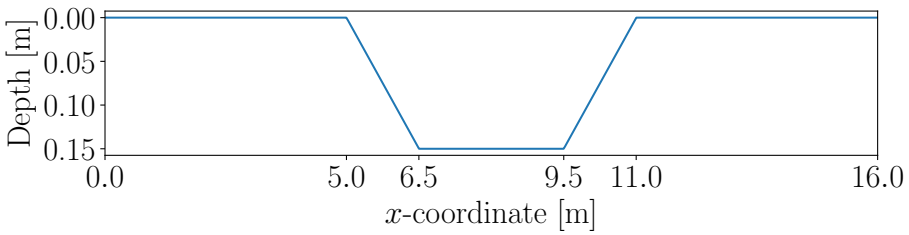


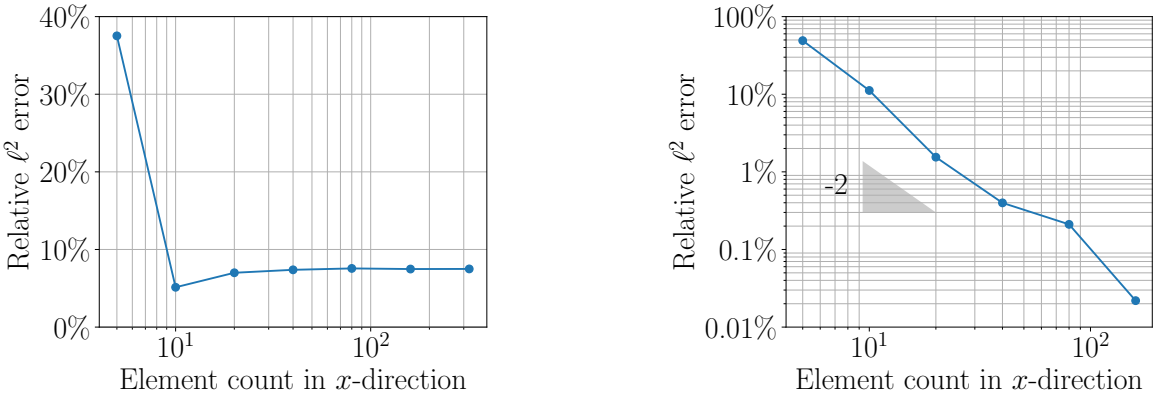
FIGURE 6.7. Trench profile for the 'Migrating Trench' test case.

The problem is effectively 1D because the bathymetry only changes in the x-direction and the boundary conditions are symmetric in the y-direction. However, we adopt a 2D domain for two reasons: (a) for consistency with the experimental data in [Van Rijn, 1980], which were obtained in a $16\,\mathrm{m} \times 1.1\,\mathrm{m}$ tank, meaning the dynamics were not fully 1D; and (b) to allow for a straightforward extension to the test case with variation in the y-direction presented in Section 4.2 of [Clare et al., 2020].

The test case was also used in [Clare et al., 2021] for hydro-morphodynamic model validation on a fixed uniform mesh; see that work for details on the problem specification and calibration of the diffusion coefficient. For the purposes of computational efficiency, a morphological scale factor of 100 is used in order to artificially accelerate the dynamics one hundred-fold. For consistency, the same timestep $\Delta t = 0.25\,\mathrm{s}$ is used for all experiments, unless otherwise stated. Implicit Euler time integration is applied.

We assess the approximation quality of mesh adaptive simulations using a discrete QoI obtained as the ℓ^2 error between the solution bathymetry field b and the experimental data b^{obs} at the terminal time. Quality is assessed when varying a number of different parameters. The ℓ^2 error is evaluated at the same 31 data points along a cross-section in the x-direction as sampled in [**Van Rijn, 1980**].

Overall Mesh Size. Figure 6.8 illustrates the convergence of the QoI under uniform mesh refinement. The (total) ℓ^2 error presented in Subfigure 6.8a does not converge to zero. This is because model error is at play, as well as discretisation error (recall (3.60)). This is acceptable because discretisation error is what we ultimately care about in the mesh adaptation context. Subfigure 6.8b shows the corresponding plot for discretisation error alone. A convergence rate close to second order is observed.



(A) Total error (B) Discretisation error

FIGURE 6.8. QoI convergence of the 'Migrating Trench' test case on uniform meshes. Modified based on plots presented in [**Clare et al., 2020**].

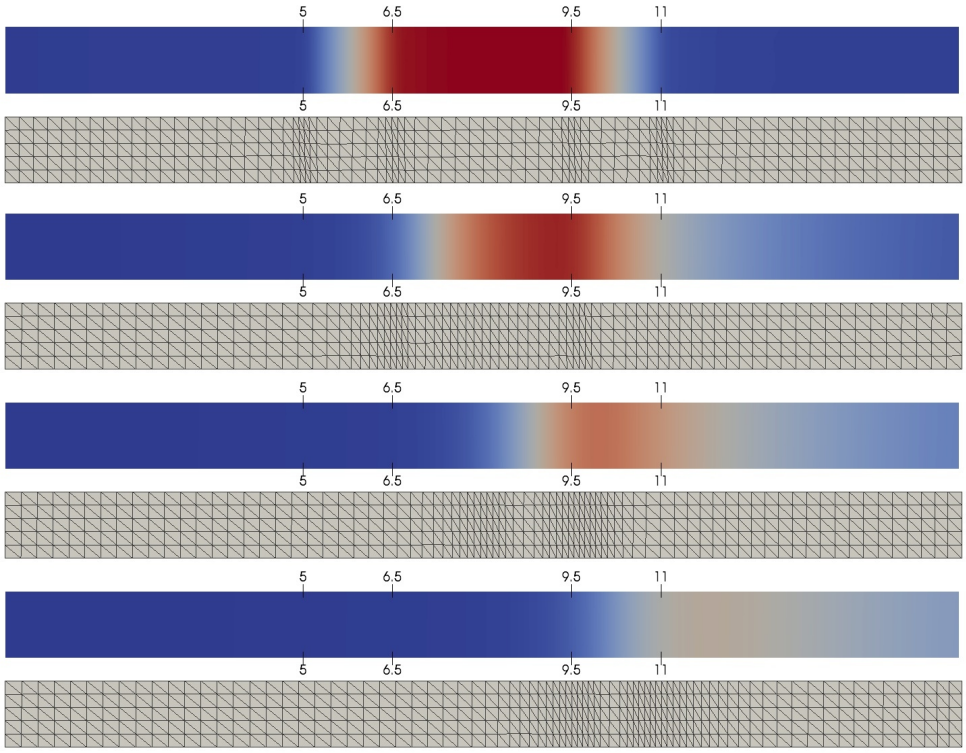
In Subfigure 6.8b, observe that, with 40 elements in the x-direction (i.e. $N_x = 40$), discretisation error has been reduced one hundred-fold. As such, in the following calibration experiments for the mesh movement algorithm, we do not attempt to tune the parameters for $N_x > 40$, since the accuracy is already very good. Recall that the x-extent is more than 14 times the y-extent. Five elements are used in the y-direction across all cases considered; only the resolution in the x-direction changes.

Figure 6.9 shows snapshots of the mesh movement applied to a mesh with $N_x = 40$, in the case $\alpha = 2$. It also shows the corresponding bathymetry field. Initially, the bathymetry profile is flat in the regions $[0, 5)$, $(6.5, 9.5)$ and $(11, 16]$ and uniformly sloped in the regions $(5, 6.5)$ and $(9.5, 11)$, as shown in the cross section in Figure 6.7. Accordingly, the mesh is fairly uniform for most of both flat and sloped regions. Because the monitor function is based on the Hessian and the sloped regions have a constant gradient, the initial mesh is only refined around the transitions between flat and sloped sections, at $\{5, 6.5, 9.5, 11\}$, where the *recovered* curvature is greatest.[1]

As the simulation progresses, bedload transport acts to fill in the trench. Through processes of erosion and deposition, it effectively becomes shallower and moves down the

[1]The curvature is technically ill-defined at the transitions, due to discontinuous gradients. However, the recovered gradient field applies a continuous approximation, which is rapidly changing at the transitions.

channel. Indeed, by the final time, the trench starts where it previously ended, as shown by comparing Subfigures 6.9a and 6.9d. As the bathymetry flattens, curvature is reduced and the adapted meshes become near uniform over larger proportions of the domain.



(A) $t = 0\,\mathrm{h}$

(B) $t = 5\,\mathrm{h}$

(C) $t = 10\,\mathrm{h}$

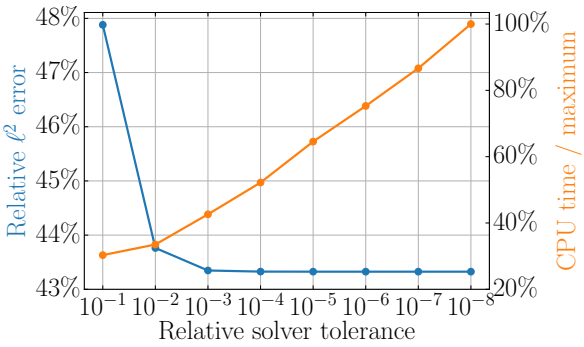
(D) $t = 15\,\mathrm{h}$

FIGURE 6.9. Snapshots of a moving mesh for the 'Migrating Trench' test case with $\alpha = 2$. The underlying mesh has 40 elements in the x-direction and 5 in the y-direction. Modified based on plots presented in [**Clare et al., 2020**].

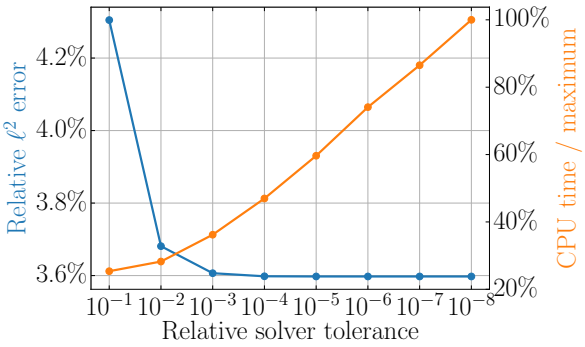
A monitor function based on the Hessian deploys mesh resolution most densely wherever there is high curvature, meaning that the trench transitions should be well captured. It is important to accurately resolve these transitions because they are the most notable bed erosion features in this test case. In [**Clare et al., 2020**], we used $m_\alpha^{\text{Hessian}}$, although we could also have used the Hessian metric density in place of the Frobenius norm. As mentioned in Section 6.3.4, the Frobenius norm was re-interpreted in the element-wise sense, to give a smoothing effect.

Monge-Ampère Solver Tolerance. Frequent mesh movement leads to heightened computational cost, as shown later in this section. One reason for this is that if the Monge-Ampère equation is solved to a high tolerance, then it can take many iterations for the relaxation to converge. The tolerance here refers to a relative tolerance on the residual of (6.10), which determines to what extent equilibrium has been reached. In Section 6.3, it was suggested that it is often sufficient to only solve this equation to a low relative tolerance, `tol`, such as $\texttt{tol} = 10^{-4}$, or even $\texttt{tol} = 10^{-3}$. Figure 6.10 shows the accuracy vs. computational cost trade-off due to solver tolerance, to see how large a tolerance may be taken. Relative discretisation error against the converged high resolution run is computed in the case $\alpha = 2$.

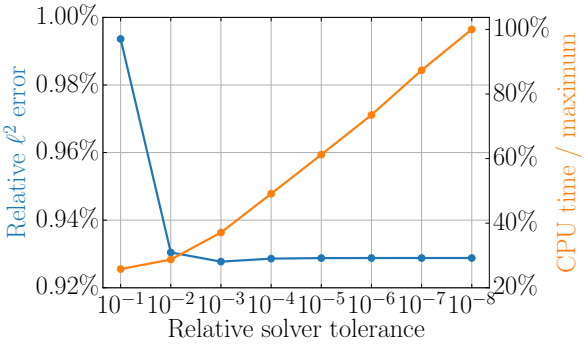
As the tolerance is tightened, ℓ^2 error converges to 43.33%, 3.598% and 0.9288% (to 4 significant figures) for the cases $N_x = 5$, $N_x = 10$ and $N_x = 20$, respectively. In the



(A) $N_x = 5$



(B) $N_x = 10$



(C) $N_x = 20$

FIGURE 6.10. Trade-off between discretisation error and CPU time with Monge-Ampère solver tolerance for the 'Migrating Trench' test case. Here there are 40 timesteps per mesh movement and $\alpha = 2$.

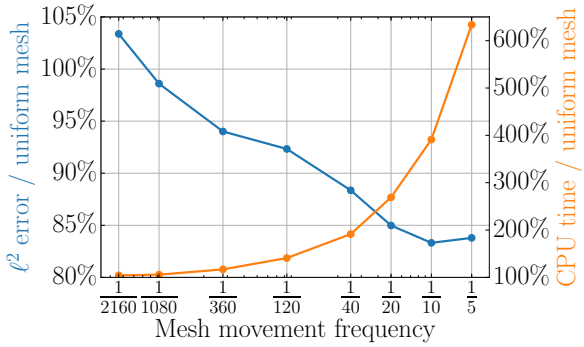
first case, convergence is attained at $\texttt{tol} = 10^{-3}$, which comes with less than half of the computation time required at $\texttt{tol} = 10^{-8}$. In the second and third cases, convergence is attained at $\texttt{tol} = 10^{-4}$, with around half the computation time than $\texttt{tol} = 10^{-8}$. Whilst the error has not fully converged at $\texttt{tol} = 10^{-3}$, it could be claimed that it is sufficiently close for practical purposes. Similar patterns were observed for other values of N_x.

Whilst a tight tolerance comes with a high computational cost, it does not necessarily yield a significant improvement in accuracy. the largest improvement displayed in Figure 6.10 is that due to tightening the tolerance from 10^{-1} to 10^{-3} in Subfigure 6.10a and even that is only an improvement of 5%. As such, we argue that $\texttt{tol} = 10^{-3}$ provides a good compromise between accuracy and computational efficiency over a range of cases, cutting the CPU time in half compared with $\texttt{tol} = 10^{-8}$, at least. It is used henceforth.

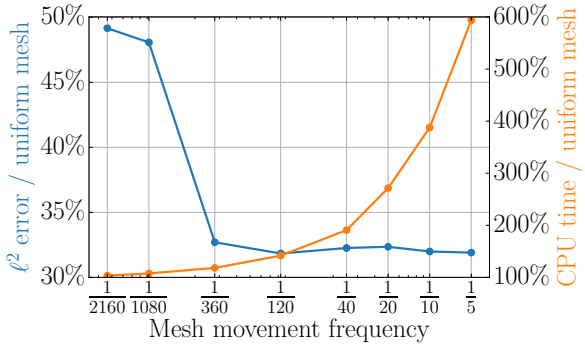
Mesh Movement Frequency. Another trade-off between accuracy and computational cost is that due to the mesh movement frequency. Figure 6.11 shows this for three choices of overall mesh size in the case $\alpha = 2$. As mesh movement frequency is increased, computation time increases monotonically, as might be expected. The relationship between frequency and ℓ^2 error is not as simple and is different for each mesh size considered.

In all three cases, we observe that adapting the mesh every five timesteps is prohibitively expensive, requiring as much as six times the fixed mesh simulation time. Accordingly, the discretisation error is reduced by at least 15%, 65% and 40% in the cases $N_x = 5$, $N_x = 10$ and $N_x = 20$, respectively.

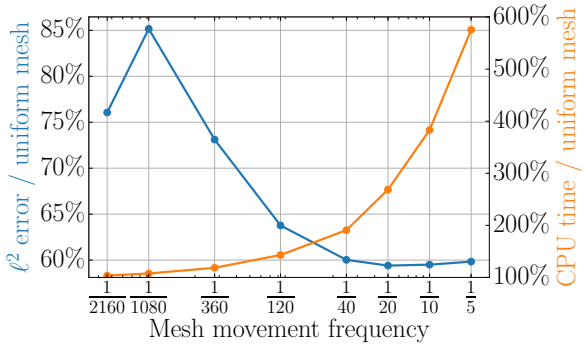
Moving the coarsest mesh often yields reduced error, although not much. This is because mesh movement goes some way to improve the suitability of the discretisation for the problem at hand, but there are insufficient DoFs to yield a highly accurate solution. The



(A) $N_x = 5$



(B) $N_x = 10$



(C) $N_x = 20$

FIGURE 6.11. Trade-off between discretisation error and CPU time with mesh movement frequency for the 'Migrating Trench' test case with $\alpha = 2$. Errors and times are shown as a percentage relative to the fixed mesh case. Modified based on plots presented in [**Clare et al., 2020**].

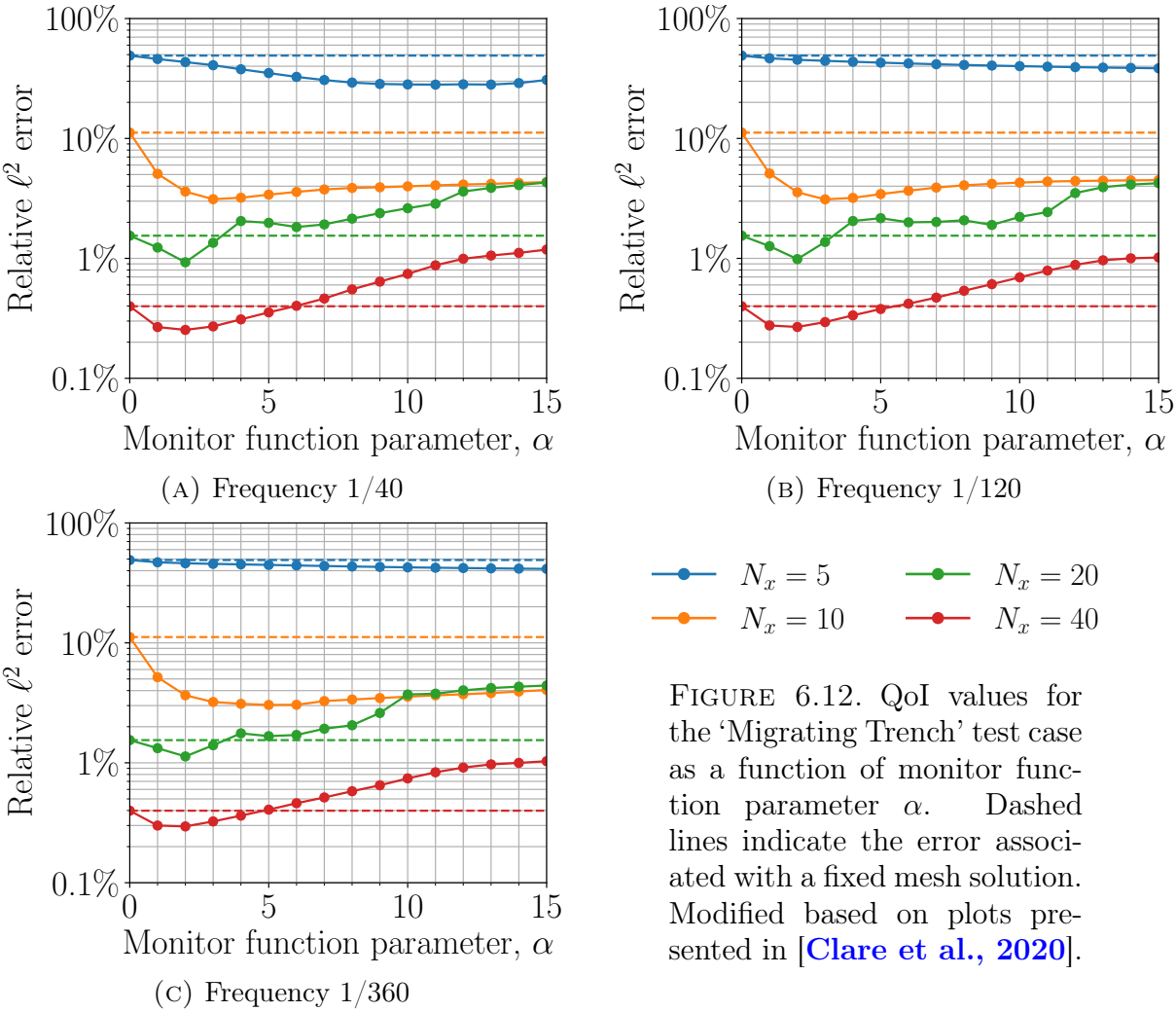
opposite problem appears on the finest mesh, where the accuracy is already quite good, due to there being more DoFs. The case $N_x = 10$ provides an example where mesh movement is very effective, with just a small number of applications. For frequencies 1/360 or larger, the error effectively plateaus, indicating that more frequent adaptation is not worth the additional cost. In this particular case, frequency 1/360 is very attractive because it has a comparable cost to the fixed mesh simulation, yet reduces the error by almost 70%.

Interestingly, adapting the $N_x = 20$ mesh once before the simulation begins yields a better error reduction than doing the same in addition to adapting halfway through. A similar pattern is observed for $N_x = 40$. This is likely due to the fact that monitor function $m_\alpha^{\text{Hessian}}$ is not normalised in time, i.e. it is only really valid for the time level at which it was evaluated. Whilst adapting the mesh one more time improves its suitability at that particular timestep, it does not guarantee suitability for subsequent timesteps. Similarly, adapting just once at $N_x = 5$ leads to a lower accuracy than using a uniform mesh.

For the mesh sizes shown, there is no obvious mesh movement frequency which simultaneously ensures significantly reduced error and comparable computational cost, across all resolutions. In the case $N_x = 5$, it is a fairly direct trade-off between cost and accuracy. In the case $N_x = 10$, adapting every 360 timesteps provides an excellent compromise. In the case $N_x = 20$, it is costly to attain error reductions of 40% or more (requiring twice the CPU time of a fixed mesh simulation), so a frequency 1/120 is perhaps suitable. For this particular setup, it appears that mesh movement is more effective on meshes of small overall size, but not so small that there are insufficient DoFs to resolve the trench profile. An important takeaway is that the application of mesh movement always leads to a reduction in discretisation error, provided it is applied again after the initial remeshing.

Monitor Function. Recall that the parameter α in (6.13) determines the extent to which high curvature solution data are to be resolved. It is not obvious what an effective parameter choice would be for this test case. One approach would be to apply a gradient-based optimisation technique. However, the implementation of this was beyond the scope of [**Clare et al., 2020**], because the resulting gradient computation requires differentiation through the mesh adaptation routine. This implies the differentiation of mesh-to-mesh interpolation, which was not currently supported in Firedrake when the associated experiments were conducted. We propose this as future work. Instead, a range of parameter values were sampled in order to deduce an effective choice.

Figure 6.12 shows discretisation errors for $\alpha \in [0, 15]$, for a range of mesh resolution levels and mesh movement frequencies. At a glance, the three plots, which consider mesh movement frequencies 1/40, 1/120 and 1/360, have broadly similar features.



(A) Frequency 1/40

(B) Frequency 1/120

(C) Frequency 1/360

FIGURE 6.12. QoI values for the 'Migrating Trench' test case as a function of monitor function parameter α. Dashed lines indicate the error associated with a fixed mesh solution. Modified based on plots presented in [**Clare et al., 2020**].

Differences are more striking when comparing across mesh resolutions. At $N_x = 5$, small improvements in accuracy are attained for larger values of α. At $N_x = 10$, taking $\alpha \geq 2$ implies a significant reduction in error, although smaller reductions are achieved for larger values. At $N_x = 20$ and $N_x = 40$, modest reductions can be made for $\alpha \approx 2$, but values larger than this are less effective. Worse still, taking $\alpha \geq 4$ can actually lead to lower accuracy than holding the mesh fixed ($\alpha = 0$). Eventually the Monge-Ampère solver fails to converge for $\alpha > 15$. As observed previously, this is because the monitor function becomes too focused on the most curved regions and becomes insufficiently smooth.

It appears that the 'optimal' choice of α is more dependent on DoF count than on mesh movement frequency and that, the more DoFs there are, the smaller α should be taken. A choice $\alpha \approx 2$ seems most appropriate for $N_x = 20$ and $N_x = 40$, whilst $\alpha \approx 4$ is suitable for $N_x = 10$. In the case $N_x = 5$, taking $\alpha \in [10, 15]$ appears best, although the resulting mesh movement algorithm does not reduce error significantly overall, because the overall DoF count is too low to achieve this.

As discussed above, [**Clare et al., 2020**] contains an extension of this effectively 1D test case so that the initial bathymetry varies in the y-direction. The same experiments as above were performed, with similar results. The 2D results are omitted here for brevity.

Summary. As documented above, tuning the parameters associated with a mesh adaptation routine can be expensive in terms of both labour and computational cost. Even once effective parameter choices have been established for a particular problem, there is no guarantee that they will be of any use for others. If such studies are to be avoided, lengthy experience with mesh adaptation tools and an in-depth understanding of the problem at hand are essential in order to make effective parameter choices. Nonetheless, we did make some conclusions based on evidence as to what might be good parameter choices in the context of hydro-morphodynamic modelling in [**Clare et al., 2020**]. Indeed, it was established that some parameters are far more impactful than others.

The next and final chapter of this thesis discusses a mesh adaptation paradigm which goes some way to reduce the number of tunable parameters, making for a simpler user experience. Whilst remeshing frequency and other mesh adaptation parameters such as minimum tolerated element size remain important, the 'assessment' step is largely accounted for. This means that parameters such as α above are no longer of concern.

CHAPTER 7

Goal-Oriented Mesh Adaptation

Motivation. The first part of this thesis demonstrates how the computational geoscientist is able to obtain useful information on the propagation of sensitivities using adjoint methods, assuming the existence of some *quantity of interest (QoI)*. A broad range of possible uses and applications are discussed, some of which are considered in detail. The second part highlights mesh adaptation methods as providing means for controlling the distribution of DoFs in both space and time, with the potential to not only improve accuracy, but also reduce the overall computational cost of a simulation. This, the final chapter, is where the two parts come together most clearly. It focuses on a particular class of mesh adaptation methods which uses the adjoint method to drive mesh adaptation routines so that PDE solutions obtained on the resulting meshes admit accurate approximations to the QoI.

Goal-oriented mesh adaptation can be viewed in the context of optimisation, except that, whilst the gradient-based optimisation routines considered in Chapters 3 and 4 have the aim of minimising the QoI itself (since it represents e.g. a solution misfit against observations, which we seek to reduce), the aim of goal-oriented mesh adaptation is to minimise *the error incurred during its evaluation*. There are, however, some other major differences between gradient-based optimisation and goal-oriented mesh adaptation. Firstly, the mesh adaptation process is not differentiable in general. This means that, it is not usually possible to obtain formulae for gradients of the QoI error w.r.t. parameters defining the adaptation scheme. However, given that goal-oriented mesh adaptation seeks to minimise *QoI error* and the true QoI is clearly unknown, it is not obvious how a gradient-based method would be used, even in cases where the adaptation is differentiable. That is, it is not possible to evaluate the quantity which we seek to minimise, except in cases where analytical solutions are available. Moreover, where previously considered routines perturb control parameters which appear – either explicitly or implicitly – in the statement of the continuous form problem, here the control parameters are intrinsically linked to the *discretised* problem and do not appear at all in the continuous version.

Instead of using gradient information, goal-oriented mesh adaptation methods are typically derived from error estimates for the QoI error. The most commonly deployed class of goal-oriented error estimates are those of *dual weighted residual (DWR)* type, pioneered in the works of [**Becker and Rannacher, 1996**] and [**Becker and Rannacher, 2001**]. In the fundamental DWR error estimator, the QoI error is approximated by the weak residual of the forward problem, evaluated with the adjoint solution in place of a test function. The adjoint error is clearly unknown, so the evaluation of goal-oriented error estimates requires a strategy for reliably estimating the error, or else application of further error results in order to reformulate the estimator. Alternative goal-oriented error estimates exist, such as the one derived in [**Loseille et al., 2010**].

Of course, minimising QoI error is not the only goal; mesh adaptation should also be applied in such a way that excessive resolution is avoided where it is not required, enabling

a more efficient simulation. In this thesis, we opt to construct goal-oriented mesh adaptation methods using the Riemannian metric framework. The main motivation for this is the ability to incorporate anisotropy. Used properly, mesh anisotropy avoids excessive resolution in directions for which flow features are relatively unchanging, whilst allowing sufficient resolution in cross-stream directions, where there is greater variation. With the inclusion of adjoint information, relatively coarse mesh resolution can be deployed in large portions of the domain, where the QoI is deemed to be insensitive to the resulting dynamics.

Novel Contributions. Goal-oriented error estimates for the shallow water equations are detailed and used to drive both isotropic and anisotropic goal-oriented mesh adaptation methods for an idealised steady-state tidal turbine farm test case. Enhanced QoI convergence properties compared with uniform meshing are demonstrated. The anisotropic method performs particularly well for this advection-dominated problem.

Tracer transport problems are also considered. For an established steady-state test case with an analytical solution, various aspects of goal-oriented mesh adaptation methods are compared, including metric formulation, interpretation of the adjoint error, type of error estimate and whether the continuous or discrete adjoint method is used. Resulting meshes are also compared, both qualitatively and quantitatively, in order to glean better understanding of how the goal-oriented metrics distribute DoFs. One of the metrics is based on an alternative to the DWR error estimate, derived in [**Loseille et al., 2010**]. The corresponding adjoint error estimate is derived for linear PDEs for the first time in this thesis; the combination of the resulting metrics from the forward and adjoint estimators is shown to have excellent convergence properties for the tracer transport test case considered.

The extension to three dimensions is considered, with one resulting mesh demonstrably multi-scale. The extension to time-dependent problems is also considered, in the context of an idealised desalination plant outfall scenario. This setup, presented in [**Wallwork et al., 2021**], is the first application of goal-oriented methods to this particular application area. Finally, goal-oriented mesh adaptation is applied to a tsunami hazard assessment problem in a realistic domain. Although adjoint-based mesh adaptation has previously been used in [**Davis and LeVeque, 2016**], this presents the first use of truly goal-oriented methods to solve this type of problem. That is, a dual weighted residual error indicator is deployed, as opposed to a simple weighting of the forward solution with the adjoint solution.

Chapter Organisation. This chapter is organised as follows. The mathematical background for DWR type error estimation is provided in Section 7.1 and applied to the tracer transport and hydrodynamics models considered in this thesis in Section 7.2. Section 7.3 compares a number of strategies for evaluating such error estimates, both in terms of effectivity and computational efficiency. Five methods for constructing Riemannian metrics from goal-oriented error estimates are discussed in Section 7.4, two of which give rise to isotropic meshes and three of which give rise to anisotropic meshes. Whilst based on metric construction strategies from the literature, a number of novel modifications due to the author of this thesis are proposed and indicated. Mesh adaptation algorithms built upon goal-oriented metrics are presented for the steady-state and time-dependent cases in Sections 7.5 and 7.6. Section 7.5 includes steady-state tracer transport and shallow water test cases which act to validate the proposed goal-oriented mesh adaptation strategy and compare different aspects thereof. A non-depth-averaged tracer transport problem

is considered to demonstrate the extension of the goal-oriented framework to the three dimensional case. Finally, goal-oriented mesh adaptation is applied to an idealised desalination plant outfall scenario and a tsunami hazard assessment problem in a realistic domain in Sections 7.7 and 7.8, respectively.

7.1. The Dual Weighted Residual

For the purposes of this section, consider a PDE which may be written in variational formulation as

$$a(w; v) = L(v), \qquad \forall v \in V, \tag{7.1}$$

where $a : W \times V \to \mathbb{R}$ is linear in the second argument, $L : V \to \mathbb{R}$ and W and V are function spaces. We assume (7.1) to have a unique solution $w \in W$. In addition, consider a finite element approximation,

$$a(w_h; v) = L(v), \qquad \forall v \in V_h, \tag{7.2}$$

with solution $w_h \in W_h$, for finite-dimensional subspaces $W_h \subset W$ and $V_h \subset V$.

In addition, consider a QoI, $J : W \to \mathbb{R}$ and the corresponding adjoint problem,

$$a^*(w^*, w) = J^*(w), \qquad \forall w \in W, \tag{7.3}$$

with $a^* : V \times W \to \mathbb{R}$ and $J^* : W \to \mathbb{R}$ and for which we assume a unique solution $w^* \in V$. In the linear case, a and a^* are bilinear forms and L and J are linear forms. Moreover, we drop the semicolon notation and have that $a^*(v, \cdot) = a(\cdot, v)$ for any $v \in V$ and $J^*(w) = J(w)$ for any $w \in W$. Otherwise, a^* and/or J^* are obtained by linearisation and taking the transpose. The finite element approximation to (7.3) is given by

$$a^*(w_h^*, w) = J^*(w), \qquad \forall w \in W_h \tag{7.4}$$

and has solution $w_h^* \in V_h$.

Weak residuals of the forward and adjoint problem are denoted,

$$\rho(\cdot; v) := L(v) - a(\cdot; v) \quad \text{and} \quad \rho^*(\cdot, w) := J^*(w) - a^*(\cdot, w), \qquad v \in V, \quad w \in W. \tag{7.5}$$

Note that the LHS of the adjoint equation is the negative transposed linearised forward residual:

$$a^*(v, \cdot) = -\partial_w \rho(\cdot, v, w), \qquad \forall v \in V. \tag{7.6}$$

For now, suppose that both PDE and QoI are linear. As before, the error in the forward solution is denoted $e := w - w_h$. Similarly, $e^* := w^* - w_h^*$ is the error in the adjoint solution. Exploiting identities (7.1)–(7.4), linearity of the QoI and RHS and Galerkin orthogonality, we have the identity [**Becker and Rannacher, 2001**]

$$J(w) - J(w_h) = J(e) = a(e, w^*) = a(e, e^*) = a(w, e^*) = L(e^*) = L(w^*) - L(w_h^*). \tag{7.7}$$

Applying Galerkin orthogonality again, we find that

$$J(w) - J(w_h) = \rho(w_h, w^* - \phi_h) = a(e, e^*) = \rho^*(w_h^*, w - \psi_h), \quad \forall \phi_h, \psi_h \in V_h. \tag{7.8}$$

In particular, (7.8) holds in the cases $(\phi_h, \psi_h) = (0, 0)$ and $(\phi_h, \psi_h) = (w_h^*, w_h)$.

Returning to the more general case where the PDE and QoI are nonlinear, remainder terms are introduced into (7.7).

Theorem 2. In the notation introduced above,

$$J(w) - J(w_h) = \rho(w_h; w^* - w_h^*) + R^{(2)}, \tag{7.9}$$

$$J(w) - J(w_h) = \rho^*(w_h^*, w - w_h) + R^{(2)}, \tag{7.10}$$

$$J(w) - J(w_h) = \frac{1}{2}\rho(w_h; w^* - w_h^*) + \frac{1}{2}\rho^*(w_h^*, w - w_h) + R^{(3)}, \tag{7.11}$$

where $R^{(2)}$ and $R^{(3)}$ are quadratic and cubic remainders[1] in the primal and adjoint errors.

Proof. See Propositions 2.2, 2.3 and 2.4 and their proofs on pp.10–12 of [**Becker and Rannacher, 2001**]. □

Error results (7.9)–(7.11) lie at the heart of much of the literature on goal-oriented error estimation and mesh adaptation. The formulations shown here (with the choice $(\phi_h, \psi_h) = (w_h^*, w_h)$) are useful because they are amenable to further error estimation including bounds on norms of e or e^*, such as Céa's lemma.

The expression $\rho(w_h, e^*)$ on the RHS of (7.9) is what we refer to as the *dual weighted residual (DWR)*. It combines a measure of approximation quality for the forward problem (the residual) with a measure of approximation quality for the adjoint problem (the approximation error).

Since its remainder term is quadratic, the weak residual in (7.9) is termed the *first order forward* DWR error estimator. Similarly, that in (7.10) is a first order DWR estimator; since that residual is for the adjoint equation, we refer to it as the *first order adjoint* DWR estimator. Finally, observe from (7.11) that the average of the first order dual weighted residuals is a *second order* DWR estimator, in the sense that the remainder is cubic.

In the case where both PDE and QoI are linear, the remainders in (7.9)–(7.10) are zero, meaning that the DWR estimators are *exact*, as demonstrated in (7.8). Exactness may be maintained in (7.11) for a quadratic QoI. If forward and adjoint errors are sufficiently small then the remainder terms are also, in which case the DWR estimators provide accurate approximations to the QoI error for nonlinear problems, as well as linear ones. Note that the above presentation has not accounted for stabilisation terms; for the treatment of such terms, we refer to pp.15–17 of [**Becker and Rannacher, 2001**].

Given the finite element context, the weak residuals may most naturally be decomposed as sums over the contributions from each mesh element. That is,

$$\rho(w_h, e^*) = \sum_{K\in\mathcal{H}} \rho(w_h, e^*)|_K, \tag{7.12}$$

$$\rho^*(w_h^*, e) = \sum_{K\in\mathcal{H}} \rho^*(w_h^*, e)|_K. \tag{7.13}$$

These spatial decompositions are useful for mesh adaptation, where local error contributions are much more important than global quantities. By considering local contributions

[1]Note that the remainders differ in (7.9) and (7.10), despite the (standard) notation.

on each element, we are able to establish regions of the domain where the application of mesh adaptation will more likely improve our estimate of the QoI. The local DWR indicator $\rho_K(w_h, e^*)$ will take large values wherever there is significant approximation error for the forward problem *and* the error in the adjoint solution is large.

It should be noted that it is only possible to assemble DWR estimators exactly in test cases for which the true forward and/or adjoint solutions are known. This is certainly not the case for practical applications, so the direct error terms should be approximated somehow. A variety of methods for achieving this are discussed in Section 7.3.

7.2. Goal-Oriented Error Estimation

In most cases, integration by parts is applied when defining the weak residual. This might be to reduce the order of derivatives or to apply Neumann boundary conditions, for instance. When evaluating element-based error indicators, it is often beneficial to perform an additional integration by parts, *on each element*. Doing so will return the tested strong residual, in addition to boundary contributions and inter-element flux terms. How the error indicators come out becomes clearer when considering the examples in Subsections 7.2.1 and 7.2.2. However, their general form for the first order forward case is

$$\rho_K(w_h, e^*) = \langle \Psi(w_h), e^* \rangle_K + \langle \psi^\partial(w_h), e^* \rangle_{\partial K \cap \partial\Omega} + \langle \psi^{\text{flux}}(w_h), e^* \rangle_{\partial K \setminus \partial\Omega}\,, \tag{7.14}$$

where Ψ is the strong residual for the PDE, ψ^∂ accounts for errors due to the weak enforcement of boundary conditions and ψ^{flux} accounts for flux terms introduced by the second integration by parts. Note that strongly enforced boundary conditions do not contribute errors and therefore ψ^∂ is zero on corresponding boundary segments. The expansion (7.14) is substituted into (7.12) in order to perform error estimation and mesh adaptation.

For stabilised finite element methods, additional terms are contributed to (7.14). Whilst the terms above are all obtained by testing by the adjoint error, the stabilisation term does not always take such a simple form, as becomes clear in Subsections 7.2.1 and 7.2.2.

In the remainder of this section, we use the theory from Section 7.1 to formulate goal-oriented error estimators for the two main equation sets considered in this thesis: tracer transport and shallow water modelling.

7.2.1. Tracer Transport.

This subsection contains goal-oriented error estimates for tracer transport problems which were published in [**Wallwork et al., 2020a**].

In this section, we present goal-oriented error estimators for a CG discretisation of tracer transport problem with SUPG stabilisation.

Using the notation of (3.19), the steady-state problem may be written as

$$\begin{cases} \mathbf{u}\cdot\nabla c - \nabla\cdot(\underline{\boldsymbol{D}}\nabla c) = S & \text{in } \Omega \\ c = g_D & \text{on } \partial\Omega_D \\ \underline{\boldsymbol{D}}\nabla c\cdot\widehat{\mathbf{n}} = g_N & \text{on } \partial\Omega_N \end{cases}, \tag{7.15}$$

with appropriate compatibility conditions on $\underline{\boldsymbol{D}}$. That is, we have an advection-diffusion problem with a forcing term, Dirichlet conditions on $\partial\Omega_D$ and Neumann conditions on $\partial\Omega_N$.

Recall the SUPG stabilised weak formulation for the forward problem, as described in Section 2.7. In order to formulate first order forward DWR error indicators for a finite element approximation c_h and adjoint error e^*, we apply integration by parts on an element-by-element basis, giving

$$\begin{aligned}\rho_h(c_h,e^*)|_K =& \langle S+\nabla\cdot(\underline{\boldsymbol{D}}\nabla c_h)-\mathbf{u}\cdot\nabla c_h, e^*\rangle_K \\ &+\langle\underline{\boldsymbol{D}}\nabla c_h\cdot\widehat{\mathbf{n}}-g_N, e^*\rangle_{\partial K\cap\partial\Omega_N} \\ &+\langle\underline{\boldsymbol{D}}\nabla c_h\cdot\widehat{\mathbf{n}}_K^+, e^*\rangle_{\partial K\setminus\partial\Omega} \\ &+\langle S+\nabla\cdot(\underline{\boldsymbol{D}}\nabla c_h)-\mathbf{u}\cdot\nabla c_h, \tau\mathbf{u}\cdot\nabla e^*\rangle_K\,.\end{aligned} \tag{7.16}$$

Unlike in the previous notation, the symbol + stands for the interior side of the facet and − stands for the exterior side. As such, $\widehat{\mathbf{n}}_K^+$ denotes the set of outward pointing normals and $\widehat{\mathbf{n}}_K^- = -\widehat{\mathbf{n}}_K^+$ the set of inward pointing normals.

The corresponding error indicator on element K is obtained by taking the modulus. The terms in (7.16) are decomposed as contributions from element interiors, boundary condition implementation, inter-element fluxes and stabilisation, respectively. As noted in [**Wallwork et al., 2020a**], this error indicator collapses to zero if c_h coincides with c. This desirable property follows from having a Petrov-Galerkin method. The first order forward DWR error estimator is obtained by summation over all mesh elements.

Similarly, applying integration by parts to the SUPG-stabilised continuous adjoint problem based on a QoI J yields

$$\begin{aligned}\rho_h^*(c_h^*,e)|_K =& \langle\partial J_c-\nabla\cdot(\underline{\boldsymbol{D}}\nabla c_h^*)+\nabla\cdot(\mathbf{u}c_h^*), e\rangle_K \\ &+\langle\underline{\boldsymbol{D}}^T\nabla c_h^*\cdot\widehat{\mathbf{n}}+c_h^*\mathbf{u}\cdot\widehat{\mathbf{n}}, e\rangle_{\partial K\cap\partial\Omega_D} \\ &+\langle\underline{\boldsymbol{D}}^T\nabla c_h^*\cdot\widehat{\mathbf{n}}_K^+ + c_h^*\mathbf{u}\cdot\widehat{\mathbf{n}}_K^+, e\rangle_{\partial K\setminus\partial\Omega} \\ &+\langle\partial J_c-\nabla\cdot(\underline{\boldsymbol{D}}\nabla c_h^*)+\nabla\cdot(\mathbf{u}c_h^*), \tau\nabla\cdot(\mathbf{u}e)\rangle_K\,,\end{aligned} \tag{7.17}$$

where c_h^* is a finite element approximation to the adjoint solution and e is the error in the forward solution. Again the error indicator on element K is obtained by taking the modulus. Terms of (7.17) are decomposed as in (7.16) and we again observe the property that the estimator collapses to zero at the exact adjoint solution. The first order adjoint DWR error estimator is obtained by summation over all mesh elements. Further, the second order DWR estimator and corresponding indicators are given by averaging the results from (7.16) and (7.17), summing or taking the modulus, as appropriate.

7.2.2. Shallow Water. *This section contains a goal-oriented error estimate for shallow water problems solved using a* $\mathbb{P}1_{DG}-\mathbb{P}2$ *discretisation which was first published in* [**Wallwork et al., 2020b**].

For the DG finite element discretisations used for shallow water modelling, flux terms are much more significant, as may be expected. As in Subsection 2.6.2, the terms of the shallow water equations are treated individually. Following the same notation, $\mathbf{q}_h = (\mathbf{u}_h,\eta_h)$ denotes the finite element shallow water solution, $\mathbf{q}^{\text{ext}} = (\mathbf{u}^{\text{ext}},\eta^{\text{ext}})$ any external values enforced on the domain boundary and $\boldsymbol{\xi} = (\boldsymbol{\phi},\zeta)$ test functions from the mixed space. First, we consider the $\mathbb{P}1_{DG}-\mathbb{P}2$ element pair.

Integrating by parts the Lax-Friedrichs stabilised advection term (2.62) yields

$$\begin{aligned}\rho_{\text{adv}}(\mathbf{q}_h, \boldsymbol{\xi}) = &\int_K \boldsymbol{\phi} \cdot (\mathbf{u}_h \cdot \nabla \mathbf{u}_h) \,\mathrm{d}x - \int_{\partial K} ((\mathbf{u}_h \boldsymbol{\psi}^T)\mathbf{u}_h) \cdot \widehat{\mathbf{n}}_K^+ \,\mathrm{d}S \\ &+ \int_{\partial K \setminus \partial\Omega} [\![\boldsymbol{\phi}(\mathbf{u}_h \cdot \widehat{\mathbf{n}})]\!] \cdot \{\!\{\mathbf{u}_h\}\!\} \,\mathrm{d}S + \int_{\partial K \cap \partial\Omega} \left(\boldsymbol{\phi} \cdot \frac{1}{2}(\mathbf{u}_h + \mathbf{u}^{\text{ext}})\right) \mathbf{u}^{\text{rie}} \cdot \widehat{\mathbf{n}} \,\mathrm{d}s \\ &+ \int_{\partial K \setminus \partial\Omega} \beta [\![\boldsymbol{\phi}]\!] \cdot [\![\mathbf{u}_h]\!] \,\mathrm{d}S + \int_{\partial K \cap \Gamma_{\text{freeslip}}} \beta \boldsymbol{\phi} \cdot (\mathbf{u}_h - \mathbf{u}^{\text{ext}}) \,\mathrm{d}s,\end{aligned} \tag{7.18}$$

where the last four terms are as seen previously. Similarly for the SIPG-stabilised viscosity term (2.63):

$$\begin{aligned}\rho_{\text{vis}}(\mathbf{q}_h, \boldsymbol{\xi})|_K = &- \int_K \boldsymbol{\phi} \cdot (\nabla \cdot (\underline{\boldsymbol{\nu}} \nabla \mathbf{u}_h)) \,\mathrm{d}x + \int_{\partial K \setminus \partial\Omega} \boldsymbol{\phi} (\widehat{\mathbf{n}}_K^+)^T : \underline{\boldsymbol{\nu}} \nabla \mathbf{u}_h \,\mathrm{d}S \\ &- \int_{\partial K \setminus \partial\Omega} \left[\!\left[\boldsymbol{\phi} \widehat{\mathbf{n}}^T\right]\!\right] : \{\!\{\underline{\boldsymbol{\nu}} \nabla \mathbf{u}_h\}\!\} \,\mathrm{d}S - \int_{\partial K \setminus \partial\Omega} \{\!\{\nabla \boldsymbol{\phi}\}\!\} : \{\!\{\underline{\boldsymbol{\nu}}\}\!\} \left[\!\left[\mathbf{u}_h \widehat{\mathbf{n}}^T\right]\!\right] \,\mathrm{d}S \\ &- \int_{\partial K \cap \partial\Omega} \nabla \boldsymbol{\phi} : (\underline{\boldsymbol{\nu}}(\mathbf{u}_h - \mathbf{u}^{\text{ext}})) \widehat{\mathbf{n}}^T \,\mathrm{d}s + \int_{\partial K \setminus \partial\Omega} \sigma \left[\!\left[\boldsymbol{\phi} \widehat{\mathbf{n}}^T\right]\!\right] : \{\!\{\underline{\boldsymbol{\nu}}\}\!\} \left[\!\left[\mathbf{u}_h \widehat{\mathbf{n}}^T\right]\!\right] \,\mathrm{d}S \\ &+ \int_{\partial K \cap \partial\Omega} (\sigma \boldsymbol{\phi} \widehat{\mathbf{n}}^T) \cdot (\underline{\boldsymbol{\nu}}(\mathbf{u}_h - \mathbf{u}^{\text{ext}}) \widehat{\mathbf{n}}^T) \,\mathrm{d}s - \int_{\partial K \cap \partial\Omega} \boldsymbol{\phi} \widehat{\mathbf{n}}^T : \underline{\boldsymbol{\nu}} \nabla \mathbf{u}_h \,\mathrm{d}s,\end{aligned} \tag{7.19}$$

where the last six terms are as seen previously. Whilst the continuity term (2.54) does not contain inter-element fluxes (due to the continuity of the elevation space), its integration over K introduces flux terms:

$$\begin{aligned}\rho_{\text{cty}}(\mathbf{q}_h, \boldsymbol{\xi})|_K = &\int_K \zeta \nabla \cdot ((\eta_h + b)\mathbf{u}_h) \,\mathrm{d}x + \int_{\partial K \cap \partial\Omega} \phi(\eta^{\text{rie}} + b)\mathbf{u}^{\text{rie}} \cdot \widehat{\mathbf{n}} \,\mathrm{d}s \\ &- \int_{\partial K} \zeta(\eta_h + b)\mathbf{u}_h \cdot \widehat{\mathbf{n}}_K^+ \,\mathrm{d}S.\end{aligned} \tag{7.20}$$

No integration by parts is performed for the pressure gradient, Coriolis or drag terms.

Having integrated by parts, the goal-oriented error estimator is obtained by substituting instances of the test function tuple $\boldsymbol{\xi}$ with the adjoint error $\mathbf{e}^* = (\mathbf{e}^*_{\mathbf{u}}, e^*_\eta)$ and summing as in (2.55). When restricted to the boundary of an element K, the identities

$$\{\!\{v\}\!\}|_{\partial K} = \frac{1}{2} \mathbb{1}_{\partial K} \, v \quad \text{and} \quad \left[\!\left[\mathbf{v} \widehat{\mathbf{n}}^T\right]\!\right]|_{\partial K} = \mathbb{1}_{\partial K} \, \mathbf{v} (\widehat{\mathbf{n}}_K^+)^T \tag{7.21}$$

follow immediately from definitions (2.44) and (2.45).

The integrals over K sum to return the inner product of the shallow water strong residual, $\boldsymbol{\Psi}(\mathbf{q}_h)$, with the test function tuple. Collecting boundary terms not related to stabilisation or symmetrisation gives

$$\begin{aligned}\left\langle \boldsymbol{\psi}^{\partial}(\mathbf{q}_h), \, \mathbf{e}^* \right\rangle_{\partial K \cap \partial\Omega} := &\left\langle (\eta^{\text{rie}} + b)\mathbf{u}^{\text{rie}} \cdot \widehat{\mathbf{n}}, e^*_\eta \right\rangle_{\partial K \cap \partial\Omega} + \left\langle g(\eta^{\text{rie}} - \eta_h)\widehat{\mathbf{n}}, \mathbf{e}^*_{\mathbf{u}} \right\rangle_{\partial K \cap \partial\Omega} \\ &+ \left\langle \frac{\mathbf{u}_h + \mathbf{u}^{\text{ext}}}{2} \mathbf{u}^{\text{rie}} \cdot \widehat{\mathbf{n}}, \mathbf{e}^*_{\mathbf{u}} \right\rangle_{\partial K \cap \partial\Omega}.\end{aligned} \tag{7.22}$$

As before, these terms convey the extent to which the boundary conditions are satisfied. Ignoring stabilisation and symmetrisation, the inter-element flux terms are given by

$$\begin{aligned}\left\langle \boldsymbol{\psi}^{\text{flux}}(\mathbf{q}_h),\, \mathbf{e}^* \right\rangle_{\partial K} :=& \langle [\![\mathbf{u}_h \cdot \widehat{\mathbf{n}}]\!] \{\!\{\mathbf{u}_h\}\!\}, \mathbf{e}^*_{\mathbf{u}} \rangle_{\partial K \setminus \partial\Omega} - \left\langle (\mathbf{u}_h \cdot \widehat{\mathbf{n}}^+_K)\mathbf{u}_h, \mathbf{e}^*_{\mathbf{u}} \right\rangle_{\partial K} \\ &+ \left\langle \underline{\boldsymbol{\nu}} \nabla \mathbf{u}_h \cdot \widehat{\mathbf{n}}^+_K, \mathbf{e}^*_{\mathbf{u}} \right\rangle_{\partial K \setminus \partial\Omega} - \left\langle \{\!\{\underline{\boldsymbol{\nu}} \nabla \mathbf{u}_h\}\!\} \cdot \widehat{\mathbf{n}}^+_K, \mathbf{e}^*_{\mathbf{u}} \right\rangle_{\partial K \setminus \partial\Omega} \\ &- \left\langle (\eta_h + b)\mathbf{u}_h \cdot \widehat{\mathbf{n}}^+_K, e^*_\eta \right\rangle_{\partial K \setminus \partial\Omega}.\end{aligned} \tag{7.23}$$

The penalisation component of SIPG contributes

$$\left\langle \boldsymbol{\psi}^{\text{IP}}(\mathbf{q}_h), \mathbf{e}^* \right\rangle_{\partial K} := \left\langle \sigma \{\!\{\underline{\boldsymbol{\nu}}\}\!\} [\![\mathbf{u}_h \widehat{\mathbf{n}}^T]\!] \widehat{\mathbf{n}}^+_K, \mathbf{e}^*_{\mathbf{u}} \right\rangle_{\partial K \setminus \partial\Omega} + \left\langle \sigma(\underline{\boldsymbol{\nu}}(\mathbf{u}_h - \mathbf{u}^{\text{ext}})), \mathbf{e}^*_{\mathbf{u}} \right\rangle_{\partial K \cap \partial\Omega}. \tag{7.24}$$

All of the terms above take the form of an inner product whose second argument is the adjoint error. However, it is not the case for the stabilisation terms introduced by the Lax-Friedrichs method:

$$\mathcal{E}^{\text{LF}}_K(\mathbf{q}_h, \mathbf{e}^*) := \langle \beta [\![\mathbf{u}_h]\!], [\![\mathbf{e}^*_{\mathbf{u}}]\!] \rangle_{\partial K \setminus \partial\Omega} + \left\langle \beta(\mathbf{u}_h - \mathbf{u}^{\text{ext}}), \mathbf{e}^*_{\mathbf{u}} \right\rangle_{\partial K \cap \Gamma_{\text{freeslip}}}. \tag{7.25}$$

Nor is it true for the symmetrisation terms introduced by the SIPG method. As in the case of SUPG stabilisation (7.16), symmetrisation introduces integrals containing gradients of the adjoint error:

$$\begin{aligned}\left\langle \boldsymbol{\psi}^{\text{S}}(\mathbf{q}_h), \nabla \mathbf{e}^* \right\rangle_{\partial K} :=& -\frac{1}{2} \left\langle \{\!\{\underline{\boldsymbol{\nu}}\}\!\} [\![\mathbf{u}_h \widehat{\mathbf{n}}^T]\!], \nabla \mathbf{e}^*_{\mathbf{u}} \right\rangle_{\partial K \setminus \partial\Omega} \\ &- \left\langle (\underline{\boldsymbol{\nu}}(\mathbf{u}_h - \mathbf{u}^{\text{ext}}))\widehat{\mathbf{n}}^T, \nabla \mathbf{e}^*_{\mathbf{u}} \right\rangle_{\partial K \cap \partial\Omega}.\end{aligned} \tag{7.26}$$

In summary, the goal-oriented error estimate associated with a $\mathbb{P}1_{DG} - \mathbb{P}2$ discretisation of the shallow water equations is given by

$$\begin{aligned}\rho(\mathbf{q}_h, \mathbf{e}^*)|_K =& \langle \boldsymbol{\Psi}(\mathbf{q}_h), \mathbf{e}^* \rangle_K + \left\langle \boldsymbol{\psi}^{\partial}(\mathbf{q}_h), \mathbf{e}^* \right\rangle_{\partial K \cap \partial\Omega} + \left\langle \boldsymbol{\psi}^{\text{flux}}(\mathbf{q}_h), \mathbf{e}^* \right\rangle_{\partial K} \\ &+ \mathcal{E}^{\text{LF}}_K(\mathbf{q}_h, \mathbf{e}^*) + \left\langle \boldsymbol{\psi}^{\text{IP}}(\mathbf{q}_h), \mathbf{e}^* \right\rangle_{\partial K} + \left\langle \boldsymbol{\psi}^{\text{S}}(\mathbf{q}_h), \nabla \mathbf{e}^* \right\rangle_{\partial K},\end{aligned} \tag{7.27}$$

as defined in (7.22)–(7.26).

Extending this error estimator to the $\mathbb{P}1_{DG} - \mathbb{P}1_{DG}$ case amounts to replacing $(\eta^{\text{rie}} - \eta)$ with η^{rie} in (7.22) and adding in extra terms to (7.23):

$$\left\langle g(\eta^\star - \eta)\widehat{\mathbf{n}}^+_K, \mathbf{e}^*_{\mathbf{u}} \right\rangle_{\partial K} + \left\langle \{\!\{H\}\!\} \mathbf{u}^\star \cdot \widehat{\mathbf{n}}^+_K, e^*_\eta \right\rangle_{\partial K}, \tag{7.28}$$

with $\mathbf{u}^\star$ and $\eta^\star$ as defined in (2.58).

7.2.3. Time-Dependent Extension. The extension of goal-oriented error estimation to the time-dependent case remains an open problem for general discretisations. If the time discretisation is also finite element, giving rise to a *space-time* FEM, then the framework as discussed applies directly. If the time discretisation uses the method of lines, an approximate method must be used. In the following, we propose to simply apply the same time integration method to the error indicators as is applied for the underlying PDE.

Where in the spatial decomposition we consider the contribution from a single element, K, for the temporal decomposition we consider the contribution from a single timestep, $(t^{(k)}, t^{(k+1)}]$. Suppose we have a PDE which is time integrated using Crank-Nicolson with implicitness θ. Evaluating error indicators means taking the instantaneous values and applying the appropriate quadrature rule in time. On pp.54–57 of [**Becker and**

Rannacher, 2001], a space-time FEM treatment of DWR error estimation is shown to be equivalent to Crank-Nicolson for a $\mathbb{P}1$ discretisation in time. For either of the first order forward DWR indicators, (7.16) or (7.27), derived so far, this suggests the choice of error estimator

$$\begin{aligned}\rho_K^{(k)}(w_h, e^*) :=& \frac{1}{\Delta t}\left\langle w_h^{(k+1)}, (e^*)|_{t=t^{(k+1)}}\right\rangle_K - \frac{1}{\Delta t}\left\langle w_h^{(k)}, (e^*)|_{t=t^{(k)}}\right\rangle_K \\ &- \theta\rho_K\left(w_h^{(k+1)}, (e^*)|_{t=t^{(k+1)}}\right) - (1-\theta)\rho_K\left(w_h^{(k)}, (e^*)|_{t=t^{(k)}}\right),\end{aligned} \tag{7.29}$$

where w_h is the forward finite element solution and e^* is the adjoint error. In the default case where $\theta = \frac{1}{2}$, this is simply the trapezium rule. The first order adjoint indicator based on (7.17) may be obtained by exchanging $(w_h, e^*, \rho(\cdot,\cdot))$ with $(w_h^*, e, \rho^*(\cdot,\cdot))$.

7.2.4. A Priori Error Estimates. A posteriori DWR type error results are not the only ones relevant to goal-oriented error estimation. The authors of [Loseille et al., 2010] derive an alternative a priori result. We reformulate it here for consistency with the notation used in this thesis. Whilst DWR estimates contain errors in the forward and/or adjoint solutions, this estimator contains errors incurred when the weak residual is represented in finite element space. For this reason, we distinguish between the weak residual in infinite dimensions and its finite-dimensional approximation using a subscript h in the following.

Proposition 1. Consider function spaces W and V and a PDE with weak residual $\rho : W \times V \to \mathbb{R}$ and unique solution $w \in W$ satisfying $\rho(w; v) = 0$ for any $v \in V$. In addition, consider a finite element weak residual $\rho_h : W_h \times V_h \to \mathbb{R}$ which acts on finite-dimensional subspaces $W_h \subset W$ and $V_h \subset V$ and is formed by discretising all of the associated coefficients and forcing terms appropriately. For any linear QoI $J : W \to \mathbb{R}$, the adjoint solution $w^* \in V$ of the resulting adjoint problem satisfies

$$J(w) - J(w_h) = (\rho - \rho_h)(w; w^*) + \widetilde{R}, \tag{7.30}$$

where the remainder term $\widetilde{R}$ is quadratic in the adjoint error, e^*, and interpolation errors on W_h.

Proof. The proof is modified based on that presented in [Loseille et al., 2010] to use the notation adopted herein.

Firstly, note the identity

$$\rho_h(w; v) - \rho_h(w_h; v) = \rho_h(w; v) - \rho(w; v) = (\rho_h - \rho)(w; v), \qquad \forall v \in V_h, \tag{7.31}$$

which implies that

$$\rho_h(\Pi_h w; w_h^*) - \rho_h(w_h; w_h^*) = \rho_h(\Pi_h w; w_h^*) - \rho_h(w; w_h^*) + (\rho_h - \rho)(w; w_h^*), \tag{7.32}$$

where $\Pi_h : W \to W_h$ projects into the finite element space.

Since we have a linear QoI, the definition of the adjoint problem, the QoI error may be decomposed as

$$\begin{aligned}J(w) - J(w_h) &= J(w - \Pi_h w) + J(\Pi_h w - w_h) \\ &= -\partial_w\rho(w, w^*, w - \Pi_h w) - \partial_w\rho_h(w_h, w_h^*, \Pi_h w - w_h),\end{aligned} \tag{7.33}$$

thanks to identity (7.6).

Consider Taylor expanding $\rho_h(\cdot;w_h^*)$ at w_h, giving the second order remainder term,

$$R_1 := \rho_h(\Pi_h w; w_h^*) - \rho_h(w_h; w_h^*) - \partial_w\rho_h(w_h, w_h^*, \Pi_h w - w_h). \tag{7.34}$$

Combining this with (7.32), the error decomposition (7.33) becomes

$$\begin{aligned} J(w) - J(w_h) = &- \partial_w\rho(w, w^*, w - \Pi_h w) \\ &- \rho_h(\Pi_h w; w_h^*) + \rho_h(w; w_h^*) + (\rho - \rho_h)(w; w_h^*) + R_1. \end{aligned} \tag{7.35}$$

Consider also Taylor expanding $\rho_h(\cdot;w_h^*)$ at w, giving the second order remainder term,

$$R_2 := \partial_w\rho_h(w, w_h^*, \Pi_h w - w) - \rho_h(\Pi_h w; w_h^*) + \rho_h(w; w_h^*). \tag{7.36}$$

Substituted into (7.35), this yields

$$\begin{aligned} J(w) - J(w_h) = &- \partial_w\rho(w, w^*, w - \Pi_h w) - \partial_w\rho_h(w, w_h^*, \Pi_h w - w) \\ &+ (\rho - \rho_h)(w; w_h^*) + R_1 + R_2. \end{aligned} \tag{7.37}$$

Defining the second order remainder terms,

$$\begin{aligned} D_1 &= (\partial_w\rho - \partial_w\rho_h)(w, w_h^*, \Pi_h w - w), \\ D_2 &= \partial_w\rho(w, w^* - w_h^*, \Pi_h w - w), \\ D_3 &= (\rho_h - \rho)(w; w^* - w_h^*), \end{aligned} \tag{7.38}$$

substituting into (7.37) and setting $\widetilde{R} := R_1 + R_2 + D_1 + D_2 + D_3$, we arrive at (7.30), as required. □

The above result has been successfully applied to steady-state Euler problems in [**Loseille et al., 2010**] and has been extended to the time-dependent case in [**Belme et al., 2012**].

As in the DWR case, (7.30) can be interpreted as a 'first order forward' result, since it involves residuals of the forward problem. Whilst the result assumes a linear QoI, it is successfully applied to problems with nonlinear QoIs in an approximate sense in [**Loseille et al., 2010**] and [**Belme et al., 2012**]. Below, we derive the corresponding first order adjoint error estimator for linear PDEs for the first time.

Corollary 1. Consider function spaces W and V and a linear PDE with unique solution $w \in W$, as well as a linear QoI $J : W \to \mathbb{R}$. Denote the weak residual of the associated adjoint problem by $\rho^* : V \times W \to \mathbb{R}$ and its unique solution by $w^* \in V$. In addition, consider a finite element weak residual $\rho_h^* : V_h \times W_h \to \mathbb{R}$ for the adjoint problem, which acts on finite-dimensional subspaces $W_h \subset W$ and $V_h \subset V$ and is formed by discretising all of the associated coefficients and forcing terms appropriately. Then we have the a priori error result

$$J(w) - J(w_h) = (\rho^* - \rho_h^*)(w^*, w) + \widehat{R}, \tag{7.39}$$

where the remainder term $\widehat{R}$ is quadratic in the forward error, e, and interpolation errors on V_h.

Proof. Suppose the forward problem has the weak form

$$a(w, v) = L(v), \qquad \forall v \in V, \tag{7.40}$$

for a bilinear form $a : W \times V \to \mathbb{R}$ and linear form $L : V \to \mathbb{R}$. Then the adjoint problem takes the form

$$a(v, w^*) = J(v), \qquad \forall v \in W. \tag{7.41}$$

Given some additional QoI, we can obtain the adjoint of (7.41). For the specific choice of linear QoI given by $L : V \to \mathbb{R}$, the adjoint of (7.41) is given by (7.40) and has unique solution w. The application of Proposition 1 to (7.41) implies that

$$L(w^*) - L(w_h^*) = (\rho^* - \rho_h^*)(w^*; w) + \widehat{R}, \tag{7.42}$$

where the remainder term $\widehat{R}$ involves the forward error, e, as well as interpolation errors on V_h. The result follows due to (7.7). □

7.3. Approximating Forward and Adjoint Errors

The goal-oriented error estimates (7.16), (7.17) and (7.27) contain terms which are not known: the forward and adjoint errors; more specifically, the true forward and adjoint solutions. Thus, evaluating the error estimators derived above first requires means for obtaining 'better' approximations to the true forward and/or adjoint solutions than afforded by the 'base' finite element space, V_h. There are a number of possible approaches, including:

(1) Solving the adjoint equation again in a globally enriched space;
(2) Solving the adjoint equation again in locally enriched spaces;
(3) Applying superconvergent patch recovery techniques;
(4) Bounding the error estimator.

This list and the majority of the following discussion refer to approximation of the adjoint error. However, the same logic applies for approximating the forward error, except where otherwise stated.

Approaches (1)–(4) are discussed and compared in the following subsections. It should be noted that they do not comprise an exhaustive list. For example, recent research in [**Roth et al., 2021**] has performed enrichment for the adjoint problem using neural networks.

7.3.1. Global Enrichment. The simplest means of gaining a more accurate approximation of w^* is to solve the adjoint problem again in a globally enriched finite element space. That is, another finite dimensional space with more DoFs. This could be the result of a global uniform (h-)refinement of the mesh or a global increase in polynomial order (p-refinement), for example. In these examples, the enriched space is often a superspace of V_h, although this depends on the finite element of choice and need not be the case in general. Provided the adjoint solution is sufficiently regular, it is reasonable to expect that the enriched adjoint solution will provide an improved approximation [**Rognes and Logg, 2013**].

Even with the lowest order global h- and p-refinements, the resulting system is considerably more computationally expensive to solve than the 'base' adjoint system. If the base space is $\mathbb{P}p_{DG}$ on a triangular mesh then a p-refined space has $\frac{p+3}{p+1}$ times more DoFs and an iso-$\mathbb{P}2$ h-refined space has $2^n = 4$ times as many. We do not consider applying both h- and p-refinement, since this would result in *eight* times more DoFs for a $\mathbb{P}1_{DG}$ space, which is deemed to be too high a cost to be feasible. The statements above hold in an approximate sense for $\mathbb{P}p$ spaces on uniform or Delaunay meshes with at least tens of thousands of elements. The discussion above assumes the 2D case; the situation is even worse in 3D.

Under iso-$\mathbb{P}2$ refinement, we denote the enrichment of V_h by $V_{h/2}$. In this case, prolongation may be applied to transfer w_h^* into the enriched space, where it is subtracted from the enriched adjoint solution $w_{h/2}^*$. That is,

$$e^* \approx w_{h/2}^* - \Pi_{h/2} w_h^*, \tag{7.43}$$

where $\Pi_{h/2} : V_h \to V_{h/2}$ denotes prolongation. Error indicators are assembled on the enriched space, before being either injected or projected back to the base space, as desired.

Consider now the case of nonlinear problems, for which the adjoint is dependent on the forward solution. A naïve approach to global enrichment is to solve the forward problem in the enriched space in order to form the adjoint problem in enriched space by a linearisation about $w_{h/2}$, the forward solution in $V_{h/2}$. Given that it is nonlinear, this comes with a significant cost, putting the worth of the whole goal-oriented mesh adaptation routine in question. However, it is standard practice in the literature (see [**Rognes and Logg, 2013**] and [**Roth et al., 2021**], for example) to instead form the adjoint equations using a linearisation about the prolongation $\Pi_{h/2} w_h$ of the solution from the base space. Doing so means that only the adjoint equation need be solved in the enriched space; the cost of prolongation is negligible in comparison. This approach is validated for a steady-state tidal turbine test case in Subsection 7.5.4.

It should be remarked that enrichments involving p-refinement do not necessarily result in sensible discretisations and care should be taken that the resulting system remains stable. For example, the Crouzeix-Raviart $\mathbb{P}1$-nonconforming (CR) element is an effective finite element choice for certain problems [**Crouzeix and Raviart, 1973**], but its p-refinement is not even unisolvent. For a triangular element, DoFs of CR1 are at the midpoints of each edge, whilst CR2 has two DoFs which are spaced $\frac{1}{3}$ and $\frac{2}{3}$ along each edge. The reason that the latter is not unisolvent is that it is possible to find non-zero quadratic polynomials which vanish at these points. For example, take an equilateral triangular element, centred at the origin, in which case each node of the CR2 element is equidistant from the origin, by r units (say). The non-zero quadratic $x^2 + y^2 - r^2$ vanishes at all of those points.

Returning to the elements used in this thesis, p-refinement for the SUPG-stabilised tracer problem gives a $\mathbb{P}2$ space with SUPG stabilisation. That the resulting system is stable is confirmed in [**DeBlois, 1996**]; in fact, that work found SUPG-stabilised $\mathbb{P}2$ to be more accurate than SUPG-stabilised $\mathbb{P}1$. For the mixed shallow water discretisation, p-refinement yields a $\mathbb{P}2_{DG} - \mathbb{P}3$ pair, which inherits the desirable properties of $\mathbb{P}1_{DG} - \mathbb{P}2$. Similarly, $\mathbb{P}2_{DG} - \mathbb{P}2_{DG}$ is a viable element pair choice. Lax-Friedrichs can just as well be applied to a quadratic velocity space as a linear one, implying stability of the resulting finite element method. In summary, p-refinements may be applied to the discretisations described in this thesis without concern.

7.3.2. Local Enrichment. Whilst simple to implement, the method described in Subsection 7.3.1 is undoubtedly the most expensive means of constructing an approximation of the adjoint error. A related approach is to enrich the finite element space *locally*, instead of over the whole domain. Such a strategy was first advocated in [**Babuška and Rheinboldt, 1978b**].

The method described in [**Dolejší and Roskovec, 2017**] applies to DG discretisations, taking a local, element-based approach. That is, the enriched solution $(w_h^*)^+$ is obtained

by solving the adjoint problem an element-by-element fashion. Given an adjoint weak residual $\rho^*(\cdot,\cdot)$, the local problem on element $K \in \mathcal{H}$ takes the form

$$\rho^*((w_K^*)^+, v) = 0, \quad \forall v \in V_h^+(K), \qquad (w_K^*)^+|_{K'} = w_h^*|_K, \quad \forall K' \neq K, \tag{7.44}$$

where $V_h^+(K) \ni (w_K^*)^+|_K$ is the enriched finite element space on element K. The global higher order adjoint approximation is then given by setting $(w_h^*)^+|_K := (w_K^*)^+$ on each element. Clearly, this construction is not valid for continuous discretisations. Partitions of unity other than the element indicator function $\mathbb{1}_K$ are required for that case.

For the implementation in [**Dolejší and Roskovec, 2017**], $V_h^+(K)$ involves only p-refinement, meaning that neither refined meshes, nor mesh-to-mesh interpolation need to be considered. The facility to solve local problems is available in Firedrake through PETSc's `PCPATCH` preconditioner type. However, it has not been considered in this thesis and is proposed as future work. The same framework could also be used to automatically derive goal-oriented error estimates. This was achieved in the FEniCS project in [**Rognes and Logg, 2013**] and would be greatly advantageous, with the consequence that derivations such as those presented in Sections 7.2.1 and 7.2.2 could be avoided.

7.3.3. Patch Recovery. Recall the 'ultraconvergent' Zienkiewicz-Zhu recovery technique described in Subsection 5.3.1. The idea is to take a $\mathbb{P}p_{DG}$ field and recover its gradient in $\mathbb{P}(p+1)$ space. However, it can equally be applied in order to generate a higher order approximation of the field itself. In this case, $N = 1$ in (5.25), meaning we have a matrix-vector linear system, rather than a matrix-matrix one.

A DG variant is considered in [**Dolejší and Solin, 2016**], where the construction is again element-based. The $\mathbb{P}(p+1)$ field recovered over the patch around an element K is restricted onto this element in order to contribute to the global $\mathbb{P}(p+1)_{DG}$ approximation. This method is used for goal-oriented hp-adaptation in [**Dolejší and Roskovec, 2017**], along with the method described in Subsection 7.3.2. A weighting term is included for cells with differing polynomial degrees to account for p-adaptation.

Patch recovery can also be done in CG spaces by either choosing an appropriately wide stencil (see [**Rognes and Logg, 2013**]) or using a different mesh with overlapping elements (see [**Carpio et al., 2013**]).

An advantage of patch recovery type methods is that they are independent of the underlying PDE, unlike global and local enrichment methods. They take as input solution fields and provide as output higher order approximations thereof. Further, such methods do not require the solution of global linear systems, only small local ones. Patch recovery may therefore be implemented relatively cheaply by solving discrete systems, without the need to utilise much of the finite element machinery.

7.3.4. Difference Quotients. Each of the preceding approximation techniques assumes the standard form of a DWR type error estimator, with the adjoint error e^* to be approximated in an enriched space. The 'difference quotient' approach introduced in [**Becker and Rannacher, 2001**] instead evaluates a modified form of the error estimator, which is obtained by applying upper bounds.

Recall the DWR error indicator $\rho_K(w_h, e^*)$ as decomposed into contributions from element interior, domain boundary and inter-element fluxes, as in (7.14). We have the upper bound

$$\begin{aligned}\rho_K(w_h, e^*) &= \langle \Psi(w_h), e^* \rangle_K + \langle \psi(w_h), e^* \rangle_{\partial K} \\ &\leq \|\Psi(w_h)\|_K \|e^*\|_K + \|\psi(w_h)\|_{\partial K} \|e^*\|_{\partial K} \\ &\leq \left(\|\Psi(w_h)\|_K + h_K^{-\frac{1}{2}} \|\psi(w_h)\|_{\partial K} \right) \underbrace{\left(\|e^*\|_K + h_K^{\frac{1}{2}} \|e^*\|_{\partial K} \right)}_{\omega_K},\end{aligned} \tag{7.45}$$

where $\psi(\cdot) := \psi^{\partial}(\cdot) + \psi^{\text{flux}}(\cdot)$ combines flux and boundary terms and the first inequality follows by Cauchy-Schwarz. Let us restrict attention to the case of an elliptic problem solved in $\mathbb{P}1$ space. Following [**Ciarlet, 1978**] and [**Brenner and Scott, 2007**], we have

$$\begin{aligned}\omega_K &:= \|w^* - w_h^*\|_K + h_K^{\frac{1}{2}} \|w^* - w_h^*\|_{\partial K} \\ &\leq C' \max \left\{ \|w^* - \Pi_h w^*\|_K, h_K^{\frac{1}{2}} \|w^* - \Pi_h w^*\|_{\partial K} \right\} \\ &\leq C h_K^2 |w^*|_{H^2(K)} = C h_K^2 \|\nabla^2 w^*\|_K,\end{aligned} \tag{7.46}$$

for some constants $C' > 0$ and $C > 0$.

The interpolation constant C is difficult to determine in practice, meaning that we cannot use the difference quotient approach to evaluate a global error estimator. However, it may be localised, giving rise to the error indicator

$$\eta_K^{\text{DQ}} := \left(\|\Psi(w_h)\|_K + h_K^{-\frac{1}{2}} \|\psi(w_h)\|_{\partial K} \right) \|\nabla \cdot \mathbf{R}(w_h^*)\|_K \tag{7.47}$$

on element K. Here $\mathbf{R}(w_h^*)$ is a $\mathbb{P}1$ recovery operator for the gradient of the adjoint solution, such as L^2 projection. Taking the finite element derivative yields a $\mathbb{P}0$ field.

In the context of tracer transport problems solved in $\mathbb{P}1$ space with SUPG stabilisation, the fact that we have a Petrov-Galerkin method means that we may account for the stabilisation term as

$$\eta_K^{\text{DQ}} := \left(\|\Psi(c_h)\|_K + h_K^{-\frac{1}{2}} \|\psi(c_h)\|_{\partial K} \right) \|\nabla \cdot \mathbf{R}(c_h^* + \tau \mathbf{u} \cdot \nabla c_h^*)\|_K. \tag{7.48}$$

A disadvantage of the difference quotient method is that it is not straightforwardly extensible to other stabilisation methods or discretisations. The method must also be modified when applied to vector or mixed problems.

7.3.5. Comparison. In this subsection, we compare two global enrichment strategies and the difference quotient method:

$\mathbf{GE_h}$: Global enrichment with h-refinement;
$\mathbf{GE_p}$: Global enrichment with p-refinement;
DQ: Difference quotients.

Comparison is made in terms of the computational cost to generate an error indicator field and in terms of the *effectivity index* of the global error estimator. The effectivity index, I_{eff}, is an overestimation/underestimation measure for error estimators. First introduced

in [Babuška and Rheinboldt, 1978a], the version used here is expressed by the formula

$$I_{\text{eff}} := \frac{|\eta(w_h)|/|J(w_h)|}{|J(w) - J(w_h)|/|J(w)|}, \tag{7.49}$$

for a global error estimator η of the QoI error $J(w) - J(w_h)$. Ideally, we seek an error estimator such that $I_{\text{eff}} = 1$, in which case the error estimator is exact. For the purposes of mesh adaptation, it is usually sufficient to have I_{eff} converge to some positive constant, since are concerned with local contributions to the error, rather than overall scale factors. We expect the DQ estimator to be an overestimate, because we have made no effort to establish a value for the interpolation constant C.

In order to evaluate the effectivity index, we need to consider a problem with a known analytical solution. To avoid model errors, we consider a Poisson problem whose source term is an analytic function of the mesh coordinates. Such a problem could represent tracer transport without advection, for example. The test case was introduced in [Micheletti et al., 2010]:

$$-\Delta c = f \quad \text{in} \quad \Omega = [0,1]^2, \qquad c|_{\partial\Omega} = 0, \tag{7.50}$$

where $c \in H_0^2(\Omega) =: V$ and the forcing term is chosen to give the manufactured analytical solution $c(x,y) = \sin(\pi x)\sin(\pi y)$. The weak form of (7.50) is given by

$$\langle \nabla c, \nabla \phi \rangle = \langle f, \phi \rangle, \qquad \forall \phi \in H_0^1(\Omega) \subset V \tag{7.51}$$

and the QoI is chosen as

$$J(c) = \langle \nabla c, \nabla g_i \rangle, \quad \text{where} \quad \begin{cases} g_1(x,y) := xy(x-1)(y-1) \\ g_2(x,y) := (\frac{\pi}{2} + \arctan(40(x - \frac{1}{2})))g_1(x,y) \end{cases} \tag{7.52}$$

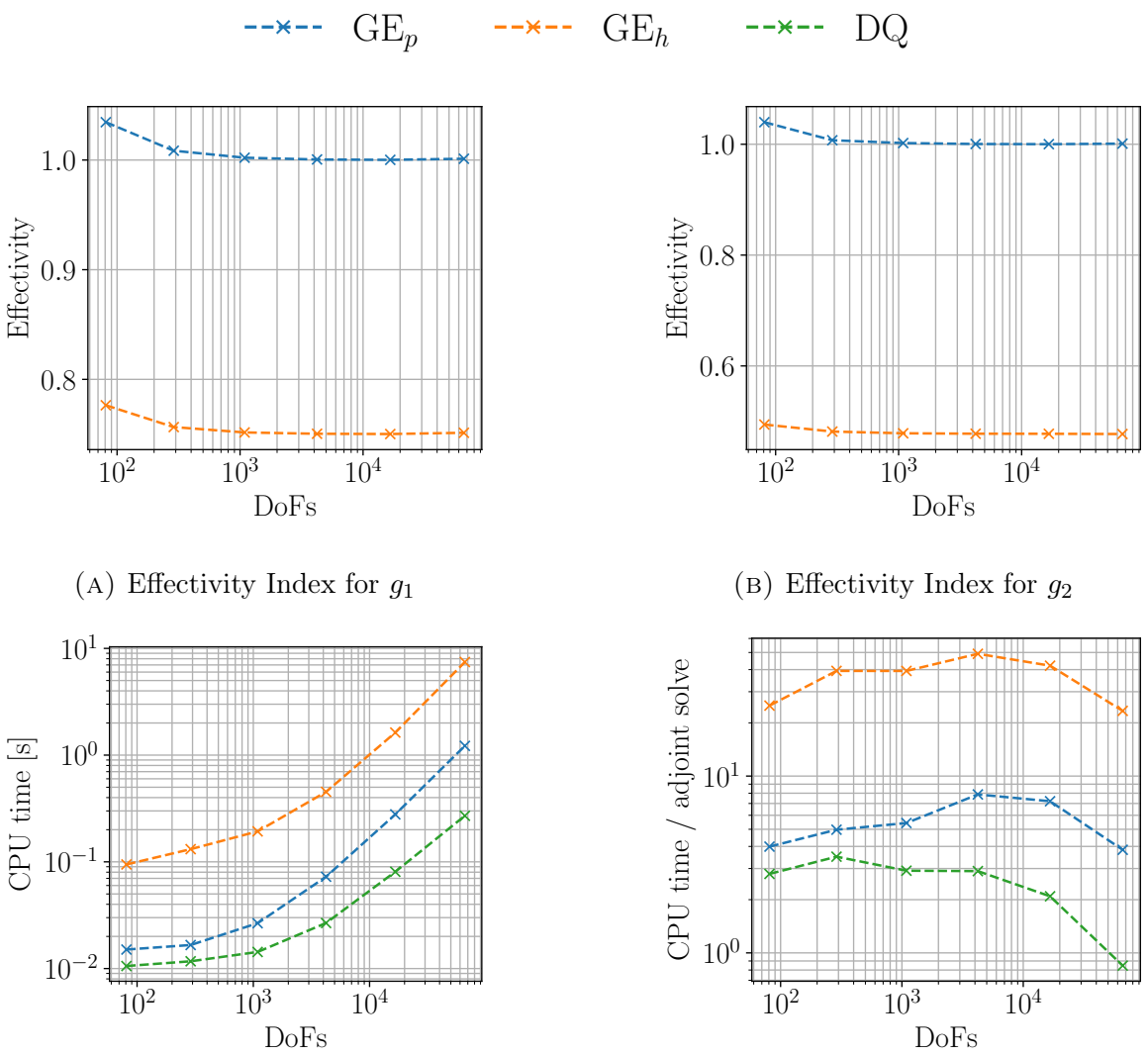
are two choices for the Riesz representation. For g_1, we have a 'bubble function' on $[0,1]^2$, whereas g_2 uses a smooth approximation to an indicator function in order to 'zero out' values on the left-hand side of the domain. In either case, the exact QoI value is available.

For a sequence of increasingly refined meshes, we solve (7.51) and its adjoint in a $\mathbb{P}1$ space $V_h \subset H_0^1(\Omega)$ and evaluate (7.49) for each enrichment method (which requires auxiliary PDE solves). In this instance, we actually evaluate the first order adjoint error estimates, since we have the analytical solution for the forward problem, rather than the adjoint.

Subfigures 7.1a and 7.1b illustrates the effectivity indices on a sequence of uniformly refined meshes. In evaluating (7.49), we consider the absolute value of the sum of contributions over all elements. The main purpose of this experiment is to assess the approximation quality of the enriched forward solution.

Out of the two global enrichment methods considered, GE_p has the best effectivity properties, with I_{eff} converging to unity for both g_1 and g_2 as resolution is increased. The effectivity of GE_h also converges to a fixed value, although there is a slight underestimation. The underestimation is worse for g_2.

Subfigure 7.1c illustrates the CPU time associated with each method. Note that forward and adjoint solves in the base space are not included in the timing, only the error indicator evaluation and any enriched solves required for it. In addition, Subfigure 7.1d shows the same timings, but relative to the CPU time associated with an adjoint solve on the base mesh.



(A) Effectivity Index for g_1

(B) Effectivity Index for g_2

(C) Absolute CPU time

(D) Relative CPU time

FIGURE 7.1. Comparison of enrichment methods for a Poisson test case with an analytical solution in terms of effectivity index and CPU time. 'CPU time' corresponds to the time taken to evaluate the corresponding DWR error estimator. Values are computed on a sequence of uniformly refined meshes.

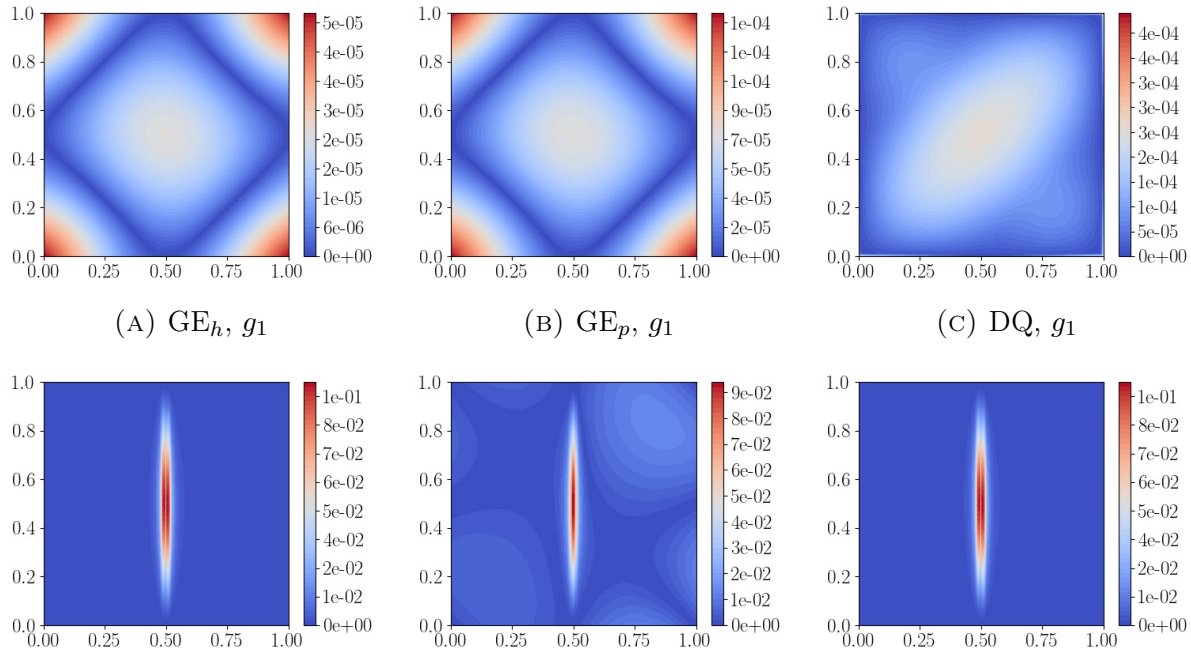
As discussed in Subsection 7.3.1, for a $\mathbb{P}1$ base space, GE$_h$ and GE$_p$ involve adjoint solves in spaces with approximately 4 and 2 times as many DoFs, respectively. However, Subfigure 7.1d demonstrates that the computational cost is not all due to DoF count, since none of the curves have constant gradient. As such, there are other costs, such as interpolation and the construction of the enriched spaces (which is included in the cost analysis).

The relative cost of GE$_h$ and GE$_p$ appear to grow slightly with increasing DoF count and then decrease for higher resolutions. DQ follows a similar pattern and actually becomes cheaper than an adjoint solve on the base space for the finest mesh considered. Recall that this method does not require enriched spaces at all; its main cost is the recovery step. Consequently, it is by far the cheapest method considered. Whilst, as has been stated,

DQ is not useful for global error estimation, its cheapness is very desirable in the context of mesh adaptation.

Of the two global enrichment methods, GE_p stands out as being the best suited for goal-oriented error estimation, due to its desirable effectivity properties and having lower cost.

Figure 7.2 shows the error indicator fields generated by each of the enrichment methods. A 131,072 element uniform mesh is used and both Riesz representations are considered. Qualitatively, the global enrichment methods yield similar error indicator fields for both cases, although GE_p indicates some background features which GE_h does not, which could explain its enhanced effectivity properties. Error is estimated to be high at the corners in the first case and along the line $x = 0.5$ (where g_2 has a sharp gradient) in the second case. The DQ indicator has a similar structure to the global enrichment indicators for g_2, but not g_1, where it fails to indicate large errors committed at the corners. This is likely because the DQ formulation (7.48) involves the Laplacian of the adjoint solution, which also lives in $H_0^1(\Omega)$ and therefore goes to zero at the domain boundaries.



(A) GE_h, g_1 (B) GE_p, g_1 (C) DQ, g_1

(D) GE_h, g_2 (E) GE_p, g_2 (F) DQ, g_2

FIGURE 7.2. DWR indicator fields resulting from three different enrichment methods for a Poisson test case with an analytical solution. Presented on a uniform mesh with 131,072 elements.

7.4. Goal-Oriented Metric-Based Mesh Adaptation

The simplest means of generating a metric from a scalar goal-oriented error indicator field is to scale an identity matrix of appropriate dimension. This gives rise to an isotropic metric, as described in Subsection 7.4.1. In advection-dominated problems, it can be advantageous to use meshes whose anisotropy aligns with features of the flow which are deemed to be important. As such, we seek to develop anisotropic Riemannian metrics for the purposes of goal-oriented mesh adaptation.

The literature contains many possible ways to generate anisotropic goal-oriented metrics. To the best of the author's knowledge, the first published use of an anisotropic metric derived from DWR error estimators is [**Formaggia et al., 2004**]. In that work, an element-wise interpolation error bound is derived for a scalar field, in terms of its Hessian, as well as elemental anisotropy coefficients. It is argued that an element-wise anisotropic goal-oriented metric may be established by reconstructing an orthogonal eigendecomposition using the Hessian matrix and DWR error indicators. DWR indicators fit naturally with this approach, because they are also defined in an element-wise basis. The methodology of [**Formaggia et al., 2004**] was later extended to higher order elements and strong anisotropy in [**Carpio et al., 2013**].

Hessian matrices provide natural means of incorporating anisotropy into a metric, since they contain curvature information in each coordinate direction. One simple but effective Hessian-based goal-oriented metric is that proposed in [**Power et al., 2006**]. Instead of using DWR indicators directly, this approach reinterprets the error result in order to formulate a metric as the weighting of the Hessian of the adjoint solution by the strong residual of the forward equation. A metric based on the first order adjoint DWR error estimator may also be derived. An advantage of this approach is that it does not require the (often computationally expensive) approximation techniques for the forward and adjoint error terms described in Section 7.3.

Not all goal-oriented metrics are based on (a posteriori) dual weighted residual error results. The anisotropic metric proposed in [**Loseille et al., 2010**] instead uses a related a priori result. Like the approach of [**Power et al., 2006**], this method does not require the evaluation of DWR error indicators. Also in a similar vein, a reinterpretation of the error result is deployed in order to argue that Hessian matrices may be used to represent interpolation errors in an anisotropic manner. The approach was shown to be effective in the application of Euler equations to aerospace problems in [**Loseille et al., 2010**]. Its extension to the time-dependent case was considered and validated in [**Alauzet et al., 2011**, **Belme et al., 2012**].

One approach which is notable in that it generates an anisotropic metric *without* using Hessian matrices is that described in [**Rogé and Martin, 2008**]. The authors instead introduce anisotropy by sequentially refining the mesh in each coordinate direction and evaluating error indicators accordingly. These indicators are interpolated back onto the base mesh, whereby orthogonal matrices may be assembled, whose columns correspond to the major axes of anisotropy. An anisotropic metric may be constructed using an orthogonal eigendecomposition with eigenvalues corresponding to residuals in the major axes' directions. Whilst this approach does not require the recovery of Hessian matrices, it does use gradient information. This approach is deemed to be too computationally expensive and complex to be considered further, since we seek efficient ways to generate goal-oriented metrics.

As revealed above, there are many ways to introduce anisotropy into Riemannian metrics, each of which deserves investigation in its own right. For the remainder of this section, we focus on modified versions of four of the methods mentioned in the literature review above. These methods include an isotropic approach (*'Isotropic DWR'*), as well as the anisotropic approaches from [**Formaggia et al., 2004**] (*'Anisotropic DWR'*), [**Power et al., 2006**] (*'Weighted Hessian'*) and [**Loseille et al., 2010**] (*'Weighted Gradient'*).

7.4.1. Isotropic DWR. Let ρ_K denote the first order forward DWR indicator (7.9) on mesh element K and ρ_K^* denote the first order adjoint indicator (7.10).

Vertex-Based. Creating an isotropic metric from ρ_K is as simple as scaling the appropriate identity matrix, as in (5.15). However, recall that Riemannian metrics are defined point-wise and Pragmatic interprets them vertex-wise. Thus, we require metrics which live in tensor $\mathbb{P}1$ space and need apply a projection:

$$\mathcal{M}^{\mathrm{DWR}} := \Pi_1\left(\sum_{K\in\mathcal{H}} \rho_K \mathbb{1}_K\right) \underline{\boldsymbol{I}}_n. \tag{7.53}$$

Metric (7.53) may be normalised using the L^p normalisation techniques discussed in Subsection 5.4.1. We use the label 'DWR' because the metric is built directly from the forward DWR error indicator on each element. The corresponding isotropic metric built from ρ_K^* is denoted $\mathcal{M}^{\mathrm{DWR}^*}$.

In line with the second order DWR result (7.11), another possible isotropic metric may be obtained by combining $\mathcal{M}^{\mathrm{DWR}}$ and $\mathcal{M}^{\mathrm{DWR}^*}$. Usually, it is recommended to normalise metrics before combination. However, in this special case, an average of the unnormalised metrics is the isotropic metric derived from the second order DWR indicator. The non-commutativity of averaging and normalisation is demonstrated in Subsection 7.5.2.

Element-Based. An alternative normalisation procedure for the element-based metric framework is presented in [**Carpio et al., 2013**] and is described in the following. For an element K with volume $|K|$, we seek the 'optimal' element volume $|\widetilde{K}|$ which minimises interpolation error under the constraint that the metric complexity is smaller than a target value, $\mathcal{C}_T > 0$. In [**Braack, 1998**], it is proved that the optimum is achieved by

$$|\widetilde{K}| = |K| \left(\frac{\sum_{K\in\mathcal{H}} \rho_K^{\frac{1}{\alpha+1}}}{\mathcal{C}_T}\right) \rho_K^{-\frac{1}{\alpha+1}}, \tag{7.54}$$

for a parameter $\alpha > 1$ which is not known a priori. A vertex-wise isotropic metric may be obtained by projection as

$$\mathcal{M}_\alpha^{\mathrm{DWR}} := \Pi_1\left(\sum_{K\in\mathcal{H}} \frac{|\widehat{K}|}{|\widetilde{K}|} \mathbb{1}_K\right) \underline{\boldsymbol{I}}_n, \tag{7.55}$$

where $|\widehat{K}|$ is the volume of the (fixed) reference element. In practice, α may be understood as akin to the norm order in L^p normalisation methods, except that resulting meshes tend to be more multi-scale for *large* values of α.

Observe that metric (7.55) is mesh-dependent, meaning that meshes generated using it are heavily influenced by the meshes on which these metrics are defined. However, since we consider metric-based mesh adaptation to be driven by a fixed-point iteration (with the goal-oriented extension presented in Subsection 7.5.1), the repeated mesh adaptation and metric redefinition steps act to ensure that the dependence on the initial mesh diminishes with iteration count [**Wallwork et al., 2020b**]. Provided the convergence criteria are chosen sufficiently strictly, a suitable number of iterations are performed for mesh independence to be achieved.

7.4.2. Anisotropic DWR. The alternative normalisation (7.54) can also be used to define an anisotropic metric, which builds upon the element-based framework described in Subsection 5.1.5.

On an element $K \in \mathcal{H}$, suppose we have a constant Hessian metric $\underline{\boldsymbol{H}}_K$ recovered from a solution component, or combination thereof. As in the vertex-based case, $\underline{\boldsymbol{H}}_K$ admits an orthogonal eigendecomposition,

$$\underline{\boldsymbol{H}}_K = \underline{\boldsymbol{V}}_K \, \underline{\boldsymbol{\Lambda}}_K \, \underline{\boldsymbol{V}}_K^T, \qquad \underline{\boldsymbol{\Lambda}}_K = \mathrm{diag}(\lambda_{K,1}, \dots, \lambda_{K,n}). \tag{7.56}$$

Without loss of generality (because the eigendecomposition ordering is not unique), assume $|\lambda_{K,1}| \geq |\lambda_{K,2}| \geq \cdots \geq |\lambda_{K,n}| > 0$. As with the SPD part of the cell Jacobian, stretching factors $\{s_{K,i}\}_{i=1}^n$ for the Hessian may be defined as in (5.11), but with the eigenvalues taken in modulus. A vertex-wise metric may be constructed by projection:

$$\mathcal{M}_\alpha^{\mathrm{ADWR}} := \Pi_1 \left(\sum_{K \in \mathcal{H}} \frac{|\widehat{K}|}{|\widetilde{K}|} \underline{\boldsymbol{V}}_K \, \mathrm{diag}\left(s_{K,1}^{\frac{1}{2}}, \dots, s_{K,n}^{\frac{1}{2}} \right) \underline{\boldsymbol{V}}_K^T \, \mathbb{1}_K \right), \tag{7.57}$$

where, again, $|\widehat{K}|$ is the volume of the (fixed) reference element. This construction is very appealing as an anisotropic metric, since element size is controlled by $|\widetilde{K}|$, orientation is controlled by $\underline{\boldsymbol{V}}_K$ and shape is controlled by the stretching factor matrix. Adjoint information is contained within $|\widetilde{K}|$. Note that the goal-oriented error indicator impacts the size, but not the anisotropy, whilst the solution field impacts anisotropy but not size.

The label 'ADWR' refers to the fact we have an anisotropic metric, constructed directly from the DWR error indicator on each element. The subscript α acts as a reminder that this metric is normalised using (7.54). Notice that if each $s_{K,i} = 1$ then we recover the element-based isotropic metric (7.55).

For the version of (7.57) derived from the first order adjoint DWR error indicator, $\mathcal{M}_\alpha^{\mathrm{ADWR}^*}$, $|\widetilde{K}|$ is constructed from ρ_K^* and the eigendecomposition of the *adjoint* solution Hessian is modified.

7.4.3. 'Weighted Hessian' Approach. Another approach involving Hessians of solution fields is presented in [**Power et al., 2006**]. In this case, however, the Hessian of the *adjoint* solution is used for the metric constructed from the first order forward DWR estimator and the Hessian of the *forward* solution is used for the adjoint version.

As in Subsection 7.3.4, the DWR estimators (7.9) and (7.10) may be re-interpreted as interpolation error results using Céa's lemma (2.28). Ignoring flux terms, this gives

$$\begin{aligned} |J(w) - J(w_h)| &\leq \|\Psi_h(w_h)\| \, \|w^* - w_h^*\| \leq C_1 \|\Psi_h(w_h)\| \, \|w^* - \Pi_h w^*\|, \\ |J(w) - J(w_h)| &\leq \|\Psi_h^*(w_h^*)\| \, \|w - w_h\| \leq C_2 \|\Psi_h^*(w_h^*)\| \, \|w - \Pi_h w\|, \end{aligned} \tag{7.58}$$

where $C_1, C_2 > 0$ are constants and the Cauchy-Schwarz inequality is applied to obtain the first pair of upper bounds.

Given the relationship between interpolation errors and Hessians revealed in Section 5.3, the authors of [**Power et al., 2006**] suggest deriving anisotropic metrics from (7.58) by scaling Hessians of the adjoint and forward solutions, respectively. That is, we can construct anisotropic metrics by weighting the Hessian of the adjoint/forward solution by the forward/adjoint strong residual. In practice, we project into $\mathbb{P}1$ space, as this is

required by the mesh adaptation toolkit used herein:

$$\begin{aligned}\mathcal{M}^{\mathrm{WH}} &:= \Pi_1\left(\sum_{K\in\mathcal{H}}\|\Psi(w_h)\|_K\mathbb{1}_K\right)|\underline{\boldsymbol{H}}(w^*)|,\\ \mathcal{M}^{\mathrm{WH}^*} &:= \Pi_1\left(\sum_{K\in\mathcal{H}}\|\Psi^*(w_h^*)\|_K\mathbb{1}_K\right)|\underline{\boldsymbol{H}}(w)|,\end{aligned} \tag{7.59}$$

where Ψ and Ψ^* are strong residuals for the forward and adjoint problems and $\Pi_1 : \mathbb{P}0 \to \mathbb{P}1$ is an appropriate projection operator. Clearly, anisotropy is inherited from the Hessian.

In practice, the Hessians of the exact forward and adjoint solutions are approximated by Hessians recovered from the corresponding finite element solutions using methods described in Subsection 5.3.1. Of course, the Hessian metrics arising from (7.59) should be normalised before deployment.

If the finite element space is not scalar, the Hessians arising from each component of the adjoint/forward solution should be combined before assembling weighted Hessian metrics. Again, a metric related to the second order DWR error estimate may be obtained by combining the metrics in (7.59).

Note that, as presented, the weighted Hessian approach accounts for contributions to the DWR from element interiors, but not those from inter-element fluxes, boundary conditions or stabilisation. As noted in [**Wallwork et al., 2021**], in the context of a tracer transport model stabilised using an SUPG method with parameter τ, the metric can be modified by exploiting the fact that we have a Petrov-Galerkin method:

$$\mathcal{M}^{\mathrm{WH}}_{\mathrm{SUPG}} := \Pi_1\left(\sum_{K\in\mathcal{H}}\|\Psi(c_h)\|_K\mathbb{1}_K\right)|\underline{\boldsymbol{H}}(c_h^* + \tau\mathbf{u}\cdot\nabla c_h^*)| \tag{7.60}$$

and similarly for the adjoint version, $\mathcal{M}^{\mathrm{WH}^*}_{\mathrm{SUPG}}$. Note that this approach is very similar to that which accounts for SUPG stabilisation in the difference quotient (7.48).

7.4.4. 'Weighted Gradient' Approach. Unlike the goal-oriented metrics above, the approach of [**Loseille et al., 2010**] is based on the a priori error estimate (7.30) derived therein. Where the DWR may be interpreted as the strong residual weighted by the adjoint error, (7.30) can be viewed as the interpolation error of the strong residual, weighted by the adjoint solution. That is,

$$J(w) - J(w_h) \approx (\rho - \Pi_h\rho)(w; w^*), \tag{7.61}$$

assuming the remainder term to be small. For simplicity of presentation, assume that we have a scalar PDE. For the vector PDE case, see pp.3–5 of [**Loseille et al., 2010**].

If the strong form PDE can be written in conservative form with a potential functional $\boldsymbol{\mathcal{F}}$ as

$$\Psi(w) = \nabla\cdot\boldsymbol{\mathcal{F}}(w) = 0, \tag{7.62}$$

then (7.61) takes the form

$$J(w) - J(w_h) \approx \left\langle\widehat{\mathbf{n}}\cdot(\overline{\boldsymbol{\mathcal{F}}} - \Pi_h\overline{\boldsymbol{\mathcal{F}}})(w), w^*\right\rangle_{\partial\Omega} - \left\langle(\boldsymbol{\mathcal{F}} - \Pi_h\boldsymbol{\mathcal{F}})(w), \nabla w^*\right\rangle. \tag{7.63}$$

The trace term $\overline{\boldsymbol{\mathcal{F}}}$ results from the boundary integral contributions in the integration by parts, along with any Neumann boundary conditions. Moreover, by the triangle inequality,

$$|J(w) - J(w_h)| \leq \langle |(\boldsymbol{\mathcal{F}} - \Pi_h \boldsymbol{\mathcal{F}})(w)|, |\nabla w^*| \rangle + \left\langle \left| \widehat{\mathbf{n}} \cdot (\overline{\boldsymbol{\mathcal{F}}} - \Pi_h \overline{\boldsymbol{\mathcal{F}}})(w) \right|, |w^*| \right\rangle_{\partial\Omega}. \tag{7.64}$$

Forward Weighted Gradient Metric. Note the presence of interpolation errors in $\boldsymbol{\mathcal{F}}$ and $\overline{\boldsymbol{\mathcal{F}}}$ in (7.64). Using the bound on the L^p-norm of the interpolation error provided by (5.19) as justification, we may construct an anisotropic metric by an appropriate normalisation of

$$\underline{\boldsymbol{H}}^{\text{volume}} := \sum_{i=1}^{n} |\underline{\boldsymbol{H}}(\mathcal{F}_i(w))| \left| \frac{\partial w^*}{\partial x_i} \right|, \qquad \underline{\boldsymbol{H}}^{\text{surface}} := |w^*| \left| \underline{\boldsymbol{H}} \left(\overline{\boldsymbol{\mathcal{F}}}(w) \cdot \widehat{\mathbf{n}} \right) \right|. \tag{7.65}$$

Following the notation used in Chapter 5, the modulus sign indicates that we have restricted eigenvalues to be positive, in order to ensure the Hessian is indeed a metric. To use these metrics in practice, we first normalise them using the approach documented in Section 5.4 and then intersect them on the boundary:

$$\mathcal{M}^{\text{WG}} := \begin{cases} \underline{\boldsymbol{H}}^{\text{volume}}_{L^p} & \text{in} \quad \Omega \\ \underline{\boldsymbol{H}}^{\text{volume}}_{L^p} \cap \underline{\boldsymbol{H}}^{\text{surface}}_{L^p} & \text{on} \quad \partial\Omega \end{cases}. \tag{7.66}$$

It is not immediately obvious how to choose a target metric complexity so that (7.66) attains it, because the volume and surface components are of different dimension. Making an appropriate choice of normalisation constant involves solving a small nonlinear algebraic system, whose solution depends on the density of the constituent metrics. See [**Loseille et al., 2010**] for details on the algebra problem and its solution.

For some problems, the PDE cannot be captured in conservative form (7.62). Suppose instead it can be expressed as

$$\Psi(w) = \nabla \cdot \boldsymbol{\mathcal{F}}(w) = f, \tag{7.67}$$

where the forcing term f is independent of w. In (7.64) this yields an additional interpolation error term for the forcing:

$$\begin{aligned} |J(w) - J(w_h)| \leq{}& \langle |(\boldsymbol{\mathcal{F}} - \Pi_h \boldsymbol{\mathcal{F}})(w)|, |\nabla w^*| \rangle \\ &+ \left\langle \left| \widehat{\mathbf{n}} \cdot (\overline{\boldsymbol{\mathcal{F}}} - \Pi_h \overline{\boldsymbol{\mathcal{F}}})(w) \right|, |w^*| \right\rangle_{\partial\Omega} \\ &+ \langle |f - \Pi_h f|, |w^*| \rangle. \end{aligned} \tag{7.68}$$

In [**Wallwork et al., 2020a**], we recommended modifying the volume Hessian to give

$$\underline{\boldsymbol{H}}^{\text{volume}}_f := \sum_{i=1}^{n} |\underline{\boldsymbol{H}}(\mathcal{F}_i(w))| \left| \frac{\partial w^*}{\partial x_i} \right| + |\underline{\boldsymbol{H}}(f)| \, |w^*|. \tag{7.69}$$

Of course, there are many PDEs which cannot be expressed in the form (7.67) with f independent of w. However, (7.65) and (7.69) are sufficient for many tracer transport problems.

Adjoint Weighted Gradient Metric. Suppose the adjoint problem can also be written in conservative form, for some potential $\mathcal{G}$, modulo a forcing term g:

$$\Psi^*(w^*) = \nabla \cdot \mathcal{G}(w^*) = g. \tag{7.70}$$

In [Wallwork et al., 2020a], we proposed an adjoint weighted gradient metric $\mathcal{M}^{\mathrm{WG}^*}$, constructed by intersecting

(7.71)
$$\underline{\boldsymbol{H}}^{\mathrm{volume}^*} := \sum_{i=1}^{n} |\underline{\boldsymbol{H}}(\mathcal{G}_i(w^*))| \left|\frac{\partial w}{\partial x_i}\right| + |\underline{\boldsymbol{H}}(g)|\,|w^*|, \qquad \underline{\boldsymbol{H}}^{\mathrm{surface}^*} := |w|\left|\underline{\boldsymbol{H}}\left(\overline{\boldsymbol{\mathcal{G}}}(w^*)\cdot\widehat{\mathbf{n}}\right)\right|,$$

as in (7.66). In that work, the volume component of this metric was demonstrated to be effective in improving QoI convergence properties, but the corresponding error estimator was neither proven, nor claimed to exist. That it is indeed based on a goal-oriented error estimate is proved by Corollary 1.

7.4.5. Other Methods. The goal-oriented metrics discussed in Subsections 7.4.1–7.4.4 are derived from a posteriori and a priori goal-oriented error estimates. There also exist goal-oriented methods which do not make use of these results and are instead based on the QoI alone.

One such method is presented in [Micheletti et al., 2010] and makes use of the Riesz representation theorem (2.4.7) to express a QoI $J : H_0^1(\Omega) \to \mathbb{R}$ for a standard Poisson problem in the form

$$J(v) := \int_\Omega \nabla v \cdot \nabla g \,\mathrm{d}x, \qquad v \in H_0^1(\Omega), \tag{7.72}$$

where $g = \mathcal{R}_{H^1}(J)$. By Galerkin orthogonality of the Poisson problem, (7.72) satisfies

$$J(e) = \langle \nabla w - \nabla w_h, \nabla g - \nabla \Pi_h g\rangle\,, \tag{7.73}$$

where w is the exact solution, w_h is the finite element solution in $\mathbb{P}p$ space and Π_h interpolates into that space. Gradients of the discrete quantities, w_h and $\Pi_h g$, may be computed directly, in the finite element sense. Gradients of w and g, on the other hand, must be approximated using a recovery method, such as described in Subsection 7.3.3. The authors propose isotropic and anisotropic element-based adaptation schemes involving integrals of the directly computed and recovered gradients over patches. Effectively, we have a 'QoI-based' method which exploits its structure.

A great advantage of the above approach is that it does not require solution of the adjoint problem; metrics are constructed by post-processing of the forward solution alone. This implies a significant reduction in computational cost and memory requirements compared with the goal-oriented metrics described in previous subsections. Further, an anisotropic framework is available, as proposed in [Micheletti et al., 2010]. A disadvantage of the method as stated is that it only ensures optimal convergence for the Poisson problem, otherwise (7.73) is not exact. However, the extension to advection-diffusion-reaction problems is considered in [Micheletti and Perotto, 2013].

Whilst the approach above avoids solving the adjoint problem, it still requires enrichment methods. The approach of [Davis and LeVeque, 2016], on the other hand, avoids enrichment, but still requires solution of the adjoint problem. It is focused on tsunami propagation modelling using the linear shallow water equations, for a continuous, linear QoI of the form

$$J(\mathbf{u},\eta) := \int_{T_{\mathrm{start}}}^{T_{\mathrm{end}}} \int_R \eta(\mathbf{x},T)\,\mathrm{d}x\,\mathrm{d}t, \quad R \subset \Omega. \tag{7.74}$$

That is, free surface displacement is integrated over a coastal region, R, which is held to be of importance. By accurately approximating J, it is possible to make a hazard assessment for a hypothetical or past tsunami. Such a QoI admits an L^2 Riesz representation given by the indicator function $\mathcal{R}_{L^2}(J) = (\mathbf{0}, \mathbb{1}_{R\times\{T\}})$. The authors show that, under particular choices of boundary conditions, the adjoint solution associated with (7.74) satisfies

$$\langle \mathbf{q}^*(\mathbf{x}, t_1), \mathbf{q}(\mathbf{x}, t_1)\rangle = \langle \mathbf{q}^*(\mathbf{x}, t_2), \mathbf{q}(\mathbf{x}, t_2)\rangle\,, \qquad \forall\, 0 \le t_1 \le t_2 \le T. \tag{7.75}$$

As such, they advocate a mesh adaptation indicator which is simply the spatial L^2 inner product of the approximate forward and adjoint shallow water solution tuples, $\mathbf{q}_h$ and $\mathbf{q}_h^*$.

The problem-dependence of this 'adjoint-based' method means that it is not straightforwardly extensible to general problems. However, it is demonstrated in Section 7.8 that it largely mimics the isotropic DWR approach, at a reduced computational cost. Given that we have an inner product of the forward ('primal') and adjoint ('dual') solutions, we refer to this error indicator as *'dual weighted primal' (DWP)*.[2]

7.5. Steady-State Goal-Oriented Mesh Adaptation

7.5.1. Goal-Oriented Mesh Adaptation Loop. As discussed in Chapter 5, a fixed-point iteration is used for metric-based mesh adaptation methods. Convergence is achieved if the change in mesh element count crosses a pre-specified threshold. In the goal-oriented case, convergence may also be checked based on relative tolerances for values of the QoI or error estimator. Algorithm 3 shows the modifications in the steady-state case. For the version which uses a target interpolation error, rather than a target metric complexity, see Algorithm 1 of [**Wallwork et al., 2020a**].

Given target metric complexity $\mathcal{C}_T > 0$;
Given initial mesh $\mathcal{H}_0$;
Set $i := 0$;
while *not converged* **do**
  Solve forward problem on $\mathcal{H}_i$;
  Evaluate the QoI and check for its convergence;
  Solve adjoint problem on $\mathcal{H}_i$;
  Extract metric $\mathcal{M}_i$, normalising for target complexity $\mathcal{C}_T$;
  Apply metric gradation to $\mathcal{M}_i$;
  Adapt the mesh using $\mathcal{M}_i$ to obtain $\mathcal{H}_{i+1}$;
  Increment i;
end

Algorithm 3: Goal-oriented mesh adaptation routine for time-independent problems, as presented in [**Wallwork et al., 2021**].

Algorithm 3 is applied to steady-state test cases in the following three subsections.

[2]This term was not used by the authors of [**Davis and LeVeque, 2016**].

7.5.2. Point Discharge with Diffusion in 2D. *This subsection contains experimental results which were published in the conference paper* [**Wallwork et al., 2020a**] *and later extended in* [**Wallwork et al., 2021**]. *The results presented are as in* [**Wallwork et al., 2021**].

Consider again the 'Point Discharge with Diffusion' test case introduced in Subsection 3.6.1. Recall that we have two source-receiver problems, whose receiver regions differ: one which is directly downstream of the source (R_1) and another which is offset (R_2). Recall that the adjoint problems – whose solutions are presented in Figure 3.4 – are also advection-diffusion problems, with opposite flow direction.

In [**Wallwork et al., 2020a**], we compared QoI convergence curves for the isotropic, weighted Hessian (WH) and weighted gradient (WG) metrics, as well as qualitative features of the resulting meshes. In addition, we investigated convergence properties and mesh features given by averaging or intersecting each metric with its adjoint counterpart.

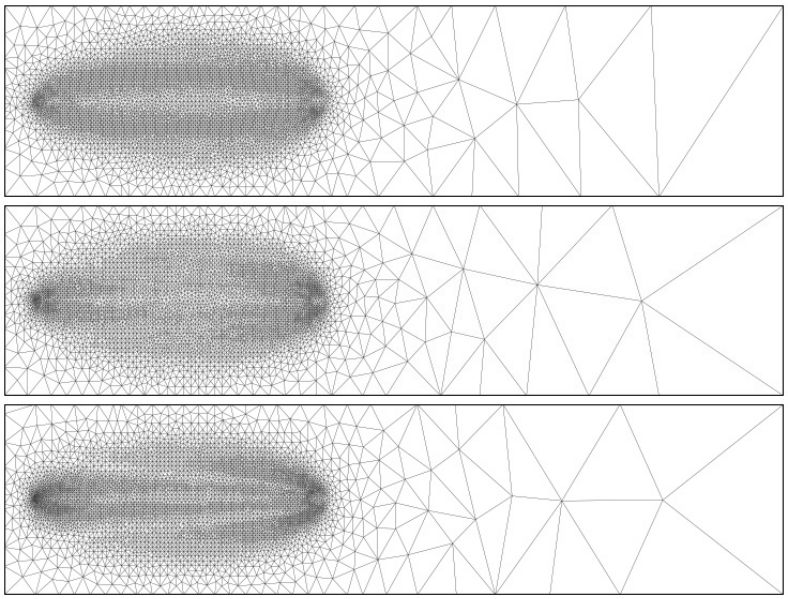
The implementation used to generate the results presented in [**Wallwork et al., 2020a**] contained a number of simplifications, as follows. As mentioned in Subsection 3.6.1, the point source was represented using a indicator function with a small radius which was calibrated by trial-and-error. Regarding the isotropic metric, an interpretation of DWR error indicators as a sum of moduli was used, rather than the modulus of a sum, in line with the 'cell facet split' approach in [**Rognes and Logg, 2010**]. As such, cross terms between the element interior and inter-element/boundary terms were exacerbated, rather than cancelled. The 'error representation' approach in [**Rognes and Logg, 2010**] (modulus of a sum) was found to be more effective in later work. Regarding the WH approach, SUPG stabilisation was not accounted for. Finally, regarding the WG approach, Hessian surface terms were neglected, leading to potential under-resolution near the domain boundary.

The numerical experiments were later reimplemented in [**Wallwork et al., 2021**]. In that paper, the Gaussian point source parametrisation presented in Subsection 3.6.1 was adopted and calibrated using a gradient-based optimisation method, making for a more accurate representation. DWR indicators were computed using the difference quotient formulation, for purposes of efficiency. In addition, the anisotropic DWR metric was included in the comparison. The WH approach was modified to account for SUPG stabilisation using (7.60). Boundary Hessian metrics were recovered for WG and combined with those on the domain interior using the methodology of [**Loseille et al., 2010**]. In terms of discretisation differences for the underlying PDE, a SUPG stabilisation parameter whose element size measure accounts for anisotropy was adopted, rather than the circumradius (see Subsection 2.7.3). Finally, the discrete adjoint of the SUPG stabilised forward problem was utilised, rather than the SUPG stabilised continuous adjoint problem. This decision was informed by the Taylor test in Subsection 3.6.1.

In the following, we present the results of [**Wallwork et al., 2021**] (which compared the four goal-oriented metrics), along with a number of other comparisons: enrichment methods for the isotropic DWR metric; continuous vs. discrete adjoint methods; and combination of first order forward and adjoint metrics by averaging or intersection. For brevity, we refer to [**Wallwork et al., 2020a**] for the original results.

Comparison of Enrichment Methods. First, we consider the isotropic DWR metric, $\mathcal{M}^{\mathrm{DWR}}$, with error indicator evaluated using different enrichment methods: GE_h, GE_p and DQ. The discrete adjoint method is used to obtain adjoint solutions.

Figure 7.3 shows example meshes in the aligned configuration. They have in common bands of high mesh resolution upstream of the receiver region and coarse resolution downstream. This is consistent with the test case being advection-dominated, whereby the QoI value is insensitive to the dynamics downstream of the receiver. In some cases the mesh is extremely coarse near the outflow boundary.



(A) GE_h
(11,586 elements,
5,830 DoFs,
max. aspect ratio 1.8)

(B) GE_p
(10,593 elements,
5,334 DoFs,
max. aspect ratio 1.8)

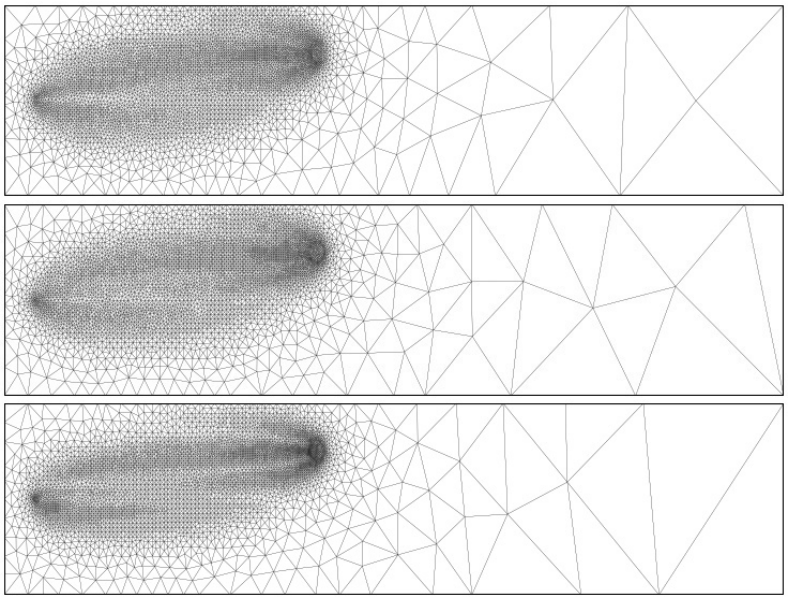
(C) DQ
(11,253 elements,
5,660 DoFs,
max. aspect ratio 2.7)

FIGURE 7.3. Example meshes for isotropic goal-oriented mesh adaptation applied to the 'Point Discharge with Diffusion' test case with aligned source and receiver, using different enrichment methods. L^1 normalisation is applied to all metrics.

The mesh arising from the DQ method takes a slightly different form from the others, with bands of resolution spreading upstream from the receiver. This is likely due to the inclusion of the Laplacian of the adjoint solution in the error indicator. For the global enrichment methods, the bands appear to be aligned with the axis from source to receiver. GE_h gives rise to the mesh which is closest to being symmetric in the bisector of the source-receiver axis. An exception is that, as with the other meshes, more resolution is deployed around the point source than around the receiver region.

The meshes presented in Subfigures 7.3a–7.3b have elements with aspect ratio two or smaller, meaning that they are effectively isotropic, as may be expected from the use of an isotropic metric. Examining the aspect ratio field in more detail, the DQ method has just one element where the value is larger than two, in the top left corner. In the offset configuration presented in Figure 7.4, it is the GE_p mesh which exhibits a single element with aspect ratio greater than two, on the right hand boundary. Other than these exceptions, all of the elements are effectively isotropic. The qualitative observations made above carry over to the offset case, except that, for GE_h and, especially, DQ, more resolution is deployed in the receiver region than before. This is because the receiver is now outside of the centre of the flow, so smaller tracer concentrations are to be detected there.

By increasing the target complexity in the normalisation strategies, we are able to obtain goal-oriented metrics of increased complexity and hence meshes with more DoFs. Figure



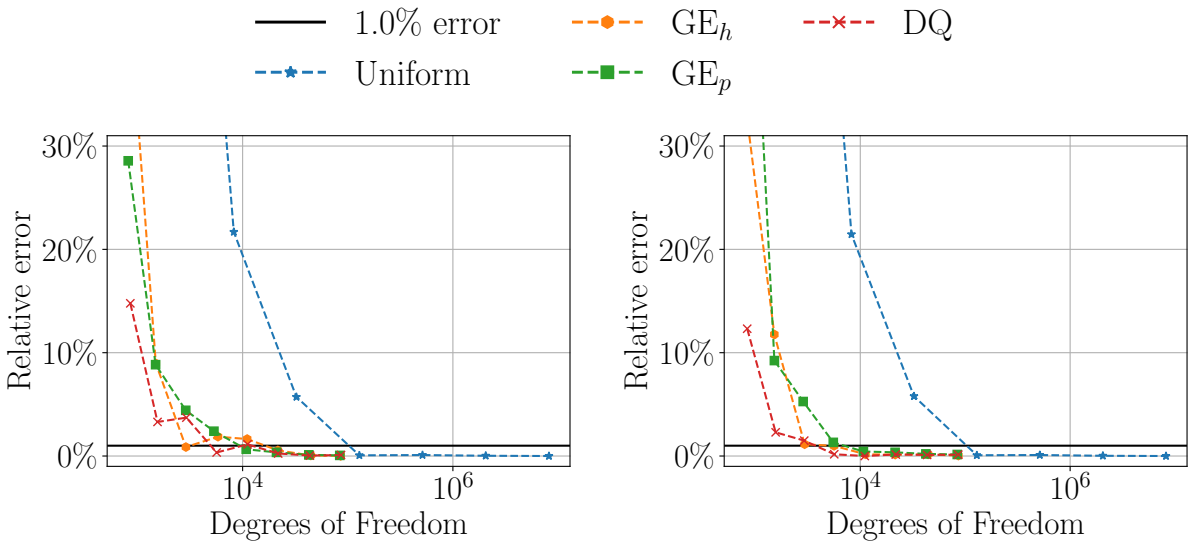
(A) GE_h
(11,151 elements,
5,621 DoFs,
max. aspect ratio 1.6)

(B) GE_p
(10,968 elements,
5,528 DoFs,
max. aspect ratio 2.7)

(C) DQ
(11,215 elements,
5,647 DoFs,
max. aspect ratio 1.6)

FIGURE 7.4. Example meshes for isotropic goal-oriented mesh adaptation applied to the 'Point Discharge with Diffusion' test case with offset source and receiver, using different enrichment methods. L^1 normalisation is applied to all metrics.

7.5 shows convergence curves for the four methods, as well as iso-ℙ2 (uniform) refinement. For both aligned and offset cases, uniform refinement requires over 100,000 DoFs in order to obtain relative error smaller than 1%. Under each enrichment method, an order of magnitude fewer DoFs are required in order to cross this threshold. Moreover, all approaches yield errors consistently below 1% thereafter.



(A) Aligned source and receiver

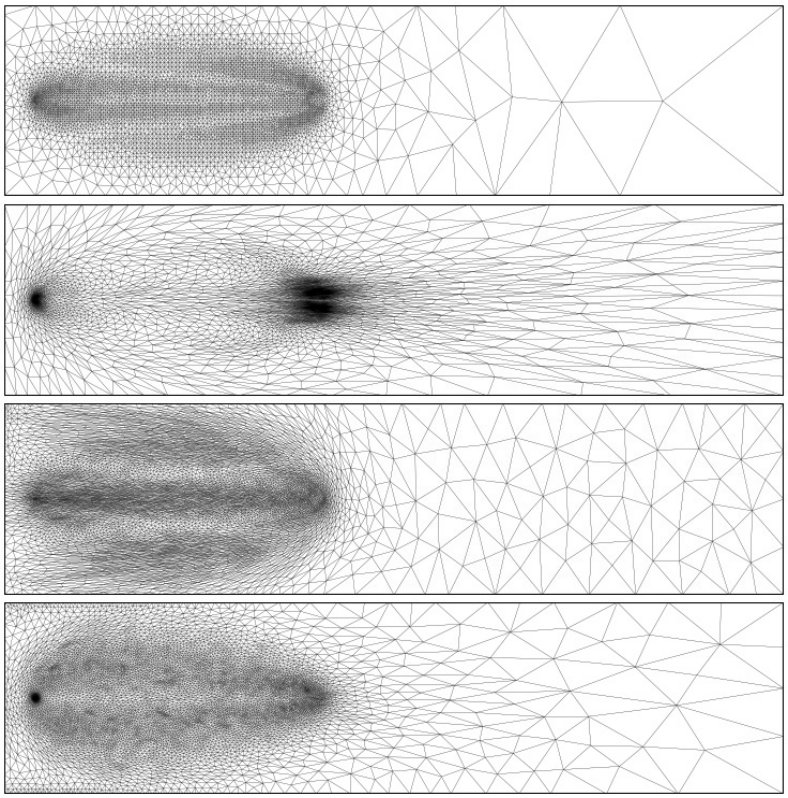
(B) Offset source and receiver

FIGURE 7.5. QoI convergence plots for the 'Point Discharge with Diffusion' test case, for an isotropic goal-oriented metric, under four different enrichment methods. L^1 normalisation is applied to all metrics.

In the aligned case shown in Subfigure 7.5a, all three methods appear to perform well, ensuring a fairly consistent decrease in error with increasing DoF count, although there are some exceptions, where the error rises again temporarily. In the offset case, all methods enable errors less than 1% with 10,000 DoFs or more. Somewhat surprisingly, the DQ method stands out as having very desirable convergence properties – especially in the offset configuration – despite having by far the lowest computational cost. Given that it is also competitive in the aligned case, this method of accounting for the adjoint error is advocated for first order forward error indicators. It is not possible to come to a conclusion regarding which of the global enrichment methods provides the best option for this particular test case, based on accuracy alone. Given that p-refinement is the cheapest and has the best effectivity properties, it is likely that it would be the most effective choice in practice.

Comparison of First Order Forward Metrics. Having established an enrichment method of choice, we proceed to compare the metrics $\mathcal{M}^{\text{DWR}}$, $\mathcal{M}^{\text{ADWR}}$, $\mathcal{M}^{\text{WH}}$ and $\mathcal{M}^{\text{WG}}$, each of which involves residuals of the forward problem.

In the case of aligned source and receiver, the goal-oriented metric generation methods yield meshes such as those presented in Figure 7.6. Again, the meshes each have bands of high resolution upstream of the receiver and coarse resolution downstream.



(A) Isotropic DWR (11,253 elements, 5,660 DoFs, max. aspect ratio 2.7)

(B) Anisotropic DWR (13,871 elements, 6,967 DoFs, max. aspect ratio 19.8)

(C) Weighted Hessian (11,775 elements, 5,946 DoFs, max. aspect ratio 23.1)

(D) Weighted gradient (17,908 elements, 9,041 DoFs, max. aspect ratio 8.1)

FIGURE 7.6. Example meshes for isotropic and anisotropic goal-oriented mesh adaptation applied to the 'Point Discharge with Diffusion' test case with aligned source and receiver. The anisotropic DWR metric is normalised using $\alpha = 2$; L^1 normalisation is used in all other cases. As presented in [**Wallwork et al., 2021**].

The mesh due to the isotropic DWR metric presented in Subfigure 7.6a is fairly isotropic, with maximum aspect ratio 2.7 across all elements. This is in fact the same mesh as

presented in Subfigure 7.3c, whose sole element with aspect ratio greater than two is in the top left corner. The other meshes are at least moderately anisotropic, with the most anisotropic elements (maximum aspect ratio 23.1) being due to WH.

Comparing Subfigures 7.6b and 7.6c, it is clear that the ADWR mesh retains anisotropy in the downstream region, whereas the elements there are fairly isotropic for WH. This may be explained by noting that WH inherits anisotropy from derivatives of the adjoint tracer concentration, which is uniformly zero downstream. Unlike WH, WG inherits its anisotropy from the Hessian of the potential functional, which is not uniformly zero downstream. As such, all elements in the resulting mesh are moderately anisotropic.

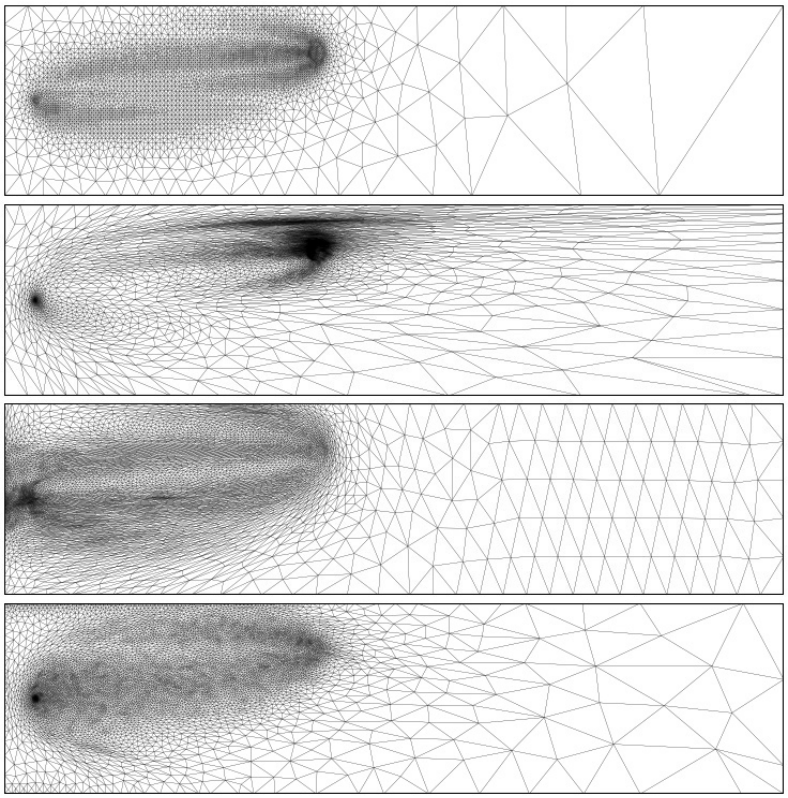
As discussed in Subsection 5.8.2, normalisation parameters play an important role regarding the distribution of resolution in meshes arising from a particular metric. For the (vertex-based) isotropic DWR, WH and WG methods, L^1 normalisation is applied, in order to give multi-scale meshes. For the (element-based) ADWR method, the parameter choice $\alpha = 2$ was found to be effective. For these choices of normalisation parameters, ADWR deploys higher mesh resolution around the source and receiver. However, it is difficult to compare differently normalised metrics and this observation would not necessarily hold for all values of α.

Recall that the WG metric is comprised of a combination of four metrics: two 'volume' metrics, one 'surface' metric and an additional metric which accounts for the source term (see (7.69)). The source term manifests in the mesh as high resolution around the point source. The surface term only contributes on the top and bottom boundaries, where Neumann conditions are imposed. It conveys the extent to which these boundary conditions are imposed. This manifests in the mesh as increased resolution at the top and bottom boundaries of the upstream region. The same increased resolution is not apparent downstream because of the weighting by the (uniformly zero) adjoint solution. In addition, the fact that WG consists of a combination of normalised metrics explains why the mesh presented in Subfigure 7.6d has considerably more elements and DoFs than the other meshes, despite using the same target metric complexity.

Now consider the offset case, for which example meshes are presented in Figure 7.7. All of the above qualitative observations made in the aligned case carry over, with the meshes having comparable anisotropy. In the case of ADWR, the maximal aspect ratio of 74.1 is significantly higher than in the aligned case. This is likely due to the skew introduced by the offset problem.

Convergence of QoI error with increased mesh element count is demonstrated for all four approaches in Figure 7.8. For the metrics considered, an order of magnitude fewer DoFs are required in order to cross the 1% error threshold than uniform refinement, in both configurations. Moreover, errors are consistently below 1% thereafter.

It is difficult to draw any conclusions about the optimality of one particular metric, since the differences between convergence curves are not significant or consistent. However, it is remarkable that errors smaller than 1% can be attained with almost *two* orders of magnitude fewer elements than are required using uniform refinement – WG in the aligned case and ADWR in the offset case. These are perhaps coincidentally small errors due to cancellations, since larger QoI errors are reported on meshes with more DoFs. That the ADWR metric performs comparably to the other metrics suggests that the DQ formulation works well in the anisotropic, as well as isotropic construction. This is encouraging in terms of the development of efficient goal-oriented mesh adaptation strategies, due to its



(A) Isotropic DWR (11,215 elements, 5,647 DoFs, max. aspect ratio 1.6)

(B) Anisotropic DWR (13,643 elements, 6,853 DoFs, max. aspect ratio 74.1)

(C) Weighted Hessian (13,066 elements, 6,611 DoFs, max. aspect ratio 22.4)

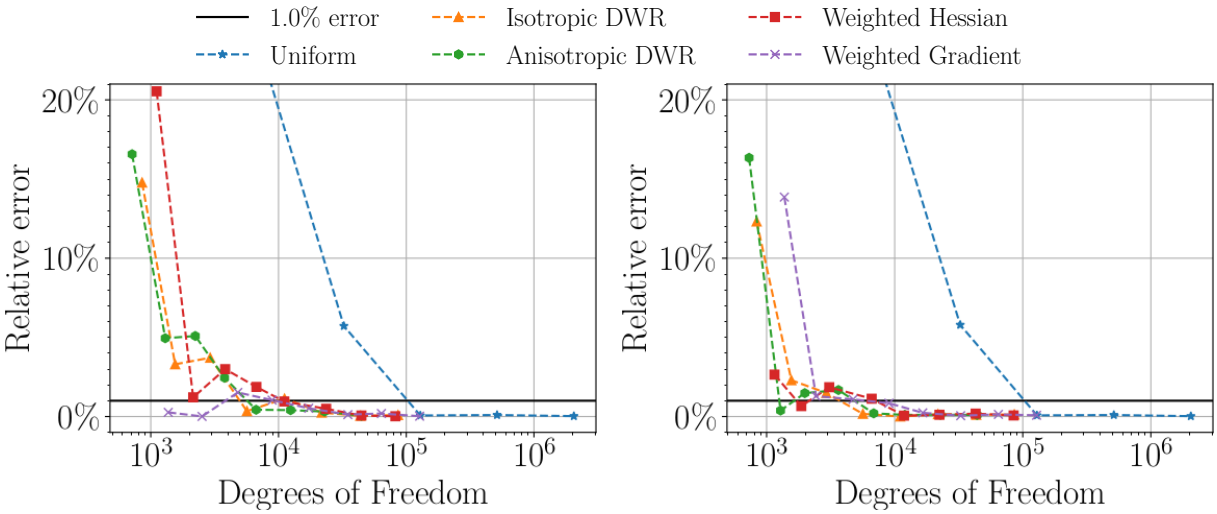
(D) Weighted gradient (17,717 elements, 8.972 DoFs, max. aspect ratio 8.0)

FIGURE 7.7. Example meshes for isotropic and anisotropic goal-oriented mesh adaptation applied to the 'Point Discharge with Diffusion' test case with offset source and receiver. The anisotropic DWR metric is normalised using $\alpha = 2$; L^1 normalisation is used in all other cases. As presented in [**Wallwork et al., 2021**].

low computational cost. It is an advantage of the ADWR method that it is sufficiently flexible to allow for different error estimator formulations; this is not possible with WH or WG.

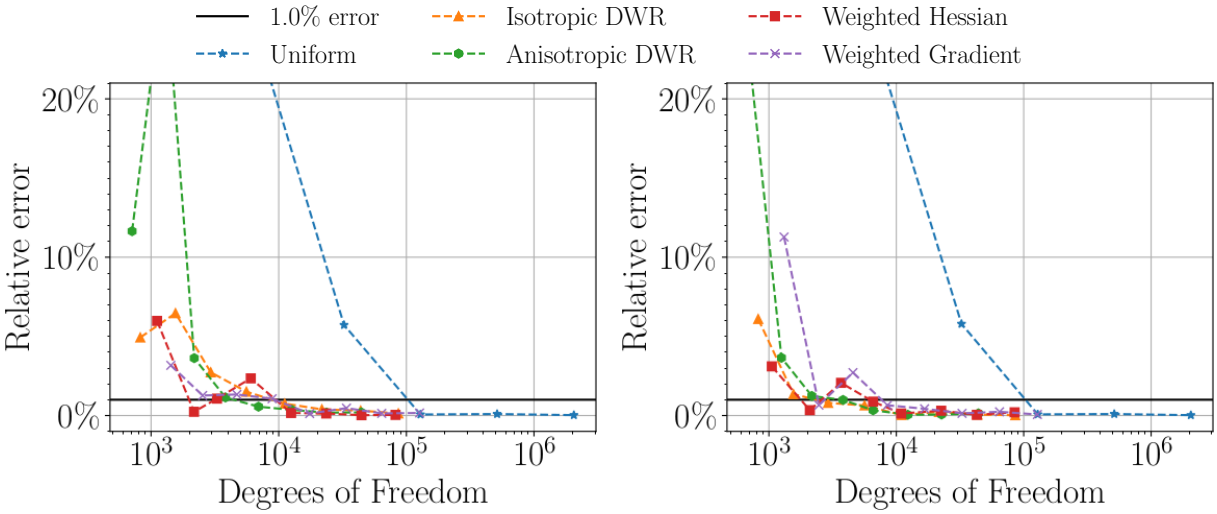
It may be that this test case makes it 'too easy' for the goal-oriented metrics, enabling them all to achieve a high accuracy with relatively few DoFs, after which the error plateaus. Indeed, the convergence analysis experiments for the steady-state shallow water test case presented in Subsection 7.5.4 show a clearer distinction between different goal-oriented metrics. In conclusion, it is not possible to hold any one of the metrics to be best suited to the tracer transport test case presented here, since they all perform well.

Comparison of Continuous and Discrete Adjoint. Following the conclusions of the Taylor test in Subsection 3.6.1, the discrete adjoint is used in the results presented thus far in this subsection. For comparison, Figure 7.9 shows the corresponding convergence curves assuming the continuous adjoint method. As in the previous comparison, the continuous adjoint problem is stabilised using SUPG, whereas the discrete adjoint is applied to the SUPG-stabilised forward problem. It is not obvious that discrete adjoint or continuous adjoint leads to lower QoI errors overall, except that there are more notable increases in error with increasing DoF count under the continuous adjoint (see WG in the offset configuration and WH in both cases). Both discrete and continuous adjoint methods yield errors consistently below 1% with more than 10,000 DoFs, for all goal-oriented metrics considered. As such, we can proceed to use either of the two methods.



(A) Aligned source and receiver (B) Offset source and receiver

FIGURE 7.8. QoI convergence plots for the 'Point Discharge with Diffusion' test case, for four different goal-oriented metrics constructed using the discrete adjoint method. The anisotropic DWR metric is normalised using $\alpha = 2$; L^1 normalisation is used in all other cases. As presented in [**Wallwork et al., 2021**].



(A) Aligned source and receiver (B) Offset source and receiver

FIGURE 7.9. QoI convergence plots for the 'Point Discharge with Diffusion' test case, for four different goal-oriented metrics constructed using the continuous adjoint method. The anisotropic DWR metric is normalised using $\alpha = 2$; L^1 normalisation is used in all other cases.

Comparison of Second Order metrics. Four goal-oriented metrics built upon first order forward goal-oriented error indicators have been compared. Next, consider metrics constructed from the corresponding second order goal-oriented error indicators. Recall that the second order DWR indicator is simply the average of the first order forward and adjoint versions, (7.12)–(7.13), the latter of which contains residuals in the adjoint equation. The automatic decomposition of error indicators (such as the discrete adjoint weak residual) into cell and flux components in Firedrake, as done in FEniCS in [**Rognes**

and Logg, 2013], is proposed as future work. For the time being, we opt to use the continuous adjoint in the following. The convergence curves presented in Figure 7.9 suggest that doing so does not result in a significant loss of accuracy. In addition, we find that the DQ formulation is not as effective when applied to the first order adjoint version of the ADWR metric. Instead, we choose the GE_p method for all subsequent experiments, since this is the next most computationally efficient method.

The results below repeat those performed in [**Wallwork et al., 2020a**], but with simplifications of that work overcome, as described above. For both aligned and offset configurations, the goal-oriented metrics already presented are combined with $\mathcal{M}^{\mathrm{DWR},*}$, $\mathcal{M}^{\mathrm{ADWR},*}$, $\mathcal{M}^{\mathrm{WH},*}$ or $\mathcal{M}^{\mathrm{WG},*}$, as appropriate. Combination by averaging and intersection are compared. In addition, we include an isotropic metric constructed directly from the second order DWR error estimator, which differs from the averaged first order metrics because normalisation and combination are applied in reverse order.

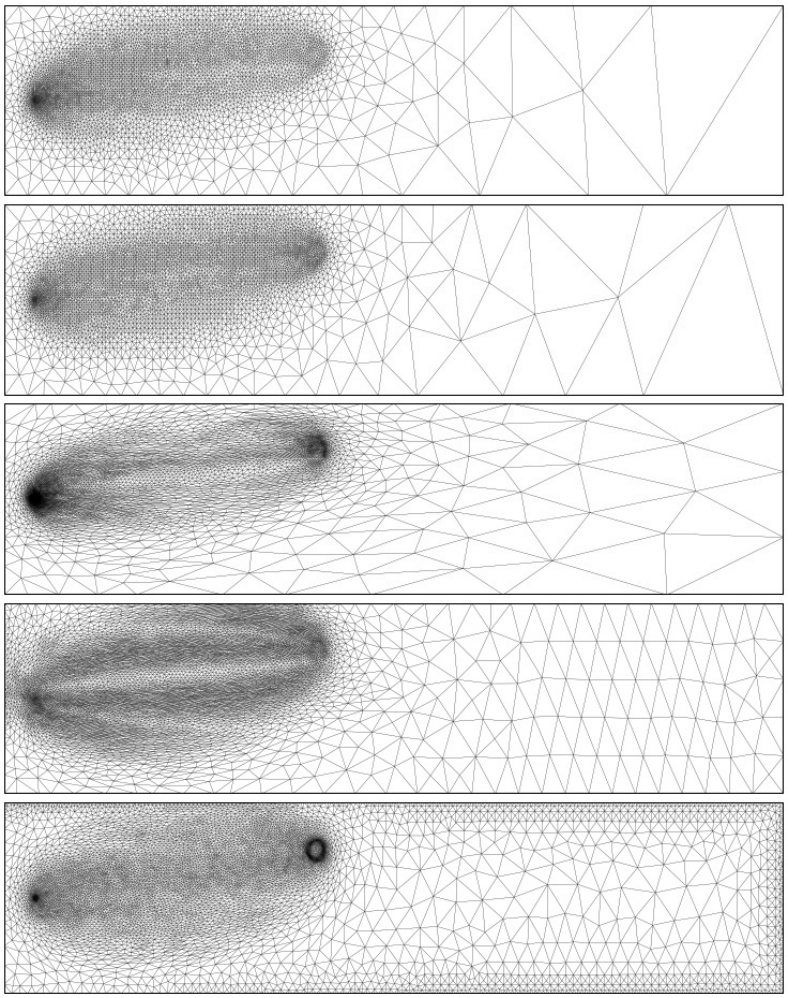
Figure 7.10 shows example meshes arising from averaging forward and adjoint metrics. The meshes associated with the second order estimate (Subfigure 7.10a) and average of first order metrics (Subfigure 7.10b) are qualitatively very similar, with resolution most focused around the source.

The ADWR mesh (Subfigure 7.10c) differs from the version based on the forward estimator alone (Subfigure 7.7b) in that the elements of highest anisotropy are contained within the region between source and receiver, rather than extending along the top boundary. This is likely because the adjoint metric involves the Hessian of the adjoint solution, which is uniformly zero downstream of the receiver.

Whilst the mesh due to the averaged WH metric (Subfigure 7.10d) is qualitatively very similar to what was seen previously, the averaged WG metric leads to meshes with a rather different structure (Subfigure 7.10e). The metric focuses mesh resolution along all boundaries except the inflow, especially in the downstream region. This is due to the boundary component for the adjoint WG metric involving a term which assessed how well the Robin condition is weakly satisfied. Where the mesh in Subfigure 7.7d focuses its resolution most clearly on the source for the forward problem, here it is also focused on the source for the adjoint problem. Since the adjoint source is an indicator function, many DoFs are deployed to resolve its discontinuities.

Note that the final metrics used to generate both forms of averaged isotropic metric attain the target complexity, 4000. The intersected metric presented in Subfigure 7.11a, on the other hand, has complexity 5690.1. This is because the metrics are normalised *before* intersection, which generally acts to increase this quantity (as demonstrated in Subsection 5.8.3). The same can be observed of the other meshes presented in Figure 7.11, which bear a strong likeness of the meshes in Figure 7.10, but have more elements and DoFs than their counterparts. Other qualitative observations of the above hold true for these meshes, too.

Convergence plots are presented in Figure 7.12. Under the first order error estimator alone, no error curves were consistently below 1% for function spaces with 2,000 DoFs and above. The combined WG metrics achieve this in every case. That the combined WG metrics perform so well demonstrates that the first order adjoint a priori error estimator (7.39) derived in this thesis includes information which enables a high quality QoI estimate. Whilst the ADWR and WH metrics do not yield comparably small errors using few DoFs, they do perform well on the whole across all four cases.



(A) 2^{nd} order isotropic DWR
(10,755 elements,
5,422 DoFs,
max. aspect ratio 1.7)

(B) Isotropic DWR
(11,413 elements,
5,754 DoFs,
max. aspect ratio 2.1)

(C) Anisotropic DWR
(15,626 elements,
7,838 DoFs,
max. aspect ratio 10.6)

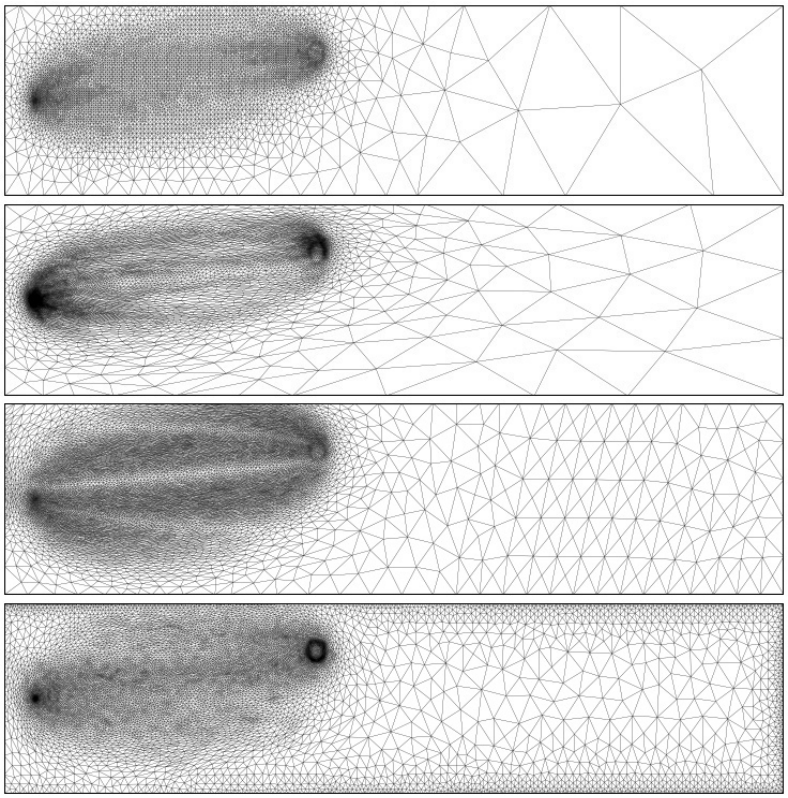
(D) Weighted Hessian
(13,788 elements,
6,957 DoFs,
max. aspect ratio 15.4)

(E) Weighted gradient
(17,422 elements,
8,955 DoFs,
max. aspect ratio 6.6)

FIGURE 7.10. Example meshes for metric-averaged goal-oriented mesh adaptation applied to the 'Point Discharge with Diffusion' test case with offset source and receiver. The anisotropic DWR metric is normalised using $\alpha = 2$; L^1 normalisation is used in all other cases.

In the aligned configuration, neither of the error curves corresponding to averaged isotropic metrics or averaged error indicators are particularly attractive, since the error does not decrease monotonically. In the offset configuration, averaging metrics yields consistently small errors using fewer DoFs. As such, it could be argued that useful information may be obtained by treating the second order DWR error estimate in terms of its two first order components. Intersecting first order isotropic metrics gives more favourable results than either averaging approach in the aligned configuration, with errors being consistently small over a broader range of DoF counts. However, this is not the case in the offset configuration, meaning we are not able to draw a clear conclusion of which isotropic approach is most effective.

Summary. In conclusion, we find that goal-oriented mesh adaptation enables convergence to the exact QoI values using (often significantly) fewer DoFs than required under uniform meshing, for this particular time-independent test case. An isotropic metric and three anisotropic metrics are validated, each of which enabling enhanced convergence properties, compared with uniform refinement. Moreover, metrics based on forward and



(A) Isotropic DWR (15,906 elements, 8,004 DoFs, max. aspect ratio 2.1)

(B) Anisotropic DWR (26,296 elements, 13,175 DoFs, max. aspect ratio 12.1)

(C) Weighted Hessian (20,875 elements, 10,501 DoFs, max. aspect ratio 14.4)

(D) Weighted gradient (25,715 elements, 13,216 DoFs, max. aspect ratio 6.6)

FIGURE 7.11. Example meshes for metric-intersected goal-oriented mesh adaptation applied to the 'Point Discharge with Diffusion' test case with offset source and receiver. The anisotropic DWR metric is normalised using $\alpha = 2$; L^1 normalisation is used in all other cases.

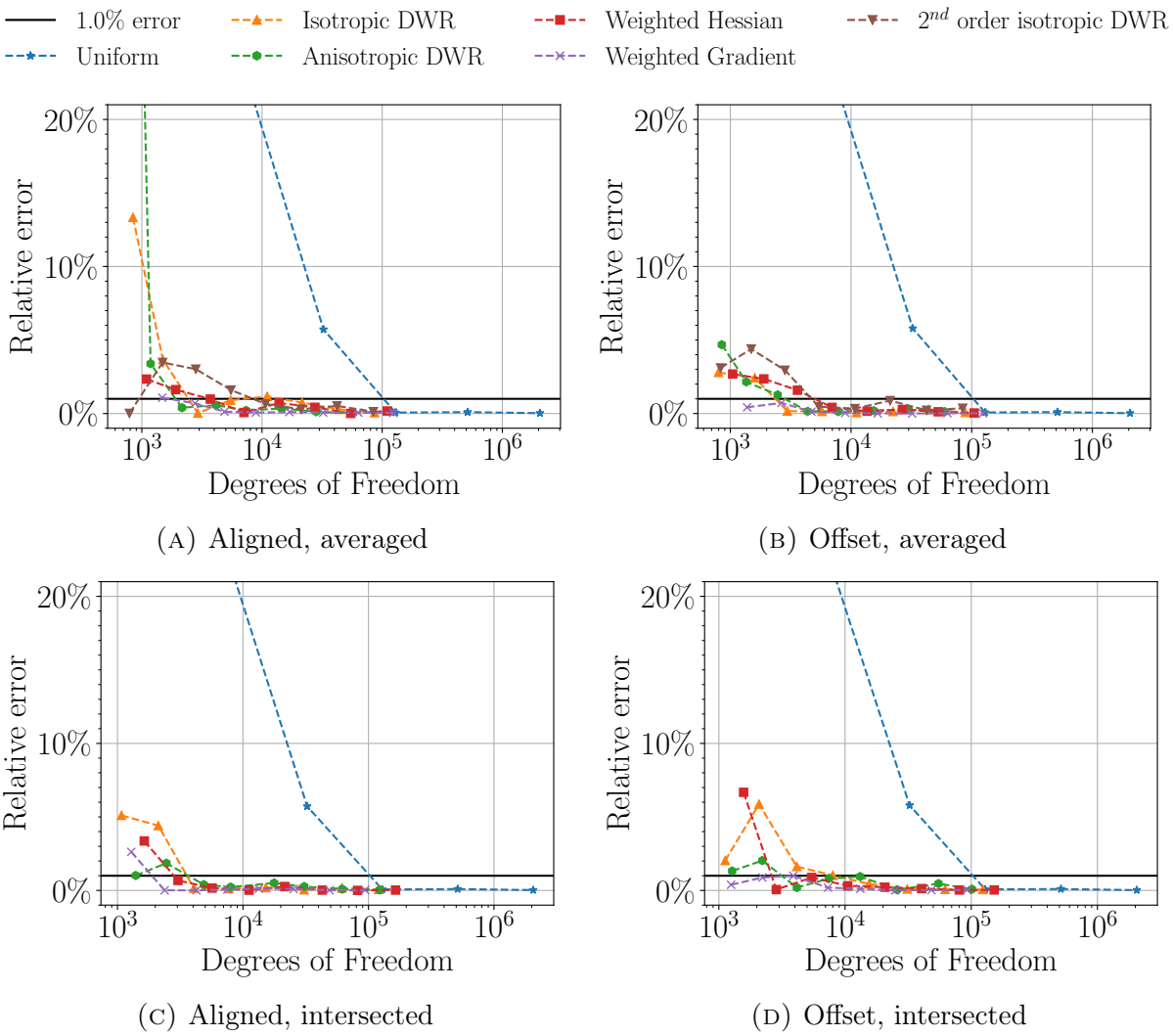
adjoint DWR error results (7.9) and (7.10) may be combined in order to obtain further enhancements, albeit with heightened computational cost. We find that metric intersection generally leads to more favourable convergence properties than averaging.

It has already been concluded that it is not possible to say which first order metric is best suited to the test case considered here, since they all perform very well. The experiments focused on second order metrics, on the other hand, suggest that the weighted gradient metric offers the best convergence properties, in that it achieves consistently small QoI error, from as few as $2,000$ elements.

7.5.3. Point Discharge with Diffusion in 3D. *This subsection contains experimental results which were published in* [**Wallwork et al., 2020a**].

Whilst the test case considered in Subsection 7.5.2 fulfils its purpose as a validation experiment for the goal-oriented mesh adaptation framework, the two dimensional approximation is insufficient for many realistic CFD applications. For instance, in the desalination outflow modelling context, the vertical component is essential for accurately capturing the small scales associated with the inflow and outflow pipes, which are orders of magnitude smaller than the dominant dynamical processes and spatial domain size.

In this subsection, we extend the 'Point Discharge with Diffusion' test case to three dimensions by making the following modifications: domain $\Omega = [0, 50] \times [0, 10] \times [0, 10]$, fluid velocity $\mathbf{u} = (1, 0, 0)$, point source location $\mathbf{x}_0 = (2, 5, 5)$ and region of interest



(A) Aligned, averaged

(B) Offset, averaged

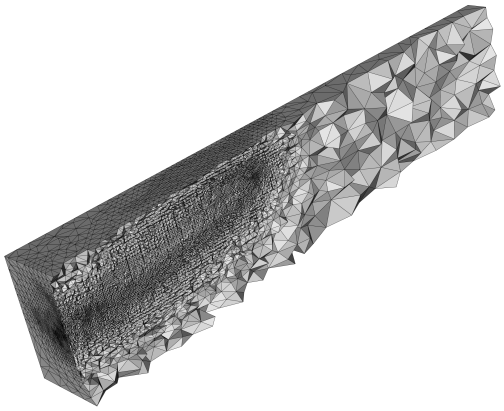
(C) Aligned, intersected

(D) Offset, intersected

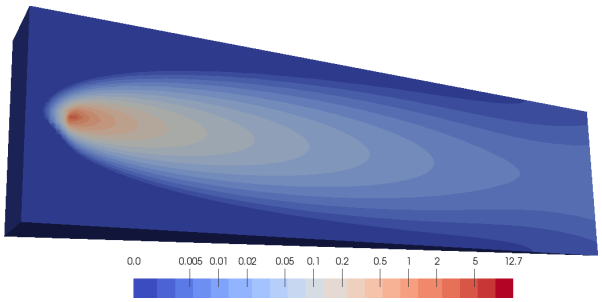
FIGURE 7.12. QoI convergence plots for the 'Point Discharge with Diffusion' test case, for four different goal-oriented metrics constructed by combining metrics due to first order error estimates. The anisotropic DWR metric is normalised using $\alpha = 2$; L^1 normalisation is used in all other cases.

centred at $(20, 7.5, 7.5)$. Unlike in the other test cases considered in this thesis, the tracer concentration is *not* depth-averaged. Figure 7.13 compares the numerical solution of this problem using an SUPG stabilised $\mathbb{P}1$ method defined on both a 1,920,000 element uniform mesh and an anisotropic goal-oriented mesh generated by averaging forward and adjoint weighted Hessian metrics.

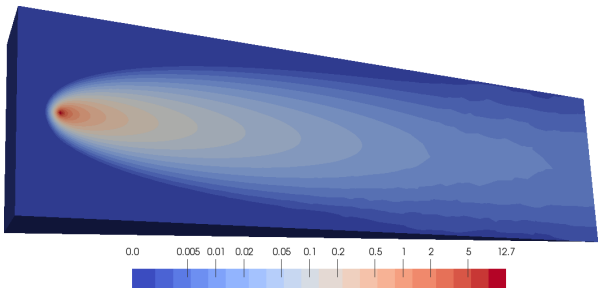
The goal-oriented mesh presented in Subfigure 7.13a is remarkably multi-scale, with cell volumes ranging from as small as 1.25×10^{-10} cubic units near to the source and receiver to as large as 4.81 cubic units downstream of the receiver. That is, the mesh element sizes cover *ten cubic orders of magnitude*, which corresponds to variations of factors as much as 1,000 in each spatial dimension. For the domain considered here, a uniform mesh with cell volumes 1.25×10^{-10} would require 4×10^{13} mesh elements, which is clearly prohibitively large. This simple calculation reveals why a multi-scale approach is essential for realistic 3D applications where the region of interest is much smaller than the domain size.



(A) Clip of a mesh with 1,776,396 elements resulting from goal-oriented mesh adaptation.



(B) Finite element solution on a uniform mesh with 1,920,000 elements.



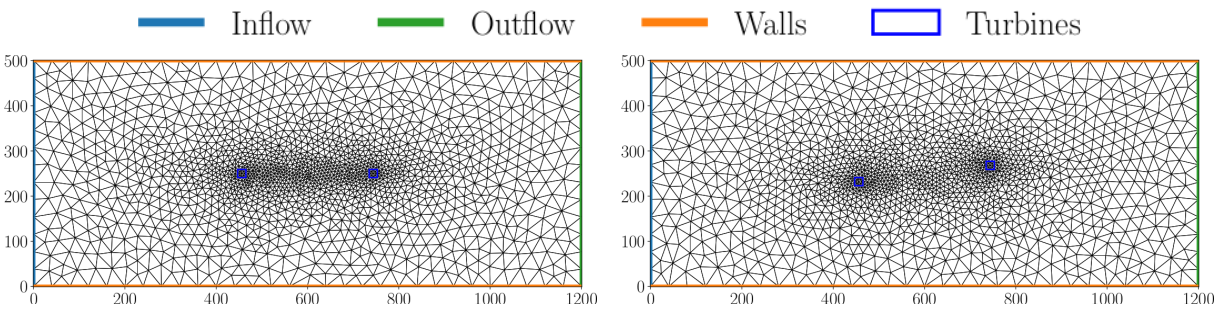
(C) Finite element solution on the goal-oriented mesh shown in Subfigure 7.13a.

FIGURE 7.13. Slices of the solution to the 3D extension of the 'Point Discharge with Diffusion' test case discussed in Subsection 7.5.2, presented on a uniform mesh and a mesh generated using averaging of forward and adjoint weighted Hessian metrics. Slices were taken on the plane intersecting source and receiver which is orthogonal to the z-axis. As presented in [**Wallwork et al., 2020a**].

The isosurface plots shown in Subfigures 7.13b–7.13c reveals some interesting aspects of the goal-oriented approach. Firstly, we observe that the source term is well resolved, due to the high levels of resolution deployed in its vicinity. This is clearly not the case for the uniform mesh, where numerical diffusion appears to have smoothed out the source term somewhat. Secondly, whilst the uniform mesh yields a relatively smooth solution across the entire domain, grid-scale features can be observed downstream of the receiver region for the goal-oriented mesh. As noted in Subsection 7.5.2, this is permitted because the QoI is independent of the downstream dynamics in this advection-dominated problem. In conclusion, the goal-oriented approach allows us to more accurately capture what we deem to be the important aspects of the fluid dynamics, whilst using fewer elements than found in the uniform mesh.

7.5.4. Tidal Array. *This subsection contains experimental results which were published in* [**Wallwork et al., 2020b**]. *After the paper was published, a minor bug was detected; the results presented here are therefore different than those presented in the paper. Whilst these updated results are slightly different, the conclusions remain the same.*

The steady-state shallow water problem described in this subsection is solved using a $\mathbb{P}1_{DG} - \mathbb{P}2$ mixed discontinuous/continuous SIPG method with Lax-Friedrichs stabilisation. The linear systems arising during the numerical solution of this nonlinear equation are solved using a direct method.



(A) Aligned (3,328 elements, 1,708 vertices, max. aspect ratio 1.8)

(B) Offset (3,170 elements, 1,629 vertices, max. aspect ratio 1.7)

FIGURE 7.14. Base meshes used for the aligned and offset configurations of the 'Tidal Array' test case. Modified based on plots presented in [**Wallwork et al., 2020b**].

Consider a simple two turbine tidal farm, $\mathcal{F} = \{T_1, T_2\}$, in an idealised rectangular channel domain, $\Omega = [0, 1200] \times [0, 500]$, with units of metres. Two different configurations are assessed for the farm: one where T_1 is directly upstream of T_2 and another where T_1 and T_2 are offset by one turbine diameter to South and North, respectively. Given turbines of diameter 18 m, the former 'aligned' configuration has turbines centred at $\{(456, 250), (744, 250)\}$, whereas the latter 'offset' configuration has turbines centred at $\{(456, 232), (744, 218)\}$.

As considered in Subsection 2.1.2, the QoI is given by a proxy for the total power output of the array. In the steady state case, we have (as in (2.11)),

$$J(\mathbf{u}, \eta) := \int_\Omega \rho_{\text{sea}}\, C_t \|\mathbf{u}\|^3 \,\mathrm{d}x, \tag{7.76}$$

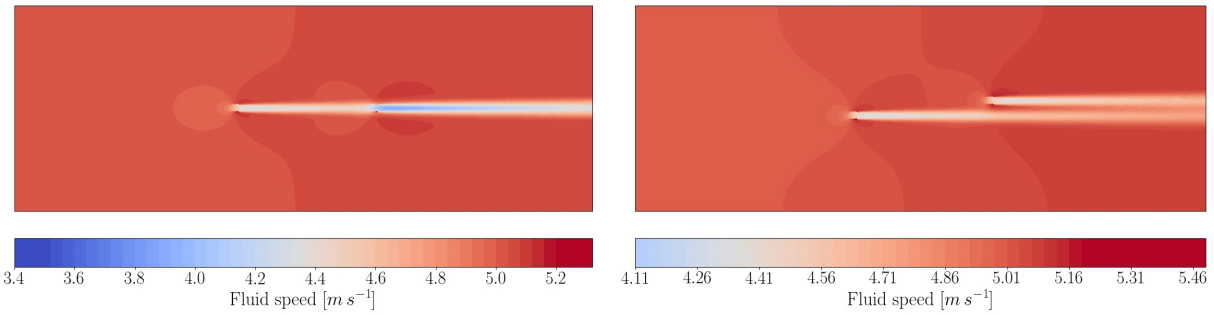
which has units of Watts. Here the turbine drag coefficient C_t is as given in (2.7) and depends on the configuration of the farm. Whilst realistic tidal turbine applications are inherently time-dependent, it is useful to consider steady-state test cases in order to understand the impact of the array configuration on its power output. The total power output (7.76) in the two array configurations are denoted by J_0 and J_1, with the subscript indicating the South/North offset in terms of turbine diameters.

We consider a flat bathymetry $b = 40\,\text{m}$ and flow driven by an inflow condition, $\mathbf{u} = (5, 0)\,\text{m}\,\text{s}^{-1}$, on the Western boundary. These depths and fluid speeds are representative of the Pentland Firth, Scotland – one of the UK's greatest tidal resources [**du Feu et al., 2019**]. A uniform isotropic viscosity $\nu = 0.5\,\text{m}^2\,\text{s}^{-1}$ is specified, resulting in a moderately advection-dominated problem. Free-slip conditions are imposed on the channel walls and an outflow condition $\eta = 0\,\text{m}$ on the Eastern boundary closes the system.

The nonlinear system is linearised about the inflow velocity and zero elevation.

Base meshes taking account of the turbines are generated using *gmsh* [**Geuzaine and Remacle, 2009**], where the mesh resolution inside the turbine footprint is higher than in the rest of the domain. These meshes are displayed in Figure 7.14. As for the 'Point Discharge with Diffusion' test case, a mesh hierarchy is obtained by iso-$\mathbb{P}2$ refinement.

Figure 7.15 shows the magnitude of the velocity (i.e. the speed) given by solving the test case on the third mesh of this hierarchy. Examining Subfigure 7.15a, we observe that the



(A) Aligned (B) Offset

FIGURE 7.15. Fluid speed [$\mathrm{m\,s^{-1}}$] as computed on meshes generated by two uniform refinements of the meshes shown in Subfigures 7.14a–7.14b. Modified based on plots presented in [**Wallwork et al., 2020b**].

momentum deficit at T_2 is more significant than at T_1 when the turbines are aligned. This occurs because the wake of T_1 has not fully recovered at T_2 and so this turbine overall experiences slower flow than T_1. As a consequence, turbine T_2 generates less power. In the offset case displayed in Subfigure 7.15b, the momentum deficit is similar at T_1 and T_2. This suggests that the presence of the first turbine is less impactful upon the flow speed meeting T_2 than in the aligned case. As a result, the total power output of the offset array configuration is higher than in the aligned configuration.

QoI Convergence on Uniform Meshes. Table 7.1 illustrates the convergence of QoI values under iso-ℙ2 refinement. The values displayed on the final row of Table 7.1 exhibit convergence to four significant figures and therefore present benchmark values for later comparison. In agreement with the discussion above, the offset configuration achieves a 15% higher total power output than the aligned configuration, illustrating how turbine positioning is an important factor in tidal farm design.

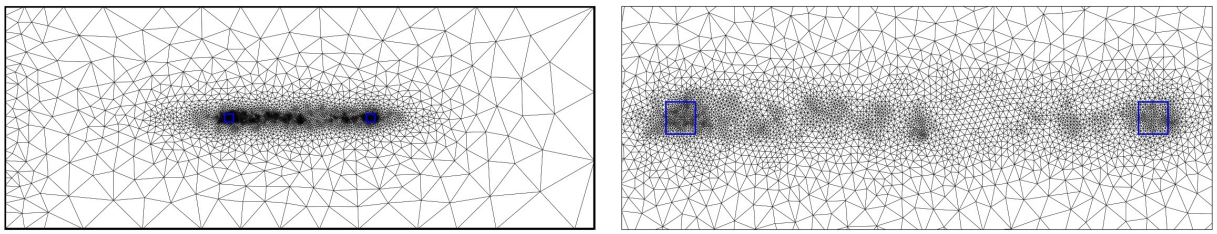
Aligned			Offset		
Elements	DoFs	J_0	Elements	DoFs	J_1
3,328	26,711	22.0185 MW	3,170	25,447	25.2703 MW
13,312	106,669	21.8757 MW	12,680	101,613	25.1891 MW
53,248	426,329	21.8270 MW	50,720	406,105	25.1492 MW
212,992	1,704,625	21.8127 MW	202,880	1,623,729	25.1359 MW
851,968	6,817,121	21.8088 MW	811,520	6,493,537	25.1322 MW

TABLE 7.1. Convergence of QoIs J_0 and J_1 evaluated at finite element solutions on a hierarchy of meshes generated by uniform refinement of the initial mesh. Modified based on the table presented in [**Wallwork et al., 2020b**].

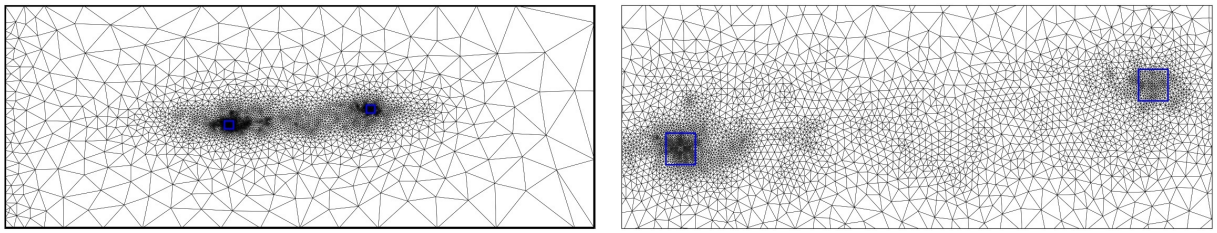
Goal-Oriented Mesh Adaptation. In this subsection, we apply the isotropic and anisotropic metric-based mesh adaptation methods described in Subsection 7.4.2. The goal-oriented error indicators used within the metrics are those derived in Subsection 7.2.2.

Figure 7.16 compares sample isotropic and anisotropic meshes, for both aligned and offset array configurations. Enrichment was performed using global h-refinement, with the forward solution prolonged into the enriched space, rather than the forward problem being solved. Each mesh deploys significant resolution surrounding the turbines – in

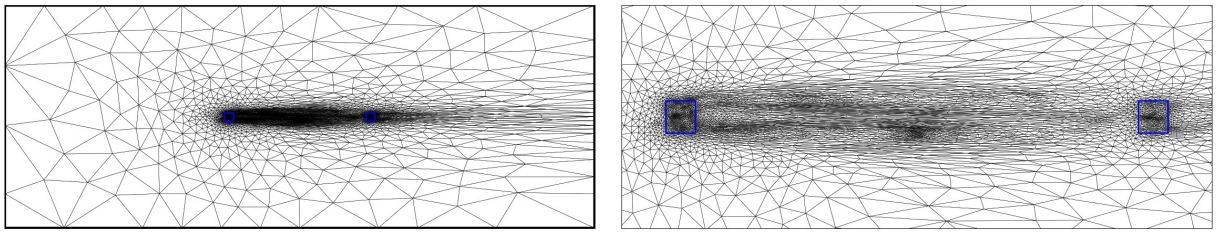
particular the first one. This is required for an accurate QoI approximation. The QoI may be approximated accurately without deploying significant resolution downstream of the second turbine, since it is independent of the dynamics in that region for an advection-dominated problem. Accordingly, each mesh is coarse in the region downstream of the second turbine. In the 'farm region' surrounding and inbetween the turbines, the mesh resolution is higher, meaning that the local dynamics will be better resolved than those downstream. The meshes presented in Subfigures 7.16c–7.16d are clearly more anisotropic than those in Subfigures 7.16a–7.16b, as we should expect, and the maximum aspect ratios stated are an order of magnitude larger.



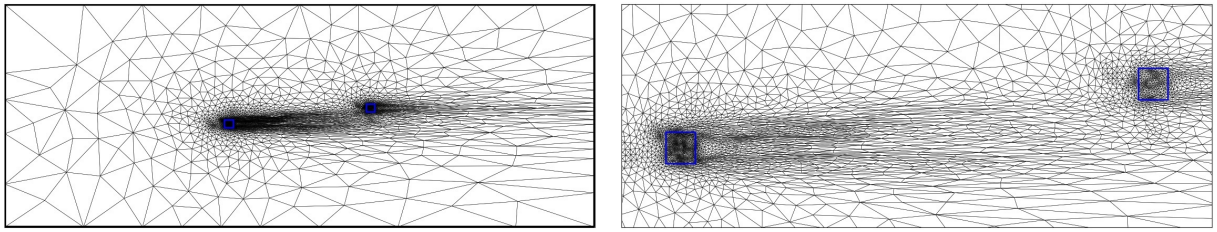
(A) Aligned, isotropic (9,385 elements, 4,720 vertices, max. aspect ratio 2.6)



(B) Offset, isotropic (9,155 elements, 4,608 vertices, max. aspect ratio 2.8)



(C) Aligned, anisotropic (7,192 elements, 3,619 vertices, max. aspect ratio 33.2)

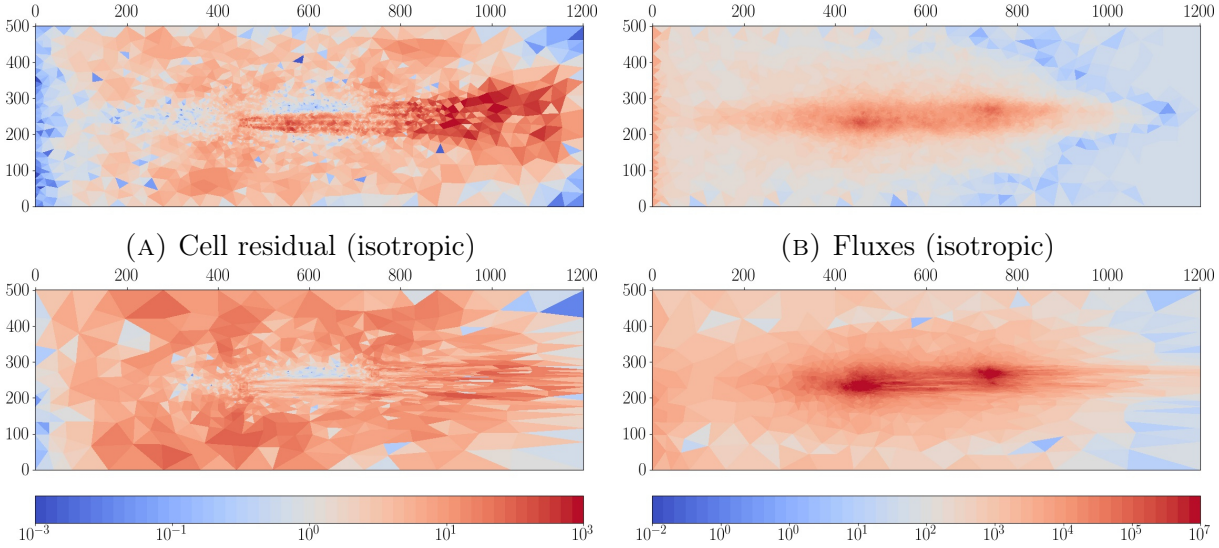


(D) Offset, anisotropic (6,801 elements, 3,424 vertices, max. aspect ratio 19.7)

FIGURE 7.16. Example meshes generated using goal-oriented mesh adaptation, including zooms on the farm region. Subfigures 7.16a and 7.16c correspond to the aligned configuration, whilst Subfigures 7.16b and 7.16d correspond to the offset configuration. Modified based on plots presented in [**Wallwork et al., 2020b**].

Components of the DWR error indicator due to cell residual and flux contributions are shown in Figure 7.17. Given the colour scales, the contributions from fluxes are much

more significant than those from cell residuals. However, the cell residual is more significant in the isotropic case than the anisotropic case, largely because the latter has higher mesh resolution downstream. Flux terms, on the other hand, are more significant in the anisotropic case, especially in the farm region. Errors due to the inflow condition are apparent for the flux terms on the isotropic mesh, which manifests in heightened resolution on the left hand boundary.



(A) Cell residual (isotropic)

(B) Fluxes (isotropic)

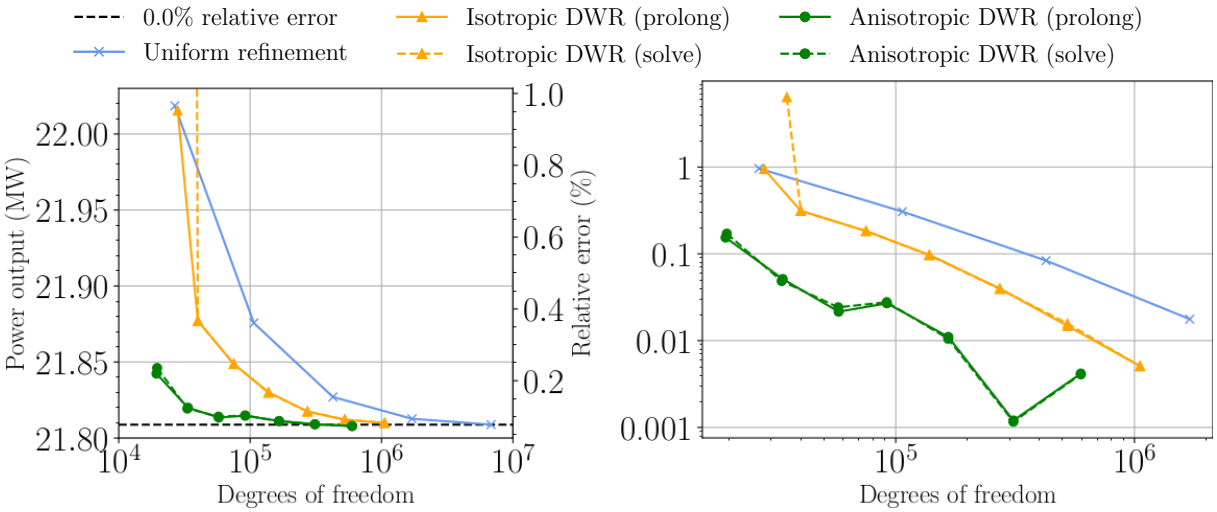
(C) Cell residual (anisotropic)

(D) Fluxes (anisotropic)

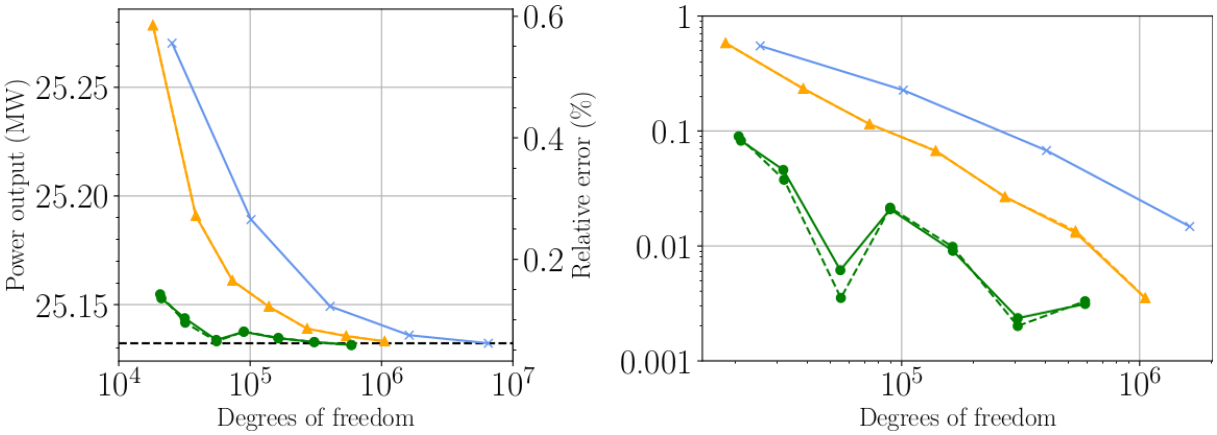
FIGURE 7.17. Cell residual and flux error indicators for the 'Tidal Array' test case on meshes generated using goal-oriented mesh adaptation. All plots are in the offset configuration. Modified based on plots presented in [**Wallwork et al., 2020b**].

Figure 7.18 illustrates the convergence of power output on meshes generated from isotropic and anisotropic goal-oriented metrics with increasing complexity. For each adaptation strategy and array configuration, we observe convergence of the power output to the benchmark values established in Table 7.1. For a given DoF count, the goal-oriented adaptive strategies yield estimates with consistently smaller error than uniform refinement, except for the first isotropic metric data point in the aligned configuration. Further, observe that the anisotropic approach consistently out-performs the isotropic approach in this regard. Whilst the convergence of QoI error with increasing DoFs is monotonic under uniform refinement and the isotropic metric, this is not the case for the anisotropic metric.

Regarding the GE_h enrichment method, two cases are considered for the enriched adjoint problem definition, as discussed. The naïve method is to first solve the forward problem in the enriched space and then form the adjoint problem using a linearisation about its solution (i.e. 'solve'). The standard method used in the literature (see [**Rognes and Logg, 2013**] and [**Roth et al., 2021**], for example) is to simply linearise about the prolongation of the forward solution from the base space (i.e. 'prolong'). Given that the nonlinear iteration typically takes four iterations to converge, the overall cost of the former method is effectively five times that of the latter (since the cost of prolongation is negligible in comparison to performing a nonlinear solve). However, as demonstrated in Figure 7.18, there is very little gained in terms of accuracy by taking the more expensive route, if anything at all. In fact, there are data points where the latter method happens to



(A) Aligned configuration



(B) Offset configuration

FIGURE 7.18. QoI convergence analysis under uniform refinement, isotropic adaptation and anisotropic adaptation for (a) the aligned and (b) the offset array configurations. Modified based on plots presented in [**Wallwork et al., 2020b**].

be more accurate. In conclusion, we validate the prolongation approach for this particular test case.

The non-monotonic convergence observed in both this and the previous test case is likely due to stopping criteria not being sufficiently tight. Whilst relative convergence of the QoI value, mesh element count and error estimator are all checked, the adaptation loop breaks if just one is satisfied. This could lead the algorithm to accept a mesh which gives rise to a substantially different QoI value, but happens to have a similar number of elements as the previous iteration. In future work, it would be better practice to assert that all three quantities converge.

7.6. Time-Dependent Goal-Oriented Mesh Adaptation

The literature contains many applications of goal-oriented mesh adaptation to the solution of steady-state PDEs, wherein the computational complexity requirements for accurately

estimating the QoI have been demonstrated to be lower than with other other approaches (see previous section). Whilst goal-oriented mesh adaptation requires multiple solves of the forward and adjoint problems (and in some cases also problems solved in enriched finite element spaces), these systems may usually be formulated as a single linear or nonlinear system. This means that the computational overheads associated with steady-state goal-oriented mesh adaptation are generally not cumbersome, especially if convergence of the fixed point iteration loop is achieved in a small number of iterations.

The computational overheads associated with a single forward or adjoint solve in a time-dependent problem are typically much more significant. In addition, the reversed causality in the adjoint problem and the fact that it depends on the forward solution at each time level (for nonlinear problems) means that it is not possible to apply goal-oriented mesh adaptation in an 'on-the-fly' fashion. The forward, adjoint and any enriched versions thereof must be solved over the whole time interval before goal-oriented error estimation can be performed. As such, the development of general goal-oriented mesh adaptation algorithms for time-dependent problems which ensure both minimal QoI error and low computational complexity remains an open problem. The work of [**Belme et al., 2012**] presents an effective approach for the metric-based case, which is adopted here, albeit with a different choice of metric.

In this thesis we do not account for error due to time discretisation. Instead, we focus on errors arising from the discretisation of the spatial domain and in particular those spatial discretisation errors due to the choice of mesh.

Assuming some time integration scheme, a simple approach is to simply apply the steady-state approach as stated in Algorithm 3, with error indicators evaluated as integrals over single cells and the entire time period (i.e. $K \times (0, T]$). That is, we seek to obtain a single mesh which is to be used used over the whole time period. This approach is advantageous in that the mesh is only adapted once per iteration. Similarly, mesh-to-mesh interpolations are not required at all, meaning interpolation error is only introduced when interpolating analytic functions and data into finite element spaces (which is unavoidable) and in a single projection from $\mathbb{P}0$ to $\mathbb{P}1$. However, a major disadvantage is that evaluating error indicators as integrals over all time has the effect of 'smoothing out' information which changes from timestep to timestep. This is particularly problematic if there are flow features which move rapidly across the domain, since their significance at any particular location and time is not given the precedence it deserves. Shockwaves and tsunamis are examples of inherently time-dependent flow features whose spatial location is extremely important.

A similar simple approach also reapplies Algorithm 3, but evaluates error indicators as integrals over single cells and timesteps (i.e. $K \times (t^{(k)}, t^{(k+1)}]$). Metrics are computed *at each timestep* and then normalised using space-time normalisation. Whilst the computational cost and interpolation error associated with such an approach are likely higher than the one described above, it may be readily extended to a case with multiple meshes, as described in the following subsection. By evaluating error indicators on a timestep-by-timestep basis, the evolution of error in time is accounted for, enabling the spatial mesh size to change during the simulation, as required. Nonetheless, smoothing still occurs if the metrics are accumulated by time integration (i.e. L^1 normalisation). Metric intersection could also be applied for the time accumulation, which reduces the extent to which features are smoothed. However, the size of a mesh constructed by intersecting

a large number of metrics is unpredictable and likely very high, implying a significant computational cost.

7.6.1. Adaptation on Time Subintervals. Recall the notation introduced in Subsection 5.7.2, where $\{\mathcal{W}_i\}_{i=1}^k$ provides a partition of the temporal domain $(0,T]$ into subintervals. As described in that section, we seek to optimise the mesh used on each of these k subintervals. The difference here is that the mesh adaptation is driven by goal-oriented error estimation and, as such, the optimisation of mesh $\mathcal{H}_i$ requires the solution of both forward and adjoint equations on subinterval $\mathcal{W}_i$.

Algorithm 4 provides one method for generating a sequence of goal-oriented meshes, for a target space-time metric complexity. It may also be straightforwardly applied in the case of a target interpolation error. It is based on the algorithm presented on p.6338 of [**Belme et al., 2012**].

Given target space-time complexity $\mathcal{C}_T > 0$;
Given initial meshes $\{\mathcal{H}_0^j\}_{j=1}^k$;
Set $i := 0$;
while *not converged* **do**
  for $j = 1, \dots, k$ **do**
    Interpolate fields onto $\mathcal{H}_i^j$ from formulae/data/previous mesh;
    Solve forward problem over subinterval $\mathcal{W}^j$ on $\mathcal{H}_i^j$;
    Store forward solution from the final timestep of $\mathcal{W}^j$;
  end
  Evaluate the QoI and check for its convergence;
  for $j = k, \dots, 1$ **do**
    Load forward solution for first timestep of $\mathcal{W}^j$;
    Solve forward problem over subinterval $\mathcal{W}^j$ on $\mathcal{H}_i^j$;
    Solve adjoint problem over subinterval $\mathcal{W}^j$ on $\mathcal{H}_i^j$;
    Construct metric $\mathcal{M}_i^j$ over subinterval $\mathcal{W}^j$;
  end
  Normalise $\{\mathcal{M}_i^j\}_{j=1}^k$ based on target complexity $\mathcal{C}_T$;
  for $j = 1, \dots, k$ **do**
    Apply gradation to $\mathcal{M}_i^j$;
    Adapt mesh $\mathcal{H}_i^j$ using $\mathcal{M}_i^j$ to obtain $\mathcal{H}_{i+1}^j$;
  end
  Increment i;
end

Algorithm 4: Mesh adaptation routine for time-dependent problems, as presented in [**Wallwork et al., 2021**].

The number and length of subintervals to be used is chosen before application of Algorithm 4, for simplicity. Under explicit timestepping methods, the algorithm is more complicated. This is because the timestep restriction due to the CFL criterion may force a change of timestep at different time levels. A fixed point algorithm which accounts for explicit timestep restrictions in the case of 'weighted gradient' type metrics is presented in [**Belme et al., 2012**].

Another simplification of the method as presented is that static metrics are constructed at every timestep and then accumulated over each subinterval. This can be expensive in practice, especially if the metric requires Hessian recovery. As such, a more efficient implementation only computes metrics at a subset of timesteps and the accumulation approach is modified accordingly. For example, under L^1 time normalisation, a longer step length should be used in the quadrature scheme.

7.7. Desalination Plant Outfall Modelling

This section contains experimental results as presented in [**Wallwork et al., 2021**].

As a first demonstration of time-dependent goal-oriented metric-based mesh adaptation, consider the idealised desalination plant outfall scenario introduced in [**Wallwork et al., 2021**]. The problem consists of a channel domain $\Omega = [-1500, 1500] \times [-500, -500]$ with units of metres and free-slip conditions on the North and South boundaries, $\partial\Omega_N$. Idealised tides are generated by simple sinusoidal forcings of the water depth on the East and West boundaries, $\partial\Omega_D$. On those boundaries, a constant background salinity of $c_b = 39\,\mathrm{g\,l^{-1}}$ is imposed. This value is also used as an initial condition for salinity.

For given bathymetry, viscosity and drag fields, the hydrodynamics are driven by tidal forcings of the free surface elevation. Typically, a hydrodynamics model – for example, shallow water – is used to give the fluid velocity, which, in turn, drives the tracer transport. Since we consider a flat bathymetry and constant viscosity and drag parameters, the hydrodynamics may be approximated by a simple sinusoidal forcing of the *velocity*:

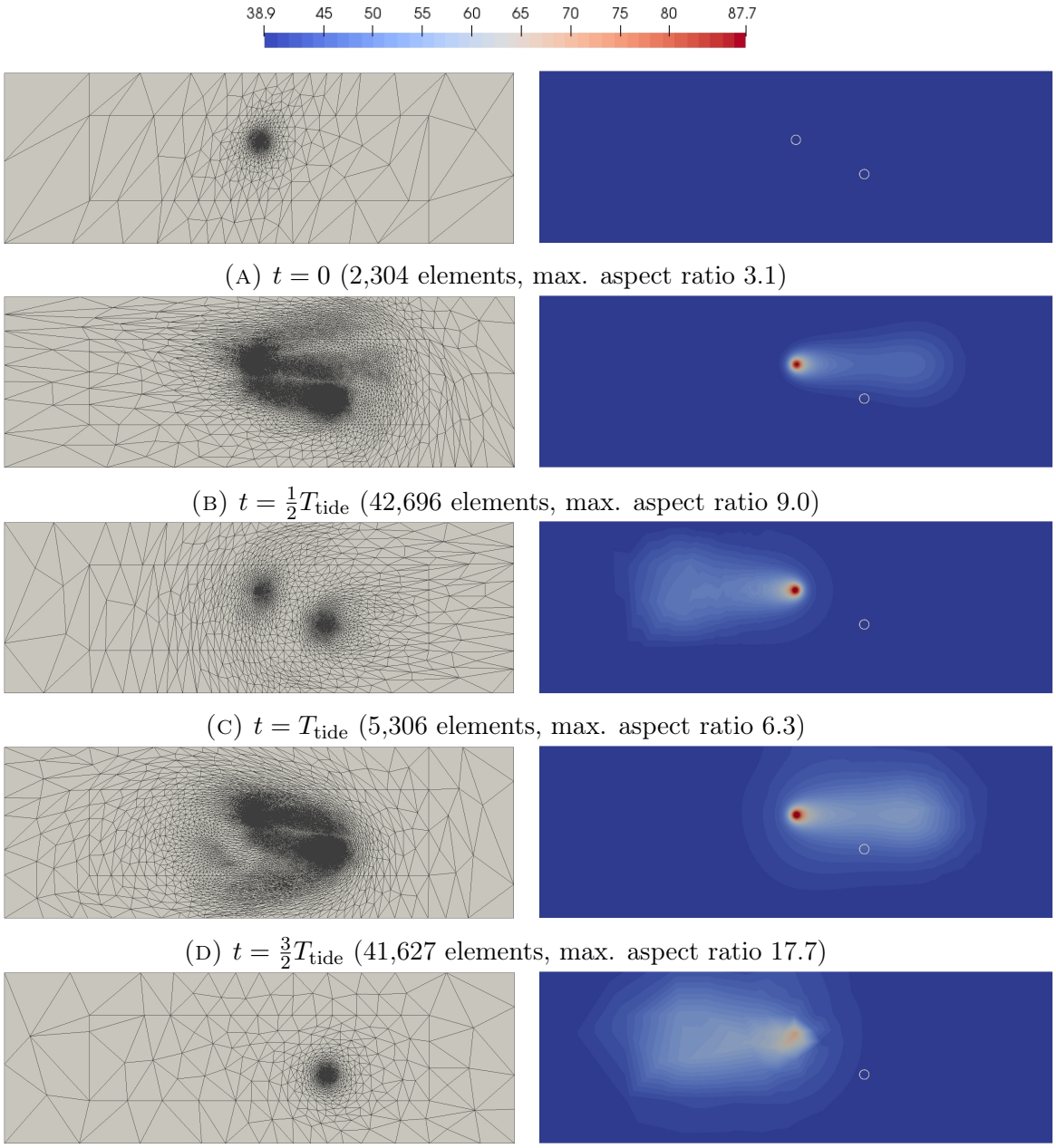
$$\mathbf{u}(\mathbf{x}, t) = (U \sin(\omega t), 0), \qquad \omega = \frac{2\pi}{T_{\text{tide}}}, \quad U = 1.15\,\mathrm{m\,s^{-1}}. \tag{7.77}$$

Doing so implies a massive reduction in the computational complexity of the model, meaning we need only solve the (linear) advection-diffusion equation at each timestep. In order to speed up the tidal cycles twenty-fold, we use a period, T_{tide}, which is just 5% of the M2 tidal constituent. Two tidal periods are to be simulated: $T = 2T_{\text{tide}} = 4464.0\,\mathrm{s}$.

Suppose the inlet and outlet pipes are of radius $r_{\text{in}} = r_{\text{out}} = 25\,\mathrm{m}$ and are positioned at $\mathbf{x}_{\text{in}} = (400, -100)$ and $\mathbf{x}_{\text{out}} = (0, 100)$, respectively. That is, we have a source term in the circular region, $R_{\text{out}} := B_{r_{\text{out}}}(\mathbf{x}_{\text{out}})$, and a receiver region, $R_{\text{in}} := B_{r_{\text{in}}}(\mathbf{x}_{\text{in}})$. A discharge rate of 2 units per second is used for the source. The difference between the modelled salinity at the inlet pipe (i.e. receiver) and the background value is integrated over time to yield the QoI

$$J(c) := \int_0^{2T_{\text{tide}}} \int_\Omega \mathbb{1}_{R_{\text{in}}}(c - c_b)\,\mathrm{d}x\,\mathrm{d}t = \int_0^{2T_{\text{tide}}} \int_{R_{\text{in}}} (c - c_b)\,\mathrm{d}x\,\mathrm{d}t. \tag{7.78}$$

Consider a timestep $\Delta t = 2.232$ and 100 uniform subintervals, so that each mesh is used for 20 timesteps. In the first iteration, a uniform mesh is used for all subintervals. However, after convergence of the fixed point iteration, each mesh is adapted to the WH metric, accumulated over the 20 timestep subinterval. The WH metric is normalised in space and time using L^{10} normalisation. The continuous adjoint method is used, with the mesh-to-mesh data transfer performed using the same conservative projection operator in both directions. Meshes corresponding to a selection of subintervals are shown in Figure 7.19, along with snapshots of the salinity solution field.



(A) $t = 0$ (2,304 elements, max. aspect ratio 3.1)

(B) $t = \frac{1}{2}T_{\text{tide}}$ (42,696 elements, max. aspect ratio 9.0)

(C) $t = T_{\text{tide}}$ (5,306 elements, max. aspect ratio 6.3)

(D) $t = \frac{3}{2}T_{\text{tide}}$ (41,627 elements, max. aspect ratio 17.7)

(E) $t = 2T_{\text{tide}}$ (2,409 elements, max. aspect ratio 3.2)

FIGURE 7.19. Example meshes for the application of the L^{10} normalised weighted Hessian metric to the idealised desalination problem. The two circles in Subfigure 7.19a indicate the locations of the inlet and outlet pipes, with radii to scale. The single circle in each of the other subfigures indicates the inlet pipe. As presented in [**Wallwork et al., 2021**]

The background salinity is enforced as the initial condition at $t = 0$. The increased resolution near the outlet pipe at $t = 0$ is due to information from 'future' timesteps, over the first subinterval. A similar pattern is apparent at the terminal time $t = 2T_{\text{tide}}$, where the salinity is only resolved near the inlet. In this advection-dominated problem, the adjoint solution is near zero away from the inlet because any saline water there does not have sufficient time to reach the inlet. As such, the mesh is coarse away from the inlet and the salinity field experiences significant artificial diffusion. This phenomenon is also apparent at $t = T_{\text{tide}}$, where the flow is in the opposite direction to the inlet.

Isolines of the salinity are fairly smooth between outlet and inlet in the other snapshots at times $t = \frac{1}{2}T_{\text{tide}}$ and $t = \frac{3}{2}T_{\text{tide}}$, with some numerical diffusion downstream of the inlet in Subfigure 7.19d. Note that the meshes at $t = \frac{1}{2}T_{\text{tide}}$ and $t = \frac{3}{2}T_{\text{tide}}$ have an order of magnitude more elements than the others shown, showing how metric-based goal-oriented mesh adaptation with L^1 normalisation allows for a discretisation which is multi-scale in both space and time.

Note that the meshes at $t = \frac{1}{2}T_{\text{tide}}$ and $t = \frac{3}{2}T_{\text{tide}}$ also have stronger anisotropy than the other times. For the WH metric, this arises because both the strong residual and adjoint solution Hessian have significant magnitudes.

7.8. Tsunami Hazard Assessment

This section contains experimental results which build upon the subject of this author's MRes thesis, "Multi-scale numerical simulation of a tsunami using mesh adaptive methods". *The results presented below are sufficiently different to be included here because: (a) the original work applied metric-based mesh adaptation in the 'on-the-fly' sense, rather than using a fixed-point iteration; (b) it completely neglected flux terms in the DWR error estimator; and (c) the model comparison was in terms of ℓ^p errors related to gauge timeseries, as opposed to QoI error.*

Section 7.7 considered goal-oriented mesh adaptation applied to a time-dependent tracer transport problem. In this final section, we instead apply it to a time-dependent shallow water problem, namely that of making an accurate tsunami hazard assessment.

7.8.1. Quantity of Interest. As discussed in Subsection 1.1.3, there are many different QoIs which may be associated with assessing a tsunami hazard. These likely differ depending on the stakeholder making the decision. For the manager of an oil rig, the QoI could be forces exerted on the structure or some quantification of waves over a certain height. For policy makers making decisions on a national scale, it might make more sense to consider flux integrals across the coast or coastal inundation. As with oil rig managers, those in charge of pieces of coastal infrastructure are likely more concerned with localised impacts. For the purposes of this subsection, suppose that the nuclear power plant at Fukushima Daiichi is of primary concern. As discussed in Chapter 4, the 2011 Tōhoku tsunami led to the meltdown of this particular plant, bringing about a mass evacuation, significant financial costs and a multi-year cleanup operation.

For models with wetting-and-drying capability, one possible QoI could be the inundation of the power plant footprint, D – which is initially dry – over a time window $(T_{\text{start}}, T_{\text{end}}] \subseteq (0, T]$:

$$J^{\text{inundation}}(\mathbf{u}, \eta) := \int_{T_{\text{start}}}^{T_{\text{end}}} \int_D \widetilde{H} \,\mathrm{d}x \,\mathrm{d}t, \tag{7.79}$$

where $\widetilde{H} > 0$ is the modified water depth (see [**Kärnä et al., 2011**] for the definition of this quantity). Since the model setup in this thesis does not use wetting-and-drying, we instead integrate free surface elevation over a circular region, $R \subset \Omega$, in the coastal ocean, centred on the plant:

$$J^{\text{coastal}}(\mathbf{u}, \eta) := \int_{T_{\text{start}}}^{T_{\text{end}}} \int_R \eta \,\mathrm{d}x \,\mathrm{d}t, \qquad R := B_r(\mathbf{x}_{\text{P}}) \cap \Omega, \tag{7.80}$$

where $r > 0$ is the radius and $\mathbf{x}_\text{P}$ is the plant location. This is a very simple means of assessing the tsunami hazard, but it does convey a sense of the water discharge entering R during the time window of interest. Tsunamis are impactful because of the sheer volume of the water they bring, as well as their wavespeed and wave height. In the integrated quantity (7.80) (with units $\text{m}^3\,\text{s}$) we choose to focus on water volume, for the purposes of a proof-of-concept simulation for the goal-oriented mesh adaptation application. More realistic means of quantifying the hazard will be addressed in future work.

As with the tracer transport examples considered in previous sections, (7.80) admits an L^2 Riesz representation given by an indicator function: $\mathcal{R}(J^\text{coastal}) = \mathbb{1}_{R\times(T_\text{start},T_\text{end}]}$. Figure 7.20 shows it (i.e. the region of interest) with radius $r = 100\,\text{km}$.

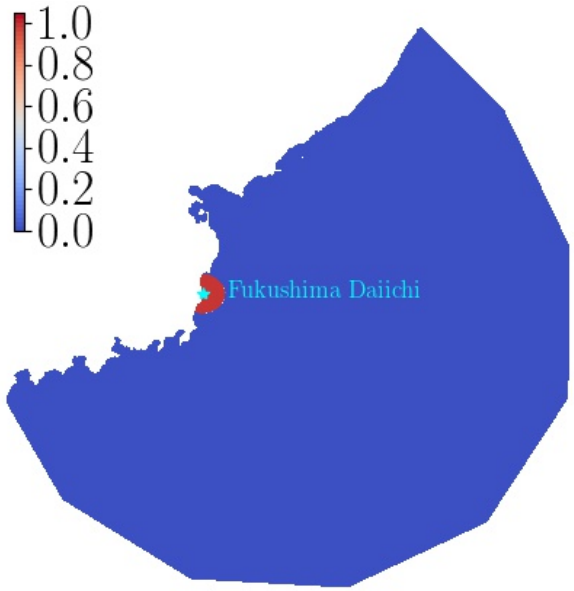
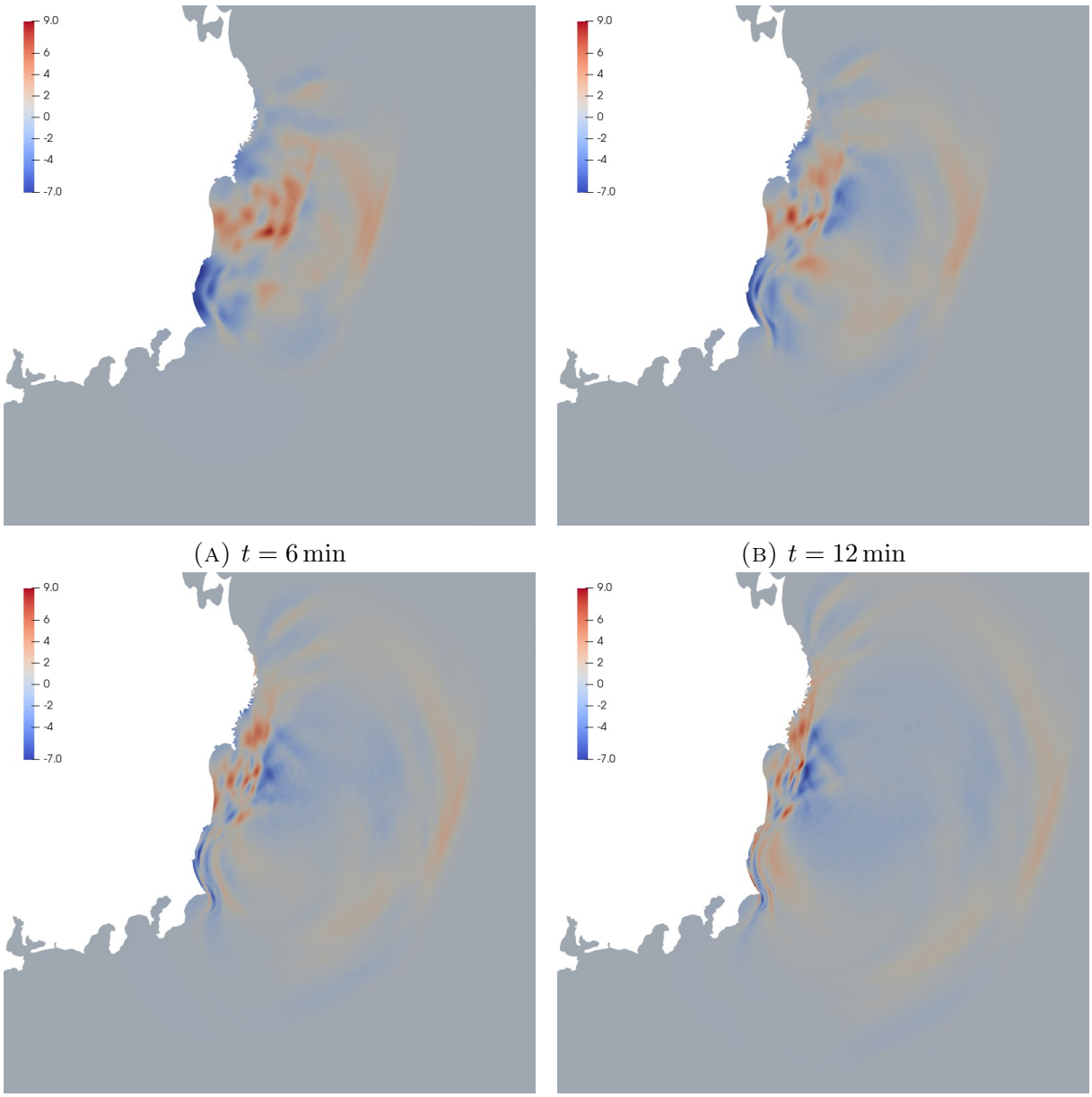


FIGURE 7.20. Region of interest centred on Fukushima Daiichi nuclear power plant. Presented in a $\mathbb{P}1$ space defined on mesh $\mathcal{H}_3$.

7.8.2. Tōhoku Tsunami. The problem setup is identical to that presented in Subsection 4.2.1, except that we only consider the first 24 minutes, by which time much of the tsunami wave reached the shore. This can be deduced from the snapshots presented in Figure 7.21. Beginning with the initial surface generated at the culmination of Section 4.5 (and zero initial velocity), the free surface perturbation causes two waves – one bound for the East coast of Japan and another bound for the open ocean. For the purposes of this experiment, we are only interested in the wave bound for the coast. The dynamics of the wave moving away from the coast do not affect the QoI, meaning we are not interested in resolving it. Shoaling is evident in the wave which approaches the coast, the central part of which becomes narrower and taller. For consistency with the inversion experiment in Subsection 4.5.3, a $\mathbb{P}1$ discretisation for the elevation is used, as part of a Taylor-Hood finite element pair.

Adaptive Simulation using DWP Indicators. Recall the DWP indicator introduced in Subsection 7.4.5, based on QoI (7.75), which evaluates the elevation in region R at the end time. In order to extend the mesh adaptation indicator to QoI (7.80), which involves a time period of interest, the point-wise ℓ^∞ norm of the spatial L^2 inner product (represented as a $\mathbb{P}1$ field) is taken over an appropriate subinterval of the temporal domain. The indicator was originally developed for hierarchical h-adaptation. As such, an element-based formulation was used, in order to do cell tagging (see [**Davis and LeVeque,**



(A) $t = 6\,\text{min}$ (B) $t = 12\,\text{min}$

(C) $t = 18\,\text{min}$ (D) $t = 24\,\text{min}$

FIGURE 7.21. Snapshots of the free surface elevation in a simulation of the Tōhoku tsunami which assumes the source obtained in Section 4.5. Presented in a $\mathbb{P}1$ space defined on mesh $\mathcal{H}_2$.

2016]). The version used here is defined at mesh vertices by

$$\rho^{DWP}(\mathbf{x},t) := \max_{\tau\in[t,\widehat{T}(t)]} |\mathbf{q}(\mathbf{x},t)\cdot\mathbf{q}^*(\mathbf{x},\tau)|\,, \qquad \widehat{T}(t) := \min\{T_{\text{end}}, t + T_{\text{end}} - T_{\text{start}}\}. \tag{7.81}$$

Given that T coincides with T_{end}, this effectively means that the maximum is taken over either the length of the time interval of interest, or until the end of the simulation – whichever comes first. Whilst the inner product is cheap to compute, the indicators can be expensive to evaluate if the time range contains many timesteps. This is especially true if multiple subintervals are used for the fixed-point iteration, because mesh-to-mesh interpolation needs to be applied in order to compute the inner product with the forward

solution at a given time level. However, since both the forward problem and QoI are linear, adjoint solves do not require storage of the forward trajectory.

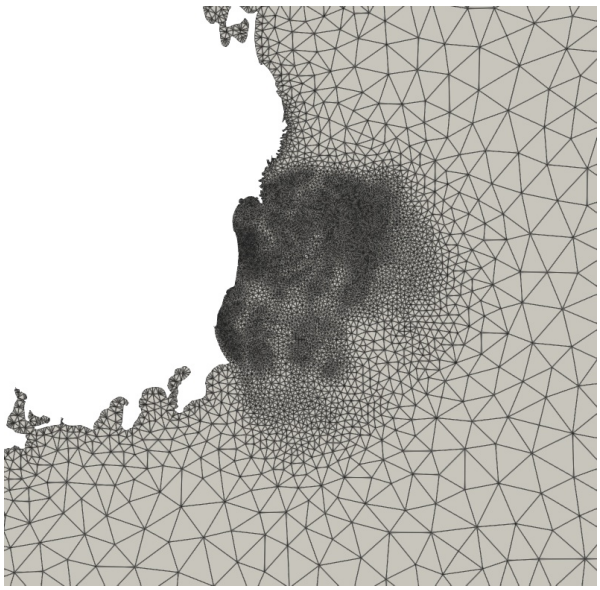
Under Hessian-based metrics, recovered from a component field such as elevation or fluid speed, the DoF count ramps up as the simulation progresses, as the tsunami waves spread across the domain. For adjoint-based DWP metrics, the opposite occurs, as shown in Figure 7.22. This makes sense, because the adjoint problem also contains a wave equation, but with the arrow of time reversed. L^1 normalisation is applied in space and time, with target metric complexity 5,000. The time interval is split into 24 subintervals. Boundary gradation is applied, in order to ensure that elements on the coast are at most 10 km in diameter. In the neighbourhood of Miyagi Prefecture (corresponding to boundary tag 300 in (4.1)) the maximal size is further constrained to 5 km. These gradations are consistent with those applied by qmesh on the base meshes. The time interval of interest begins at $T_{\text{start}} = 20\,\text{min}$.

Initially, there is a region of high mesh resolution around the tsunami source. As time progresses, this region follows the wave bound for the coast, shrinking in size accordingly. Figure 7.21 shows that this wave struck a long segment of the coast. However, only the part surrounding the region of interest is resolved under the DWP indicator.

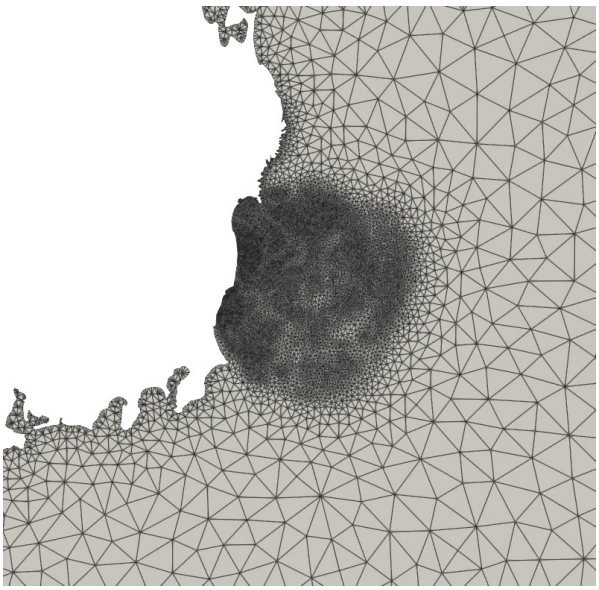
Adaptive Simulation using DWR Indicators. Figure 7.23 shows meshes due to the isotropic DWR indicator, for the same time snapshots. GE_h enrichment was used to obtain a higher order approximation to the adjoint solution.

Note that the meshes shown in Figure 7.23 are coarser than those in Figure 7.22 in the direction of the open ocean. The QoI is independent of the dynamics in that part of the domain, so this reduces unnecessary resolution. Aside from this, the DWP method tends to focus resolution more densely than DWR. It is also noteworthy that, in the final mesh, DWP deploys high mesh resolution over the entire region of interest, whereas DWR seeks only to resolve its boundary. This makes sense, because DWP focuses on magnitudes, whereas DWR focuses on residuals and fluxes.

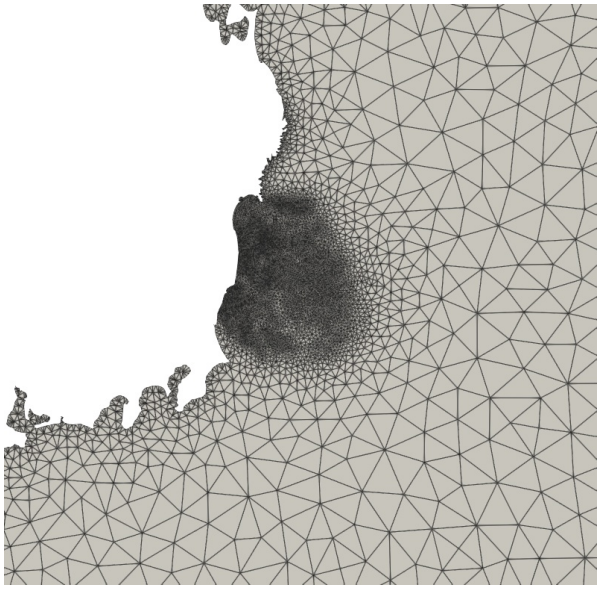
The simulation using DWP indicators took 28.4 minutes to complete three fixed point iterations on one process, whereas the simulation using DWR indicators took 44.7 minutes. The increased runtime is mainly due to the fact that adjoint errors are approximated using global enrichment. Adjoint solves in globally enriched spaces are computationally expensive and should be avoided, where possible. In future work, it would be beneficial to consider reducing the computational cost of the goal-oriented method by extending the difference quotient formulation to problems in mixed finite element spaces. It would also be useful to perform convergence analysis of the kind found in Subsection 5.8.4.



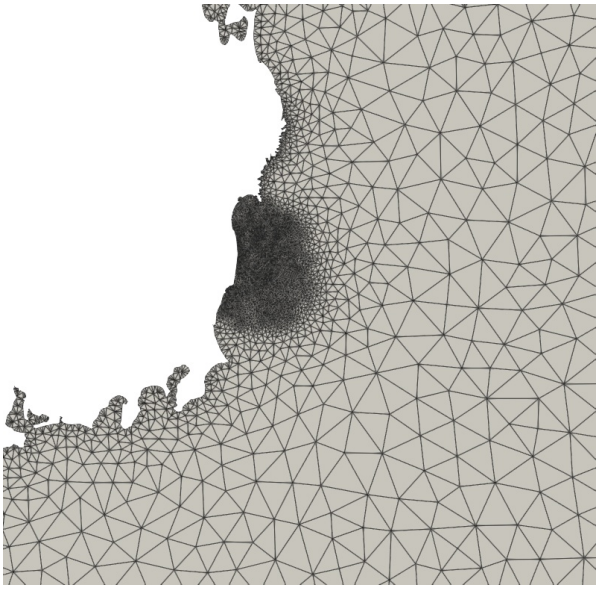
(A) $t = 0\,\text{min}$ (19,119 elements, 10,199 vertices)



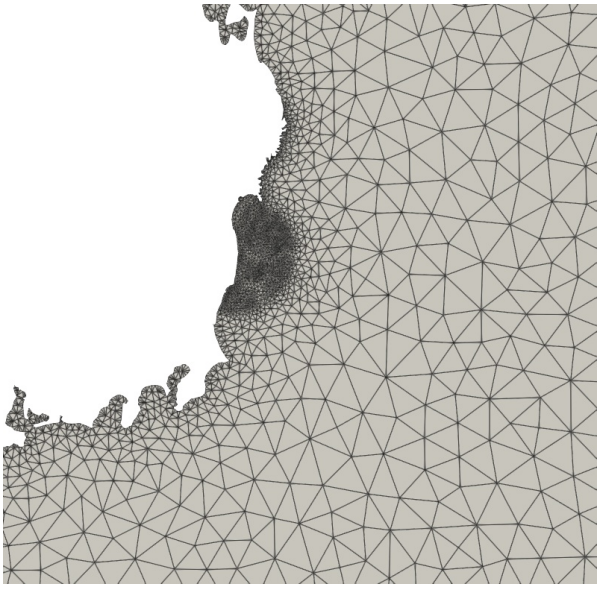
(B) $t = 6\,\text{min}$ (20,119 elements, 10,703 vertices)



(C) $t = 12\,\text{min}$ (15,487 elements, 8,389 vertices)

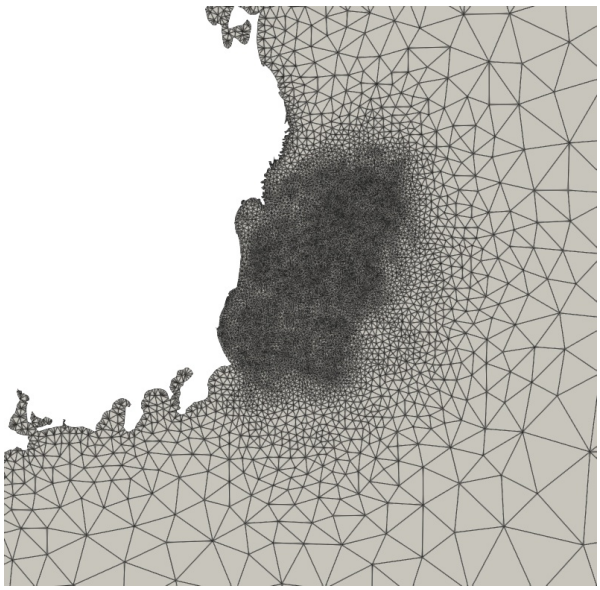


(D) $t = 18\,\text{min}$ (12,033 elements, 6,656 vertices)

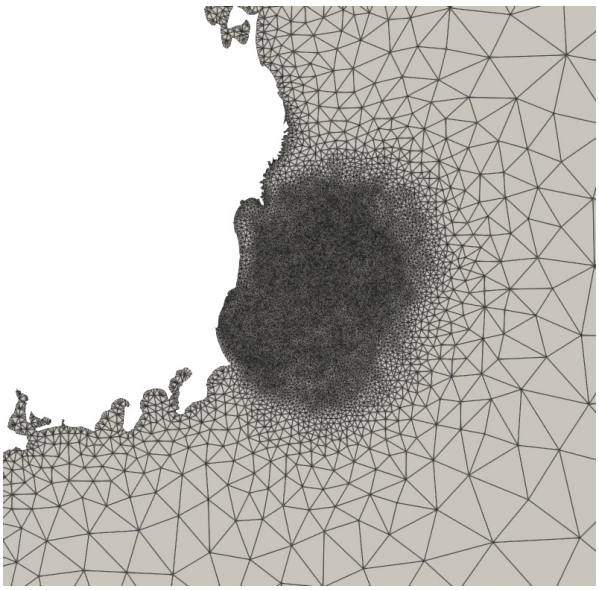


(E) $t = 24\,\text{min}$ (7,607 elements, 4,431 vertices)

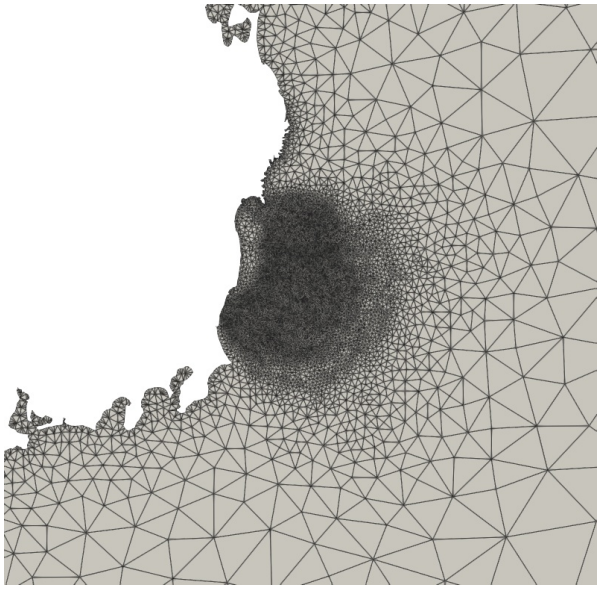
FIGURE 7.22. Mesh snapshots for an adjoint-based adaptive simulation of the Tōhoku tsunami based on DWP indicators, with Fukushima Daiichi being the location of interest.



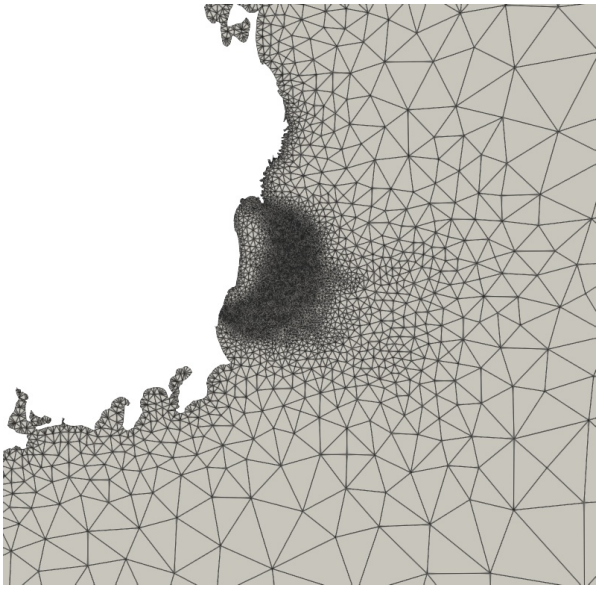
(A) $t = 0\,\text{min}$ (18,473 elements, 9,840 vertices)



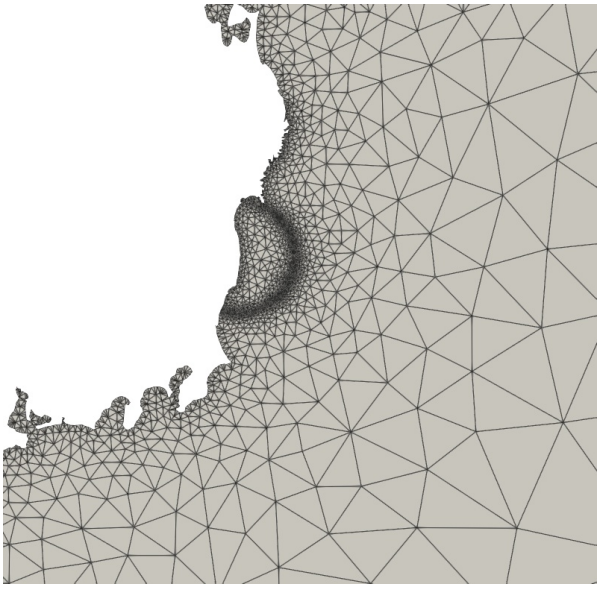
(B) $t = 6\,\text{min}$ (23,307 elements, 12,257 vertices)



(C) $t = 12\,\text{min}$ (17,937 elements, 9,572 vertices)



(D) $t = 18\,\text{min}$ (11,937 elements, 6,572 vertices)



(E) $t = 24\,\text{min}$ (5,506 elements, 3,354 vertices)

FIGURE 7.23. Mesh snapshots for an isotropic goal-oriented adaptive simulation of the Tōhoku tsunami based on DWR indicators, with Fukushima Daiichi being the location of interest.

Conclusion

Summary of Thesis Achievements

Chapter 3 contains a comparison of continuous and discrete adjoint methods. A variety of aspects are considered, including gradient accuracy, computational efficiency and compatibility with other model components or features, including mesh adaptation. Gradient-based optimisation methods are used to calibrate a Gaussian approximation to a Dirac delta point source, for an advection-diffusion test case with an analytical solution.

Chapter 4 also contains a comparison of continuous and discrete adjoint methods. To the best of the author's knowledge, it is the first such comparison in the context of shallow water modelling. The chapter also contains the first composition of two different AD tools to the task of tsunami source inversion. PyADOL-C is used to differentiate the Okada source model, whilst pyadjoint is used to differentiate a simple tsunami model which solves a linear wave equation. Despite the simplicity of the tsunami propagation model, it is demonstrated that the inversion strategy is able to capture the principal features of the timeseries profiles it seeks to assimilate.

Chapter 6 contains the first application of r-adaptation methods to a full hydro-morphodynamic model with both bedload and suspended sediment transport. Building upon the steady-state implementation of optimal transport r-adaptation presented in [**McRae et al., 2018**], a time-dependent extension was considered, with the Monge-Ampère equation solved at a subset of timesteps. Experiments were performed regarding the effective choice of a variety of parameters involved in both the adaptation method and its implementation, including mesh movement frequency, solver tolerance and a scalar parameter in the monitor function. This investigation can be used to inform later uses of optimally transported mesh movement applied to sediment transport modelling.

Chapter 7 contains a number of further achievements of this thesis. In terms of error estimation, first order forward goal-oriented error estimates for two discretisations of the shallow water equations are presented for the first time: an equal-order DG approach and the mixed discontinuous-continuous pair advocated in [**Cotter et al., 2009**]. First order forward and adjoint error estimates are also presented for an SUPG-stabilised CG discretisation of the advection-diffusion equation, similar to those which have already appeared in the literature (for example, in [**Carpio et al., 2013**]). In addition to these a posteriori error estimates, an adjoint counterpart to the first order forward a priori goal-oriented estimate derived in [**Loseille et al., 2010**] is proved for the first time.

The above error estimates are used to construct a variety of goal-oriented metrics, inspired by metrics found in the literature. One notable enhancement is given by intersecting the forward metric due to [**Loseille et al., 2010**] with its adjoint counterpart, based on the a priori error estimate derived herein. For an established advection-diffusion test case with an analytical solution, the intersection of these two metrics enables a goal-oriented mesh

adaptation strategy with attractive convergence properties. In the same experiment, it is also demonstrated that the 'difference quotient' formulation of the first order forward dual weighted residual, due to [**Becker and Rannacher, 2001**], can be used to generate isotropic meshes which admit QoI estimates of a similar accuracy to global enrichment methods, for a fraction of the computational cost.

Chapter 7 also presents the first applications of goal-oriented mesh adaptation to three distinct problem classes within coastal ocean modelling and engineering: tidal turbine farm modelling; desalination plant outfall; and tsunami hazard assessment. The corresponding examples are of increasing complexity and build the foundation for the application of the goal-oriented mesh adaptation framework to realistic problems. The tidal turbine test case reveals enhanced QoI convergence properties under an anisotropic goal-oriented method. Moreover, it demonstrates that this enhanced convergence may be attained by prolongation of the forward solution to an enriched finite element space, as opposed to solving an enriched forward problem, implying a significant reduction in computational cost. The desalination and tsunami modelling examples demonstrate the extension to the time-dependent case and reveal some desirable properties of the resulting discretisation.

Potential for Future Work

One of the main motivations for using a two dimensional, depth-averaged formulation given in Chapter 2 is that its reduced computational cost is conducive to rapid development of advanced discretisation methods. This model choice is especially useful if outer loop processes such as adjoint modelling and mesh adaptation are involved. Now that the goal-oriented mesh adaptation framework has been demonstrated for a variety of different 2D problems, as well as a simple 3D problem, future work may build upon this foundation. One coastal ocean engineering problem where goal-oriented mesh adaptation may be expected to be particularly effective is a 3D baroclinic simulations of desalination plant outfall in realistic coastal domains, wherein the dynamics are tidally varying and the radii of the inlet and outlet pipes, tidal processes and domain size vary on multiple orders of magnitude. Another challenging, multi-scale coastal engineering problem is the time-dependent simulation of a tidal turbine farm, with power or energy output as QoI.

A serial implementation of goal-oriented mesh adaptation has been sufficient for the purposes of this thesis. In order to tackle any of the large scale problems discussed above, it is imperative that a parallelised implementation be sought. This first requires the coupling of Firedrake to a mesh adaptation tool via a parallelised interface. Immediate future plans are to couple PETSc and Firedrake to the metric-based mesh adaptation toolkit Mmg [**Dapogny et al., 2014**] and develop a new module dedicated to making goal-oriented mesh adaptation technologies more widely available to the Firedrake community. Automation of the goal-oriented error estimation process (as documented in [**Rognes and Logg, 2013**] for the FEniCS project) would improve usability of goal-oriented technologies in Firedrake yet further.

In this thesis, r-adaptation methods were considered at length, but were not used within the goal-oriented framework. The only documented use of purely r-adaptive goal-oriented mesh adaptation the author could find in the literature is that of [**Bauer et al., 2014**]. This presents an interesting possible line of research. Another possible extension to the work on r-adaptation is to couple the optimal transport based adaptation with an ALE type method, so that interpolation error may be reduced.

Bibliography

[Abe, 1973] Abe, K. (1973). Tsunami and mechanism of great earthquakes. *Physics of the Earth and Planetary Interiors*, 7(2):143–153.

[Ainsworth and Oden, 2011] Ainsworth, M. and Oden, J. T. (2011). *A posteriori error estimation in finite element analysis*, volume 37. John Wiley & Sons.

[Alauzet, 2003] Alauzet, F. (2003). *Adaptation de maillage anisotrope en trois dimensions: applications aux simulations instationnaires en mécanique des fluides*. PhD thesis, Montpellier 2.

[Alauzet, 2010] Alauzet, F. (2010). Size gradation control of anisotropic meshes. *Finite Elements in Analysis and Design*, 46(1-2):181–202.

[Alauzet et al., 2011] Alauzet, F., Belme, A., and Dervieux, A. (2011). Anisotropic goal-oriented mesh adaptation for time dependent problems. In *Proceedings of the 20th International Meshing Roundtable*, pages 99–121. Springer.

[Alauzet et al., 2007] Alauzet, F., Frey, P. J., George, P.-L., and Mohammadi, B. (2007). 3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations. *Journal of Computational Physics*, 222(2):592–623.

[Alauzet and Olivier, 2010] Alauzet, F. and Olivier, G. (2010). An L^p-L^∞ space-time anisotropic mesh adaptation strategy for time dependent problems. In *Proceedings of ECCOMAS CFD*.

[Alnæs et al., 2015] Alnæs, M., Blechta, J., Hale, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M. E., and Wells, G. N. (2015). The FEniCS project version 1.5. *Archive of Numerical Software*, 3(100).

[Alnæs et al., 2014] Alnæs, M. S., Logg, A., Ølgaard, K. B., Rognes, M. E., and Wells, G. N. (2014). Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software (TOMS)*, 40(2):1–37.

[Amante and Eakins, 2020] Amante, C. and Eakins, B. W. (2020). ETOPO1 arc-minute global relief model: procedures, data sources and analysis. Accessed: 2020-08-06.

[Armijo, 1966] Armijo, L. (1966). Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3.

[Avdis et al., 2018] Avdis, A., Candy, A. S., Hill, J., Kramer, S. C., and Piggott, M. D. (2018). Efficient unstructured mesh generation for marine renewable energy applications. *Renewable Energy*, 116:842–856.

[Avriel, 2003] Avriel, M. (2003). *Nonlinear programming: analysis and methods*. Courier Corporation.

[Awanou, 2015] Awanou, G. (2015). Quadratic mixed finite element approximations of the Monge–Ampère equation in 2D. *Calcolo*, 52(4):503–518.

[Babuška and Rheinboldt, 1978a] Babuška, I. and Rheinboldt, W. C. (1978a). A-posteriori error estimates for the finite element method. *International Journal for Numerical Methods in Engineering*, 12(10):1597–1615.

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

[Babuška and Suri, 1994] Babuška, I. and Suri, M. (1994). The p and hp versions of the finite element method, basic principles and properties. *SIAM Review*, 36(4):578–632.

[Babuška et al., 1981] Babuška, I., Szabo, B. A., and Katz, I. N. (1981). The p-version of the finite element method. *SIAM Journal on Numerical Analysis*, 18(3):515–545.

[Barral, 2015] Barral, N. (2015). *Time-accurate anisotropic mesh adaptation for three-dimensional moving mesh problems*. PhD thesis, Université Pierre et Marie Curie.

[Barral et al., 2016] Barral, N., Knepley, M. G., Lange, M., Piggott, M. D., and Gorman, G. J. (2016). Anisotropic mesh adaptation in Firedrake with PETSc DMPlex. *arXiv preprint arXiv:1610.09874*.

[Bauer et al., 2014] Bauer, W., Baumann, M., Scheck, L., Gassmann, A., Heuveline, V., and Jones, S. C. (2014). Simulation of tropical-cyclone-like vortices in shallow-water ICON-hex using goal-oriented r-adaptivity. *Theoretical and Computational Fluid Dynamics*, 28(1):107–128.

[Becker and Rannacher, 1996] Becker, R. and Rannacher, R. (1996). *A feed-back approach to error control in finite element methods: basic analysis and examples*. IWR.

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

[Beda et al., 1959] Beda, L. M., Korolev, L. N., Sukkikh, N. V., and Frolova, T. S. (1959). Programs for automatic differentiation for the machine besm. Technical report, Technical Report, Institute for Precise Mechanics and Computation Techniques.

[Behrens and Zimmermann, 2000] Behrens, J. and Zimmermann, J. (2000). Parallelizing an unstructured grid generator with a space-filling curve approach. In *European Conference on Parallel Processing*, pages 815–823. Springer.

[Belme et al., 2012] Belme, A., Dervieux, A., and Alauzet, F. (2012). Time accurate anisotropic goal-oriented mesh adaptation for unsteady flows. *Journal of Computational Physics*, 231(19):6323–6348.

[Bifulco et al., 2009] Bifulco, I., Raiconi, G., and Scarpa, R. (2009). Computer algebra software for least squares and total least norm inversion of geophysical models. *Computers & Geosciences*, 35(7):1427–1438.

[Bischof et al., 1992] Bischof, C., Carle, A., Corliss, G., Griewank, A., and Hovland, P. (1992). ADIFOR–generating derivative codes from fortran programs. *Scientific Programming*, 1(1):11–29.

[Blaise et al., 2013] Blaise, S., St-Cyr, A., Mavriplis, D., and Lockwood, B. (2013). Discontinuous Galerkin unsteady discrete adjoint method for real-time efficient tsunami simulations. *Journal of Computational Physics*, 232(1):416–430.

[Bowman, 2020] Bowman, W. (2020). Python distribution of the MatLab package UTide. *Python Package Index*. URL: `https://pypi.org/project/UTide/`.

[Braack, 1998] Braack, M. (1998). *An adaptive finite element method for reactive-flow problems*. PhD thesis, IWR.

[Brenner and Scott, 2007] Brenner, S. and Scott, R. (2007). *The mathematical theory of finite element methods*, volume 15. Springer Science & Business Media.

[Brooks and Hughes, 1982] Brooks, A. N. and Hughes, T. J. R. (1982). Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32(1-3):199–259.

[Broyden, 1965] Broyden, C. G. (1965). A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19(92):577–593.

[Broyden et al., 1973] Broyden, C. G., Dennis Jr, J., and Moré, J. J. (1973). On the local and superlinear convergence of quasi-Newton methods. *IMA Journal of Applied Mathematics*, 12(3):223–245.

[Budd et al., 2013] Budd, C. J., Cullen, M. J. P., and Walsh, E. J. (2013). Monge–Ampére based moving mesh methods for numerical weather prediction, with applications to the Eady problem. *Journal of Computational Physics*, 236:247–270.

[Budd et al., 2009] Budd, C. J., Huang, W., and Russell, R. D. (2009). Adaptivity with moving grids. *Acta Numerica*, 18(1):111–241.

[Budd and Williams, 2009] Budd, C. J. and Williams, J. F. (2009). Moving mesh generation using the parabolic Monge–Ampère equation. *SIAM Journal on Scientific Computing*, 31(5):3438–3465.

[Capinski and Kopp, 2013] Capinski, M. and Kopp, P. E. (2013). *Measure, integral and probability*. Springer Science & Business Media.

[Carnarius et al., 2011] Carnarius, A., Thiele, F., Özkaya, E., Nemili, A., and Gauger, N. R. (2011). Optimal control of unsteady flows using a discrete and a continuous adjoint approach. In *IFIP Conference on System Modeling and Optimization*, pages 318–327. Springer.

[Carpio et al., 2013] Carpio, J., Prieto, J. L., and Bermejo, R. (2013). Anisotropic "goal-oriented" mesh adaptivity for elliptic problems. *SIAM Journal on Scientific Computing*, 35(2):A861–A885.

[Cho et al., 2007] Cho, Y.-S., Sohn, D.-H., and Lee, S. O. (2007). Practical modified scheme of linear shallow-water equations for distant propagation of tsunamis. *Ocean Engineering*, 34(11-12):1769–1777.

[Choi et al., 2004] Choi, J., Margetis, D., Squires, T. M., and Bazant, M. Z. (2004). Steady advection-diffusion around finite absorbers in two-dimensional potential flows. *arXiv preprint cond-mat/0403740*.

[Ciarlet, 1978] Ciarlet, P. (1978). The finite element method for elliptic problems.

[Clare et al., 2021] Clare, M. C. A., Percival, J. R., Angeloudis, A., Cotter, C. J., and Piggott, M. D. (2021). Hydro-morphodynamics 2D modelling using a discontinuous Galerkin discretisation. *Computers & Geosciences*, 146:104658.

[Clare et al., 2020] Clare, M. C. A., Wallwork, J. G., Kramer, S. C., Weller, H., Cotter, C. J., and Piggott, M. D. (2020). Multi-scale challenges hydro-morphodynamic modelling using mesh movement methods. *EarthArXiv Preprint*. DOI: `10.31223/osf.io/tpqvy`. URL: `https://doi.org/10.31223/osf.io/tpqvy`.

[Clawpack Development Team, 2020] Clawpack Development Team (2020). Clawpack version 5.7.0. DOI: `10.5281/zenodo.3764278`. URL: `https://www.clawpack.org`.

[Codiga, 2020] Codiga, D. (2020). UTide unified tidal analysis and prediction functions. *MATLAB Central File Exchange*. URL: `https://mathworks.com/matlabcentral/fileexchange/46523-utide-unified-tidalanalysis-and-prediction-functions`.

[Cotter et al., 2009] Cotter, C. J., Ham, D. A., and Pain, C. C. (2009). A mixed discontinuous/continuous finite element pair for shallow-water ocean modelling. *Ocean Modelling*, 26(1-2):86–90.

[Crank and Nicolson, 1947] Crank, J. and Nicolson, P. (1947). A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 43, pages 50–67. Cambridge University Press.

[Crouzeix and Raviart, 1973] Crouzeix, M. and Raviart, P.-A. (1973). Conforming and nonconforming finite element methods for solving the stationary stokes equations i. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 7(R3):33–75.

[Dao and Tkalich, 2007] Dao, M. and Tkalich, P. (2007). Tsunami propagation modelling: a sensitivity study. *Natural Hazards and Earth System Science*, 7(6):741–754.

[Dapogny et al., 2014] Dapogny, C., Dobrzynski, C., and Frey, P. (2014). Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. *Journal of Computational Physics*, 262:358–378.

[Davis and LeVeque, 2016] Davis, B. N. and LeVeque, R. J. (2016). Adjoint methods for guiding adaptive mesh refinement in tsunami modeling. In *Global Tsunami Science: Past and Future, Volume I*, pages 4055–4074. Springer.

[D'Azevedo, 1991] D'Azevedo, E. F. (1991). Optimal triangular mesh generation by coordinate transformation. *SIAM Journal on Scientific and Statistical Computing*, 12(4):755–786.

[D'Azevedo and Simpson, 1991] D'Azevedo, E. F. and Simpson, R. B. (1991). On optimal triangular meshes for minimizing the gradient error. *Numerische Mathematik*, 59(1):321–348.

[DeBlois, 1996] DeBlois, B. M. (1996). Quadratic, streamline upwinding for finite element method solutions to 2-D convective transport problems. *Computer Methods in Applied Mechanics and Engineering*, 134(1):107–115.

[Delzanno et al., 2008] Delzanno, G. L., Chacón, L., Finn, J. M., Chung, Y., and Lapenta, G. (2008). An optimal robust equidistribution method for two-dimensional grid adaptation based on Monge–Kantorovich optimization. *Journal of Computational Physics*, 227(23):9841–9864.

[Divett et al., 2013] Divett, T., Vennell, R., and Stevens, C. (2013). Optimization of multiple turbine arrays in a channel with tidally reversing flow by numerical modelling with adaptive mesh. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1985):20120251.

[Dolejší and Roskovec, 2017] Dolejší, V. and Roskovec, F. (2017). Goal-oriented error estimates including algebraic errors in discontinuous Galerkin discretizations of linear boundary value problems. *Applications of Mathematics*, 62(6):579–605.

[Dolejší and Solin, 2016] Dolejší, V. and Solin, P. (2016). hp-discontinuous Galerkin method based on local higher order reconstruction. *Applied Mathematics and Computation*, 279:219–235.

[Donea and Huerta, 2003] Donea, J. and Huerta, A. (2003). *Finite element methods for flow problems*. John Wiley & Sons.

[Donea et al., 2017] Donea, J., Huerta, A., Ponthot, J.-P., and Rodríguez-Ferran, A. (2017). Arbitrary Lagrangian–Eulerian methods. *Encyclopedia of Computational Mechanics Second Edition*, pages 1–23.

[du Feu et al., 2019] du Feu, R. J., Funke, S. W., Kramer, S. C., Hill, J., and Piggott, M. D. (2019). The trade-off between tidal-turbine array yield and environmental impact: A habitat suitability modelling approach. *Renewable Energy*, 143:390–403.

[Ekelschot et al., 2017] Ekelschot, D., Moxey, D., Sherwin, S. J., and Peiró, J. (2017). A p-adaptation method for compressible flow problems using a goal-based error indicator. *Computers & Structures*, 181:55–69.

[Epshteyn and Rivière, 2007] Epshteyn, Y. and Rivière, B. (2007). Estimation of penalty parameters for symmetric interior penalty Galerkin methods. *Journal of Computational and Applied Mathematics*, 206(2):843–872.

[Exner, 1920] Exner, F. M. (1920). *Zur physik der dünen*. Hölder.

[Farhat et al., 1998] Farhat, C., Degand, C., Koobus, B., and Lesoinne, M. (1998). Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering*, 163(1-4):231–245.

[Farrell et al., 2013] Farrell, P. E., Ham, D. A., Funke, S. W., and Rognes, M. E. (2013). Automated derivation of the adjoint of high-level transient finite element programs. *SIAM Journal on Scientific Computing*, 35(4):C369–C393.

[Farrell and Maddison, 2011] Farrell, P. E. and Maddison, J. R. (2011). Conservative interpolation between volume meshes by local Galerkin projection. *Computer Methods in Applied Mechanics and Engineering*, 200(1-4):89–100. DOI: 10.1016/j.cma.2010.07.015.

[Farrell et al., 2009] Farrell, P. E., Piggott, M. D., Pain, C. C., Gorman, G. J., and Wilson, C. R. (2009). Conservative interpolation between unstructured meshes via supermesh construction. *Computer Methods in Applied Mechanics and Engineering*, 198(33-36):2632–2642.

[Formaggia et al., 2004] Formaggia, L., Micheletti, S., and Perotto, S. (2004). Anisotropic mesh adaptation in computational fluid dynamics: application to the advection–diffusion–reaction and the Stokes problems. *Applied Numerical Mathematics*, 51(4):511–533.

[Formaggia and Perotto, 2001] Formaggia, L. and Perotto, S. (2001). New anisotropic a priori error estimates. *Numerische Mathematik*, 89(4):641–667.

[Frey and Alauzet, 2005] Frey, P.-J. and Alauzet, F. (2005). Anisotropic mesh adaptation for CFD computations. *Computer Methods in Applied Mechanics and Engineering*, 194(48-49):5068–5082.

[Funke et al., 2014] Funke, S. W., Farrell, P. E., and Piggott, M. D. (2014). Tidal turbine array optimisation using the adjoint approach. *Renewable Energy*, 63:658–673.

[Funke et al., 2017] Funke, S. W., Farrell, P. E., and Piggott, M. D. (2017). Reconstructing wave profiles from inundation data. *Computer Methods in Applied Mechanics and Engineering*, 322:167–186.

[Garel et al., 2014] Garel, F., Goes, S., Davies, D., Davies, J. H., Kramer, S. C., and Wilson, C. R. (2014). Interaction of subducted slabs with the mantle transition-zone: A regime diagram from 2-D

thermo-mechanical models with a mobile trench and an overriding plate. *Geochemistry, Geophysics, Geosystems*, 15(5):1739–1765.

[Gebremedhin et al., 2005] Gebremedhin, A. H., Manne, F., and Pothen, A. (2005). What color is your Jacobian? Graph coloring for computing derivatives. *SIAM review*, 47(4):629–705.

[George et al., 1991] George, P., Hecht, F., and Vallet, M. (1991). Creation of internal points in Voronoi's type method. Control and adaptation. *Advances in Engineering Software*, 13(5-6):303–312.

[George et al., 2002] George, P. L., Borouchaki, H., and Laug, P. (2002). An efficient algorithm for 3D adaptive meshing. *Advances in Engineering Software*, 33(7-10):377–387.

[Geuzaine and Remacle, 2009] Geuzaine, C. and Remacle, J.-F. (2009). Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *International Journal on Numerical Methods in Engineering*, 79(11):1309–1331.

[Giering and Kaminski, 1998] Giering, R. and Kaminski, T. (1998). Recipes for adjoint code construction. *ACM Transactions on Mathematical Software (TOMS)*, 24(4):437–474.

[Giles et al., 2005] Giles, M., Ghate, D., and Duta, M. C. (2005). Using automatic differentiation for adjoint CFD code development. Technical report, Oxford University Computing Laboratory.

[Gingold and Monaghan, 1977] Gingold, R. A. and Monaghan, J. J. (1977). Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389.

[Godunov, 1959] Godunov, S. K. (1959). A difference scheme for numerical solution of discontinuous solution of hydrodynamic equations. *Math. Sbornik*, 47:271–306.

[Griewank et al., 1996] Griewank, A., Juedes, D., and Utke, J. (1996). Algorithm 755: ADOL-C: a package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software (TOMS)*, 22(2):131–167.

[Griewank and Walther, 2000] Griewank, A. and Walther, A. (2000). Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software (TOMS)*, 26(1):19–45.

[Gunzburger, 2002] Gunzburger, M. D. (2002). *Perspectives in flow control and optimization*. SIAM.

[Ham et al., 2021] Ham et al. (2021). Software used in 'Mesh Adaptation and Adjoint Methods for Finite Element Coastal Ocean Modelling'. DOI: `10.5281/zenodo.4561836`. URL: `https://doi.org/10.5281/zenodo.4561836`.

[Hascoet and Pascual, 2013] Hascoet, L. and Pascual, V. (2013). The Tapenade automatic differentiation tool: Principles, model, and specification. *ACM Transactions on Mathematical Software (TOMS)*, 39(3):1–43.

[Herzog and Kunisch, 2010] Herzog, R. and Kunisch, K. (2010). Algorithms for pde-constrained optimization. *GAMM-Mitteilungen*, 33(2):163–176.

[Hiester et al., 2011] Hiester, H. R., Piggott, M. D., and Allison, P. A. (2011). The impact of mesh adaptivity on the gravity current front speed in a two-dimensional lock-exchange. *Ocean Modelling*, 38(1):1–21.

[Hossen et al., 2018] Hossen, M. J., Gusman, A., Satake, K., and Cummins, P. R. (2018). An adjoint sensitivity method applied to time reverse imaging of tsunami source for the 2009 Samoa earthquake. *Geophysical Research Letters*, 45(2):627–636.

[Huang and Kamenski, 2015] Huang, W. and Kamenski, L. (2015). A geometric discretization and a simple implementation for variational mesh generation and adaptation. *Journal of Computational Physics*, 301:322–337.

[Huang et al., 1994] Huang, W., Ren, Y., and Russell, R. D. (1994). Moving mesh partial differential equations (MMPDEs) based on the equidistribution principle. *SIAM Journal on Numerical Analysis*, 31(3):709–730.

[Hückelheim et al., 2019] Hückelheim, J., Kukreja, N., Narayanan, S. H. K., Luporini, F., Gorman, G., and Hovland, P. (2019). Automatic differentiation for adjoint stencil loops. *arXiv preprint arXiv:1907.02818*.

[Johnson, 1929] Johnson, R. A. (1929). *Modern geometry: an elementary treatise on the geometry of the triangle and the circle*. Houghton, Mifflin Company.

[Kabanikhin et al., 2014] Kabanikhin, S., Hasanov, A., Marinin, I., Krivorotko, O., and Khidasheli, D. (2014). A variational approach to reconstruction of an initial tsunami source perturbation. *Applied Numerical Mathematics*, 83:22–37.

[Kajiura, 1970] Kajiura, K. (1970). Tsunami source, energy and the directivity of wave radiation. *Bulletin of the Earthquake Research Institute, University of Tokyo*, 48:835–869.

[Kärnä et al., 2011] Kärnä, T., De Brye, B., Gourgue, O., Lambrechts, J., Comblen, R., Legat, V., and Deleersnijder, E. (2011). A fully implicit wetting–drying method for DG-FEM shallow water models, with an application to the Scheldt Estuary. *Computer Methods in Applied Mechanics and Engineering*, 200(5-8):509–524.

[Kärnä et al., 2018] Kärnä, T., Kramer, S. C., Mitchell, L., Ham, D. A., Piggott, M. D., and Baptista, A. M. (2018). Thetis coastal ocean model: discontinuous Galerkin discretization for the three-dimensional hydrostatic equations. *Geoscientific Model Development*, 11(11):4359–4382.

[Kärnä et al., 2021] Kärnä, T., Kramer, S. C., Mitchell, L., Wallwork, J. G., Angeloudis, A., Clare, M. C. A., Ham, D. A., Barral, N., McRae, A. T. T., and Warder, S. C. (2021). Thetis coastal ocean model. DOI: `10.5281/zenodo.4560054`. URL: `https://doi.org/10.5281/zenodo.4560054`.

[Kärnä et al., 2013] Kärnä, T., Legat, V., and Deleersnijder, E. (2013). A baroclinic discontinuous Galerkin finite element model for coastal flows. *Ocean Modelling*, 61:1–20.

[Kirby et al., 2013] Kirby, J. T., Shi, F., Tehranirad, B., Harris, J. C., and Grilli, S. T. (2013). Dispersive tsunami waves in the ocean: Model equations and sensitivity to dispersion and Coriolis effects. *Ocean Modelling*, 62:39–55.

[Kramer and Piggott, 2016] Kramer, S. C. and Piggott, M. D. (2016). A correction to the enhanced bottom drag parameterisation of tidal turbines. *Renewable Energy*, 92:385–396.

[Lange et al., 2016] Lange, M., Mitchell, L., Knepley, M. G., and Gorman, G. J. (2016). Efficient mesh management in Firedrake using PETSC DMPlex. *SIAM Journal on Scientific Computing*, 38(5):S143–S155.

[LeVeque et al., 2011a] LeVeque, R., George, D. L., and Berger, M. J. (2011a). Adaptive mesh refinement techniques for tsunamis and other geophysical flows over topography. *Acta Numerica*, pages 211–289.

[LeVeque, 1996] LeVeque, R. J. (1996). High-resolution conservative algorithms for advection in incompressible flow. *SIAM Journal on Numerical Analysis*, 33(2):627–665.

[LeVeque et al., 2011b] LeVeque, R. J., George, D. L., and Berger, M. J. (2011b). Tsunami modelling with adaptively refined finite volume methods. *Acta Numerica*, 20:211.

[Liu et al., 2009] Liu, Y., Shi, Y., Yuen, D. A., Sevre, E. O., Yuan, X., and Xing, H. L. (2009). Comparison of linear and nonlinear shallow wave water equations applied to tsunami waves over the China Sea. *Acta Geotechnica*, 4(2):129–137.

[Loseille, 2008] Loseille, A. (2008). *Adaptation de maillage 3D anisotrope multi-échelles et cibléa une fonctionnelle. Applicationa la prédiction haute-fidélité du bang sonique*. PhD thesis, PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France.

[Loseille and Alauzet, 2011a] Loseille, A. and Alauzet, F. (2011a). Continuous mesh framework part I: well-posed continuous interpolation error. *SIAM Journal on Numerical Analysis*, 49(1):38–60.

[Loseille and Alauzet, 2011b] Loseille, A. and Alauzet, F. (2011b). Continuous mesh framework part II: validations and applications. *SIAM Journal on Numerical Analysis*, 49(1):61–86.

[Loseille et al., 2010] Loseille, A., Dervieux, A., and Alauzet, F. (2010). Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations. *Journal of computational physics*, 229(8):2866–2897.

[Lotto et al., 2017] Lotto, G. C., Nava, G., and Dunham, E. M. (2017). Should tsunami simulations include a nonzero initial horizontal velocity? *Earth, Planets and Space*, 69(1):1–14.

[Lucy, 1977] Lucy, L. B. (1977). A numerical approach to the testing of the fission hypothesis. *The astronomical journal*, 82:1013–1024.

[MacInnes et al., 2013] MacInnes, B. T., Gusman, A. R., LeVeque, R. J., and Tanioka, Y. (2013). Comparison of earthquake source models for the 2011 Tohoku event using tsunami simulations and near-field observations. *Bulletin of the Seismological Society of America*, 103(2B):1256–1274.

[Maddison et al., 2016] Maddison, J. R., Farrell, P. E., and Panourglas, I. P. (2016). Parallel supermeshing for multimesh modelling. Technical Report eCSE03-08, ARCHER. URL: `https://www.archer.co.uk/community/eCSE/eCSE03-08/eCSE03-08-TechnicalReport.pdf`.

[McRae et al., 2018] McRae, A. T. T., Cotter, C. J., and Budd, C. J. (2018). Optimal-transport-based mesh adaptivity on the plane and sphere using finite elements. *SIAM Journal on Scientific Computing*, 40(2):1121–1148.

[Micheletti and Perotto, 2013] Micheletti, S. and Perotto, S. (2013). Anisotropic recovery-based a posteriori error estimators for advection-diffusion-reaction problems. In *Numerical Mathematics and Advanced Applications 2011*, pages 43–51. Springer.

[Micheletti et al., 2010] Micheletti, S., Perotto, S., and Farrell, P. E. (2010). A recovery-based error estimator for anisotropic mesh adaptation in CFD. *SeMA Journal*, 50(1):115–137.

[Micheletti et al., 2003] Micheletti, S., Perotto, S., and Picasso, M. (2003). Stabilized finite elements on anisotropic meshes: a priori error estimates for the advection-diffusion and the Stokes problems. *SIAM Journal on Numerical Analysis*, 41(3):1131–1162.

[Mitusch, 2018] Mitusch, S. K. (2018). An algorithmic differentiation tool for FEniCS. Master's thesis, University of Oslo.

[Mori et al., 2012] Mori, N., Takahashi, T., and 2011 Tohoku Earthquake Tsunami Joint Survey Group (2012). Nationwide post event survey and analysis of the 2011 Tohoku earthquake tsunami. *Coastal Engineering Journal*, 54(1):1250001–1.

[Mostaghimi et al., 2016] Mostaghimi, P., Kamali, F., Jackson, M. D., Muggeridge, A. H., Pain, C. C., et al. (2016). Adaptive mesh optimization for simulation of immiscible viscous fingering. *SPE Journal*, 21(06):2–250.

[Mulia et al., 2018] Mulia, I. E., Gusman, A. R., Jakir Hossen, M., and Satake, K. (2018). Adaptive tsunami source inversion using optimizations and the reciprocity principle. *Journal of Geophysical Research: Solid Earth*, 123(12):10–749.

[Nadarajah and Jameson, 2000] Nadarajah, S. and Jameson, A. (2000). A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. In *38th Aerospace Sciences Meeting and Exhibit*, page 667.

[Nadarajah and Jameson, 2001] Nadarajah, S. and Jameson, A. (2001). Studies of the continuous and discrete adjoint approaches to viscous automatic aerodynamic shape optimization. In *15th AIAA Computational Fluid Dynamics Conference*, page 2530.

[Nadarajah and Jameson, 2007] Nadarajah, S. K. and Jameson, A. (2007). Optimum shape design for unsteady flows with time-accurate continuous and discrete adjoint method. *AIAA journal*, 45(7):1478–1491.

[National Police Agency of Japan, 2020] National Police Agency of Japan (2020). Police countermeasures and damage situation associated with 2011 Tōhoku district – off the Pacific Ocean earthquake. `https://www.npa.go.jp/news/other/earthquake2011/pdf/higaijokyo_e.pdf`. Accessed: 2020-07-16.

[NOAA, 2021] NOAA (2021). Center for tsunami research: Pacific marine environmental laboratory. `https://nctr.pmel.noaa.gov/propagation-database.html`. Accessed: 2021-03-03.

[Oishi et al., 2013] Oishi, Y., Piggott, M. D., Maeda, T., Kramer, S. C., Collins, G. S., Tsushima, H., and Furumura, T. (2013). Three-dimensional tsunami propagation simulations using an unstructured mesh finite element model. *Journal of Geophysical Research: Solid Earth*, 118(6):2998–3018.

[Okada, 1985] Okada, Y. (1985). Surface deformation due to shear and tensile faults in a half-space. *Bulletin of the Seismological Society of America*, 75(4):1135–1154.

[Olivier, 2011] Olivier, G. (2011). *Anisotropic metric-based mesh adaptation for unsteady CFD simulations involving moving geometries*. PhD thesis, Paris 6.

[Pain et al., 2001] Pain, C. C., Umpleby, A. P., De Oliveira, C. R. E., and Goddard, A. J. H. (2001). Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations. *Computer Methods in Applied Mechanics and Engineering*, 190(29-30):3771–3796.

[Parkinson et al., 2014] Parkinson, S. D., Hill, J., Piggott, M. D., and Allison, P. A. (2014). Direct numerical simulations of particle-laden density currents with adaptive, discontinuous finite elements. *Geoscientific Model Development*, 7(5):1945–1960.

[Piggott et al., 2009] Piggott, M. D., Farrell, P. E., Wilson, C. R., Gorman, G. J., and Pain, C. C. (2009). Anisotropic mesh adaptivity for multi-scale ocean modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1907):4591–4611.

[Pires and Miranda, 2001] Pires, C. and Miranda, P. M. A. (2001). Tsunami waveform inversion by adjoint methods. *Journal of Geophysical Research: Oceans*, 106(C9):19773–19796.

[Power et al., 2006] Power, P. W., Pain, C. C., Piggott, M. D., Fang, F., Gorman, G. J., Umpleby, A. P., Goddard, A. J. H., and Navon, I. M. (2006). Adjoint a posteriori error measures for anisotropic mesh optimisation. *Computers & Mathematics with Applications*, 52(8-9):1213–1242.

[Rathgeber et al., 2016] Rathgeber, F., Ham, D. A., Mitchell, L., Lange, M., Luporini, F., McRae, A. T., Bercea, G.-T., Markall, G. R., and Kelly, P. H. (2016). Firedrake: automating the finite element method by composing abstractions. *ACM Transactions on Mathematical Software (TOMS)*, 43(3):1–27.

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

[Ren et al., 2018] Ren, Z., Ji, X., Wang, P., Hou, J., Shan, D., and Zhao, L. (2018). Source inversion and numerical simulation of 2017 M w 8.1 Mexico earthquake tsunami. *Natural Hazards*, 94(3):1163–1185.

[Ren et al., 2015] Ren, Z.-Y., Zhao, X., and Liu, H. (2015). Dispersion effects on tsunami propagation in South China Sea. *Journal of Earthquake and Tsunami*, 9(05):1540001.

[Rhufat, 2020] Rhufat, D. (2020). triangle. *Python Package Index*. URL: `https://pypi.org/project/triangle/`.

[Riadh et al., 2014] Riadh, A., Cedric, G., and Jean, M. H. (2014). TELEMAC modeling system: 2D hydrodynamics TELEMAC-2D software release 7.0 user manual. *Paris: R&D, Electricite de France*, page 134.

[Rogé and Martin, 2008] Rogé, G. and Martin, L. (2008). Goal-oriented anisotropic grid adaptation. *Comptes Rendus Mathématique*, 346(19-20):1109–1112.

[Rognes and Logg, 2010] Rognes, M. E. and Logg, A. (2010). Exploring automated adaptivity and error control. In *AIP conference proceedings*, volume 1281, pages 794–797. American Institute of Physics.

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

[Rossa and Coutinho, 2013] Rossa, A. L. and Coutinho, A. L. (2013). Parallel adaptive simulation of gravity currents on the lock-exchange problem. *Computers & Fluids*, 88:782–794.

[Roth et al., 2021] Roth, J., Schröder, M., and Wick, T. (2021). Neural network guided adjoint computations in dual weighted residual error estimation. *arXiv preprint arXiv:2102.12450*.

[Saito et al., 2014] Saito, T., Inazu, D., Miyoshi, T., and Hino, R. (2014). Dispersion and nonlinear effects in the 2011 Tohoku-Oki earthquake tsunami. *Journal of Geophysical Research: Oceans*, 119(8):5160–5180.

[Saito et al., 2011] Saito, T., Ito, Y., Inazu, D., and Hino, R. (2011). Tsunami source of the 2011 Tohoku-Oki earthquake, Japan: Inversion analysis based on dispersive tsunami simulations. *Geophysical Research Letters*, 38(7).

[Satake, 1987] Satake, K. (1987). Inversion of tsunami waveforms for the estimation of a fault heterogeneity: Method and numerical experiments. *Journal of Physics of the Earth*, 35(3):241–254.

[Satake, 1995] Satake, K. (1995). Linear and nonlinear computations of the 1992 Nicaragua earthquake tsunami. In *Tsunamis: 1992–1994*, pages 455–470. Springer.

[Schwedes et al., 2017] Schwedes, T., Ham, D. A., Funke, S. W., and Piggott, M. D. (2017). Mesh dependence in PDE-constrained optimisation. In *Mesh Dependence in PDE-Constrained Optimisation*. Springer.

[Shao et al., 2011] Shao, G., Li, X., Ji, C., and Maeda, T. (2011). Focal mechanism and slip history of the 2011 M w 9.1 off the Pacific coast of Tohoku earthquake, constrained with teleseismic body and surface waves. *Earth, Planets and Space*, 63(7):559–564.

[Shao et al., 2020] Shao, G., Li, X., Ji, C., and Maeda, T. (2020). Okada parameters for the Tōhoku tsunami, as determined in Shao et al. 2011. `https://ji.faculty.geol.ucsb.edu/big_earthquakes/2011/03/0311_v3/result_c/static_out`. Accessed: 2020-08-07.

[Smith, 2016] Smith, R. C. (2016). *Numerical modelling of tsunami generated by deformable submarine slides*. PhD thesis, Imperial College London.

[Smith et al., 2016] Smith, R. C., Hill, J., Collins, G. S., Piggott, M. D., Kramer, S. C., Parkinson, S. D., and Wilson, C. (2016). Comparing approaches for numerical modelling of tsunami generation by deformable submarine slides. *Ocean Modelling*, 100:125–140.

[Stees and Shontz, 2018] Stees, M. and Shontz, S. M. (2018). An angular approach to untangling high-order curvilinear triangular meshes. In *International Meshing Roundtable*, pages 327–342. Springer.

[Stumm and Walther, 2009] Stumm, P. and Walther, A. (2009). Multistage approaches for optimal offline checkpointing. *SIAM Journal on Scientific Computing*, 31(3):1946–1967.

[Teukolsky et al., 1992] Teukolsky, S. A., Flannery, B. P., Press, W., and Vetterling, W. (1992). Numerical recipes in C. *SMR*, 693(1):59–70.

[Ulrich et al., 2019] Ulrich, T., Vater, S., Madden, E. H., Behrens, J., van Dinther, Y., Van Zelst, I., Fielding, E. J., Liang, C., and Gabriel, A.-A. (2019). Coupled, physics-based modeling reveals earthquake displacements are critical to the 2018 Palu, Sulawesi tsunami. *Pure and Applied Geophysics*, 176(10):4069–4109.

[Van Rijn, 1980] Van Rijn, L. C. (1980). *Storm surge barrier Oosterschelde-computation of siltation in dredged trenches: Semi-empirical model for the flow in dredged trenches*. Deltares, Delft, The Netherlands.

[Virtanen et al., 2020] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). Scipy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272.

[Wallwork et al., 2020a] Wallwork, J. G., Barral, N., Ham, D. A., and Piggott, M. D. (2020a). Anisotropic goal-oriented mesh adaptation in Firedrake. In *Proceedings of the 28th International Meshing Roundtable*, pages 83–100. Zenodo. DOI: `10.5281/zenodo.3653101`. URL: `https://doi.org/10.5281/zenodo.3653101`.

[Wallwork et al., 2021] Wallwork, J. G., Barral, N., Ham, D. A., and Piggott, M. D. (2021). Goal-oriented error estimation and mesh adaptation for tracer transport modelling. *EarthArXiv Preprint*. DOI: 10.31223/X56021. URL: https://doi.org/10.31223/X56021.

[Wallwork et al., 2020b] Wallwork, J. G., Barral, N., Kramer, S. C., Ham, D. A., and Piggott, M. D. (2020b). Goal-oriented error estimation and mesh adaptation for shallow water modelling. *Springer Nature Applied Sciences*, 2:1053–1063. DOI: 10.1007/s42452-020-2745-9. URL: https://rdcu.be/b35wZ.

[Wallwork and Clare, 2021] Wallwork, J. G. and Clare, M. C. A. (2021). Simulation code for 'Mesh Adaptation and Adjoint Methods for Finite Element Coastal Ocean Modelling' (updated following examiner corrections). DOI: 10.5281/zenodo.5147988. URL: https://doi.org/10.5281/zenodo.5147988.

[Wallwork et al., 2019] Wallwork, J. G., Hovland, P. D., Zhang, H., and Marin, O. (2019). Computing derivatives for PETSc adjoint solvers using algorithmic differentiation. *arXiv preprint arXiv:1909.02836 [cs.MS]*. URL: https://arxiv.org/abs/1909.02836.

[Walter, 2014] Walter, S. F. (2014). PyADOLC: a wrapper for ADOL-C. *GitHub repository*. URL: https://github.com/b45ch1/pyadolc.

[Wang et al., 2009] Wang, Q., Moin, P., and Iaccarino, G. (2009). Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation. *SIAM Journal on Scientific Computing*, 31(4):2549–2567.

[Warder, 2020] Warder, S. C. (2020). *Sensitivity analysis, uncertainty quantification and parameter estimation for a numerical tide and storm surge model*. PhD thesis, Imperial College London.

[Warder et al., 2020] Warder, S. C., Angeloudis, A., Kramer, S. C., Cotter, C. J., and Piggott, M. D. (2020). A comparison of Bayesian inference and gradient-based approaches for friction parameter estimation. *EarthArXiv*.

[Weller et al., 2016] Weller, H., Browne, P., Budd, C., and Cullen, M. (2016). Mesh adaptation on the sphere using optimal transport and the numerical solution of a Monge–Ampère type equation. *Journal of Computational Physics*, 308:102–123.

[Wengert, 1964] Wengert, R. E. (1964). A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):463–464.

[Wesseling, 2009] Wesseling, P. (2009). *Principles of computational fluid dynamics*, volume 29. Springer Science & Business Media.

[White, 1979] White, Jr, A. B. (1979). On selection of equidistributing meshes for two-point boundary-value problems. *SIAM Journal on Numerical Analysis*, 16(3):472–502.

[Wolfe, 1969] Wolfe, P. (1969). Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235.

[Wolfe, 1971] Wolfe, P. (1971). Convergence conditions for ascent methods. II: Some corrections. *SIAM Review*, 13(2):185–188.

[Yerry and Shephard, 1983] Yerry, M. and Shephard, M. (1983). A modified quadtree approach to finite element mesh generation. *IEEE Computer Graphics and Applications*, pages 39–46.

[Zhang et al., 2019] Zhang, H., Constantinescu, E. M., and Smith, B. F. (2019). PETSc TSAdjoint: a discrete adjoint ODE solver for first-order and second-order sensitivity analysis. *arXiv preprint arXiv:1912.07696*.

[Zhang and Sandu, 2014] Zhang, H. and Sandu, A. (2014). FATODE: a library for forward, adjoint, and tangent linear integration of ODEs. *SIAM Journal on Scientific Computing*, 36(5):C504–C523.

[Zhou et al., 2019] Zhou, T., Meng, L., Xie, Y., and Han, J. (2019). An adjoint-state full-waveform tsunami source inversion method and its application to the 2014 Chile-Iquique tsunami event. *Journal of Geophysical Research: Solid Earth*, 124(7):6737–6750.

[Zhu et al., 1997] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560.

[Zienkiewicz and Zhu, 1987] Zienkiewicz, O. C. and Zhu, J. Z. (1987). A simple error estimator and adaptive procedure for practical engineering analysis. *International Journal for Numerical Methods in Engineering*, 24(2):337–357.

[Zienkiewicz and Zhu, 1992a] Zienkiewicz, O. C. and Zhu, J. Z. (1992a). 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.

[Zienkiewicz and Zhu, 1992b] Zienkiewicz, O. C. and Zhu, J. Z. (1992b). 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.

Code Availability

The Firedrake and Thetis installations used to generate the experimental results presented in this thesis are archived on Zenodo at [**Ham et al., 2021**] and [**Kärnä et al., 2021**]. The adaptive solver framework and all simulation scripts are archived at [**Wallwork and Clare, 2021**]. See the `README` file therein for installation instructions and details on the location of test cases which appear in this thesis.