Goal-Oriented Error Estimation and Mesh Adaptation for Tracer Transport Modelling

Joseph G. Wallwork\textsuperscript{a,*}, Nicolas Barral\textsuperscript{b,c}, David A. Ham\textsuperscript{d}, Matthew D. Piggott\textsuperscript{a}

\textsuperscript{a}Department of Earth Science and Engineering, Imperial College London, London, SW7 2AZ, U.K.
\textsuperscript{b}University of Bordeaux, CNRS, Bordeaux INP, IMB, UMR 5251, 33400 Talence, France.
\textsuperscript{c}INRIA, IMB, UMR 5251, 33400 Talence, France.
\textsuperscript{d}Department of Mathematics, Imperial College London, London, SW7 2AZ, U.K.

Abstract

This paper applies metric-based mesh adaptation methods to advection-dominated tracer transport modelling problems in two and three dimensions, using the finite element package Firedrake. In particular, the mesh adaptation methods considered are built upon goal-oriented estimates for the error incurred in evaluating a diagnostic quantity of interest (QoI). In the motivating example of modelling to support desalination plant outfall design, such a QoI could be the salinity at the plant inlet, which could be negatively impacted by the transport of brine from the plant’s outfall. Four approaches are considered, one of which yields isotropic meshes. The focus on advection-dominated problems means that flows are often anisotropic; thus, three anisotropic approaches are also considered. Meshes resulting from each of the four approaches yield solutions to the tracer transport problem which give better approximations to QoI values than uniform meshing, for a given mesh size. The methodology is validated using an existing 2D tracer transport test case with a known analytical solution. Goal-oriented meshes for an idealised time-dependent desalination outfall scenario are also presented.

Keywords: anisotropy, mesh adaptation, error estimation, desalination outfall, Firedrake

1. Introduction

1.1. The Goal-Oriented Framework

Coastal ocean modelling is primarily concerned with understanding the hydrodynamics of coastal processes through numerical simulations. This includes the transport of passive and active tracers such as sediment, pollutants and salinity. Such tracer fields are typically modelled using equations of advection-diffusion type. In some cases, such as sediment transport, there is a two-way coupling with the hydrodynamics – the active case. In other cases, such as with pollutants suspended in water, it is sufficient to model the tracer as passive, with its transport being purely driven by the flow velocity. For problems in either category, there is often a relevant

\textsuperscript{*}Corresponding author

Email addresses: j.wallwork16@imperial.ac.uk (Joseph G. Wallwork), nicolas.barral@inria.fr (Nicolas Barral), david.ham@imperial.ac.uk (David A. Ham), m.d.piggott@imperial.ac.uk (Matthew D. Piggott)

Preprint submitted to Computer Aided Design January 4, 2022
quantity of interest (QoI), which a user of a model wishes to deduce from the solution field. For sediment transport problems, this may be the erosion of a particular bank. For pollutants, it might be the concentration in a region of particular environmental importance. The scenario considered in this paper is the outfall of a desalination plant, where the salinity at the inlet pipe is a useful QoI when using modelling to support the design process of locating the inlet and outlet locations. The goal of this work is to generate adaptive meshes, which minimise the error incurred in evaluating the chosen QoI. In turn, this optimises the speed vs accuracy of the calculation required at each step of an iterative design optimisation process.

Mesh adaptation typically requires two ingredients: an error estimator (defined on the current mesh) and a method which uses this error estimator to obtain an adapted mesh. Each of these components involves a design choice.

In line with the above discussion, the error estimation approach used in this work is framed around the QoI error. The class of error estimates associated with the QoI error are termed goal-oriented (or goal/output-based) in the literature. Many goal-oriented error estimates are based on the a posteriori error result established in [1, 2] – the dual-weighted residual (DWR). This result gives rise to first and second order error estimates for the QoI error, in terms of residuals and errors related to forward and adjoint equations. Other related formulations exist, such as the fully discrete ‘error correction’ approach in [3] and the a priori estimate due to [4].

By considering local contributions to the DWR error estimator, it is possible to identify regions of the domain where the use of increased mesh resolution will reduce the overall QoI error. Similarly, regions of the domain may be determined where it is admissible to decrease mesh resolution, with little impact on accuracy. Thus, goal-oriented mesh adaptation routines are often built upon element-wise error indicator fields. Dual weighted residual error estimation has been used to drive effective meshing strategies for a number of tracer transport applications (such as [5, 6, 7, 8, 9]). This work includes a comparison of three such approaches, as well as one based on an alternative error estimate. It also includes extensions of those goal-oriented approaches to the time-dependent case, as well as the first application of goal-oriented mesh adaptation to desalination outfall modelling.

In terms of the mesh adaptation model, element-wise error indicators lend themselves well to hierarchical adaptation methods, which refine on an element-by-element basis, such as quadtree and oct-tree type methods (see [2], for example). In these methods, mesh elements are refined or coarsened based on whether the local error indicator value meets pre-specified thresholds. Hierarchical mesh adaptation methods have been combined with an adaptive polynomial degree (p-adaptation), yielding hybrid hp-adaptation methods. See [8] for an example of goal-oriented hp-adaptation in the context of tracer transport modelling.

1.2. Accounting for Anisotropy

Standard hierarchical adaptation methods do not incorporate directionality, producing meshes whose elements are as isotropic as those in the original mesh. We focus on advection-dominated tracer transport, wherein the fluid flows (and/or regions of high tracer concentration variation) are anisotropic. Using appropriate anisotropic meshes, these direction-dependent features may be accurately resolved using relatively few (spatial) degrees of freedom (DoFs). As such, it makes sense to use a mesh adaptation method which can also account for this anisotropy. The literature contains a number of papers in which anisotropic adaptation algorithms are applied to time-dependent tracer transport problems, such as [10, 11, 12]. One approach is to perform hierarchical adaptation in an anisotropic manner, making single cuts (slices), instead of refining uniformly (see [13, 14, 10], for example). However, such an approach cannot fully control
element shape or orientation. This level of control is provided by the metric-based framework [15, 16, 17, 18, 11], which is used throughout this paper. In the metric-based framework, Riemannian metric fields act as continuous analogues for (inherently discrete) meshes [18].

Assuming that mesh adaptation is to be driven by metric fields, one of the main research questions of this paper is how to effectively construct such a metric from goal-oriented error estimates. In particular, effective approaches to anisotropic metric construction are sought.

The literature contains a number of notable efforts in the development of goal-oriented metrics. The simplest approach is to multiply the identity matrix by an appropriately scaled projection of the (scalar) DWR error (for examples, see [7, 19]). Whilst straightforward to construct, the resulting metrics do not account for anisotropy.

The first documented use of dual-weighted residual error estimation techniques for constructing anisotropic metrics is presented in [5]. Therein, an error result is derived which bounds the interpolation error of a scalar field on a mesh element by an expression involving element anisotropy coefficients and the Hessian of the field. The eigenvalues of a Hessian matrix are modified in order to enforce element sizes and include anisotropic stretching factors, as dictated by local error indicators. Hessian matrices are useful because they already contain information relating to the anisotropy of variables of interest. The work of [5] was extended to higher order elements and problems with strongly anisotropic features in [7] (2D) and [20] (3D). An element-based formulation is used, which fits naturally with the fact that the DWR yields element-based error indicators.

A simple approach which does not use the DWR estimator directly was introduced in [21]. By reinterpreting the error in the adjoint solution as an interpolation error, the authors of [21] argue for a formulation in which the Hessian of the approximate adjoint solution may be weighted by the strong residual of the forward equation. On its own, the formulation does not account for errors due to flux terms or stabilisation. In this paper, we propose a method for accounting for errors due to SUPG stabilisation. The results are encouraging, both in terms of mesh anisotropy and convergence of the QoI.

Like the approach of [21], the authors of [4] again reinterpret errors, leading to another Hessian-based formulation, which involves the combination of anisotropic metrics for both domain interior and domain boundary. Unlike those described above, this approach is not based on the (a posteriori) DWR error result; instead it is based upon an a priori error result, first derived in [4]. The formulation was extended to the time-dependent case in [22, 23]. A modification to account for source terms was proposed in [19], although that work did not account for boundary contributions. Both source terms and boundary contributions are properly accounted for herein.

Another notable metric-based mesh adaptation method that makes use of the goal-oriented framework is MOESS (Metric Optimization through Error Sampling and Synthesis) [24]. Given an error quantity (such as a DWR error estimator), MOESS performs local solves in order to establish how it varies across the domain. The gradient of the error function w.r.t. the Riemannian metric is approximated using an affine-invariant tensor manipulation framework. Making use of the duality between meshes and metrics [4], MOESS then solves a continuous optimisation problem in metric space, starting from a metric induced by the current mesh. The objective of the optimisation problem is to minimise local error contributions. This approach differs from those above, which instead derive anisotropic metrics based on error estimation results and heuristics.

1.3. Paper Summary

This paper builds upon the work of [19] and [25], which implemented goal-oriented mesh adaptation techniques in the Python-based finite element package, Firedrake [26]. The four goal-
oriented mesh adaptation routines implemented in those works are presented again here, with a number of simplifications in the original implementations overcome and with stabilisation terms properly accounted for. The methods are validated and compared against uniform refinement and standard 'Hessian-based' anisotropic mesh adaptation in the context of both 2D and 3D test cases with analytical solutions. This is the first time these approaches have been compared in the same software package. As mentioned above, a major novelty of this paper is its extension of the goal-oriented framework to the time-dependent case. This extension is a necessary step towards solving realistic tracer transport problems.

The implementation makes use of the tracer transport modelling framework developed in the Thetis ocean modelling project [27]. Software versions used for numerical experiments presented in this work are archived through Zenodo as [28, 29, 30]. The simulation code used to generate all results is archived at [31].

The remainder of the paper is arranged as follows. The metric-based mesh adaptation strategy is introduced in Section 2. Section 3 applies goal-oriented error estimation to an advection-diffusion problem. In addition, four strategies for the construction of goal-oriented metrics are presented and later used for numerical experiments in Sections 4 and 5. Section 4 considers the steady-state case, including validation experiments based on the TELEMAC-2D 'Point Discharge with Diffusion' test case, for which an analytical solution exists [32]. The 3D case is also considered. Section 5 considers the extension to time-dependent problems, presenting meshes and QoI convergence curves for an idealised desalination plant outfall scenario. Finally, conclusions are drawn in Section 6, with potential future work discussed.

2. Metric-Based Mesh Adaptation

As discussed in Section 1, this work utilises a metric-based approach. This means that mesh adaptation processes are driven by Riemannian metric fields of dimension $n \times n$, where $\Omega \subset \mathbb{R}^n$ is the spatial domain of interest. A Riemannian metric field, or metric, denoted $M = \{M(x)\}_{x \in \Omega}$, is a collection of symmetric positive-definite (SPD) linear forms defined pointwise. Encoded within the metric is local information on distances, from which anisotropic edge lengths and element volumes can be derived.

Metric-based mesh adaptation was first introduced in [15]. It uses a Riemannian metric space within the mesher in order to compute the necessary geometrical quantities. The aim of the adaptation process is to generate a unit mesh with respect to this Riemannian metric space. That is, all mesh elements are in some sense close to being isotropic, when viewed in the metric space. A major advantage of the metric-based approach is that it enables control of mesh anisotropy, meaning that not only element size, but also shape and orientation may be dictated.

Given an error estimator, we seek a mesh of the spatial domain such that the numerical solution of a PDE either: (a) achieves a certain level of error [16, 21]; or (b) minimises the interpolation error for a given number of mesh vertices [4, 33]. In this work we follow the latter approach. For this, we need to establish the error estimator and express it as a metric.

As mentioned in the introduction, the metric-based framework provides a continuous analogue for the (inherently discrete) mesh. One example where this is relevant is in the notion of metric complexity – the continuous analogue of the mesh vertex count, defined by [18]

$$C(M) := \int_{\Omega} \sqrt{\det(M(x))} \, dx.$$  \hspace{1cm} (1)

See [18] for further details on other correspondences.
Throughout this paper, we use the notation $\mathcal{H}$ when referring to meshes and $K$ to denote elements thereof. Occasionally, a cell size function $h$ is mentioned as a subscript, $\mathcal{H}_h$, if instructive. The edge set of element $K$ is denoted $\partial K$, with outward normal vectors $\hat{n}_K$. The indicator function which is unity on element $K$ and zero elsewhere is denoted $1_K$.

2.1. Hessian-Based Metric

Suppose that we are interested in bounding the $L^p$ interpolation error associated with a piecewise linear and continuous ($P^1$) approximation, $\Pi_h u$, of a scalar field of interest, $u$, for some $p \in [1, \infty)$. Suppose $u$ is sufficiently smooth that its Hessian, $H(u) = \begin{bmatrix} \frac{\partial^2 u}{\partial x^2} & \frac{\partial^2 u}{\partial x \partial y} \\ \frac{\partial^2 u}{\partial y \partial x} & \frac{\partial^2 u}{\partial y^2} \end{bmatrix}$, (in 2D) (2) is well defined. Building upon earlier works, including [34, 35, 36], it is demonstrated in [17] that the interpolation error is related to an approximation of the Hessian, $H$, by

$$
\|u - \Pi_h u\|_{L^p(\Omega)} \leq n C_T^{-\frac{2}{p}} \left( \int_{\Omega} \det(|H|) \frac{dx}{\|\mathbf{w}\|} \right)^{\frac{1}{2p}},
$$

(3)

provided that we have a unit mesh with respect to a metric of complexity $C_T > 0$. That is, a metric $\mathcal{M}$ such that $C(\mathcal{M}) = C_T$.

Since $H$ is symmetric, it has an orthogonal eigen-decomposition, $H = \mathbf{V} \Lambda \mathbf{V}^T$, meaning it makes sense to take the absolute value as

$$
|H| = \mathbf{V} |\Lambda| \mathbf{V}^T.
$$

(4)

Taking the absolute value implies that $|H|$ is SPD and is therefore a valid metric. The justification for doing so is that we are primarily interested in the magnitude of errors, as opposed to their sign.

Note that formula (3) breaks down in the case where $u$ is locally linear, since this implies a null Hessian. In order to account for this degeneracy, it is common practice to threshold the metric’s eigenvalues before assembling (4). Typically, minimum and maximum tolerated metric magnitudes are imposed (see [16, eq. (17)]). It is also possible to enforce a maximum tolerated anisotropy (see [16, eq. (16)]).

In practice, $u$ is only known in a discrete sense, meaning its Hessian must be approximated using a recovery technique, typically involving the solution of an auxiliary PDE. In this work, we recover gradient information using Clément interpolation [37] (i.e. volume-averaging the finite element gradient over adjacent elements). We recover Hessian information by applying the same procedure again.

For further details on Hessian-based mesh adaptation, see [15, 16, 35, 17, 18, 38].

2.2. $L^p$-Normalisation

For a metric to be applicable to any problem, it is necessary to scale it appropriately. The scaling process also allows a degree of control over various properties of the resulting meshes, such as the number of vertices and the extent to which the meshes have a multi-scale nature.

An $L^p$-normalisation strategy of order $p \in [1, \infty)$ for a metric $\mathcal{M}$ is typically governed by either a target interpolation error level, $\epsilon_T > 0$, or a target metric complexity, $C_T > 0$ [17]. In the
former case, a metric is sought which admits interpolation error smaller than $\epsilon_T$. Here, we use the latter approach, whereby we seek a metric whose complexity matches the target value, $C_T$.

For time-independent problems, this constraint is met by applying the formula (5):

$$M_{L^p} := C_T^2 \left( \int_{\Omega} \det(M)^{\frac{p}{2}} \ dx \right)^{\frac{2}{p}} \det(M)^{\frac{-1}{2p}} M. \quad (5)$$

By specifying larger values for the target complexity $C_T$ in (5), we allow heightened mesh size in return for reduced interpolation error. Note that the metric normalisation (5) is closely related to the earlier presentations in [34, eq. (67)] and [35, eq. (5)].

Taking the limit $p \to \infty$ yields the strategy referred to as $L^\infty$ normalisation. Whilst commonly used, the $L^\infty$ normalisation strategy focuses resolution around only the most prominent features, using very high levels of mesh refinement surrounding them. Lower order $L^p$ normalisation strategies yield more multi-scale meshes, with $p \approx 1$ able to capture all scales.

In the case of time-dependent problems, the normalisation approach should be applied over the time period, $(T_{\text{start}}, T_{\text{end}}]$, too. Assuming a (not necessarily fixed) timestep, $\Delta t > 0$, this amounts to introducing a time integral in (5) as [17]

$$M_{L^p} := C_T^2 \left( \int_{T_{\text{start}}}^{T_{\text{end}}} \frac{1}{\Delta t} \left( \int_{\Omega} \det(M)^{\frac{p}{2}} \ dx \right) \ dr \right)^{\frac{2}{p}} \det(M)^{\frac{-1}{2p}} M. \quad (6)$$

In the time-dependent case, it should be noted that $C_T$ is now a space-time complexity, given by integrating (1) in time. The discrete analogue is the number of mesh vertices summed over all timesteps. As such, the target space-time complexity can be thought of as a ‘DoF budget’ for the simulation.

2.3. Combining Metric Information

Suppose now that we have finitely many Riemannian metrics, each of which captures some aspect of the PDE solution, or an error estimate thereof. For instance, they might be Hessians of different components of a vector solution field.

The most straightforward way to combine these metrics is to take the average (on an entry-by-entry basis). That this yields a metric follows from the definition of positive-definiteness. In some cases, it is appropriate to do this in an affine-invariant way, as in [24]. Alternatively, metrics can be combined using intersection (also known as superposition). Unlike averaging, this combination method always yields meshes with more elements than those which would arise from the constituent metrics. For details on metric intersection, see pp.3778–3779 of [16].

An investigation into the properties of metric intersection and averaging was made on pp.131-138 of [39]. In the context of unsteady adaptation applied to advection problems, metric intersection was found to deal better with shocks. However, metric averaging was found to be more effective at resolving sharp angles and small scale features.

It is necessary to scale metrics (using the methods in Subsection 2.2 for example) before combining them. In the case where two metrics have been constructed from the Hessians of different fields, for example, there is no guarantee that these second derivative matrices are of the same order of magnitude. Normalising before combination means that the Hessians may be averaged or superimposed in a meaningful way. This comes at the cost of the metrics being constrained by the target complexity.
2.4. Metric Gradation

Metric gradation techniques are usually applied to metric fields to ensure that the sizes they prescribe at neighbouring vertices do not vary by more than a specified threshold, $\beta$. That is, the ratio of the prescribed sizes is bounded from above by $\beta$. This has the effect that meshes resulting from the metric-based mesh adaptation routine do not have sudden changes in resolution, which could act as artificial internal boundaries in the flow. For details on the metric gradation algorithm used, see [33]. In this work, we set $\beta = 1.4$, following the recommendations of that paper.

2.5. Software

Anisotropic mesh adaptation is achieved in Firedrake using Pragmatic [40]. This thread-parallel C++ toolkit takes as input a triangular or tetrahedral mesh and a metric field with DoFs defined at its vertices and returns a mesh which is adapted to the metric. The adapted mesh is derived from the input one through a series of local mesh modifications (refinement, coarsening, edge/face swapping and Laplacian smoothing) that optimise the lengths of the edges, as well as a quality functional. For details on the parallel implementation of Pragmatic, see [41]. For model inter-comparison studies involving Pragmatic, see [42, 43, 44].

Pragmatic and Firedrake are interfaced through PETSc [40, 45, 46, 47], which provides the unstructured mesh representation that underpins Firedrake’s own mesh concept and which is used to solve the linear and nonlinear systems associated with finite element problems. In detail, high-level error estimate information is defined based on Riemannian metric fields constructed in Firedrake from FEM solutions of the PDE and its adjoint. Along with the mesh, the data associated with this field is passed to PETSc, where it is represented as a PETSc Vec. Pragmatic receives the data as an array of doubles. The mesh is adapted accordingly and propagated back to the Firedrake level.

Mesh-to-mesh data transfer is necessary for transient mesh adaptation problems. In this work, we use conservative projection [48], enabled in Firedrake by its interface with libsupermesh [49, 50]. A major advantage of this approach is that integrated quantities are conserved across mesh adaptation steps.

Goal-oriented error estimation and mesh adaptation is facilitated by Pyroteus [30]. Pyroteus uses Pragmatic as the adaptation backend and allows the automatic solution of forward and adjoint problems over sequences of meshes, with discrete adjoint equations solved using dolfin-adjoint [51] and mesh-to-mesh data transfer using libsupermesh.

3. Goal-Oriented Error Estimation

3.1. Tracer Transport Model

Advection-Diffusion System. This paper focuses on the application of goal-oriented error estimation and mesh adaptation to tracer transport problems. The prognostic variable, $c$, is the concentration of a passive tracer immersed in a fluid with velocity, $\mathbf{u}$, and SPD diffusivity tensor, $D$. For a source term $S$, consider the steady-state tracer transport problem,

$$
\begin{aligned}
\mathbf{u} \cdot \nabla c - \nabla \cdot (D \nabla c) &= S & \text{in } \Omega \\
\frac{\nabla c \cdot \hat{n}}{c} &= g_{N} & \text{on } \partial \Omega_{N} \\
\end{aligned}
$$

(7)

The domain boundary, $\partial \Omega$, is assumed to be piecewise smooth, decomposed into a disjoint union of Dirichlet and Neumann components as $\partial \Omega = \partial \Omega_{D} \cup \partial \Omega_{N}$. We also allow outflow boundaries,
given by natural boundary conditions, but these correspond to Neumann conditions in this case and therefore may be absorbed into $\partial \Omega_N$.

**Finite Element Method.** Let $V$ be a function space containing the exact solution, $c$, of (7). For a Continuous Galerkin (CG) discretisation, we have a weak formulation

$$a(c, v) = L(v), \quad \forall v \in V,$$

comprised of a bilinear form $a : V \times V \to \mathbb{R}$ and linear form $L : V \to \mathbb{R}$,

$$a(c, v) := (c(u \cdot \hat{n}_K), v)_{\partial \Omega} - (u \cdot c, \nabla v) + \langle D \nabla c, \nabla v \rangle - \langle D \nabla c \cdot \hat{n}_K, v \rangle_{\partial \Omega},$$

$$L(v) := (S, v) - (g_D(u \cdot \hat{n}_K), v)_{\partial \Omega} + (g_N, v)_{\partial \Omega_N}.$$

Here $\langle \cdot, \cdot \rangle$ denotes the usual $L^2$ inner product on $\Omega$. For other spaces $S$, we use the notation $\langle \cdot, \cdot \rangle_S$. In addition, denote $\| \cdot \| := \langle \cdot, \cdot \rangle$ and $\| \cdot \|_S := \langle \cdot, \cdot \rangle_S$.

Integration by parts has been applied in deriving (9), to apply boundary conditions and reduce the order of derivatives. Whilst it is common practice to implement Dirichlet boundary conditions in a strong sense, Thetis is primarily a discontinuous Galerkin code and so they are imposed weakly in (9). The weak formulation is also important for the construction of goal-oriented error estimators in Subsection 3.3.

Equation (7) gives rise to the strong residual,

$$\Psi(c) := S - u \cdot \nabla c + \nabla \cdot (D \nabla c).$$

Similarly, equation (8) gives rise to the weak residual,

$$\rho(c, v) := L(v) - a(c, v).$$

For a finite dimensional subspace $V_h \subset V$, the Galerkin approximation,

$$\rho(c_h, v) = L(v) - a(c_h, v) = 0, \quad \forall v \in V_h,$$

admits a finite element solution, $c_h \in V_h$. In this work we use a CG method with polynomial degree 1, i.e. $V_h = P_1$. That is, the tracer concentration is assumed to be piecewise linear on each mesh element, with continuity imposed across inter-element boundaries.

**Stabilisation.** Stabilisation terms are usually added to (8) in order to control under- and overshoots that are typical in advection-dominated problems. *Streamline Upwind Petrov-Galerkin (SUPG) stabilisation* is applied, implying an additional term $\langle \Psi(c) \cdot (\tau u \cdot \nabla v) \rangle$, where $\tau > 0$ is a stabilisation parameter.

In this work, we make the standard choice of stabilisation parameter [52],

$$\tau_K := \frac{h_K}{2\|u\|_2} \min \left(1, \frac{\text{Pe}_h}{3}\right), \quad \text{where} \quad \text{Pe}_h := \frac{h_K\|u\|_2}{2D}.$$

is the mesh Péclet number and $h_K$ is a cell size measure for element $K$. We only consider the case of constant isotropic diffusion, so $D = D_I$ for some $D > 0$. A typical choice of cell size is the cell diameter, i.e. the circumradius for triangular and tetrahedral elements. For isotropic elements, this is a suitable choice and it scales with element volume. However, it can give very large values for highly anisotropic elements. In the triangular case, for example, the area of a
highly anisotropic element is much smaller than an element with all edges the same length as the anisotropic element’s longest edge. For anisotropic elements, a more suitable measure of cell size is that advocated in [53]. If \( J_K \) is the Jacobian of the map from a reference element, \( \hat{K} \), to some mesh element \( K \in \mathcal{H} \), then cell size may be defined as the minimum eigenvalue of \( \sqrt{J_K^T J_K} \). For further details on the SUPG stabilisation strategy, see [52][53].

**Linear Solver Strategy.** This paper strongly advocates the use of anisotropic meshes for advection-dominated transport problems with direction-dependent features. The goal-oriented metrics presented in Subsection 3.6 can result in meshes whose elements are isotropic (aspect ratio \( O(1) \)), fairly anisotropic (aspect ratio \( O(10) \)), strongly anisotropic (aspect ratio \( O(100) \)) and extremely anisotropic (aspect ratio \( O(1,000) \), or higher). At the point at which elements become extremely anisotropic, the linear solver strategy for finite element problems built upon such meshes can be compromised. As such, as mentioned in Subsection 2.1, it is common practice to impose a maximum tolerated anisotropy value such that elements are substantially anisotropic, but not extremely so, whereby solver issues could arise. The numerical experiments in this paper allow extremely high anisotropy in order to get a sense for the anisotropic manner that different Riemannian metrics seek to adapt the mesh. Subfigure 4d, for example, shows an example of a mesh with maximum aspect ratio over 3,000. If extremely anisotropic meshes are to be used, the linear solver strategy must be appropriately robust. As such, in this work we use a direct method, which applies a full LU decomposition [54, 55]. This is not a scalable approach, but ensures robustness and allows us to avoid the difficult problem of finding effective preconditioning strategies for tracer transport problems defined on such meshes. Doing so is proposed as future work. One promising solver strategy, proposed in [56], involves using an incomplete LU decomposition as a preconditioner and adopting an adaptive stopping criterion.

### 3.2. Adjoint Tracer Transport Problem

Given the strong residual (10), the ‘forward equation’ (in this case the advection-diffusion equation) may be written in ‘residual form’ as

\[
\Psi(c) = 0.
\]

In addition, we introduce a scalar diagnostic which is held to be important for some use case – the QoI, \( J : V \rightarrow \mathbb{R} \). The adjoint equation associated with (14) and \( J \) is given by

\[
\left( \frac{\partial \Psi}{\partial c} (c) \right)^T c^* = \frac{\partial J^T}{\partial c}, \quad \text{where} \quad c^* \in V
\]

is the adjoint solution. Due to the linearity of the advection-diffusion equation, the derivative term on the LHS of (15) is actually independent of the forward solution. Therefore, the adjoint equation may be solved independently of the forward equation, provided the QoI is also linear. This is not true for nonlinear problems.

The derivation of an adjoint problem (15) involves making a choice: should the equation be differentiated and then discretised (i.e. the continuous adjoint approach), or discretised and then differentiated (i.e. the discrete adjoint approach)? In this work, we make use of the discrete adjoint functionality made available in Firedrake by the dolfin-adjoint library [51]. Doing so avoids tedious and error-prone manual calculations and ensures continuity of functional gradients [57].
3.3. Estimating QoI Error

Due to the goal-oriented error analysis of [2], we have the classical result,

\[ J(c) - J(c_h) = \rho(c_h, e^* - c_h^*) + R, \]

(DWR)

where the remainder term, \( R \), is quadratic in the forward and adjoint errors, \( e = c - c_h \) and \( e^* = c^* - c_h^* \). Equation (DWR) may be used to define goal-oriented error estimators, as outlined in Subsections 5.3 and 5.4.

A ‘second-order’ result may also be derived [2]: error estimates deduced from this result are applied to anisotropic goal-oriented mesh adaptation for tracer transport problems in [19]. In this paper, however, we focus on the first order result (DWR), termed the dual-weighted residual. Note that the remainder term vanishes in the case where the prognostic equation is linear and the QoI is linear [2]. By linearity of (7), (DWR) gives an exact representation of the QoI error, provided that the QoI is linear.

The error result stated above is global, in the sense that it quantifies QoI error incurred over the whole domain. Its use within a mesh adaptation routine requires localisation, so that we may seek to deduce the error incurred in each element, \( K \in \mathcal{H} \). Clearly, the weak residual may be decomposed as a sum of contributions from each mesh element. Therefore, element-wise error indicators, \( \eta_K \), may be extracted from (DWR) as

\[ \eta_K := |\rho(c_h, e^* - c_h^*)|_K, \quad K \in \mathcal{H}. \]

(16)

Summing over all elements yields a global error estimator. This provides an upper bound for the QoI error, by the triangle inequality.

Recall that integration by parts was applied over the whole domain when deriving the variational formulation [9]. Error indicators of the form (15) may be decomposed using a second integration by parts. Now the domain of integration is each mesh element, since we seek element-wise error indicators. Inter-element flux terms are introduced, as well as boundary terms,

\[ \rho(c_h, e^*)_K = \langle \nabla \psi^0(c_h), e^* \rangle_K + \langle \psi^0(c_h), e^* \rangle_{\partial K \cap \partial \Omega}, \]

(17)

where \( \psi^0 \) is the residual of any weakly imposed boundary conditions and \( \psi^{flux} \) corresponds to flux terms between elements. The magnitude of the first term tells us how well the PDE is solved across the domain. The second term conveys how well the boundary conditions have been satisfied. The third term is related to the smoothness of the solution field across inter-element boundaries. For CG methods (such as used here) the flux terms arise purely due to the integration by parts. Discontinuous discretisations contribute additional flux terms, by construction (for example, see [25]).

For the stabilised version of the CG formulation given in [8], integrating by parts yields goal-oriented error indicators

\[ \rho(c_h, e^*)_K = \langle \Psi(c_h), e^* \rangle_K + \langle \nabla c_h \cdot \hat{n}_K - g_N, e^* \rangle_{\partial K \cap \partial \Omega} + \langle (g_D - c) \mathbf{u} \cdot \hat{n}_K, e^* \rangle_{\partial K \cap \partial \Omega} + \langle \Psi(c_h), \mathbf{u} \cdot \nabla e^* \rangle_K. \]

(18)

That is, the terms on the RHS correspond to those given in (17), along with an additional stabilisation error term. When summed over all elements of the mesh, the inner product over \( \partial K \setminus \partial \Omega \) in (18) corresponds to a flux jump term. Observe that if \( c_h \) is replaced with the (assumed smooth) exact solution of (7) then this estimator vanishes, as we might hope.
3.4. Error Estimate Evaluation

An important aspect of goal-oriented error estimation is the treatment of the (clearly unknown) adjoint error term, \( e^* \), which appears in [DWR]. One approach is to approximate \( e^* \) by solving the adjoint problem again in a globally enriched space. Enrichment can mean incrementing the polynomial degree (\( p \)-refinement), subdividing elements (\( h \)-refinement) or a combination thereof. In any case, the enrichment process leads to a heightened DoF count. In [19], an enriched adjoint solution was obtained using both \( h \)- and \( p \)-refinement. This was found to be effective for goal-oriented error estimation and mesh adaptation for the same steady-state advection diffusion test case as considered in the following section. However, it comes with a far higher computational cost than solving the forward problem in the base space, which is clearly undesirable.

In this work, we instead utilise the ‘difference quotient’ approach described in [2]. No enrichment is required; instead, the error indicator is modified and evaluated using only solution fields from the base space. Combining flux and boundary terms as \( \psi(\cdot) := \psi^f(\cdot) + \psi^{ff}(\cdot) \), application of the Cauchy-Schwarz inequality yields the upper bound

\[
|\rho(c_h, e^*)|_K \leq \left( \|\Psi(c_h)\|_K + h_K^{\frac{1}{2}} \|\phi(c_h)\|_{\partial K} \right) \left( \|e^*\|_K + h_K^{\frac{1}{2}} \|e^*\|_{\partial K} \right),
\]

Again, we may interpret \( c_h^* \) as the projection of \( e^* \) into the finite element space. Thus, for some interpolation constant, \( C > 0 \), it satisfies [58]

\[
\omega_K = \|e^* - \Pi_h e^*\|_K + h_K^2 \|e^* - \Pi_0 e^*\|_{\partial K} \leq Ch_K^2 \|\nabla^2 e^*\|_K .
\]

We follow [2] in approximating the Laplacian \( \nabla^2 e^* \) by that the Laplacian \( \nabla^2 c_h^* \), as obtained using a recovery technique. In practice, it is difficult to determine \( C \), meaning that (19) cannot be used for the evaluation of global error estimates. Nonetheless, its localisation to individual elements can be readily used in a mesh adaptation algorithm, where overall scaling constants are irrelevant. One possible localisation reads

\[
\eta_K^{DQ} := \left( \|\Psi(c_h)\|_K + h_K^{\frac{1}{2}} \|\phi(c_h)\|_{\partial K} \right) \|\nabla^2 c_h^*\|_K , \quad K \in \mathcal{H} .
\]

In order to account for stabilisation, we make use of the fact that SUPG is a Petrov-Galerkin method and use the modified error indicator

\[
(\eta_{SUPG}^{DQ})_K := \left( \|\Psi(c_h)\|_K + h_K^{\frac{1}{2}} \|\phi(c_h)\|_{\partial K} \right) \|\nabla^2 (c_h^* + \tau u \cdot \nabla c_h^*)\|_K .
\]

Further, we interpret \( h_K \) as the anisotropic cell size measure used within the SUPG scheme.

3.5. ‘Isotropic DWR Metric’

The simplest way to create a metric from a scalar error indicator such as (16) is to use it to scale an identity matrix. Doing so yields a \( P_0 \) tensor field. In particular, it takes the form of a diagonal matrix with positive diagonal entries in every mesh element. Therefore, the field is SPD and hence a valid metric. However, the metric-based mesh adaptation formulation used in this paper requires the metric to be defined in \( P_1 \) space. Therefore, an additional interpolation step is required. Given an appropriate interpolation operator \( \Pi_1 : P_0 \rightarrow P_1 \), an isotropic metric may be defined as

\[
\mathcal{M}^{DWR} := \Pi_1 \left( \sum_{K \in \mathcal{H}} \eta_K \mathbb{I}_K \right) \mathbb{I}_n ,
\]

\[
\eta_K := \left( \|\Psi(c_h)\|_K + h_K^{\frac{1}{2}} \|\phi(c_h)\|_{\partial K} \right) \|\nabla^2 (c_h^* + \tau u \cdot \nabla c_h^*)\|_K .
\]
where $I_n$ is the identity matrix in $n$ dimensions. In this work, we again use Clément interpolation [37].

Metrics of the form (23) allow control of element sizes under mesh adaptation, but not their shape or orientation. Therefore, we refer to (23) as the isotropic DWR metric. Metrics should always be normalised (for example, using the methods outlined in Subsection 2.2), in order to make them relevant to the problem at hand.

3.6. Anisotropic Goal-Oriented Error Estimation

As discussed in Section 1, the literature contains a number of approaches for extending the goal-oriented error estimation framework to obtain anisotropic meshes. Some notable contributions are applied to the tracer transport modelling problem in the following.

3.6.1. ‘Anisotropic DWR Metric’

The first anisotropic metric we consider follows the work of [5, 7]. Like (23), this approach is based on the a posteriori error result (DWR) and uses an element-wise formulation. Where in Section 2 Hessians are constructed on a vertex-wise basis, let $H_K$ denote the Hessian of the forward solution constructed on element $K$. It also admits an orthogonal eigen-decomposition with eigenvector matrix $V_K$ and eigenvalues $\{\lambda_{K,i}\}_{i=1}^n$. Without loss of generality, assume $|\lambda_{K,1}| = \cdots = |\lambda_{K,n}| > 0$. The stretching factors [20, eq. (28)]

$$s_{K,i} := |\lambda_{K,i}|^{-1} \left( \prod_{j=1}^n |\lambda_{K,j}| \right)^{\frac{1}{n}}, \quad i = 1, \ldots, n$$

are element shape deformations in the directions defined by the eigenvectors. In the 2D case, $s_{K,1} = \sqrt{|\lambda_{K,2}|/\lambda_{K,1}}$ and $s_{K,2} = \sqrt{|\lambda_{K,1}|/\lambda_{K,2}}$ are the minimal and maximal anisotropy ratios in the element-wise Hessian.

Whilst the normalisation strategy used for this metric is also focused on metric complexity, it differs from the approach described in Subsection 2.2 in that it incorporates error indicators. As described in [7], an element volume $|\tilde{K}|$ is sought such that the metric complexity is smaller than some target complexity, $C_T$. Within the element-wise framework, this optimisation problem is shown in [59] to be solved by the choice

$$|\tilde{K}| := \frac{|K|}{C_T} \left( \sum_{K \in H} \eta^{-\alpha}_{K} \right)^{-\frac{1}{\alpha}},$$

where $|K|$ is the volume of element $K$. The tunable parameter $\alpha \geq 1$ may be interpreted as akin to the normalisation degree in $L^p$ normalisation. As with the isotropic metric, we use the SUPG-adjusted form of the error indicator [22] for $\eta_K$ in (25).

An anisotropic metric is constructed from (25) by reassembling the element-wise Hessian with a modified eigenvalue matrix. Again, a vertex-based metric is obtained by projection as [20, eq. (33)]

$$M^{DWR} := \Pi_1 \left( \sum_{K \in H} \left( \frac{|\tilde{K}|}{|K|} \right)^{\frac{1}{\alpha}} V_K \text{diag}(s_{K,1}^{-1}, \ldots, s_{K,n}^{-1}) V_K^T \right),$$

where $|\tilde{K}|$ is the volume of the reference element, $\tilde{K}$. This decomposition is instructive, since it separates out the components which control element size ($|\tilde{K}|/|K|$), element orientation ($V_K$) and element shape (the inverse of the stretching factor matrix).
3.6.2. ‘Weighted Hessian Metric’

The second anisotropic metric we consider is again derived from the dual-weighted residual. Like the approach described above, this metric - first presented in [21] - also makes use of the Hessian to incorporate anisotropy. This is done through the following reinterpretation of (DWR):

\[ J(c) - J(c_h) \approx \langle \Psi(c_h), e^* - \Pi_h c^* \rangle \leq \|\Psi(c_h)\| \|e^* - \Pi_h c^*\|, \quad (27) \]

where the inequality follows by Cauchy-Schwarz. That is, the interface term is neglected and the approximate adjoint solution is understood as a projection of the true adjoint solution into the finite element space, \( V_h \). Using the relation (3) between interpolation error and the Hessian, an anisotropic metric may be constructed from (27) by weighting the Hessian of the adjoint solution with the projection of the forward residual into \( P_1 \) space:

\[ M_{WH} := \Pi_1 \left( \sum_{K \in H} \|\Psi_h(c_h)\|_K \|H(c_h^*)\| \right), \quad (28) \]

In practice, a \( P_1 \) Hessian is recovered for the approximate adjoint solution. Henceforth, (28) is termed the ‘weighted Hessian’ (WH) anisotropic metric.

Since the adjoint error is already approximated using the Hessian, the discussion on the evaluating error estimates in Subsection 3.4 is not of concern for this metric.

3.6.3. ‘Weighted Gradient Metric’

For the purposes of defining the final metric, use \( \Psi_s(\cdot) \) to denote the strong residual in finite element space, as opposed to the ‘true’ strong residual, \( \Psi(\cdot) \). Instead of being based on (DWR), this metric uses an alternative error result derived in [4]:

\[ J(c) - J(c_h) = \langle (\Psi_h - \Psi)(c), c^* \rangle + \tilde{R}, \quad (29) \]

where \( \tilde{R} \) is a remainder term which now involves interpolation errors on \( V_h \) and the adjoint error, \( e^* \). As with the WH metric, we begin by noting that (29) contains what may be interpreted as an interpolation error. However, where before this was an interpolation error in the adjoint solution, here it is an interpolation error in the PDE itself.

Suppose the PDE can be written in conservative form as

\[ \Psi(c) = \nabla \cdot \mathbf{F}(c) = 0, \quad (30) \]

where \( \mathbf{F} = (F_1, \ldots, F_n) \). Integrating (29) by parts yields

\[ J(c) - J(c_h) \approx \langle (\mathbf{F} - \Pi_h \mathbf{F})(c), \nabla c^* \rangle - \langle \nabla \cdot (\mathbf{F} - \Pi_h \mathbf{F})(c), c^* \rangle_{\partial \Omega}, \quad (31) \]

where \( \mathbf{F} \) is a residual for the boundary conditions. That is, \( \Psi(c) = 0 \) if and only if \( c \) solves the PDE on the domain interior and \( \mathbf{F}(c) = 0 \) if and only if the boundary conditions are satisfied exactly. Application of the Cauchy-Schwarz and triangle inequalities yields the bound

\[ |J(c) - J(c_h)| \lesssim \sum_{i=1}^n \| (\mathbf{F}_i - \Pi_h \mathbf{F}_i)(c) \| \left\| \frac{\partial c^*}{\partial x_i} \right\| + \| \nabla \cdot (\mathbf{F} - \Pi_h \mathbf{F})(c) \|_{\partial \Omega} \| c^* \|_{\partial \Omega}. \quad (32) \]
Approximate bound \([32]\) involves interpolation errors in both \(\overline{F}\) and the components \(F_i\). In \([4]\), it is argued that interpolation error bounds of the form \([3]\) may be applied on a component-by-component basis, in order to construct anisotropic metrics. Hessians are weighted by the adjoint or components of its gradient, as appropriate:

\[
H_{\text{volume}} := \sum_{i=1}^{n} |H(F_i(c))| \left| \frac{\partial c^*}{\partial x_i} \right|, \quad H_{\text{surface}} := |c^*| \left| H(\hat{n} \cdot F(c)) \right|,
\]

The resulting metrics are then intersected on the boundary, giving

\[
M_{\text{WG}} := \left\{ \begin{array}{ll}
H_{\text{volume}}(x) \cap |H_{\text{surface}}(x)| & x \in \partial \Omega \\
|H_{\text{volume}}(x)| & x \in \Omega
\end{array} \right.
\]

Note that the error result \([29]\) which this metric is based upon is \textit{a priori}, whilst all previously considered metrics are \textit{a posteriori}. We make this classification by following \([4]\) in interpreting the a posteriori standpoint as assuming knowledge of errors incurred on an existing mesh and the a priori standpoint as assuming knowledge of the PDE solution. That is, the a posteriori metrics involve residuals evaluated on the current mesh, whereas the a priori metric \([34]\) involves exact solutions for the forward and adjoint PDEs, making no reference to finite element solutions. As in the WH approach, these exact values are approximated by finite element solutions in practice, replacing instances of \(c\) and \(c^*\) with \(c_h\) and \(c^*_h\), respectively. The difference in classification has to do with the derivation of the methods, rather than implementation details. Henceforth, \([34]\) is referred to as the ‘weighted gradient’ (WG) anisotropic metric, due to the gradient term in the domain interior contribution.

Regarding the discussion on the evaluation of error estimates in Subsection 3.4, we follow the authors of \([4]\) by approximating gradients in the exact adjoint solution by gradients in the numerical approximation thereof.

### 3.6.4. Modifications

Whilst the anisotropic methods considered are inspired by the work of \([5, 21, 4]\), there are some notable differences between the original implementations and those used in this work, as outlined in the following.

The construction of the WG metric \([34]\) requires the prognostic equation \([7]\) to be in conservative form. Assuming the velocity field to be divergence-free, the tracer transport equation may be expressed in terms of a potential functional \(\mathcal{F}\) as

\[
\nabla \cdot \mathcal{F}(c) = S, \quad \mathcal{F}(c) = uc - D\nabla c.
\]

However, the source term, \(S\), requires special treatment. In \([19]\), the metric contribution in the domain interior is constructed by summing \([33]\) and adding a term that weights the Hessian of the source term with the adjoint solution. In this work, we prefer to use metric intersection:

\[
M_{\text{WG}} := |H(F_1(c))| \left| \frac{\partial c^*}{\partial x} \right| \cap |H(F_2(c))| \left| \frac{\partial c^*}{\partial y} \right| \cap |H(S)\ | |c^*|.
\]

The first two terms on the RHS correspond to \([33]\). For the boundary term, recall that we consider three types of boundary segment: Neumann, Dirichlet and open. Neumann conditions are usually weakly imposed in finite element methods, giving

\[
\hat{n} \cdot \overline{F}(c)|_{\partial \Omega_N} = g_N - D\nabla c \cdot \hat{n}.
\]
For CG methods, Dirichlet conditions are usually strongly imposed, contributing zero error. However, Thetis uses a weak formulation, which implies the contribution
\[
\hat{n} \cdot \vec{F}(c)|_{\partial \Omega_D} = (c - g_D)\hat{u} \cdot \hat{n}.
\]
(38)

As described above, (37) and (38) convey the extent to which the Neumann and Dirichlet conditions are enforced. Boundary integrals over open boundaries are treated as natural boundary conditions. Therefore, these segments do not contribute to the surface metric.

It is worth remarking that, whilst stabilisation errors have been accounted for by the DWR indicators used in both isotropic and anisotropic metric constructions, they are not considered in the WG method. Similarly as with the modified error indicator formulation (22), we can account for stabilisation in the WH metric by making use of the fact that SUPG is a Petrov-Galerkin method. That is, we use the construction

\[
M_{\text{SUPG}}^{\text{WH}} := \Pi_1 \left\{ \sum_{K \in \mathcal{H}} \| \Psi_h(c_h) \|_{K} \right\} |_{\mathcal{H}} (c_h^* + \tau \hat{u} \cdot \nabla c_h^*). \]
(39)

As with the isotropic metric, the WG (36) and WH (39) metrics are normalised using \(L^p\) normalisation, as in [5]. Unless otherwise stated, the normalisation order \(p = 1\) is assumed, since it admits multi-scale meshes, whilst being able to capture sharp features, such as point sources. The anisotropic DWR method has inbuilt normalisation, for which we follow the advice of the authors of [7] and set \(\alpha = 6\).

4. Steady-State Goal-Oriented Mesh Adaptation

First, consider the time-independent case.

4.1. Mesh Adaptation Approach

Given an initial mesh \(\mathcal{H}_0\) and a desired metric complexity \(C_T > 0\), we construct a sequence of meshes \(\{\mathcal{H}_i\}_{i \in \mathbb{N}}\) by iteratively solving the forward problem (7) and its (discrete) adjoint and constructing an appropriate metric from their solution fields. Algorithm 1 provides a workflow representation.

```
Given target metric complexity \(C_T > 0\);
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 \(\mathcal{H}_i\);
    Construct metric \(M_i\), normalising for target complexity \(C_T\);
    Apply metric gradation to \(M_i\);
    Adapt the mesh under \(M_i\) to obtain \(\mathcal{H}_{i+1}\);
    Increment \(i\);
end
```

**Algorithm 1:** Mesh adaptation routine for time-independent problems.
Convergence is attained when the relative change in one of the QoI value or mesh element count falls below some tolerance, taken here as 0.5%. A maximum iteration count of 35 is also imposed. For a relatively isotropic \( H_0 \), we find that at least three mesh iterations are required in order to introduce anisotropy into meshes produced by Algorithm 1. Therefore, three iterations are performed before the convergence criteria are checked.

To improve robustness, we ‘spin up’ the target complexity over three iterations, starting with a base value, \( C_B \), which roughly corresponds to the initial mesh. That is, at iteration \( i \), we use target complexity \( bC_B + (1 - b)C_T \), where \( b = \max((3 - i)/3, 0) \).

4.2. Point Discharge Test Case in Two Dimensions

Numerical experiments in this subsection focus on the ‘Point Discharge with Diffusion’ test case from the TELEMAC-2D validation document version 7.0 [32]. This steady-state advection-diffusion problem is defined in a rectangular domain, \( \Omega = [0, 50] \times [0, 10] \) m\(^2\), with prescribed uniform flow velocity, \( \mathbf{u} = (u_1, u_2) = (1, 0) \) m s\(^{-1}\), and isotropic diffusivity tensor, \( D = D_I \), where \( D = 0.1 \) m\(^2\)s\(^{-1}\). A tracer is injected into the flow at \( x_0 \in \Omega \), and specified in the continuous problem via a Dirac delta function, i.e. spatially a point source. An inflow (Dirichlet) condition, \( c = 0 \), is imposed at the left-hand boundary, along with Neumann conditions on the North and South boundaries and an outflow on the RHS.

The only difference between the test case presented here and the original one in [32] is that we follow [19] in locating the point source at \( x_0 = (2, 5) \), rather than \( x_0 = (1, 5) \). We consider the same initial mesh as in [32], comprised of a uniform grid of 4,000 isosceles right-angled triangles.

4.2.1. Calibration

An advantage of this test case is that it has an analytical solution, [32]

\[
c_{\text{analytical}}(x) := \frac{q}{2\pi D} \exp \left( \frac{u_1 x_1}{2D} \right) K_0 \left( \frac{u_1 \|x - x_0\|_2}{2D} \right),
\]

where \( q \) denotes the inflow discharge at the source and \( K_0 \) is the zeroth order modified Bessel function of the second kind. Following [32], we set \( q = 100 \) g l\(^{-1}\). See [60, Subsection 3.6.1] for a derivation of (40).

Note that \( K_0 \) blows up at zero, in correspondence with (40) becoming infinite at \( x_0 \). Therefore, we approximate the \( \ell_2 \) norm in (40) using

\[
d(x) := \max(||x - x_0||_2, r),
\]

which has the effect of introducing a plateau in the region \( B_r(x_0) \), for some \( r > 0 \). Here we use the notation

\[
B_r(y) := \{x \in \Omega \mid ||x - y||_2 \leq r\},
\]

for the ball of radius \( r > 0 \), centred at \( y \in \Omega \).

Point sources cannot generally be easily represented in numerical models. Therefore, we use the radius \( r \) again when representing the source. In [19], the source is represented by an indicator function of narrow radius, which effectively replaces the point source with a pipe of radius \( r \). In this work, we instead adopt a Gaussian approximation,

\[
S(x) := q \exp \left( -\frac{||x - x_0||^2}{r^2} \right),
\]

16
meaning that $S$ is smooth, rather than discontinuous.

In [19], a calibrated value of $r$ was established by trial and error. Here we take a more rigorous calibration approach using gradient-based optimisation. The aim of this optimisation is to minimise the functional

$$J_{calibration}(c) := \int_{\Omega \setminus B_r(x_0)} (c - c_{analytical})^2 \, dx,$$

subject to the constraint that $r$ remains strictly positive. The $c$ used to evaluate (44) is obtained by solving the tracer transport equation with some value of $r$. Due to the plateau enforced by (41), we do not seek to match the analytical solution within the source region.

Running the optimisation using L-BFGS-B [61] on a fine uniform mesh with 1,024,000 elements gives the calibrated radius $r = 5.605917$ cm. This high level of mesh refinement was chosen to ensure that the point source is well represented, even for very small radii. The resulting finite element solution on this fine mesh is compared against the analytical solution in Figure 1. An initial qualitative observation is that the stabilised FEM approximation on the fine uniform mesh provides an excellent approximation to the analytical solution. It appears that the only region where there is a noticeable difference between the analytical and finite element solutions is near the boundary for $x \in [30, 50]$. This is in agreement with what may be observed in the TELEMAC-2D solution given in [32]. See [60, Subsection 3.6.1] for more details on the optimisation experiment.

4.2.2. Quantities of Interest

The functional (44) used for calibration is nonlinear and does not have a simple dependence on the control parameter, $r$. For the goal-oriented mesh adaptation experiments in the following, linear functionals of the form

$$J_i(c) := \int_{\Omega \setminus R_i} c \, dx = \int_{R_i} c \, dx, \quad R_i = B_{R_i}(x_i)$$

are used. Evaluating (45) amounts to measuring the tracer concentration in a ‘receiver region’ centred at $x_i \in \Omega$. 

Figure 1: Analytical and finite element solutions for the ‘Point Discharge with Diffusion’ test case, with calibrated radius $5.605917$ cm for the source term. Each field is presented in $P_1$ space on a uniform mesh with 1,024,000 elements and 513,921 vertices.
Figure 2: Discrete adjoint solutions for the ‘Point Discharge with Diffusion’ test case corresponding to QoIs of the form (45). Each field is presented in $P^1$ space on a uniform mesh with 1,024,000 elements and 513,921 vertices.

As in the similar steady-state advection-diffusion test case examined in [21], we consider two scenarios: one where the receiver is directly downstream from the source and one where it is offset to one side of the channel. We choose $x_1 = (20, 5)$ in the former instance and $x_2 = (20, 7.5)$ in the latter. Goal-oriented mesh adaptation seeks a mesh which allows an accurate approximation of the tracer concentration integrated over the receiver region.

For a related application in environmental science, consider the inlet and outlet pipes of a desalination plant. The inlet pipe brings sea water into the plant, from region $R_i$. Salt is extracted from this water and the remaining salt residues are released back into the ocean through the outlet pipe, depositing at $x_0$. An undesirable – but entirely possible – situation is where a significant quantity of salt from the outlet pipe is later taken back into the plant at the inlet, making the task of desalination more difficult. For this application we do not necessarily care about the wider dispersal of high salinity water, but rather achieving an efficient and accurate calculation of the salinity at the inlet. If salt is interpreted as a passive tracer in the fluid flow, we can model its advection and diffusion using (7) and quantify its concentration at the inlet using (45). Through goal-oriented mesh adaptation, we can construct meshes which well approximate the salinity at the inlet, whilst retaining relatively few DoFs. An idealised (time-dependent) tidally varying desalination plant outfall scenario is considered in Subsection 5.1.

The derivative of the QoI integrand w.r.t. the tracer concentration provides a source term for the adjoint equation. In this case, the derivative is the indicator function $1_{R_i}$, meaning the adjoint solution differs between the aligned and offset cases. Figure 2 presents discrete adjoint solutions associated with the SUPG stabilised forward problem.

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.

4.2.3. Application of Goal-Oriented Mesh Adaptation

A finite element implementation has been presented and calibrated against an analytical solution. We now progress to perform goal-oriented error estimation and mesh adaptation within this setup, using the estimators and metrics described in Subsections 3.5 and 3.6. A $L^1$-normalised
‘Hessian-based’ metric is included, to facilitate comparison between goal-oriented methods and a standard (non-goal-oriented) method.

Figure 3: Example meshes for the aligned configuration of the ‘Point Discharge with Diffusion’ test case.

**Aligned Source and Receiver.** Figure 3 shows example meshes of similar overall resolution for each method, in the case of aligned source and receiver. The element count, vertex count (i.e. DoF count) and mean and maximum element aspect ratios of the mesh in each subfigure are stated in its caption.

The Hessian-based mesh is adapted based on curvature of the tracer concentration, meaning
that resolution is concentrated around the point source and in outward-propagating bands going downstream. Moderate-to-high aspect ratios can be observed across the whole domain (mean 3.9) and there are no especially coarse elements. The goal-oriented meshes, on the other hand, have low mesh resolution downstream of the receiver region, with the majority of the mesh resolution being in the region between source and receiver. These observations are in line with the fact that the QoI value is insensitive to the downstream dynamics.

As may be expected, the isotropic DWR mesh consists of triangles with low aspect ratios (mean 1.4), whilst the anisotropic meshes contain higher aspect ratios. The mesh due to the WG approach has notably high anisotropy. Its maximum aspect ratio 836.8 – observed in Subfigure 3e – is two orders of magnitude higher than in the isotropic case. The maximum anistropies for the other methods are one order of magnitude higher than the isotropic case. The anisotropic DWR and WG meshes retain anisotropy downstream because they inherit their anisotropy from the Hessian of the forward solution (i.e. from Subfigure 3a) and the Hessian of the forward potential, respectively. The WH metric, on the other hand, inherits its anisotropy from the Hessian of the adjoint solution, which is near zero downstream of the receiver and therefore has low curvature.

One interesting anisotropic feature of the WG mesh is the strongly anisotropic refinement on the top and bottom boundaries, where Neumann conditions are imposed, as well as the inflow boundary, where a Dirichlet condition is weakly imposed. The metric construction involves intersecting the ‘volume’ and ‘surface’ contributions are intersected on these boundaries, with the surface metric conveying how well these boundary conditions are satisfied. As well as including information on boundary condition error, the weighting by the adjoint solution means that fine mesh resolution is only deployed in regions where the QoI value is sensitive to perturbations in the forward solution. This explains why there is more boundary resolution in the left-hand (upstream) part of the domain than the right-hand (downstream) part. Almost all of its extremely anisotropic elements are near to these boundaries.

For the choices of normalisation parameters presented in Figure 3, all approaches focus at least some mesh resolution around the source region. The concentration is highest for WG, probably due to the additional term included in the metric in (36). The goal-oriented approaches all also deploy at least moderate mesh resolution surrounding the receiver region, helping them to better capture the QoI. In the cases of the DWR metrics, there is higher resolution around the receiver than the source. The Hessian-based method does not resolve the receiver, since it has no knowledge of the QoI. The isotropic and WH meshes in Subfigures 3b and 3d have particularly coarse elements downstream because they are weighted by second derivatives of the adjoint solution, which is uniformly zero in that region.

Offset Source and Receiver. Figure 4 shows example meshes for the four goal-oriented metric generation methods, in the case of offset source and receiver. The Hessian-based mesh is not shown, since it is independent of the QoI and so would simply be Subfigure 3a shown again. The same qualitative observations as made above can be applied again for the meshes in the offset case. An even higher maximum aspect ratio of 3166.1 is reported for the WG metric (a fivefold increase). This is likely due to the skew introduced by the offset receiver and the proximity of the receiver to the top boundary. The maximum aspect ratio of the anisotropic DWR mesh is also increased (more than threefold), likely for the same reasons. The other meshes have comparable anisotropy to the aligned configuration.

Convergence Analysis. Figure 5 shows convergence curves for the QoI in both aligned and offset cases, as well as error curves evaluated against the fixed mesh value on a very fine mesh
with 4,096,000 elements and 2,051,841 vertices. Each data point is generated by specifying a particular target complexity and running Algorithm 1 for a given metric. Whilst we could plot convergence with increasing element count, we opt to measure mesh size using the number of vertices, because this equates to the DoF count in $P_1$ space. Vertex count is also more closely related to the metric complexity (i.e. the control of the experiment).

In each configuration, repeated uniform refinement of the initial uniform mesh allows convergence to a value close to the analytical QoI value. Since there is model error at play, we choose to plot relative discretisation errors (against the converged QoI values on uniform meshes), rather than total errors (against QoIs due to the analytical formulae).

Under uniform refinement, over 100,000 DoFs are required to yield less than 1% error. All four goal-oriented approaches also exhibit convergence to these values. However, for the isotropic DWR, anisotropic DWR and WH approaches, the 1% error level is crossed using an order of magnitude fewer mesh DoFs. That is, with over 10,000 DoFs, these adaptive strategies consistently yield QoI errors smaller than 1%. The WG approach appears to require more DoFs in order to achieve this in the aligned configuration, although not as many as uniform meshing.
Its convergence properties are more desirable in the offset configuration. The reason that WG requires more DoFs than the other methods probably has to do with the amount of resolution it expends on the source term and boundaries. As target complexity (and hence DoF count) is increased, QoI errors due to the Hessian-based approach generally decrease, although the convergence to the fixed mesh value is not as strong as the goal-oriented approaches.

It is interesting to observe that, whilst the goal-oriented meshes take rather different forms, all four approaches yield improved convergence properties and the isotropic DWR, anisotropic DWR and WH approaches have error curves of remarkably similar magnitude. That it is possible to achieve this kind of improvement on uniform meshing without using enrichment methods to approximate the adjoint error demonstrates the effectiveness of the difference quotient indicator formulation. This formulation makes for a much more computationally efficient goal-oriented mesh adaptation strategy than solving the adjoint problem in a globally enriched space, as in
Other improvements on that work are that the experiments presented in this section use a formulation of the WG metric that accounts for boundary terms and that stabilisation errors are here accounted for in the DWR metrics and WH metric.

The differences between convergence curves in Subfigure 5a are not significant enough to draw conclusions regarding optimality of one particular metric. As such, we should take other factors into consideration.

Computational Cost. Tables 1 and 2 compare CPU times and fixed point iteration counts for each method. CPU time is a better proxy for the computational cost of a mesh adaptive simulation than DoF count or element count and is often what the user is more interested in.

Table 1 considers the coarsest mesh (DoF count < 10^3) for each run in the aligned configuration. In terms of total run time, using a fixed mesh is clearly the cheapest approach, as is to be expected. The adaptive runs all take 6 to 9 fixed point iterations to converge to a 0.5% relative tolerance on the element count and/or QoI value. This value tends to be lower when more DoFs are permitted. One reason for this is that 0.5% of 2,000 elements is just ten elements, meaning that two meshes must be almost identical in order to attain element count convergence. For a mesh with 200,000 elements, on the other hand, relative convergence would be attained if the mesh gained or lost fewer than 1,000 elements.

Interestingly, isotropic DWR takes less time than the Hessian-based method, even though it does not involve adjoint solves. This is probably mostly has to do with the fact that the Hessian-based approach requires more fixed point iterations to converge.

The ‘metric’ CPU time columns exclude forward solves and the mesh adaptation step, but include adjoint solves, derivative recovery and metric construction. Notice that the Hessian-based approach is the cheapest in terms of metric construction because it does not involve an adjoint solve. The isotropic and WH approaches are next cheapest, involving an adjoint solve and one Laplacian/Hessian recovery per iteration. The costs of evaluating the strong residual and flux terms and performing metric normalisation are all small in comparison to these components. Note that the derivative recovery step would not necessarily be as cheap if a different method than Clément interpolation were used. We found that doing Hessian recovery using double L^2 projection, solved with a mixed finite element method, the cost is much higher. Whilst it does not involve any more recovery steps, the anisotropic DWR approach is more expensive because it involves the solution of eigenvalue problems on each element. The WG method is most expensive because it involves multiple recovery steps: Hessians of two components of the potential on the

<table>
<thead>
<tr>
<th>Approach</th>
<th>CPU time [s]</th>
<th>#It.</th>
<th>Reason</th>
<th>CPU time per it. [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Metric</td>
<td></td>
<td>Total</td>
</tr>
<tr>
<td>Uniform</td>
<td>0.3214</td>
<td>-</td>
<td>1</td>
<td>0.3214</td>
</tr>
<tr>
<td>Hessian-based</td>
<td>4.6397</td>
<td>1.1985</td>
<td>9 Elements</td>
<td>0.5155</td>
</tr>
<tr>
<td>Isotropic DWR</td>
<td>3.9672</td>
<td>1.9412</td>
<td>6 Elements</td>
<td>0.6612</td>
</tr>
<tr>
<td>Anisotropic DWR</td>
<td>6.5124</td>
<td>3.6535</td>
<td>9 QoI</td>
<td>0.7236</td>
</tr>
<tr>
<td>Weighted Hessian</td>
<td>6.0724</td>
<td>2.7362</td>
<td>9 Elements</td>
<td>0.6747</td>
</tr>
<tr>
<td>Weighted gradient</td>
<td>11.7445</td>
<td>7.2674</td>
<td>9 Elements</td>
<td>1.3049</td>
</tr>
</tbody>
</table>

Table 1: CPU times for the first run of each method in the aligned configuration documented in Subfigure 5a (DoF count < 10^3). The total CPU time is shown (including setup, forward solves, metric construction and mesh adaptation), as well as times just for metric construction. ‘It.’ stands for iteration. Timings are generated by averaging over five runs.
Table 2: CPU times for the fifth run of each method in the aligned configuration documented in Subfigure 5a (DoF count $\approx 10^5$). The total CPU time is shown (including setup, forward solves, metric construction and mesh adaptation), as well as times just for metric construction. ‘It.’ stands for iteration. Timings are generated by averaging over five runs.

<table>
<thead>
<tr>
<th>Approach</th>
<th>CPU time [s]</th>
<th>#It. Reason</th>
<th>CPU time per it. [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total Metric</td>
<td></td>
<td>Total Metric</td>
</tr>
<tr>
<td>Uniform</td>
<td>18.0420</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Hessian-based</td>
<td>82.8931</td>
<td>7 Elements</td>
<td>11.8419 0.1564</td>
</tr>
<tr>
<td>Isotropic DWR</td>
<td>36.9570</td>
<td>5 QoI</td>
<td>7.3914 0.7103</td>
</tr>
<tr>
<td>Anisotropic DWR</td>
<td>39.0104</td>
<td>6 QoI</td>
<td>7.8021 0.8184</td>
</tr>
<tr>
<td>Weighted Hessian</td>
<td>37.9416</td>
<td>5 QoI</td>
<td>7.5883 0.6193</td>
</tr>
<tr>
<td>Weighted gradient</td>
<td>145.0380</td>
<td>5 QoI</td>
<td>29.0076 3.9362</td>
</tr>
</tbody>
</table>

domain interior, the Hessian of the source term, the Hessian of the boundary potential and the gradient of the adjoint solution. At this resolution, metric construction takes the majority of the CPU time for the anisotropic DWR and WG metrics.

Given how low the resolution is for the runs documented in Table 1, the timings are not representative of realistic cases. For example, importing Firedrake can take as much as 20% of the runtime. To get a better picture, consider Table 2, which corresponds to the finest adaptive meshes and fifth uniform mesh (DoF count $\approx 10^5$). With around 100,000 DoFs, import timings are negligible and more than half of the time is taken by mesh adaptation. It is clear from Table 2 that metric construction contributes only a small part of the overall computational cost. It is especially small for the Hessian-based approach, which does not include adjoint solves. For the goal-oriented approaches, metric construction is between 5% and 15% of the overall cost.

Interestingly, the isotropic DWR, anisotropic DWR and WH metrics are fairly competitive at this resolution, taking approximately double the time required for uniform meshing, but giving QoI errors of a similar or lower magnitude. Given that the final mesh is of around the same resolution as the uniform mesh and the adaptive runs converge in five or six iterations, this seems somewhat counter-intuitive. However, the main reasons that it is possible is that the target complexity is spun up over the first three iterations, meaning that at least the first two meshes are coarser than the final one. That the Hessian-based method takes longer than the isotropic DWR, anisotropic DWR and WH methods for the same number of iterations is probably because more mesh adaptation operations are typically required to go from a uniform mesh to a strongly anisotropic mesh than an isotropic one. This may also explain why the WG method takes by far the longest amount of time. Indeed, profiling experiments (not shown) indicate that the vast majority (around 70%) of CPU time is spent in the mesh adaptation step.

Another reason that the Hessian-based approach takes longer is that it terminates due to converged element count, rather than QoI value. If the algorithm terminates due to QoI value then the mesh adaptation step on the final iteration is skipped. It is encouraging that all four goal-oriented methods terminate due to converged QoI value, since this is what we are more interested in.

4.2.4. Summary

The convergence plots for the isotropic DWR, anisotropic DWR and WH metrics are all desirable improvements on uniform meshing and provide effective goal-oriented mesh adaptation methods for this steady-state tracer transport problem. The Hessian-based and WG metrics tend
to require more DoFs in order to attain high QoI accuracy. On the basis of CPU time comparisons, the isotropic DWR metric with a difference quotient error indicator formulation is a competitively cheap approach and should be advocated. As problem size is increased, the anisotropic DWR and WH approaches become cheaper than standard Hessian-based adaptation. As such, if anisotropic meshes are desired then we advocate either of these methods. If anisotropy is not a requirement then the isotropic DWR method should be sufficient.

4.3. Point Discharge Test Case in Three Dimensions

Problem definition. The previous subsection validated four metric-based goal-oriented mesh adaptation strategies for a 2D steady-state tracer transport problem. These metrics can also be readily applied in 3D. We demonstrate this by a simple extension of the ‘Point Discharge with Diffusion’ test case. Consider now the cuboid domain, \( \Omega = [0, 50] \times [0, 10] \times [0, 10] \text{m}^3 \), source centred at \( x_0 = (2, 5, 5) \) and offset receiver centred at \( x_R = (20, 7.5, 2.5) \). Fluid velocity remains solely in the positive \( x \)-direction and constant isotropic diffusion is again applied. The initial mesh is comprised of 30,000 uniform tetrahedra and has 6,171 vertices.

Adaptive Mesh Snapshots. Figure 6 shows examples of adapted meshes in the offset configuration. The boundary contribution has been neglected for the WG metric, so anisotropic meshing is not observed on the boundary surfaces. However, many of the other mesh features carry over from the 2D case. Firstly, all meshes focus resolution between source and receiver, with the isotropic DWR and WH meshes having particularly coarse resolution downstream. The maximum element volumes stated for these approaches are two orders of magnitude higher than for the other methods. Secondly, the isotropic mesh has elements with very low aspect ratio (mean 1.5), whereas the others have larger values. It does have at least one highly anisotropic element (max. aspect ratio 53.8), but this is likely due to geometric constraints in an awkward part of the mesh. Finally, the WG mesh again focuses resolution most strongly around the source, due to the additional term that accounts for it. Due to this, the WG mesh has a minimum element size that is two orders of magnitude smaller than any of the other approaches.

Element volumes vary by between seven and ten orders of magnitude, making the methods considered truly multi-scale approaches. Multi-scale meshes are important in desalination plant outfall modelling, because the pipe diameters, bathymetric features and tidal forcings all exist on different orders of magnitude.

Convergence Analysis. Figure 7 shows the relationship between QoI value and DoF count for the goal-oriented adaptive methods in 3D. The four methods tend to the same value with around \( 10^5 \) DoFs, thereby validating them for the 3D problem. Some of the methods converge before then, such as WG, which appears to reach the converged value with 30,000 DoFs. The curve due to the anisotropic DWR metric is also desirable because its values are relatively close even at low overall resolution.

Computational Cost. Table 3 compares CPU times and fixed point iteration counts for the final run of each method. Note that the corresponding meshes are those shown in Figure 6. The captions for that figure and the final data points in Figure 7 show that the associated DoF counts are similar, meaning that we can claim that the resolutions are similar and that it is fair to make a CPU time comparison.

Each of the methods terminates due to convergence of the QoI value, except for anisotropic DWR, which converges w.r.t. element count. For these particular runs, isotropic DWR and WH...
Figure 6: Example meshes for the offset configuration of the 3D extension of the ‘Point Discharge with Diffusion’ test case. The mean aspect ratio and minimum and maximum volumes of the elements in each mesh are stated in the captions.

are the fastest, taking both fewer iterations and shorter CPU time. Iteration count is key because target complexity is spun up over three iterations, which have a lower cost, since the corresponding meshes have lower overall resolution. That is, not all iterations are equal. This probably ac-
counts for the increased runtime of the WG approach. The fact that anisotropic DWR converges w.r.t. element count means that it actually contains as many mesh adaptation steps as for WG. Further profiling experiments (not shown) determined that approximately 91% of the anisotropic DWR runtime was taken by Pragmatic – a similar value to the other approaches. However, this does not account for the heightened cost compared with WG. The profiling technology currently available in Firedrake accounts for routines in PETSc, Firedrake and those manually labelled by the user. Understandably, its reach does not extend into external packages such as Pragmatic. In order to further investigate the expensive anisotropic DWR run, we would need to do separate profiling experiments.

Under each goal-oriented metric, the time taken to solve adjoint equations, recover derivatives and construct Riemannian metrics is a very small proportion of the overall runtime. As indicated above, the majority of the runtime is taken by the mesh adaptation routine for this 3D problem of reasonable mesh resolution. However, there are scenarios where this would not necessarily be the case. Firstly, the proportion of runtime taken by mesh adaptation would likely also differ if a different adaptation tool were used than Pragmatic. Secondly, the direct method used to solve the forward and adjoint advection-diffusion equation has computational cost $O(N^3)$, which becomes significant for problems with higher resolution than shown here. This is the reason that uniform refinement results are not shown – it is infeasible to use high enough resolution to

<table>
<thead>
<tr>
<th>Approach</th>
<th>CPU time [s]</th>
<th>#It.</th>
<th>Reason</th>
<th>CPU time per it. [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Metric</td>
<td></td>
<td>Total</td>
</tr>
<tr>
<td>Isotropic DWR</td>
<td>1151.9418</td>
<td>18.2107</td>
<td>5 QoI</td>
<td>230.3884</td>
</tr>
<tr>
<td>Anisotropic DWR</td>
<td>2012.8337</td>
<td>76.5470</td>
<td>6 Element</td>
<td>335.4723</td>
</tr>
<tr>
<td>Weighted Hessian</td>
<td>1194.4568</td>
<td>26.8204</td>
<td>5 QoI</td>
<td>238.8914</td>
</tr>
<tr>
<td>Weighted gradient</td>
<td>1989.3048</td>
<td>72.8750</td>
<td>7 QoI</td>
<td>284.1864</td>
</tr>
</tbody>
</table>

Table 3: CPU times for the final run of each method in the offset configuration documented in Figure 7 (DoF count = $O(10^5)$). The total CPU time is shown (including setup, forward solves, metric construction and mesh adaptation), as well as times just for metric construction. ‘It.’ stands for iteration. Timings are generated by averaging over five runs.
observe convergence of the QoI.

Summary. To conclude the 3D extension experiments, we find that all four goal-oriented methods yield strongly multi-scale meshes, which enable convergent QoI approximations with approximately $10^5$ DoFs. The isotropic DWR and WH metrics are found to have the lowest computational cost, although it would be beneficial to perform further investigation. Similarly as with the 2D experiments, we conclude that WH seems to be a good choice if anisotropy is desired and that the isotropic DWR metric should be sufficient otherwise.

5. Time-dependent Goal-Oriented Mesh Adaptation

We have already seen that there exist many different ways to construct Riemannian metrics from goal-oriented error estimators for steady-state problems. Moving to the time-dependent case involves making more design choices. The first is how frequently the mesh should be adapted. Whilst frequent adaptation enables close tracking of dynamical features in the flow, it comes with increased interpolation error and additional computational overheads associated with the number of adaptation steps and the computation of the required metric tensors.

Extending to the time-dependent case means adding a time derivative term in (7):

$$
\begin{align*}
\frac{\partial c}{\partial t} + u \cdot \nabla c - \nabla \cdot (D \nabla c) &= S \quad \text{in } \Omega \\
\frac{D \nabla c \cdot \hat{n}}{\nabla \cdot (D \nabla c)} &= g_N \quad \text{on } \partial \Omega_N \\
c &= g_D \quad \text{on } \partial \Omega_D.
\end{align*}
$$

(46)

Where previously the fluid velocity $u$ was prescribed as a constant, here it is time-dependent. In practice, it is usually given by the output of a hydrodynamics model. Under a depth-averaged approximation, for example, boundary forcings due to tides perturb the free surface elevation, $\eta$, which in turn drives $u$. We interpret the tracer field $c$ as passive, meaning that it does not feed back into the hydrodynamics. As such, we perform goal-oriented error estimation for the tracer model only. For its application to Thetis’ shallow water component, see [25].

We begin by decomposing the time period $(0, T]$ into a disjoint union of $k$ subintervals,

$$(0, T] = \bigcup_{j=1}^{k} W^j, \quad W^j := (t^{j-1}, t^j), \quad 0 = t^0 < t^1 < \cdots < t^k = T.$$

(47)

The idea is that each subinterval, $W^j$, is associated with exactly one mesh, $\mathcal{H}^j$. That is, we effectively have a ‘fixed mesh’ simulation on each subinterval, with mesh-to-mesh data transfer used when going to the next subinterval. This reduces the problem of time-dependent mesh adaptation to that of determining meshes that are optimal (in some sense) for their simulated subinterval, under the constraint of a given global metric complexity. As iterations of the adaptation loop progress, these meshes are updated, as indicated by a subscript index, $\mathcal{H}^j_i$. In this work, the time subintervals are chosen to have a fixed length, for simplicity. The number of subintervals is chosen so that there are enough to capture the moving dynamics, but not so many as to accrue significant interpolation error. The timestep, $\Delta t$, is also chosen to be fixed, for simplicity. However, it is possible to account for adaptive timesteping in this framework (see [23]).

Given the temporal decomposition (47), one possible extension of Algorithm 1 to the time-dependent case is presented in Algorithm 2. For each iteration of the fixed point iteration loop, we again solve the forward and adjoint problems and use their solutions to construct metrics and thereby perform mesh adaptation. Note that there is an initial forward solve over the whole time period. In general, this requires interpolation of solution fields between meshes. For this, we use
conservative interpolation [43]. Solution data are stored during the first forward solve, in order to provide initial conditions for the forward equations as solved on each subinterval in reverse.

Given target space time complexity \( N_{st} > 0 \);
Given initial meshes \( \{ \mathcal{H}_j^{0} \}_{j=1}^{k} \);
Set \( i := 0 \);
while not converged do
  for \( j = 1, \ldots, k \) do
    Interpolate fields onto \( \mathcal{H}_j^{i} \) from formulae/data/previous mesh;
    Solve forward problem over subinterval \( \mathcal{W}_j \) on \( \mathcal{H}_j^{i} \);
    Store forward solution from the final timestep of \( \mathcal{W}_j \);
  end
  Evaluate QoI and check for its convergence;
  for \( j = k, \ldots, 1 \) do
    Load forward solution for first timestep of \( \mathcal{W}_j \);
    Solve forward problem over subinterval \( \mathcal{W}_j \) on \( \mathcal{H}_j^{i} \);
    Solve adjoint problem over subinterval \( \mathcal{W}_j \) on \( \mathcal{H}_j^{i} \);
    Construct metric \( M_j^{i} \) over subinterval \( \mathcal{W}_j \);
  end
  Normalise \( \{ M_j^{i} \}_{j=1}^{k} \) based on target complexity \( N_{st} \);
  for \( j = 1, \ldots, k \) do
    Apply gradation to \( M_j^{i} \);
    Adapt mesh \( \mathcal{H}_j^{i} \) using \( M_j^{i} \) to obtain \( \mathcal{H}_j^{i+1} \);
  end
  Increment \( i \);
end

**Algorithm 2:** Mesh adaptation routine for time-dependent problems.

Since we have a sequence of metrics and a target _space-time_ complexity, this involves space-time normalisation, for which we use \( L^p \) normalisation as in formula (6). The approach is based on that proposed in [22, 23], which uses WG type metrics. Recall that goal-oriented error indicator (18) for the steady-state tracer transport problem is comprised of strong residual, boundary, flux and stabilisation terms. In the time-dependent case, it is more complicated. Applying Crank-Nicolson (for example) over timestep \( (t^{(n)}, t^{(n+1)}) \) yields

\[
\rho^{(n)}(c_h, e^*)|_K := \frac{1}{M} \left( e_h^{(n+1)}, (e^*)_{lax^*(n)} \right)_K - \frac{1}{M} \left( e_h^{(n)}, (e^*)_{lax^*(n)} \right)_K - \frac{1}{2} \rho \left( e_h^{(n+1)}, (e^*)_{lax^*(n)} \right)_K - \frac{1}{2} \rho \left( e_h^{(n)}, (e^*)_{lax^*(n)} \right)_K \tag{48}
\]

That is, we have contributions from each time level, as well as a time derivative term. In practice, we choose to interpret the adjoint error as the average of its values at times \( t^{(n)} \) and \( t^{(n+1)} \), for simplicity of the extension to the different metrics. For the isotropic DWR metric, we simply scale (48) by the identity matrix. The anisotropic DWR metric uses this indicator in the formulation of (26), but with the eigendecomposition computed from the average of the Hessians of the forward solutions over \( (t^{(n)}, t^{(n+1)}) \). For WH, we have the strong residual of the advection-diffusion
experiments, a source is released from pipe has diameter $r$. The time interval is divided into 40 uniform subintervals. $\Delta t = 232, so that there are 2,000 timesteps in total. In the terminology and notation of the previous problem definition. The final numerical experiment in this paper considers desalination outfall in an idealised tidally varying channel. Consider the rectangular domain $\Omega = [-500, -500] \times [-1500, 1500]$ m². For simplicity, the North and South boundaries, $\partial \Omega_N$, are to be interpreted as impassible cliffs. Tidal forcings are imposed on the water surface elevation on the east and west boundaries, $\partial \Omega_{E/W}$. The salinity is set to a constant background value, $c_b = 39$ g l⁻¹.

Assuming constant bathymetry, viscosity and drag parameters and simple sinusoidal boundary forcings for the surface elevation, the hydrodynamics over the whole domain may be approximated using a sinusoidal forcing for the $x$-component of the velocity:

$$u(x, t) := (U \sin(\omega t), 0), \quad \omega = \frac{2\pi}{T_{\text{tide}}}, \quad U = 1.15 \text{ m s}^{-1}. \quad (50)$$

Doing so means that we only need to solve a linear PDE (the tracer transport system) at each timestep, implying a low computational cost for the forward solve. For the purposes of this numerical experiment, $T_{\text{tide}}$ is set to 5% of the M2 tidal constituent, meaning it is artificially sped up twenty-fold. We choose a timestep of $\Delta t = 2.232$, so that there are 2,000 timesteps in total. The time interval is divided into 40 uniform subintervals.

The desalination plant has buried pipes with open ends located within the domain. The outlet pipe has diameter $r_{\text{out}} = 25$ m and is positioned at $x_{\text{out}} = (0, 100)$. The inlet pipe also has diameter $r_{\text{in}} = 25$ m, but is positioned at $x_{\text{in}} = (400, -100)$. In the terminology and notation of the previous experiments, a source is released from $R_{\text{out}} := B_{r_{\text{out}}}(x_{\text{out}})$ and received in $R_{\text{in}} := B_{r_{\text{in}}}(x_{\text{in}})$. The injection of saline water into the flow is modelled by the circular indicator function $\mathbb{I}_{R_{\text{in}}}$, scaled by discharge rate of 2. An initial mesh with 6,574 elements and 3,328 vertices is used, which has increased resolution in a box in the centre of the domain. This region is indicated in Subfigure 8a.

Tracer dispersion is modelled over two tidal periods. The QoI is given as the increase in salinity at the inlet pipe, against the background value:

$$J(c) := \int_0^{2T_{\text{tide}}} \int_{\Omega} \mathbb{I}_{R_{\text{in}}}(c - c_b) \, dx \, dt = \int_0^{2T_{\text{tide}}} \int_{R_{\text{in}}} (c - c_b) \, dx \, dt. \quad (51)$$

Adaptive Mesh Snapshots. Figure 8 displays snapshots of the salinity on a relatively fine fixed mesh at five time levels, including the initial and final times. The inlet and outlet pipes are included as annotations. At $t = 0$, the tracer concentration is initialised to the background level.

Figure 9 shows the meshes used by the isotropic DWR mesh adaptation approach, at the same time levels as shown in Figure 8. A target metric complexity of $8 \times 10^6$ is used. Note that this is a
space-time complexity, so is effectively the instantaneous metric complexity \( \mathbf{I} \) integrated over the temporal domain. Whilst the salinity is initially uniform, the mesh used at the first timestep (Subfigure 9a) is adapted to the metric computed over the following 50 timesteps, which is why there is increased mesh resolution around the outlet pipe. Similarly, the mesh at the final time \( t = 2T_{\text{tide}} \) is only well resolved around the inlet. This is because saline water released from the outlet will not reach it before the end of the simulation, meaning the adjoint solution is negligible away from the inlet. The salinity is actually relatively low near the inlet pipe at this time (as is clear from Subfigure 8e), but the higher salinity elsewhere is regarded as unimportant with regards to the goal and so is represented using coarse mesh resolution. The meshes at the other time levels have approximately an order of magnitude more elements. This observation reveals how the goal-oriented mesh adaptation can be used to construct discretisations which deploy DoFs appropriately in time, as well as in space. The mechanism which allows for such features to develop is the space-time normalisation strategy, which distributes goal-oriented metric complexity across the subintervals.

Figure 10 shows meshes due to the anisotropic DWR metric, again with \( C_T = 8 \times 10^6 \). The initial and final meshes take a similar form, except with a wider region of increased refinement around the outlet and inlet, respectively. The intermediate meshes put increased resolution in similar locations as the isotropic metric, although with a wider spread and higher anisotropy. It is interesting to observe that the highest level of resolution tends to form a band of roughly constant width, which moves side-to-side with the tide. As well as having more DoFs, the meshes
at times $t = 0.5 T_{\text{tide}}$ and $t = 1.5 T_{\text{tide}}$ (particularly the latter) have higher anisotropy than the others presented.

Figure 11 shows the meshes used by the WH metric with $C_T = 8 \times 10^6$. These meshes take a remarkably similar form to those due to isotropic DWR, albeit with greater anisotropy. This is not unsurprising, since both involve strong residuals weighted by recovered second derivatives. The maximum aspect ratios are only moderate because this is not a strongly anisotropic problem.

Finally, Figure 12 shows meshes due to the WG metric with the same target complexity as above. There is a notable feature which make these meshes stand out from the others: the inclusion of the Hessian of the source term means that a region of high resolution persists around the outlet pipe for the first half of the simulation. It ceases to exist thereafter, when there is no longer sufficient time for salinity in that location to reach the inlet pipe. Contributions to the metric from the boundary have been neglected in this simulation. If they were included, it is likely that there would be anisotropic refinement on at least some of the boundaries, as observed in the steady-state case.

**Convergence Analysis.** To conclude this experiment, we consider convergence analysis of the QoI. For simplicity of presentation, the QoI value is plotted as a function of target space-time metric complexity. For most target complexity values considered, the four goal-oriented methods differ in their approximations of the QoI, but are in reasonable agreement, given its large magnitude. With a target complexity of $2 \times 10^6$, they in good agreement and the isotropic and anisotropic DWR approaches could even be said to have converged. It is not clear that WH and
WG have fully converged at this complexity.

It is difficult to come to a conclusion regarding the optimality of any one method without more detailed convergence analyses and computational resource assessments. However, isotropic DWR stands out as having little fluctuation in QoI values, as target complexity is increased, and yields values close to the converged QoI approximation even with $C_T = 2 \times 10^6$ (corresponding to approximately three orders of magnitude fewer overall DoFs than the final run).

That the anisotropic metrics do not offer an improvement over isotropic DWR is not entirely surprising, since the test case does not have particularly strong anisotropy. It would be interesting to make the same comparison in the context of a more strongly anisotropic time-dependent problem in future work.

6. Discussion

6.1. Conclusion

This work presents the successful implementation of four approaches to goal-oriented mesh adaptation in the finite element package Firedrake. Earlier versions of three of the approaches were previously compared in Firedrake in [19] and the other was used in [25]. Many of the simplifications of the former implementation have been overcome in this work and a stabilisation strategy which accounts for mesh anisotropy has been deployed.

The numerical experiments presented in Section 4 use an established advection-diffusion test case with a known analytical solution to validate the goal-oriented strategies, demonstrat-
<table>
<thead>
<tr>
<th>$t/T_{tide}$</th>
<th>Elements</th>
<th>Vertices</th>
<th>Mean AR</th>
<th>Max. AR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>2,426</td>
<td>1,254</td>
<td>2.7</td>
<td>11.3</td>
</tr>
<tr>
<td>0.5</td>
<td>12,473</td>
<td>6,280</td>
<td>1.8</td>
<td>11.1</td>
</tr>
<tr>
<td>1.0</td>
<td>12,243</td>
<td>6,169</td>
<td>2.1</td>
<td>14.0</td>
</tr>
<tr>
<td>1.5</td>
<td>13,523</td>
<td>6,809</td>
<td>2.0</td>
<td>17.7</td>
</tr>
<tr>
<td>2.0</td>
<td>2,708</td>
<td>1,395</td>
<td>1.5</td>
<td>3.3</td>
</tr>
</tbody>
</table>

Figure 11: Example meshes for the application of the weighted Hessian metric with normalisation order $p = 1$ to the idealised desalination problem. The element count, vertex count and mean and maximum aspect ratio (AR) of each mesh are tabulated.
<table>
<thead>
<tr>
<th>$t/T_{	ext{tide}}$</th>
<th>Elements</th>
<th>Vertices</th>
<th>Mean AR</th>
<th>Max. AR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>4,794</td>
<td>2,439</td>
<td>1.8</td>
<td>4.8</td>
</tr>
<tr>
<td>0.5</td>
<td>13,755</td>
<td>6,947</td>
<td>2.1</td>
<td>19.8</td>
</tr>
<tr>
<td>1.0</td>
<td>10,826</td>
<td>5,474</td>
<td>1.8</td>
<td>7.0</td>
</tr>
<tr>
<td>1.5</td>
<td>11,560</td>
<td>5,839</td>
<td>1.8</td>
<td>13.4</td>
</tr>
<tr>
<td>2.0</td>
<td>4,077</td>
<td>2,081</td>
<td>1.6</td>
<td>3.7</td>
</tr>
</tbody>
</table>

Figure 12: Example meshes for the the application of the weighted gradient metric with normalisation order $p = 1$ to the idealised desalination problem. The element count, vertex count and mean and maximum aspect ratio (AR) of each mesh are tabulated.

Figure 13: Convergence plots of each QoI against target space-time metric complexity for the offset configuration.

are presented, revealing that goal-oriented mesh adaptation can distribute DoFs effectively in time, as well as space. Preliminary QoI convergence analyses are also included.
6.2. Outlook

In the idealised desalination outfall setup, tidal hydrodynamics were approximated using a sinusoidal velocity field. For realistic problems, complex bathymetry fields, drag coefficients and turbulent flow features mean that a prescribed velocity is insufficient; instead, a hydrodynamics model should be included in the forward problem. Hydrodynamics models are typically nonlinear and come with higher computational costs than solving the tracer transport equation alone. At the very least, the fluid velocity should be computed during the forward solve, stored and then loaded again during the adjoint solve. This is quite feasible for the relatively small problem sizes considered in this paper. However, for large-scale problems with highly resolved features, solution fields at each timestep can require significant amounts of memory, meaning storing the entire trajectory becomes infeasible. In such a situation, it becomes necessary to use checkpointing routines (such as [62]), which make a trade-off between recomputing timesteps and storing the associated data.

In this paper, salinity is treated as a passive tracer in a barotropic ocean. In the baroclinic case, density is not simply a function of pressure and depends on salinity, as well as temperature. That is, salinity is an active tracer. Future work will apply goal-oriented mesh adaptation to such flows, in which case the computational cost associated with forward and adjoint solves is significant and checkpointing routines are necessary.

Acknowledgements

Thank you to four anonymous reviewers for some excellent suggestions on how to improve the quality of this paper. We would also like to thank members of Imperial College London’s Applied Modelling and Computation Group (AMCG) and to the developers of the Firedrake and Thetis projects for their useful recommendations regarding this work. Further thanks to staff and students of the Mathematics of Planet Earth Centre for Doctoral Training (MPE CDT) for their ongoing support and advice.

Funding: This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) [grant numbers EP/L016613/1, EP/R029423/1].

References


