Goal-Oriented Mesh Adaptation for Firedrake

Joseph G. Wallwork[1], Matthew D. Piggott[1]

[1]Department of Earth Science and Engineering, Imperial College London, U.K.

Abstract

Mesh adaptation can be a very powerful tool for improving the accuracy and/or efficiency of simulations, as has been documented extensively in the literature. However, it is still not widely used. This is largely because commonly available approaches require experience and an in-depth understanding of both the problem at hand and its interaction with the discretisation method being used. Many users of numerical software lack in one of these, and sometimes both. The goal-oriented mesh adaptation framework reduces the required level of experience, since it is driven by error estimates determined by the PDE and a user-specified *quantity of interest (QoI)* that we seek to accurately approximate, such as the power output of a turbine or the drag on an aeroplane wing. This massively simplifies the challenge of using mesh adaptation in an optimal manner, since it is now only required to choose a QoI, which will generally be somewhat obvious and straightforward to implement.

The primary output of this eCSE is a new goal-oriented error estimation and mesh adaptation module, *Pyroteus*, which handles the solution of PDEs and their adjoints on sequences of meshes and enables the automatic computation of error indicators associated with such equations. In addition, both the *PETSc* solver library and *Firedrake* finite element library have been extended to support metric-based mesh adaptation functionality that can be used by Pyroteus to generate anisotropic adapted meshes based on the goal-oriented error indicators it computes.

1 Overview

This technical report documents the achievements of the ARCHER2 eCSE project "Goal-oriented mesh adaptation for Firedrake". The project had four central objectives:

1. Couple ParMmg to PETSc and provide a suite of routines to create, process and combine metric fields.
2. Make Mmg and ParMmg's mesh adaptation functionality available in Firedrake.
3. Create a Firedrake-based goal-oriented error estimation and mesh adaptation module.
4. Develop exemplar applications and training material.

These objectives are reflected in the Sections 2, 3, 4 and 5, respectively, although the training material part of the fourth objective is covered in parts at the end of the other sections.

2 Metric-Based Mesh Adaptation in PETSc

The contents of this section are based on [18], which was published as a direct output of this eCSE project.

Metric-based mesh adaptation is now a relatively mature technology, which has the capacity to produce highly anisotropic, multi-scale meshes of scientifically and industrially relevant domains in two or three dimensions. Its effectiveness as an advanced discretisation method has been demonstrated on numerous occasions, in application areas ranging from geoscience to aerospace engineering. The core idea is to use a Riemannian metric space within the mesher, to guide the concept of mesh optimality. Whilst an element of an adapted mesh might look distorted in Euclidean space, it is 'unit' when viewed in the Riemannian space – its edges having (close to) unit length. A great advantage of the metric-based approach is that it allows control of element shape and orientation, as well as size. This can be particularly beneficial for problems with strong direction-dependence and/or anisotropy, which do occur often across a range of science and engineering applications.

2.1 PETSc DMPlex

PETSc (Portable, Extensible Toolkit for Scientific Computing) [1, 2] is a widely used scientific library, written in C, which provides the linear and nonlinear solvers and associated data structures required to solve PDEs in parallel, among many other things. In PETSc, unstructured meshes are managed by the DMPlex data structure [12]. The representation uses a Hasse diagram, or a directed acyclic graph, whereby all entities are treated equally as vertices of the graph. This enables algorithms to be written independently of mesh properties such as dimension and cell type. DMPlex supports all of the operations required to handle meshing, such as generation, partitioning and distribution, creation missing edges and faces, regular refinement, extrusion, traversal, selection, manipulation and I/O [11]. The first phase of this eCSE adds Riemannian metric utilities, to support metric-based adaptation.

2.2 Mmg and ParMmg

Metric-based mesh adaptation is already available to PETSc's unstructured mesh manager, DMPlex [12], using *Pragmatic*, as documented in [3]. The software development efforts documented in this report build upon the previous work, integrating two new toolkits, *Mmg* [8] and *ParMmg* [5].

Mmg comprises a collection of tools for anisotropic metric-based adaptation of simplicial meshes. It includes a 2D local remesher (*Mmg2d*), a 3D local remesher (*Mmg3d*) and a surface remesher (*Mmgs*). The 3D local remesher has been parallelised using MPI as *ParMmg*. Parallel implementations do not currently exist in 2D or for surface remeshing. Given an input mesh and a metric defined upon it, Mmg applies a sequence of operations to the mesh in order to optimise the quality of its elements, as measured by a quality functional, for which optimality means a perfectly isotropic element with edges of unit length. In the 3D case, the operations used to optimise the mesh are (h-adaptive) node insertion, node deletion and face swapping, as well as (r-adaptive) node movement.

ParMmg performs MPI-parallel mesh adaptation iteratively, first applying Mmg3d to the mesh partition owned by each process, then repartitioning and doing the same again. Three iterations are typically sufficient to remove dependence on the initial partition.

2.3 Coupling

Given a linear simplicial mesh described using a DMPlex, a Riemannian metric field may be defined in the finite element method context as a piece-wise linear and continuous ($\mathbb{P}1$) field, with its degrees of freedom (DoFs) at mesh vertices. The values at these DoFs can be encapsulated in a PETSc Vec. The new functionality introduced by this eCSE provides routines for the construction (e.g. `DMPlexMetricCreate`) and modification (e.g. `DMPlexMetricEnforceSPD`) of such objects, as well as for combining different instances (e.g. `DMPlexMetricIntersection`). We make use of the L^p normalisation techniques introduced by [13], which both ensure that resulting adapted meshes have the desired complexity and enable them to be appropriately multi-scale (`DMPlexMetricNormalize`).

Given a metric constructed using these routines, `DMAdaptMetric` provides the interface so that ParMmg can be used to drive the mesh adaptation process. The Pragmatic interface described in [3] is also maintained. When invoked, the Vec containing the metric data is converted to an array of `PetscScalar`s (i.e. `double`s in real mode) and passed to the appropriate remesher. In addition to the metric operations above, Mmg and ParMmg have in-built metric gradation functionality, which ensures that two adjacent edges in the mesh do not differ by too large a ratio.

For further details, see [20] and the "metric-based mesh adaptation" section of [1], both of which were written as part of this eCSE.

3 Metric-Based Mesh Adaptation in Firedrake

3.1 Implementation

In addition to extending PETSc's metric-based mesh adaptation functionality, we provided *petsc4py* [7] Python bindings, so that downstream packages, such as the *Firedrake* finite element library, may take advantage of these enhancements. The new metric-based mesh adaptation functionality could then be ported to Firedrake because it uses DMPlex for its unstructured mesh representation. We introduced a new `RiemannianMetric` class to hold the function data associated with a metric field, with methods for calling the various PETSc metric utility routines. A `MetricBasedAdaptor` class was also introduced, which uses the `RiemannianMetric` instance to guide the adaptation of the underlying DMPlex.

3.2 Demonstration

To demonstrate the functionality, we solve the Poisson equation on adaptive meshes of the unit cube, $\Omega = [0,1]^d$, $d = 3$:

$$\Delta u = f \quad \text{in } \Omega, \qquad u = g \quad \text{on } \partial\Omega, \tag{1}$$

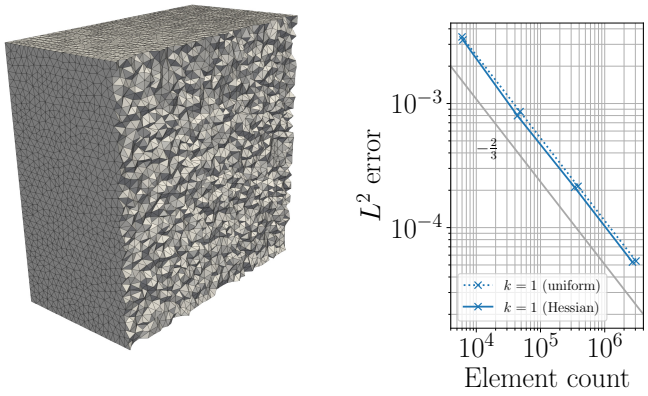

Figure 1: Adapted mesh and L^2 error convergence analysis for a metric computed using the Hessian of the weak solution of (1) with RHS $f \equiv 4$. Mesh statistics: 339,727 elements, 60,825 vertices, max. aspect ratio 4.0384, mean aspect ratio 1.4207, std. dev. 0.1985.

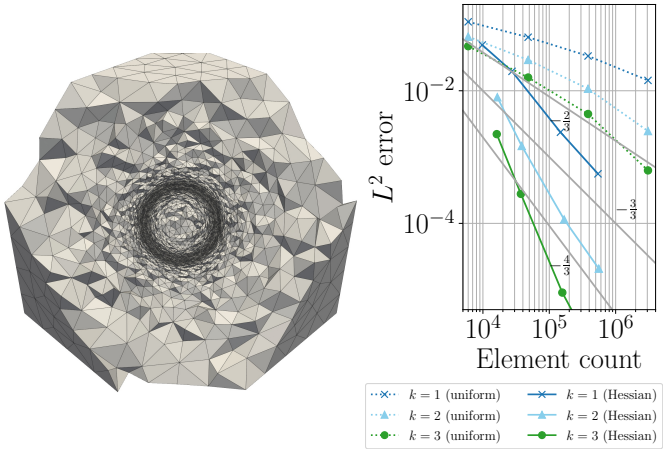

Figure 2: Adapted mesh and L^2 error convergence analysis for a metric computed using the Hessian of the weak solution of (1) with RHS [20, eq (3)]. Mesh statistics: 541,860 elements, 90,553 vertices, max. aspect ratio 12.3425, mean aspect ratio 2.6830, std. dev. 0.9973, min./max. volumes: $4.0907 \times 10^{-9}/4.2582 \times 10^{-4}$.

with solution $u \in H^2(\Omega)$, forcing $f \in L^2(\Omega)$ and Dirichlet condition defined by $g \in L^2(\partial\Omega)$. By choosing different f and g, we may construct manufactured solutions with different features. We discretize by seeking weak solutions in order $k \in \mathbb{N}$ Lagrange space, $\mathbb{P}k \subset H^1(\Omega)$. The resulting linear systems are solved using conjugate gradients, preconditioned with SOR.

Mesh adaptation is performed with the metric being an L^1 normalised Hessian of the finite element Poisson solution. This is a common basis for an error metric, since it incorporates anisotropic information related to interpolation error [13]. Its symmetry implies an orthogonal eigendecomposition at each DoF, which can be computed straightforwardly. A metric can then be constructed by simply taking the eigenvalues in modulus. This is justified because error magnitudes are typically more important than their signs. We compute Hessians using two applications of the Clément interpolant [6] – one to recover the gradient and another for the Hessian.

The experimental setup is identical to the one used to test PETSc's adaptation functionality in [20]. Here, we reproduce the results on the Firedrake level.

Hessian of a Quadratic Function First consider the quadratic manufactured solution, $u_1(\mathbf{x}) := \frac{2}{3}\mathbf{x} \cdot \mathbf{x}$, where $\mathbf{x} := (x, y, z)$. The Hessian of this solution field is just a scaling of the identity matrix, so we expect the adapted mesh to be uniformly isotropic. The mesh plot in Figure 1 verifies that that this is indeed the case. Some elements are moderately anisotropic (with aspect ratios greater than two), although this is only true for 0.96% of the elements. The presence of some slightly anisotropic elements is unavoidable in tetrahedral mesh adaptation since unit tetrahedra cannot tile space in general.

By increasing target metric complexity, we may produce increasingly refined meshes and thereby perform convergence analysis. The right-hand plot in Figure 1 demonstrates the $(k+1)/d$ convergence rate in the L^2 norm for degree $k = 1$. This corresponds to the expected $\mathcal{O}(h^{k+1})$ convergence on isotropic meshes, which is matched by standard uniform refinement.

Interface Capturing Next, we demonstrate the capability to capture interfaces. We seek the manufactured solution given in [20, eq. (3)], which is a smooth approximation of an indicator function for a sphere of radius $r = 0.15$.

The left-hand plot in Figure 2 shows the adapted mesh for target complexity 32,000. High resolution is used to capture the interface, with some fairly anisotropic elements. Notably coarse resolution is used near the domain boundaries. The caption indicates that we have a multi-scale mesh, whose element volumes span eight orders of magnitude. Elements are generally more anisotropic for this test case, with 73.22% having aspect ratio greater than two.

The Hessian-based curves in Figure 2 exhibit the same $(k+1)/d$ convergence rate as above. However, uniform refinement does not attain the optimal rate for this particular problem. As such, we demonstrate that, together, Firedrake and Mmg are able to achieve optimal L^2 convergence for a Poisson problem with a spatially varying RHS, which standard uniform refinement is unable to reproduce.

4 Goal-Oriented Mesh Adaptation using Pyroteus

Pyroteus is a new Python-based model that was created as part of this eCSE. It is designed to automate the process of goal-oriented error estimation for a given PDE problem, as described in the following.

Time-dependent problems can be particularly challenging, especially when different adapted meshes are used for different subintervals of the temporal domain. To address this case, the first fundamental object of Pyroteus is the `TimePartition`, which describes the way in which this partitioning is done:

$$(0,T] = \cup_{n=0}^{N}(t^n, t^{n+1}], \quad 0 = t^0 < \cdots < t^N = T. \tag{2}$$

Note that the subintervals need not be of uniform length.

Building upon the `TimePartition`, the core Pyroteus object is the `MeshSeq`, which associates a mesh to each subinterval, giving rise to a sequence of meshes. If we provide the `MeshSeq` with routines for constructing finite element spaces and solving PDEs on each mesh, it is able to time integrate over the entire sequence, applying mesh-to-mesh interpolation inbetween [10].

One of the key pieces of technology is provided by the subclass, `AdjointMeshSeq`: given a function for evaluating the QoI, it uses Pyadjoint [9, 14] to automatically solve the adjoint problem over all of the subintervals (in reverse). The appropriate transpose interpolation operator is used for the mesh-to-mesh transitions.

Finally, the `GoalOrientedMeshSeq` subclass uses information relating to the forward problem and QoI to facilitate automated goal-oriented error estimation. In particular, it allows the user to compute *dual weighted residual (DWR)* type error estimators, as introduced in [4]. One of the difficulties of the DWR formulation is that it involves the *true* adjoint solution, which is unknown in general. Pyroteus addresses this by approximating the true adjoint solution either using global h-refinement or global p-refinement, as is commonly done in the literature [16]. However, for simplicity, it does not take the usual approach of splitting the weak residual into strong residual and flux term components, as was done in [15]. Provided with a routine for performing mesh adaptation based on error indicators and solution data, the `GoalOrientedMeshSeq` facilitates goal-oriented mesh adaptation in a fixed point iteration type approach. That is, we repeatedly solve the forward equation, construct goal-oriented error indicators and adapt the mesh sequence until convergence criteria on the mesh complexity, QoI or error estimate values are met.

For further details and training material, see the Pyroteus webpage [21].

5 Applications

In this section we present some of the key findings of research projects that have made use of the software developed in this eCSE – in particular the new goal-oriented error estimation and mesh adaptation functionality.

5.1 Tracer Transport Modelling

In [19], goal-oriented mesh adaptation techniques are applied to idealised tracer transport modelling problems, using Pyroteus for handling the solution of forward and adjoint problems on sequences of meshes, and Mmg2d and Mmg3d for the mesh adaptation step. Steady-state problems are considered in both 2D and 3D, using various different goal-oriented metric formulations.

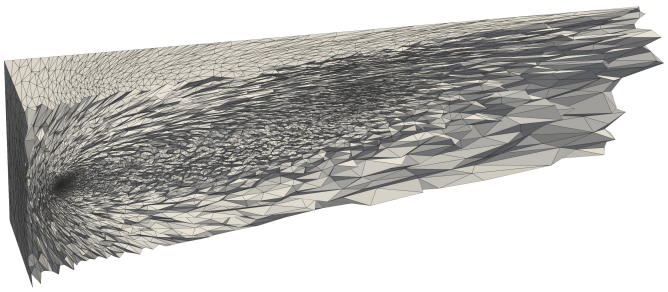

Figure 3: Adapted mesh for a steady-state tracer transport problem in 3D using an anisotropic goal-oriented metric. Mesh statistics: 749,804 elements, 136,372 vertices, max. aspect ratio 53.8, min./max. volumes 1.3×10^{-8}/1.4. Image taken from [19] with authors' permission.

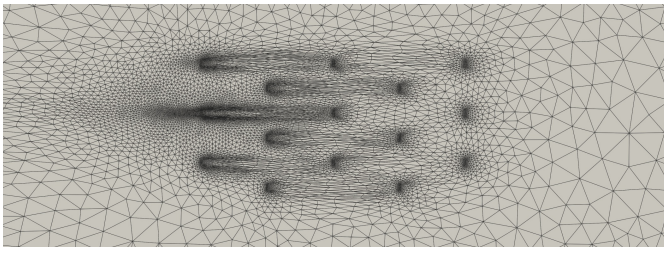

Figure 4: Adapted mesh snapshot for a time-dependent tidal farm modelling problem in 2D using an anisotropic goal-oriented metric. Mesh statistics: 23,503 elements, 211,527 DoFs, max. aspect ratio 21.8. Image taken from [17] with authors' permission.

An example 3D adapted mesh due to an anisotropic goal-oriented metric is shown in Figure 3. A source of salinity is injected at $(2, 5, 5)$, near to the left-hand boundary of the domain, $\Omega = [0, 10] \times [0, 50] \times [0, 50]$ with units of metres. It is advected to the right under a uniform fluid velocity and experiences uniform isotropic diffusion. The QoI integrates salinity over a ball 'receiver region' of radius 0.5 m centred at $(20, 7.5, 7.5)$. In this idealised setup, the source and receiver regions can be considered akin to the outlet and inlet pipes of a desalination plant which provides a motivator for this problem. For the particular metric used in this example, increased mesh resolution is deployed to capture the source term and receiver region and moderate resolution is used in bands in between. The resolution of the adapted mesh is relatively low downstream of the receiver region because we have an advection-dominated problem, wherein the QoI is effectively independent of the dynamics in this region. The wide range of element volumes reported in the caption illustrate that we have a truly multi-scale adapted mesh.

An idealised desalination plant outfall scenario is also considered, wherein mesh adaptation was applied multiple times throughout the time-dependent simulation.

5.2 Tidal Array Modelling

In [17], goal-oriented mesh adaptation techniques are applied to time-dependent tidal farm modelling problems, using Pyroteus for handling the solution of forward and adjoint problems on sequences of meshes and Mmg2d for the mesh adaptation step. Tidal hydrodynamics are modelled in 2D in the depth-averaged sense using the shallow water equations; tidal turbines within the array are represented using a drag parametrisation and the QoI is a proxy of the energy output generated by the farm over the time period.

Figure 4 shows the adapted mesh used halfway through a simulation of a single flood tide (i.e. the first half of a tidal cycle), with a zoom on the tidal farm region. During this time period, flow goes from left to right. There are five columns of turbines, arranged in a 'staggered' configuration. A 2D version of the same anisotropic metric as in Figure 3 is used. Under the guidance of this metric, the adaptive mesh focuses mesh resolution around the farm region and in particular at the turbines themselves. Anisotropic elements are used to capture the wakes of some of the upstream turbines where their accurate representation is important to the accurate calculation of the overall array energy, but not the downstream ones. As with the tracer transport example, this makes sense because of the advection-dominated nature of the problem.

6 Conclusion

In summary, this eCSE successfully completed all of the objectives set out at the start of the project. The integration of Mmg and ParMmg and provision of a range of metric utilities to support their use was merged into PETSc. Similarly, the Pyroteus functionality described herein is all available to users. The porting of PETSc's mesh adaptation functionality to Firedrake currently exists as a draft implementation. Nonetheless, it was used to generate the results in applications papers [19] and [17]. A draft implementation also exists for parallelising PETSc's mesh-to-mesh interpolation functionality.

Future work will focus on getting the new metric-based mesh adaptation functionality merged into Firedrake and the parallel mesh-to-mesh interpolation functionality merged into PETSc, to make these features available to users. Once this is done, it would be beneficial to port the parallel interpolation functionality through to Firedrake, too.

Acknowledgments

The software development work described in this report spans multiple different code bases, each having different levels of abstraction and often written in different languages. As such, its successful completion required drawing on the expertise of many developers and researchers across a number of institutions, each with deep knowledge of one or more parts of the pipeline. The project could not have been completed without this level of collaboration. The authors acknowledge the continued support of all members of the Applied Modelling and Computation Group (AMCG) at Imperial College London, as well as the development teams of the PETSc solver library, Firedrake finite element package and Thetis coastal ocean model. In particular, we would like to give special thanks to Nicolas Barral, Akhil K. Mohan, Matthew G. Knepley, Pierre R. Jolivet, Satish Balay, Lawrence Mitchell, David A. Ham, Colin J. Cotter, Stephan C. Kramer, Tuomas Kärnä, Athanasios Angeloudis and Jon Hill.

This work was funded under the embedded CSE programme of the ARCHER2 UK National Supercomputing Service (`http://www.archer2.ac.uk`).

References

[1] Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J., Brune, P., Buschelman, K., Constantinescu, E., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. D., Hapla, V., Isaac, T., Jolivet, P.,

Karpeev, D., Kaushik, D., Knepley, M. G., Kong, F., Kruger, S., May, D. A., McInnes, L. C., Mills, R. T., Mitchell, L., Munson, T., Roman, J. E., Rupp, K., Sanan, P., Sarich, J., Smith, B. F., Zampini, S., Zhang, H., Zhang, H., and Zhang, J. PETSc/TAO users manual. Tech. Rep. ANL-21/39 - Revision 3.16, Argonne National Laboratory, 2021.

[2] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. D., Karpeyev, D., Kaushik, D., Knepley, M. G., May, D. A., McInnes, L. C., Mills, R. T., Munson, T., Rupp, K., Sanan, P., Smith, B. F., Zampini, S., Zhang, H., and Zhang, H. PETSc Web page, 2021.

[3] Barral, N., Knepley, M. G., Lange, M., Piggott, M. D., and Gorman, G. J. Anisotropic mesh adaptation in Firedrake with PETSc DMPlex. In *25th Intl Meshing Roundtable* (2016).

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

[5] Cirrottola, L., and Froehly, A. Parallel unstructured mesh adaptation using iterative remeshing and repartitioning. Research Report RR-9307, INRIA Bordeaux, équipe CARDAMOM, Nov. 2019.

[6] Clément, P. Approximation by finite element functions using local regularization. *Rev fr autom inform rech opér, Anal numér 9*, R2 (1975), 77–84.

[7] Dalcin, L. D., Paz, R. R., Kler, P. A., and Cosimo, A. Parallel distributed computing using Python. *Adv Water Resour 34*, 9 (2011), 1124–1139.

[8] Dobrzynski, C., and Frey, P. Anisotropic delaunay mesh adaptation for unsteady simulations. In *17th Intl Meshing Roundtable*. Springer, 2008, pp. 177–194.

[9] Farrell, P. E., Ham, D. A., Funke, S. W., and Rognes, M. E. Automated derivation of the adjoint of high-level transient finite element programs. *SIAM Journal on Scientific Computing 35*, 4 (2013), C369–C393.

[10] Farrell, P. E., and Maddison, J. R. Conservative interpolation between volume meshes by local Galerkin projection. *Computer Methods in Applied Mechanics and Engineering 200*, 1-4 (2011), 89–100.

[11] Hapla, V., Knepley, M. G., Afanasiev, M., Boehm, C., van Driel, M., Krischer, L., and Fichtner, A. Fully parallel mesh I/O using PETSc DMPlex with an application to waveform modeling. *SIAM J Sci Comp 43*, 2 (2021), C127–C153.

[12] Knepley, M. G., and Karpeev, D. A. Mesh algorithms for pde with sieve i: Mesh distribution. *Sci Program 17*, 3 (2009), 215–230.

[13] Loseille, A., and Alauzet, F. Continuous mesh framework part ii: validations and applications. *SIAM J Numer Anal 49*, 1 (2011), 61–86.

[14] Mitusch, S. K. An algorithmic differentiation tool for FEniCS. Master's thesis, University of Oslo, 2018.

[15] Rognes, M. E., and Logg, A. Automated goal-oriented error control i: Stationary variational problems. *SIAM Journal on Scientific Computing 35*, 3 (2013), C173–C193.

[16] Wallwork, J. G. *Mesh adaptation and adjoint methods for finite element coastal ocean modelling.* PhD thesis, Imperial College London, 2021.

[17] Wallwork, J. G., Angeloudis, A., Barral, N., Mackie, L., Kramer, S. C., and Piggott, M. D. Tidal turbine array modelling using goal-oriented mesh adaptation. *Journal of Ocean Engineering and Marine Energy (under review)* (2022).

[18] Wallwork, J. G., Barral, N., Ham, D. A., and Piggott, M. D. Anisotropic goal-oriented mesh adaptation in Firedrake. In *28th Intl Meshing Roundtable* (2020), Zenodo, pp. 83–100.

[19] Wallwork, J. G., Barral, N., Ham, D. A., and Piggott, M. D. Goal-oriented error estimation and mesh adaptation for tracer transport modelling. *Computer-Aided Design* (2021), 103187.

[20] Wallwork, J. G., Knepley, M. G., Barral, N., and Piggott, M. D. Parallel metric-based mesh adaptation in PETSc using ParMmg, 2022.

[21] Wallwork, J. G., and Mohan, A. K. Pyroteus web page, 2022.