Imperial College London

Department of Earth Science and Engineering

MSc in Applied Computational Science and Engineering

Independent Research Project
Final Report

Ocean modelling using goal-oriented r-adaptation

by

Akhil Krishna Mohan

Email: akm121@imperial.ac.uk
GitHub username: acse-akm121
Repository: https://github.com/ese-msc-2021/irp-acse-akm121

Supervisors:

Prof. Matthew Piggott
Dr. Joseph Wallwork

September 2022

Acknowledgements

I would like to express my gratitude to my supervisors Dr. Joseph Wallwork, and Prof. Matthew Piggott for their help and support throughout my time as Masters' student. I hope to continue collaborating with them in the future. I would also like to thank Prof. Pablo Salinas, my faculty mentor at Imperial College, who helped me maximize my learning experience over the last year. I would like to extend a big thank you to the teaching and administrative staff from the Department of Earth Science and Engineering from Imperial College who made the graduate student experience as engaging as possible. Finally, I would like to thank my family and friends for their continued support and kindness throughout my degree.

Abstract

This project applies goal-oriented r-adaptation to motivating problems in Geophysical Fluid Dynamics (GFD). In problems like aerodynamic wing design, the solution of the underlying Partial Differential Equation can be less important than a diagnostic Quantity of Interest (QOI), such as the maximum airflow across the wing through the simulation. Goal-oriented r-adaptation describes a process to minimize errors in the QOI computation by "moving" or redistributing mesh nodes in the physical domain. This is achieved by solving the adjoint problem for our QOI estimation, defining goal-based error indicators, defining a mesh-density (monitor) function, and finally moving the mesh to equidistribute the errors in the estimation. This allows us to gain resolution in areas most relevant to the QOI computation at the expense of coarser resolution in other regions of the domain. Given that the mesh is unchanged in the computational domain, r-adaptive approaches can provide a computationally cheap means to increase accuracy of QOI estimations. We apply this approach to two motivating examples based on Burgers' Equation and the pure Advection equation, and analyse the effects of different parameters in the adaptation scheme.

1 Introduction / Literature Review

Computational Physics provides us with a wealth of interesting problems with complexities arising for a variety of reasons. While the vast majority involve solving a Partial Differential Equation (PDE) of some kind, the actual solution of the PDE can be less important than accurately simulating behaviour of some component of the physical system. For example, in modelling the behaviour of a desalination plant, we might place more emphasis on accurately computing tracer concentrations at the inlet of the system, rather than the outlets [30, 31]. Goal-oriented mesh adaptation can be used in this context, working to selectively increase resolution in regions of our domain that contribute to the "goal" computation. The salt concentration at the inlet in the desalination simulation is an example of a goal computation or Quantity of Interest (QOI).

Using the Finite Element Method (FEM), we approximate our solution fields locally using "elements" or "cells" [27]. Functions on our domain are defined by local basis functions, typically lower order polynomials. Using polynomial basis functions on our elements allows us to define derivative and integral operators in a rigorous manner, with the discretization of the method coming from the order of polynomial used. This defines a finite subspace of a function space [11], and this is where we go from continuous (in the real world) to discrete (computational representation). With a mesh of elements, and constraints to ensure continuity conditions across elements, we have a means of representing functions in space. To solve PDEs, we constrain the local functions to ensure they satisfy the conditions of the PDE.

Mesh adaptation takes advantage of the idea that using the Finite Element Method, we implicitly have two representations of the mesh that is used to solve the PDE - one in the computational domain, and one in the physical domain [12]. In 2-D, the spatial domain is tiled with either triangular

or quadrilateral elements, with tetrahedra and hexahedra being their analogues in 3-D [27]. The tessellation of the spatial domain contains vertices which define the elements in our mesh, and also nodes - points at which our functions are evaluated for interpolation. Internally, the mesh is represented by the vertices, the nodes, and their connections. Each vertex corresponds to a physical location in the spatial domain, where our PDE is defined. Thus, we have a physical representation of the mesh - the coordinate mapping of the vertices of the mesh, and the computational representation - the connections between elements at vertices and edges.

The FEM mesh allows us to use irregular geometries in space, thus opening the door to adapting the mesh based on features of the PDE solution. The three primary approaches to mesh adaptation are h-adaptation, p-adaptation, and r-adaptation [29]. h-adaptation involves changing the tessellation of mesh in the physical domain, allowing for more resolution in select areas by subdividing cells [13]. p-adaptation, by contrast, works by increasing the order of the interpolating polynomial in select areas, thus working in the computational domain [12, 13]. r-adaptation, the approach we use in this setup, works by moving mesh nodes in the physical domain, without adding or removing any nodes. This allows the mesh topology and representation in the computational domain to remain unchanged [22, 12].

While there is a wealth of literature on both goal-oriented meshing, and mesh adaptation in general, (for example [12, 13, 8, 9, 29]) the majority of these works use p- and h-adaptive methods. Our choice to use goal-oriented r-adaptation seeks to address this disparity, and create a blueprint for r-adaptation based meshing. The procedures and experiments used in this paper draw from the existing works in this niche - McManus et al.(2017) [22], Bauer et al.(2014) [7], Materna et al.(2009)[21], and Jahandari et al.(2020) [15]. Specifically, Materna et al. [21] study the effectiveness of goal-oriented r-adaptivity in a more abstract mathematical setting, whereas Bauer et al. [7] apply adaptive mesh refinement to simulating cyclone-like vortices on a given shallow water model. Jahandari et al. [15] examine the use of r-adaptation when combined with h-adaptation for the modelling of geophysical electromagnetic data. The work of McManus et al. [22] provides us with both a good overview of the theory of r-adaptation as well as a representative quantitative test case for our framework. These works provide an excellent demonstration for the scope of r-adaptive methods in goal-oriented contexts, showing promising results both in terms of accuracy and performance. In this work, we will borrow theoretical ideas from the above works, and present a more general framework to implement goal-oriented r-adaptation using Firedrake [26] and Pyroteus [30].

1.1 Summary of Novel Contributions

In this work, we validate the use of Pyroteus [30], and introduce the Pyroteus Movement library [28] in the context of goal-oriented r-adaptation. Using these libraries, we present a general framework for goal-oriented r-adaptation. Bauer et al. [7] use a single mesh and average out errors across timesteps, whereas we choose a model whereby the mesh is adapted every timestep. Jahandari et al. [15] study the hybrid hr-adaptation, and apply it to different PDEs. McManus et al. [22] do examine r-adaptation but do not explicitly use the goal-oriented context we set up. We believe that the scope for r-adaptation in goal-oriented problems is more wide ranging, and we hope to encourage more literature on this subject. We attempt to explain the effects of various parameters in the framework and discuss the qualitative and quantitative results of our experiments.

2 Mathematical Background

In this section, we briefly discuss the mathematical background of the approach used in our experiments, shaped largely by the works of Budd et al.(2009) [12], Wallwork (2021) [29], Becker et al.(2001) [8] and Bauer et al.(2014) [7]. For a more rigorous treatment of goal-oriented meshing, refer to [29, 8], and for a comprehensive study of r-adaptation, see [12].

Using the Finite Element Method (FEM), we solve our problem on a computational mesh made up of cells or "elements" that subdivide the spatial domain. We approximate our fields within these (typically triangular or quadrilateral in 2-D) elements with local polynomial approximations that are constrained to obey the physics (PDE), and also enforce continuity conditions across cells [27]. This allows us to represent irregular geometries as elements can be distorted (with some limitations [7, 29]) from their reference shapes. Inherently, this is useful on its own, but this flexibility of the FEM approach also allows us to selectively capture solution behaviours on different scales with a single computational mesh. In mesh adaptation research, the forward PDE is usually solved on a sequence of meshes that divide the time domain in equal parts. The end-time solution of one mesh is then projected onto the next mesh as an initial condition. In pure r-adaptive approaches, the computational mesh is unchanged, and the physical mesh coordinates are governed by a mesh-density or monitor function. In this sense, with a time-dependent mesh-coordinate system, we create a "moving mesh", or a sequence of meshes using which we solve our PDE [12]. In this framework, we choose to solve the adjoint problem for our QOI (goal), which allows us to define the optimal mesh for each timestep.

2.1 Forward and Adjoint Solutions

The FEM allows us to specify our problem in weak form, and solve it efficiently over our domain. When solving a PDE on a sequence of meshes, care must be taken to limit the error in mesh-to-mesh projection of solutions [29, Ch. 2.8]. In our experiments, we define a QOI as an end-time integral based on the solution field. We can also choose to use a time-integrated QOI, ensuring that the temporal nature of the QOI is maintained. In either case, to obtain information on regions in our domain most affecting the QOI calculation, we must solve the adjoint problem for our QOI [30, 8]. The adjoint solution injects the QOI as the source term and propagates backwards in time. In many cases, this is very similar to the forward solve, and once again, we must pay attention to the mesh-to-mesh projection.

Adjoint methods are used across a variety of applications in computational science [29]. They are a versatile tool typically used to obtain information on the sensitivity of the QOI calculation to perturbations in model parameters. In our case, we use the adjoint solution to define goal-based error indicators, which relates the error in QOI computation to the physical representation of the mesh [8]. At a high level, the adjoint problem is obtained by taking the complex conjugate of all the vector and tensor fields in the original PDE. Hence, it flips the time domain, resulting in a "backward-in-time" solve.

2.2 Goal-oriented Error Indicators

Once we have the forward and adjoint solutions, we can defined our goal-based error indicators. To do this, we use the method of Dual Weighted Residuals (DWRs) pioneered by Becker and Rannacher (2001) [8]. The weak residual, ρ, is "weighted" by the adjoint or "dual" solution for our QOI, thus giving us a Dual-Weighted Residual. The DWR is defined as in Wallwork et al. [31] -

$$\mathcal{J}(c) - \mathcal{J}(c_h) = \rho(c_h, c^* - c_h^*) + R \tag{1}$$

where $\mathcal{J}$ is the QOI, c, c^* are the true forward and adjoint solutions, c_h, c_h^* are the respective solutions in our finite function space, ρ is the weak residual operator, and R is the remainder term.

This is a *global* error indicator in the QOI computation [31]. We use this to define local, element-wise error indicators by decomposing this global indicator into a sum of cell-based error estimates. This decomposition gives us an upper bound on the localized errors per element, and gives us the information we need to define the monitor function.

2.3 Monitor Function

The monitor function or mesh-density function governs the resolution of the mesh in different regions of the spatial domain. Intuitively, using smaller elements to tile a region of the domain would result in lower errors in that region [7]. As such, we assume that the monitor function works this way. Using the DWR indicators as a spatial guide as to where errors run high, we can assume that the monitor function value for an element is proportional to the DWR indicators at its corresponding spatial location as well as its own measure (length in 1-D, area in 2-D, volume in 3-D) [7]. Now, to reduce the errors in our QOI computation, we can move vertices into this region, making the elements in the region smaller. This in turn, distributes the element-wise errors across all cells now moved into the said region of the domain to compensate. We can then phrase the "optimal" mesh movement for a given timestep as the mesh that most evenly distributes the localized errors across all the cells. This process, *equidistribution* [12, 23], would mean smaller cells where the indicators run high, and larger cells elsewhere to compensate. The element-wise indicators came from decomposing a sum into its cell-wise contributions, so these functions are constant within each element, and thus naturally discontinuous between elements.

In order to maintain good mesh quality [14], we must place some constraints on our monitor function [12, 7]. As per Bauer et al.(2014) [7], we must smooth and bound our monitor function. While the monitor function defined in the said paper [7] is slightly different to the mesh-density analogue we define, the bounding and smoothing for our set-up were achieved with a global standardization of the error indicators. This is implemented by subtracting the mean element-wise value, and dividing by the variance. We also add -1 times the minimum value to each element's indicator to ensure all values are positive. In this sense, we have bounded the monitor function below, by 0. The reasons for doing this are twofold. First, our DWR indicators are typically too small (around 10^{-6}) to show up on the scale for equidistribution, causing no mesh movement to occur. Second, intuitively, a mesh cannot have negative density in any region in its domain. The normalization we apply represents a process similar to the bounding of the monitor function in the work of Bauer et al.(2014) [7], ensuring that we meet a grid duality constraint. This constraint ensures that the centroid of the new triangular element stays within the element's original area each iteration. Without this constraint, we might run into the issue of mesh-tangling and sacrifice mesh quality [7, 12].

Bauer et al. also outlined a method for smoothing of the monitor function. This corresponded to local smoothing, whereby an element's monitor function value was chosen as the maximum function value in a small neighbourhood of the element. We opted to try global smoothing of two kinds - Laplacian Smoothing (as implemented in [22]), and normalization in L^p space. Interestingly, we found no discernible difference between adaptation runs with and without the smoothing applied, and as such chose not to explicitly smooth the monitor function for our benchmarks. In order to simulate smoothing in our workflow, we project the standardized element-wise indicators into a continuous scalar function space. This comes at a small memory cost, but allows us to more easily reason about differential operators applied to the monitor function. We hope to fully investigate the effects of smoothing in the r-adaptation workflow in future works.

2.4 Mesh-movement PDE

Mesh-movement PDEs can be broadly classified into *position-driven* or *velocity-driven* approaches [12]. In this work, we use position-based mesh movement, whereby mesh nodes are translated to their optimal positions to achieve QOI error reduction as per the monitor function. In velocity-driven approaches, the mesh movement is phrased as a PDE based on the solution field itself, and the moving mesh is computed after each timestep alongside the PDE [12]. It is unclear how to use the velocity-driven approach for our goal-oriented problems, then, as the mesh movement algorithm has dependence on the adjoint solution. The software we used, Pyroteus Movement [28], does not implement velocity-based approaches, so we omit them from our discussion.

As mentioned before, the driver of the mesh-movement algorithm is the equidistribution [24]. Using Ω_C as the computational domain, Ω_P as the physical domain, $\vec{\xi}$ as the mesh-coordinate representation in the computational domain, and $\vec{x}(\vec{\xi}) : \Omega_C \to \Omega_P$ as the time-dependent coordinate map, and $m(\vec{x})$ as the monitor function, we say that $\vec{x}$ equidistributes m if

$$m(\vec{x}) \det J = \theta \tag{2}$$

where θ is a normalization parameter, and J represents the Jacobian map of $\vec{x}(\vec{\xi})$. While this problem does not have a unique solution, we can place constraints on the movement to ensure uniqueness of our map. Based on optimal transport theory [31], we can achieve our movement by solving an equation of Monge-Ampère-type [10, 12], usually presented in the form

$$\det H(\phi) = f \tag{3}$$

Equivalently, we solve the equation as

$$m(\vec{x}) \det(I + H(\phi)) = 0 \tag{4}$$

where H is the Hessian of our transformation ϕ

$$\vec{x}(\vec{\xi}) = \vec{\xi} + \nabla_{\vec{\xi}}\phi(\vec{\xi}) \tag{5}$$

ϕ represents the co-ordinate map, that is constrained not to move the domain boundaries. The solution $\vec{x}$ to this equation based on our defined monitor function m is then used to define the moving mesh for each timestep. Thus, we are able to find the optimal mesh sequence to solve our PDE by solving the Monge-Ampère Equation [12, 24, 22]. Internally, this is represented as a single P1 discontinuous vector function defined at each mesh vertex.

3 Implementation Details

Overall, the development effort was to create a robust framework to achieve goal-oriented r-adaptation. The tools to solve PDEs, evaluate QOIs, and solve adjoint problems already exist in Firedrake [26] and Dolfin adjoint [20, 19]. Pyroteus [30] provides us with a framework to solve PDEs on a mesh sequence as opposed to a single static mesh, and Pyroteus Movement [28] provides us with implementations of the numerical method for r-adaptation. As such, our code contributions here are mostly on the procedural side, creating an experimental setup to benchmark the performance of our framework, and provide some insight into the parameters that go into the r-adaptation workflow.

3.1 Software

To achieve the above, we use the Firedrake [26, 2, 3, 25] Finite-Element library, built on the Unified Form Language (UFL) [1, 20] to specify our PDE setups in a high-level language. The form compiler then translates the high-level formulation to optimized, low-level C code kernels (typically built using PETSc [4, 6, 5]) which can be executed from within a Python instance. This allows us to benefit from both the efficient low-level finite element code, and also the concise formulation that the Python-based approach gives us. Firedrake uses Dolfin Adjoint [20, 19] to extend its forward-solving capability to the adjoint solutions as well.

Pyroteus [30], is a goal-oriented mesh adaptation framework built on top of Firedrake. This allows us to define our problems across a sequence of meshes by specifying the form, initial conditions, boundary conditions, and QOI computation. Once we have done this, we can use the toolkit to estimate our DWR error indicators for the QOI estimation. This is achieved by re-running the QOI computation on either a finer mesh or a mesh with higher degree interpolating polynomials. Finally, we utilize the mesh movement framework Pyroteus Movement [28, 23] to solve the mesh-movement PDE and

re-assign mesh coordinates. In addition to the goal-oriented error estimation functionality provided by Pyroteus, we are also able to compute prognostic mesh quality measures [17, 16]. The aim of our experiments will, accordingly, be to show that QOI estimation is improved while maintaining mesh quality.

3.2 Adaptation Framework

Across both experiments considered here, we are forced to make choices with regards to the r-adaptive framework used. First, we must choose if our QOI is time-integrated or is evaluated at the end-time of our simulation. While this is not strictly part of the adaptation framework, the experiments in this paper only consider end-time QOIs. Second, we must choose how frequently we would like to adapt the mesh. Adapting at every time-step can be useful, allowing us to fully benefit from the increased accuracy, however, it comes at a considerable cost. Factoring in the cost of solving the mesh-movement PDE for each mesh, and also the memory cost of storing each mesh data structure, there is a point of diminishing returns when it comes to frequency of adaptation. Due to certain software limitations, we were unable to test the effect of using fewer meshes for the solve, and instead default to adapting the mesh at every timestep. Third, we must choose how to bound and smooth our monitor function. In our experiments, we used a global standardization with a shift to ensure the function was positive everywhere in the domain. We then projected the discontinuous, element-wise errors into a $P1$ Continuous Lagrange function space. Lastly, we must choose an equidistribution tolerance for the mesh-movement. This is a measure of the variance of the monitor function value in each element, where an equidistribution of 1 represents the optimally moved mesh for our timestep(s). Once the equidistribution for a mesh changes by less than the equidistribution tolerance, we stop moving the mesh. We use an adaptation tolerance between 10^{-4} and 10^{-3} in our framework, using the Quasi-Newton solver provided by Pyroteus Movement [28, 24].

As per Wallwork (2021) [29, 31], to define our DWR indicators, we have to approximate the forward and adjoint errors for our problem. To achieve this, we use global enrichment, i.e. using another finite dimensional space with more Degrees of Freedom (DoFs). The easiest ways to achieve this are (global) h-refinement and p-refinement. Analogous to h- and p-adaptation, h-refinement refers to solving the problem on a finer mesh, and p-refinements refer to using higher order polynomials to approximate the solution fields. We use p-refinement to define our error indicators in this paper. We note that these approaches tend to be memory-limited[1] as either larger or higher-order meshes need to be stored for global enrichment. Even in the p-refined case, while the number of vertices and elements in the mesh remains unchanged, we have a global increase in the number of nodes in the mesh. In memory-limited cases, it is possible to use a difference quotient [29] approach, but its effectiveness is not studied in this paper.

4 Numerical Experiments

4.1 Burgers' Equation

As an initial proof-of-concept for the framework, we consider the viscous form of Burgers' Equation for a given vector field u on the unit square -

$$\frac{\partial u}{\partial t} + u\nabla u = \nu\nabla^2 u \tag{6}$$

with Neumann boundary conditions ($(\hat{n}\cdot\nabla)u = 0$ on Γ) where Γ is the domain boundary.

[1] We note that a solution to the memory usage of the framework would be to use disk-based checkpointing, i.e. writing objects to disk when they are not immediately required by the program. Firedrake [26] provides this functionality, however due to paucity of time, we were unable to refactor the adaptation libraries to use this feature.

We define a symmetric initial condition with velocity components only in the x-direction as

$$u(x, y, t_0) = \left[4 \sin\left(\pi\left(x - \frac{1}{2}\right)\right), 0\right] \tag{7}$$

We use P1 Lagrange elements for the field u, and first-order implicit Euler time-stepping to run the simulation from $t = 0$ to $t = 1/2$s. We define our QOI as the integral of the field u on the right-hand boundary of the domain($x = 1$). This problem is symmetric about $x = 1/2$, but the QOI being defined only at $x = 1$ makes our adaptation scheme asymmetric. If our r-adaptive framework is set up correctly, we would expect to see

1. A symmetric forward solution
2. Maxima of the error indicators and monitor function on the right-hand side of the domain
3. Adapted meshes with larger cells on the left side of the domain, and smaller cells towards the right edge of the domain

As seen in Figure 1a, the initial shock wave travels to both boundaries, and diffuses a bit before the end time. We have qualitative validation for our framework based on the results of Figure 1. Despite the simplistic nature of the test-case, the qualitative feedback provided useful inputs in terms of finding reasonable equidistribution tolerances, and also to understand the effect of smoothing on the monitor function.

4.2 Scalar Advection

We now move on to the scalar advection equation, for a test case based on the bubble shear experiment in McRae et al.(2018) [24] and Wallwork (2021) [29] considering the movement of a tracer field u defined on the unit square being advected by a time-dependent velocity field $\mathbf{c}$.

$$\frac{\partial u}{\partial t} + \nabla(c \cdot u) = 0 \tag{8}$$

where $\mathbf{c}$ is defined as

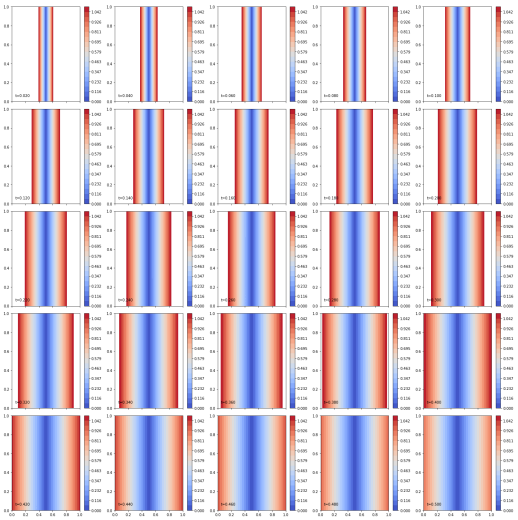
$$\mathbf{c}(x, y, t) = \begin{bmatrix} 2\sin^2(\pi x)\sin(2\pi y)\cos(2\pi t/T) \\ -2\sin(2\pi x)\sin^2(\pi y)\cos(2\pi t/T) \end{bmatrix} \tag{9}$$

We solve this equation with $T = 1$, and an initial condition where the tracer field u is 0 everywhere in the domain, except a bubble of radius 0.1 (Figure 2). The time-dependent $\cos$ term in $\mathbf{c}$ is defined so that the velocity field reverses at $t = T/4$. Thus, the analytical solution at $t = T/2$ is the initial condition. This allows gives us both a quantitative and qualitative means to demonstrate the effectiveness of our setup. For numerical stability, the pure advection equation should be solved with Streamline Upwind Petrov-Galerkin (SUPG) stabilization [29]. We discuss the effect of this stabilization in the next section.

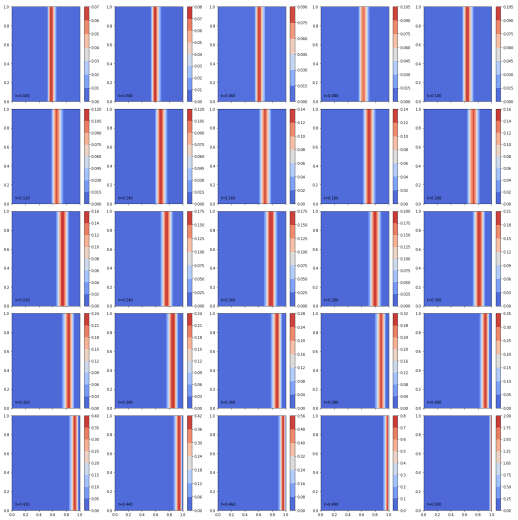
With regards to the implementation, we use continuous $P1$ Continuous Lagrange approximations of both the tracer field u and the velocity field $\mathbf{c}$. Our end-time QOI is the L^2-error between the end-time solution and the initial condition. Given that the analytic QOI is 0, we are able to quantify the errors in our QOI computation. We use implicit Euler time-stepping, with $dt = 0.02$, and run until 0.5s.

The uniform mesh is a mesh with good quality [14], and does a good job with the forward solution, shearing the bubble only slightly. The uniform mesh solution (without SUPG stabilization) has significant oscillations emanating from the bubble edge due to spurious diffusion. Both solves suffer from spurious diffusion, but the adapted solution significantly reduces this diffusion and retains the shape of the bubble better. Due to the re-purposing of resolution, the adapted meshes "hug" the

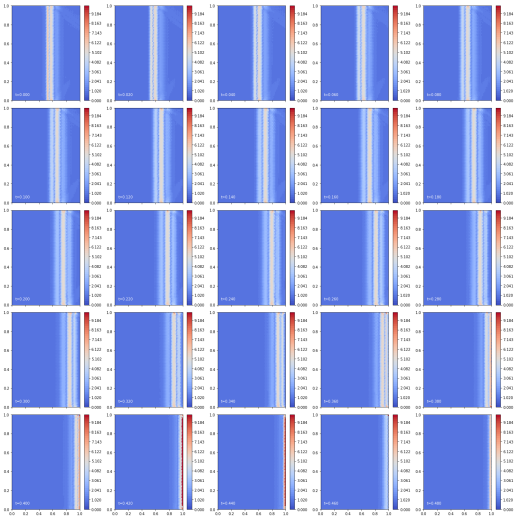
[2]The white spots in the bubble are an artifact of using a smaller colourbar to visualise the spurious diffusion



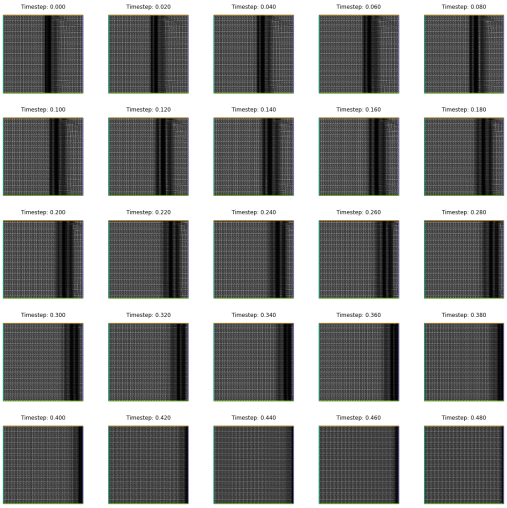
(a) Forward solution



(b) Adjoint solution



(c) Monitor Function



(d) Adapted Mesh sequence

Figure 1: Experiment Setup (Burgers' Equation)

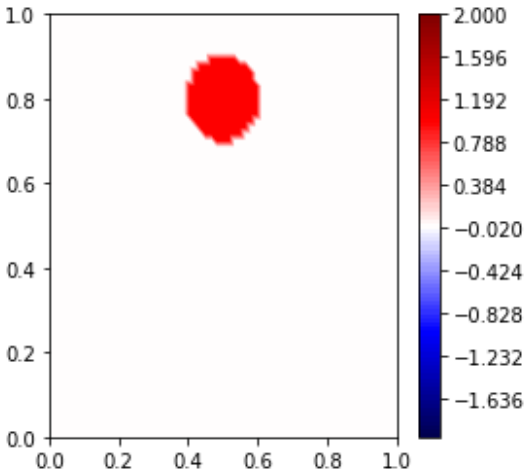
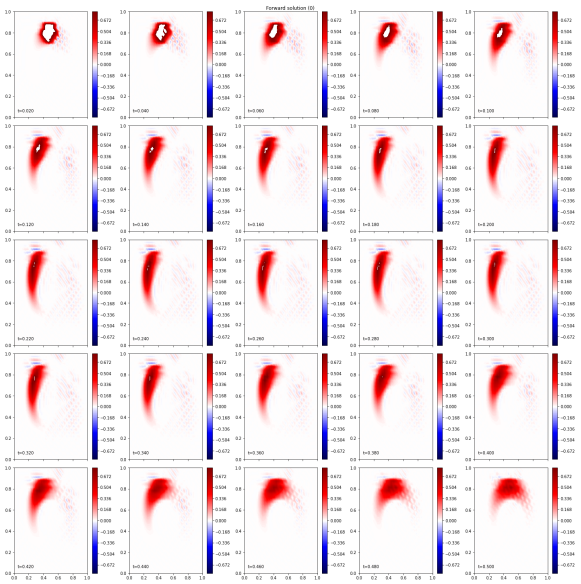
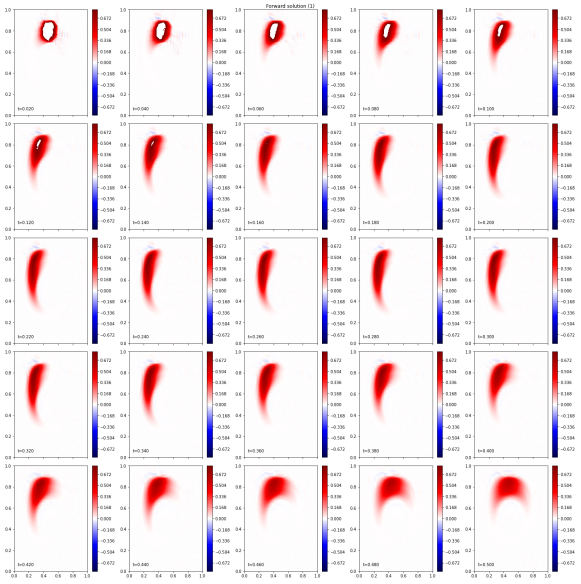


Figure 2: Initial Condition for Bubble Shear test case (64x64)



(a) Forward solution on a uniform mesh (32x32) [2]



(b) Forward solution on adapted meshes (32x32)

Figure 3: Comparison of Forward Solves for Bubble Shear test case

solution features, and add resolution to regions with sharp discontinuities. The adapted meshes can be seen in Figure 4. Our results showed some promise on this qualitative side, but did not confirm our hypothesis that we could lower the error in the QOI computation, with both the uniform and adapted QOI being within 0.1% (in terms of L^2 error) of the true analytical solution.

5 Results and Analysis

The bubble shear test case described in Section 4.2 was used to run all our experiments. We ran our experiments to benchmark our performance while adapting the mesh at every timestep, varying various parameters. Our hypothesis was that using DWR indicators as defined in Becker and Rannacher (2001) [8], we could significantly reduce error in QOI estimation. For this test case, and a uniform mesh, that would typically result in reducing the error from an already small value of 1% (depending on mesh size). As such, we were unable to significantly improve this computation, and in many of

Figure 4: Adapted Meshes (32x32)

our runs, our framework actually performed marginally worse than the uniform mesh. We suspect this is due to differences between the analytical and numerical solutions, as well as errors introduced in the many mesh-to-mesh projections we perform [29, Ch. 2.8]. Nonetheless, the effect of mitigating spurious diffusion as seen in Figure 3, as well as the qualitative result of resolving solution features better still provide us with interesting insights.

5.1 Experiment Setup

We used an Standard E4-2as v4 Microsoft Azure server, with 2 vCPUs, and 32 GiB of RAM to run all of our benchmarks. Detailed experiment data can be found at this OneDrive link. In each experiment, we used $dt = 0.02, T = 1$, and adapted the mesh at all timesteps.

In addition to re-computing the QOI on the adapted mesh sequence, we also look at certain mesh quality measures for each adapted mesh. We compute the aspect ratio (as defined in Knupp (2007) [16]), the skewness (as defined in Kurowski (2004) [18]), and the scaled jacobian (as defined in Knupp (2007) [16]). The aspect ratio relates to the geometrical difference between an element and the ideal element, skewness to an angular measure of deviation from the ideal element, and the scaled jacobian refers to a scaling factor that accounts for element orientation. For good quality meshes, we require aspect ratio between 1 and 100 [16], skewness under 5 [14], and scaled jacobian to be positive [17] at minimum.

5.2 Effect of Mesh Size

With SUPG stabilization [29], the uniform mesh and adapted mesh perform equally well on the quantitative side, both being within 1% of the true solution with a 1-5% relative difference in the magnitude of QOI error, with the uniform mesh performing better. We saw very little change in the QOI value with increase in mesh size, indicating we were converging to the numerical solution, but not the analytical solution. We used an equidsitribution tolerance of 10^{-3} for all meshes. While this

problem may not be representative of the benefits of goal-based r-adaptation, the qualitative results especially at lower resolutions were promising. Despite not improving the QOI computation in this case, the adapted meshes seen in Figure 4 show the capability of our framework to create high-quality [14] meshes that aid in resolving solution features. Table 1 demonstrates these results.

Mesh Size [3]	SUPG	Adapt Time/Mesh (s)	Max Aspect Ratio	Max Skewness	Min Scaled Jacobian
32	No	9.98	10.03	1.34	0.15
48	No	21.31	13.01	1.35	0.11
64	No	39.94	36.13	1.43	0.08
32	Yes	10.25	7.25	1.24	0.185
48	Yes	21.8	12.91	1.50	0.05
64	Yes	34.74	32.62	1.45	0.07
96	Yes	84.35	26.19	1.46	0.06
128	Yes	176.28	77.5	1.55	0.008

Table 1: Mesh Size vs Quality

5.3 Effect of Equidistribution Tolerance

The mesh movement algorithm is given a termination condition based on an equidistribution tolerance. If we set this tolerance to be too high, iterations will converge without actually moving the mesh. If we set this tolerance too low, we increase the cost of the mesh movement, and might end up with poorer quality meshes as a result of tangling. We found that an equidistribution tolerance between 10^{-4} and 5×10^{-3} gave us the best results in terms of the run-time and accuracy trade-off. Results from using different equidistribution tolerances on a 64x64 mesh can be seen in Table 2. For these experiments, we did not use SUPG stabilization.

Equidistribution Tol.	Adapt Time/Mesh (s)	Max Aspect Ratio	Max Skewness	Min Scaled Jacobian
5×10^{-3}	27.2	24.77	1.412	0.107
1×10^{-3}	39.94	36.13	1.43	0.08
5×10^{-4}	43.83	16.13	1.38	0.128
1×10^{-4}	55.96	46.40	1.467	0.07
5×10^{-5}	63.496	55.02	1.475	0.06

Table 2: Mesh Size vs Quality

With both tables in hand, we can conclude that our adaptation scheme takes time proportional to N^2 (per mesh) where the mesh is of size NxN. We can also see that lowering the equidistribution tolerance has a similar effect on the run-time for the adaptation.

5.4 Stabilization Effect

We found a more interesting use-case for the mesh adaptation algorithm when we experimented without using SUPG stabilization. What we found was that the uniform mesh would suffer from spurious diffusion, causing oscillations and noise, whereas the adapted meshes mitigated this problem to a large extent. With this in mind, one might consider the use of goal-oriented r-adaptation in order to increase a limiting timestep for stability. This can potentially be useful when running simulations at larger scales. We hope to examine this effect in future works.

[3] Mesh Size N implies there are N elements in the x- and y-directions, giving us an NxN mesh

6 Limitations and Scope for Future Work

Between a dearth of time and compute power, and inexperience in the field of FEM and mesh adaptation, this work has large scope for improvement. On the mathematical side, we consider only the viscous Burgers' Equation, and the pure scalar advection equation. While these PDEs are fundamental to the field of GFD, our experiments are still far removed from a practical application such as desalination plant outfall or vortices in a shallow water model [7, 31]. Moreover, both our experiments are defined only in two spatial dimensions, which could be unrepresentative of the effectiveness of the approach in practical applications. Future works on this topic should address this issue by solving more complex problems, like for example, the Navier-Stokes Equations for incompressible fluids in 3-D, perhaps with an eye towards more practical problem setups.

Part of the reason we were unable to achieve those practical frameworks is that the framework we finally settled on came after a long period of research and development. We had difficulty achieving stability in our model, as there are many moving parts within the framework. We were unable to test all the possible configurations of our framework for various reasons [4], but our qualitative results, especially at lower resolutions, offer motivation to pursue this framework further. We might refocus our efforts towards the potential to use this framework for upwind-like stabilization as well.

On a practical note, the cost of our framework is not quite as cheap as we expected from an r-adaptive approach. The mesh-movement algorithm is a PDE solve that takes $O(N^2)$ time, and at the moment, we must also solve our problems on a globally enriched mesh. Depending on the means of enrichment and the extent of enrichment, these solves can not only add a large factor to our forward and adjoint run-times, but also add a large memory factor to our computation. To combat this, one might use of difference quotients [29] in place of enriched forward and adjoint solutions, saving both the run-time cost and memory required for the additional forward and adjoint solves. Using different meshes for each timestep certainly doesn't help our memory usage, and was why our experiments were RAM-limited. We might look to disk-based checkpointing to reduce memory usage[5]. While not the most practical implementation, our framework provides a theoretical blueprint for applications using goal-oriented r-adaptation, and workarounds to its main flaws.

7 Conclusions

Despite the shortcomings of our approach, we contend that this paper presents not only a novel means of approaching goal-oriented problems in GFD, but also demonstrates the effectiveness of the tools used to achieved this. While Firedrake [26] has limited support for mesh adaptation, Pyroteus [30, 28] provides an API that facilitates high-level specification and efficient low-level implementations of goal-oriented error estimators and mesh adaptation algorithms.

This work provides an implementation of concepts discussed across multiple decades, and describes a general framework for goal-based problems with a fixed-size mesh. While we were unable to confirm our hypothesis that we do, in fact, reduce QOI error, we found that we produce better qualitative results than the uniform case, and a stabilization effect that could be of interest in contexts where upwind schemes are not as applicable. The ability to "re-purpose" resolution afforded to us by r-adaptation is a powerful idea, and combined with goal-based meshing, we are able to demonstrate the versatility of the Finite Element Method in this context.

[4] inexperience, time, libraries not having the functionality yet, RAM

[5] in some sense, to *equidistribute* RAM usage to avoid sharp peaks of usage

References

[1] M. S. Alnaes, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software*, 40, 2014.

[2] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[3] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136–156, 2006.

[4] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc/TAO users manual. Technical Report ANL-21/39 - Revision 3.17, Argonne National Laboratory, 2022.

[5] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil M. Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc Web page. `https://petsc.org/`, 2022.

[6] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[7] Werner Bauer, Martin Baumann, Leonhard Scheck, Almut Gassmann, Vincent Heuveline, and Sarah C. Jones. Simulation of tropical-cyclone-like vortices in shallow-water icon-hex using goal-oriented r-adaptivity. *Theoretical and Computational Fluid Dynamics*, 28:107–128, 2014.

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

[9] J. Behrens. Adaptive atmospheric modeling: scientific computing at its best. *Computing in Science & Engineering*, 7(4):76–83, 2005.

[10] Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on Pure and Applied Mathematics*, 44(4):375–417, 1991.

[11] Susanne C Brenner, L Ridgway Scott, and L Ridgway Scott. *The mathematical theory of finite element methods*, volume 3. Springer, 2008.

[12] Chris J. Budd, Weizhang Huang, and Robert D. Russell. Adaptivity with moving grids. *Acta Numerica*, pages 111–241, 2009.

[13] Krzysztof J. Fidkowski and David L. Darmofal. Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *AIAA Journal*, 49(4):673–694, 2011.

[14] Nitin S Gokhale. *Practical finite element analysis*. Finite to infinite, 2008.

[15] Hormoz Jahandari, Scott MacLachlan, Ronald D. Haynes, and Niall Madden. Finite element modelling of geophysical electromagnetic data with goal-oriented hr-adaptivity. *Computational Geosciences*, 24:1257–1283, 2020.

[16] Patrick Knupp. Remarks on mesh quality. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2007.

[17] Patrick M Knupp. Achieving finite element mesh quality via optimization of the jacobian matrix norm and associated quantities. part ii—a framework for volume mesh optimization and the condition number of the jacobian matrix. *International Journal for numerical methods in engineering*, 48(8):1165–1185, 2000.

[18] Paul Kurowski. Finite element analysis for design engineers. 2004.

[19] A. Logg and G. N. Wells. DOLFIN: automated finite element computing. *ACM Transactions on Mathematical Software*, 37, 2010.

[20] A. Logg, G. N. Wells, and J. Hake. DOLFIN: a C++/Python finite element library. In A. Logg, K.-A. Mardal, and G. N. Wells, editors, *Automated Solution of Differential Equations by the Finite Element Method*, volume 84 of *Lecture Notes in Computational Science and Engineering*, chapter 10. Springer, 2012.

[21] Daniel Materna and Franz-Joseph Barthold. Goal-oriented r-adaptivity based on variational arguments in the physical and material spaces. *Computer Methods in Applied Mechanics and Engineering*, 198(41):3335–3351, 2009.

[22] TM McManus, JR Percival, BA Yeager, N Barral, GJ Gorman, and MD Piggott. Moving mesh methods in fluidity and firedrake. Technical report, Technical Report July, Imperial College London, 2017.

[23] Andrew T. T. McRae, Colin J. Cotter, and Chris J. Budd. Optimal-transport–based mesh adaptivity on the plane and sphere using finite elements. *SIAM Journal on Scientific Computing*, 40(2):A1121–A1148, 2018.

[24] Andrew T. T. McRae, Colin J. Cotter, and Chris J. Budd. Optimal-transport–based mesh adaptivity on the plane and sphere using finite elements. *SIAM Journal on Scientific Computing*, 40(2):A1121–A1148, 2018.

[25] Lawrence Mitchell and Eike Hermann Muller. High level implementation of geometric multigrid solvers for finite element problems: applications in atmospheric modelling. *Journal of Computational Physics*, 327:1–18, 2016.

[26] Florian Rathgeber, David A. Ham, Lawrence Mitchell, Michael Lange, Fabio Luporini, Andrew T. T. McRae, Gheorghe-Teodor Bercea, Graham R. Markall, and Paul H. J. Kelly. Firedrake: automating the finite element method by composing abstractions. *CoRR*, abs/1501.01809, 2015.

[27] J. N. Reddy. *Introduction to the Finite Element Method*. McGraw-Hill Education, New York, 4th edition. edition, 2019.

[28] Joseph G. Wallwork. *Pyroteus Mesh Movement Toolkit*. https://github.com/pyroteus/movement.

[29] Joseph G. Wallwork. *Mesh Adaptation and Adjoint Methods for Finite Element Coastal Ocean Modelling*. PhD thesis, Imperial College London, 2021.

[30] Joseph G. Wallwork. Pyroteus goal-oriented mesh adaptation toolkit, September 2021.

[31] Joseph G. Wallwork, Nicolas Barral, David A. Ham, and Matthew D. Piggott. Goal-oriented error estimation and mesh adaptation for tracer transport modelling. *Computer-Aided Design*, 145:103187, 2022.