DynAMO: Multi-agent reinforcement learning for dynamic anticipatory mesh optimization with applications to hyperbolic conservation laws

T. Dzanic[a,*], K. Mittal[a], D. Kim[b], J. Yang[a], S. Petrides[a], B. Keith[b] and R. Anderson[a]

[a]*Lawrence Livermore National Laboratory, Livermore, CA 94550, United States of America*
[b]*Division of Applied Mathematics, Brown University, Providence, RI 02912, United States of America*

ARTICLE INFO

Keywords:
Adaptive mesh refinement
Finite element methods
Scientific machine learning
Reinforcement learning
Hyperbolic conservation laws

ABSTRACT

We introduce DynAMO, a reinforcement learning paradigm for Dynamic Anticipatory Mesh Optimization. Adaptive mesh refinement is an effective tool for optimizing computational cost and solution accuracy in numerical methods for partial differential equations. However, traditional adaptive mesh refinement approaches for time-dependent problems typically rely only on instantaneous error indicators to guide adaptivity. As a result, standard strategies often require frequent remeshing to maintain accuracy. In the DynAMO approach, multi-agent reinforcement learning is used to discover new local refinement policies that can anticipate and respond to future solution states by producing meshes that deliver more accurate solutions for longer time intervals. By applying DynAMO to discontinuous Galerkin methods for the linear advection and compressible Euler equations in two dimensions, we demonstrate that this new mesh refinement paradigm can outperform conventional threshold-based strategies while also generalizing to different mesh sizes, remeshing and simulation times, and initial conditions.

1. Introduction

The numerical approximation of partial differential equations (PDEs) has broadly relied on discretization techniques such as finite volume, finite difference, and finite element methods. These techniques present a general framework for targeting a wide variety of governing equations arising in engineering and scientific applications. While their formulations and numerical properties vary, the methods above share the common task of effectively discretizing the problem domain onto a mesh upon which the governing equations can be solved numerically. In many applications, mesh discretization has a considerable impact on accuracy and efficiency. In particular, for systems with a large spatio-temporal variation of scales — e.g., systems with regions of steep gradients and small-scale features that require high resolution — the use of a uniform mesh discretization can result in an exceedingly inefficient numerical approximation. These multi-scale systems are common in scientific and engineering applications, with a wide variety of examples ranging from within fluid dynamics and astrophysics to solid mechanics and bioengineering [1–5]. An efficient treatment of these problems often requires some form of adaptive mesh refinement (AMR).

AMR strategies aim to balance computational cost and solution accuracy by selectively increasing the resolution of the mesh in regions where the largest discretization errors are generated while keeping the resolution low in regions where the errors are sufficiently small. Ideally, the end result achieves maximal error reduction with minimal computational cost. AMR is widely used to solve PDEs, with the typical approach involving some form of an instantaneous *error indicator* to dictate which mesh elements are to be *refined* and/or coarsened (i.e., *de-refined*). However, a long-standing limitation of AMR strategies for time-dependent PDEs lies in the general observation that optimizing a mesh for computational errors at a fixed point in time will deliver a mesh that is suboptimal at future times. Indeed, adapted meshes can quickly become inadequate as a system evolves and introduce large errors to the discrete solution that persist indefinitely. To mitigate this effect, it is often necessary to compute with a dynamic mesh that is re-adapted frequently. Unfortunately, the computational cost associated with mesh adaptation is not negligible and can quickly

*Corresponding author
dzanic1@llnl.gov (T. Dzanic)
ORCID(s): 0000-0003-3791-1134 (T. Dzanic); 0000-0002-2062-852X (K. Mittal); 0000-0001-9892-0611 (D. Kim); 0000-0003-2234-6224 (J. Yang); 0000-0002-1284-5495 (S. Petrides); 0000-0002-6969-6857 (B. Keith); 0000-0002-3508-9944 (R. Anderson)

negate the practical benefits of AMR unless the refinement frequency is low and the solution does not vary much over the time period between re-adaptation steps.

The core belief motivating this work is that a more effective dynamic AMR paradigm can be achieved with *anticipatory* refinement. Namely, we seek a refinement strategy to predict future error propagation and respond by preemptively refining the mesh. Compared to standard AMR approaches based only on instantaneous indicators, an anticipatory refinement strategy has the potential to achieve far superior accuracy and performance owing to its ability to refine the mesh *before* future discretization errors appear. Most importantly, this would allow for longer time intervals between mesh adaptation, which is particularly beneficial for compute architectures where bandwidth is the bottleneck.

Developing an anticipatory refinement strategy for time-dependent PDEs presents significant challenges. Except for trivial governing equations for which analytic solutions exist, the future spatio-temporal behavior of a PDE solution is not known *a priori*, and directly predicting the evolution of the error is computationally infeasible. Instead of trying to directly predict the behavior of the error by techniques such as solving an adjoint problem [6], we find that using surrogate models to approximate the error evolution offers a more tractable approach. One such approach is through error transport equations of Tyson et al. [7], which present an evolution equation for the error functional [8–10], or through even simple retrospective error estimates based on re-propagating the solution and measuring the error evolution. Further within the realm of surrogate modeling, machine learning (ML) techniques have shown promise as an effective tool for error estimation [11, 12]. This success has driven efforts towards combing ML and AMR techniques, with the earliest works dating as far back as the early 1990s [13]. To date, ML has been used to design marking strategies [14–17], goal-oriented AMR techniques [18–21], and mesh density function-based mesh adaptation [13, 22–27]. However, an effective anticipatory refinement strategy for time-dependent PDEs requires optimizing long-term objectives, i.e., solution accuracy/cost over time. This in itself is a sequential decision-making problem as this objective at any given time is dependent on all of the refinement decisions leading up to that point. For this class of problems, reinforcement learning (RL) methods have shown promise as they are inherently tailored towards optimizing long-term objectives. The use of RL for AMR was first proposed in Yang et al. [28], where RL methods with variable-size global state and action spaces were applied to account for the changing mesh topology. This approach was further investigated in Yang et al. [29], where the authors used a multi-agent graph neural network with a team reward to generalize to arbitrary unstructured meshes, in Freymuth et al. [30], where local individual rewards were used to produce higher refinement levels, and in Foucart et al. [31], where AMR was formulated as a partially observable Markov decision process in which a single agent applied to each element can observe only local surrounding information. For all of these works, AMR via RL was applied to scalar linear time-dependent or stationary PDEs.

Given the potential of RL for optimizing long-term objectives in complex environments, the goal of this work is to introduce DynAMO, a new reinforcement learning paradigm for *Dynamic Anticipatory Mesh Optimization*, in order to establish and explore the feasibility of using RL to guide anticipatory mesh refinement strategies for complex nonlinear systems of PDEs. In particular, we look at learning refinement strategies for discontinuous Galerkin finite element methods (FEMs), including h-refinement and p-refinement, for general hyperbolic conservation laws, with applications to the linear advection equation and the compressible Euler equations. Due to the local domain of influence of these governing equations, we consider a decentralized partially observable Markov decision process model of dynamic mesh optimization through a multi-agent reinforcement learning viewpoint, with independent agents corresponding to elements in the mesh that observe local surrounding information. Furthermore, we introduce novel and highly generalizable observations and reward functions that 1) enable anticipatory mesh refinement for arbitrary nonlinear hyperbolic conservation laws without analytic solutions; 2) show invariance to problem scale and mesh resolution; and 3) allow for user-controlled error/cost targets at evaluation time. We demonstrate the efficacy of DynAMO in multi-dimensional numerical experiments ranging from simple linear transport to complex nonlinear shock interactions, thus establishing that anticipatory refinement can unlock levels of efficiency that have remained out of reach with previous, conventional AMR techniques.

The remainder of this manuscript is organized as follows. In Section 2, some background material on the overarching topic of this work is presented, including an overview of hyperbolic conservation laws, finite element methods, adaptive mesh refinement, and reinforcement learning. An in-depth description of the proposed DynAMO approach is given in Section 3, followed by numerical implementation details in Section 4. The results of the numerical experiments for h- and p-refinement on the advection and Euler equations are then shown in Section 5, after which we use Section 6 to draw our final conclusions.

2. Preliminaries

In this section, we present overviews of hyperbolic conservation laws, finite element methods, adaptive mesh refinement, and reinforcement learning that are necessary for the subsequent sections.

2.1. Hyperbolic conservation laws

The primary applications of this work pertain to approximations of hyperbolic conservation laws of the form

$$\begin{cases} \partial_t \mathbf{u} + \boldsymbol{\nabla}\cdot\mathbf{F}(\mathbf{u}) = \mathbf{0}, & \text{for } (\mathbf{x}, t) \in \Omega \times \mathbb{R}_+, \\ \mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}), & \text{for } \mathbf{x} \in \Omega, \end{cases} \tag{1}$$

where $\Omega \subset \mathbb{R}^d$ is a d-dimensional spatial domain, $\mathbf{u} \in \mathbb{R}^m$ is a solution of m variables, and $\mathbf{F}(\mathbf{u}) \in \mathbb{R}^{m\times d}$ is the associated flux. We consider both linear scalar conservation laws as well as nonlinear systems of equations. For the former, the quintessential example is the scalar advection equation, given as

$$\partial_t u + \boldsymbol{\nabla}\cdot(\mathbf{c}u) = 0, \tag{2}$$

where $\mathbf{c} \in \mathbb{R}^d$ is some constant velocity. For the latter, the compressible Euler equations of gas dynamics can be employed, given in the form of Eq. (1) as

$$\mathbf{u} = \begin{bmatrix} \rho \\ \mathbf{m} \\ E \end{bmatrix} \quad \text{and} \quad \mathbf{F}(\mathbf{u}) = \begin{bmatrix} \mathbf{m}^T \\ \mathbf{m} \otimes \mathbf{v} + P\mathbf{I} \\ (E + P)\mathbf{v}^T \end{bmatrix}, \tag{3}$$

where ρ is the density, $\mathbf{m} \in \mathbb{R}^d$ is the momentum vector, and E is the total energy. Furthermore, $\mathbf{I}$ denotes the identity matrix in $\mathbb{R}^{d\times d}$ and $\mathbf{v} = \mathbf{m}/\rho$ denotes the velocity. The pressure P can be computed as

$$P = (\gamma - 1)\left(E - \frac{1}{2}\rho\mathbf{v}\cdot\mathbf{v}\right), \tag{4}$$

where the specific heat ratio is taken as $\gamma = 1.4$ in this work.

The conservation law in Eq. (1) can also be represented in a quasi-linear formulation, given as

$$\partial_t \mathbf{u} + \mathbf{A}\cdot\nabla\mathbf{u} = \mathbf{0}, \tag{5}$$

where $\mathbf{A} \in \mathbb{R}^{m\times m\times d}$ is the flux Jacobian tensor defined

$$\mathbf{A} = \frac{\partial \mathbf{F}(\mathbf{u})}{\partial \mathbf{u}}, \tag{6}$$

and the product $\mathbf{A}\cdot\nabla\mathbf{u} \in \mathbb{R}^m$ is defined $(\mathbf{A}\cdot\nabla\mathbf{u})_i = \sum_{j,k} A_{ijk}(\partial u_j/\partial x_k)$. For the advection equation, this yields an identical formulation as the conservative form due to the linearity of the flux, with the flux Jacobian simply reducing to the velocity, i.e., $\mathbf{A} = \mathbf{c}$. For the Euler equations, the flux Jacobian becomes much more complex as the flux is nonlinear with respect to a vector-valued solution, and its derivation is left to Appendix A. However, this nonlinearity can cause discrepancies between the conservative formulation and the quasi-linear form, particularly so in the vicinity of discontinuities. Nevertheless, even in the nonlinear case, the structure of the flux Jacobian still possesses some useful information about the underlying system as it gives an approximation of the characteristic propagation velocity of the solution.

An important property of hyperbolic conservation laws is that this characteristic propagation velocity is finite, such that the domain of influence of any point is locally supported in space and time. An upper bound for the extent of the domain of influence $\mathcal{D}$ for some arbitrary point $\mathbf{x}_0$ over some time period $[t, t + \Delta\tau]$ can readily be estimated as

$$\mathcal{D}(\mathbf{x}_0, \Delta\tau) = \mathbb{B}^d(\mathbf{x}_0, \Delta\tau\lambda_{\max}), \tag{7}$$

where $\mathbb{B}^d(\mathbf{x}', r)$ is a d-ball centered at $\mathbf{x}'$ with radius r and $\lambda_{\max}$ is an estimate of the maximum wavespeed of the system. This domain of influence effectively ensures that the solution at $\mathbf{x}_0$ cannot be affected by the solution at any point farther than $r = \Delta\tau\lambda_{\max}$ away, which can greatly reduce the level of information necessary to adequately predict the local evolution of the system over a sufficiently small time interval.

2.2. Finite element methods

To numerically approximate hyperbolic conservation laws of the form of Eq. (1), we utilize finite element methods to discretize the governing equations, particularly the nodal discontinuous Galerkin (DG) method [32]. In this approach, the domain Ω is partitioned into N elements Ω_k such that $\Omega = \bigcup_N \Omega_k$ and $\Omega_i \cap \Omega_j = \emptyset$ for $i \neq j$. Within each element Ω_k, the discrete solution $\mathbf{u}_h(\mathbf{x})$ is approximated via a set of n_s nodal interpolating polynomials as

$$\mathbf{u}_h(\mathbf{x}) = \sum_{i=1}^{n_s} \mathbf{u}_i \phi_i(\mathbf{x}) \subset V_h, \tag{8}$$

where $\mathbf{x}_i$ is a set of solution nodes, $\phi_i(\mathbf{x})$ are their associated nodal basis functions which possess the property $\phi_i(\mathbf{x}_j) = \delta_{ij}$, and V_h is the piece-wise polynomial space spanned by the nodal basis functions. We use the short-hand notation $\mathbf{u}_i = \mathbf{u}_h(\mathbf{x}_i)$ for brevity and denote the order of the approximation as $\mathbb{P}_p$, where p is the maximal order of $\mathbf{u}_h(\mathbf{x})$.

The problem is solved through the weak formulation by applying the DG approximation and integrating Eq. (1) with respect to a test function $\mathbf{w}_h(\mathbf{x})$, which resides in the same finite element space as $\mathbf{u}_h(\mathbf{x})$, yielding

$$\sum_{k=1}^{N} \left\{ \int_{\Omega_k} \partial_t \mathbf{u}_h \cdot \mathbf{w}_h \, \mathrm{d}V + \int_{\partial\Omega_k} \hat{\mathbf{F}}(\mathbf{u}_h^-, \mathbf{u}_h^+, \mathbf{n}) \cdot \mathbf{w}_h \, \mathrm{d}S - \int_{\Omega_k} \mathbf{F}(\mathbf{u}_h) \cdot \nabla \mathbf{w}_h \, \mathrm{d}V \right\} = 0. \tag{9}$$

As the approximate solution is discontinuous across element boundaries, the flux function must be replaced by a numerical flux $\hat{\mathbf{F}}(\mathbf{u}_h^-, \mathbf{u}_h^+, \mathbf{n})$ which is dependent on the solution within the element of interest, denoted by the $\mathbf{u}_h^-$, and its face-adjacent neighbor, denoted by $\mathbf{u}_h^+$, along with the outward facing normal vector $\mathbf{n}$. For the hyperbolic conservation laws considered in this work, the Rusanov approximate Riemann solver [33] is used, such that the numerical flux can be calculated as

$$\hat{\mathbf{F}}(\mathbf{u}_h^-, \mathbf{u}_h^+, \mathbf{n}) = \frac{1}{2} \left(\mathbf{F}(\mathbf{u}_h^-) \cdot \mathbf{n} + \mathbf{F}(\mathbf{u}_h^+) \cdot \mathbf{n} \right) - \frac{1}{2} \lambda_{\max} \left(\mathbf{u}_h^+ - \mathbf{u}_h^- \right). \tag{10}$$

For the advection equation, the maximum wavespeed is simply taken as $\lambda_{\max} = |\mathbf{c} \cdot \mathbf{n}|$, which reduces to the upwind flux, whereas for the Euler equations, the Davis wavespeed estimate [34] is used.

2.3. Adaptive mesh refinement

While AMR techniques for various classes of governing equations and numerical methods differ in their specific implementations, they typically fall within the widely-used paradigm of the `SOLVE`→`ESTIMATE`→`MARK`→`REFINE` loop. A complete pass through of the above sequence for time-dependent PDEs consists of solving the governing equations over some time interval, estimating some error indicator functional (or functionals) for each element in the mesh, marking which elements are to be refined and de-refined, and applying the resulting refinement actions onto the mesh. A more in-depth overview of the steps in the AMR sequence is presented below.

`SOLVE`. In this step, the semi-discrete form of the governing time-dependent PDE is evolved using an appropriate temporal integration method, the specific choice of which is typically dependent on the characteristics of the physical system being solved. For each `SOLVE` step in the loop, the simulation is advanced over the time interval $[t, t+T]$, where T is referred to as the remesh time as this is the time period over which the mesh is fixed. We remark here that T is not the temporal integration time step Δt, the latter of which is dependent on the numerical solver and is largely irrelevant in the AMR procedure. In most applications, the remesh time interval is many times larger than the simulation time step as frequent remeshing (i.e., low T) incurs a large computational cost overhead.

`ESTIMATE`. Once the simulation has been advanced to the next remesh time, the `ESTIMATE` step consists of computing a local error indicator for each element in the mesh. The indicator functional maps the solution to a set of element-wise indicator values e_i, i.e.,

$$\mathbf{u}_h(\mathbf{x}) \mapsto e_i \quad \text{for all } i \in \{1, \ldots, N\}. \tag{11}$$

The indicator is assumed to be deterministic, such that an identical input state would result in an identical indicator distribution throughout the domain. Furthermore, the `ESTIMATE` step may result in multiple indicators for each element.

`MARK`. Given the distribution of the error indicator (or indicators) for the elements in the mesh, the `MARK` step maps this distribution to a set of refinement decisions for the elements through a policy $\pi(e_i) \rightarrow \{0, 1\} \; \forall \; i \in \{1, \ldots, N\}$, where the resulting decisions 0 and 1 correspond to de-refinement and refinement actions, respectively. We remark

here that the output of this policy may not necessarily be binary as multiple refinement methods may be allowed, a notion that will be further explained in the subsequent step of the refinement loop, and that an element's refinement decision may depend on other elements' decisions. This step of the refinement process is largely reliant on heuristics, with a wide variety of approaches, including absolute threshold-based methods, i.e.,

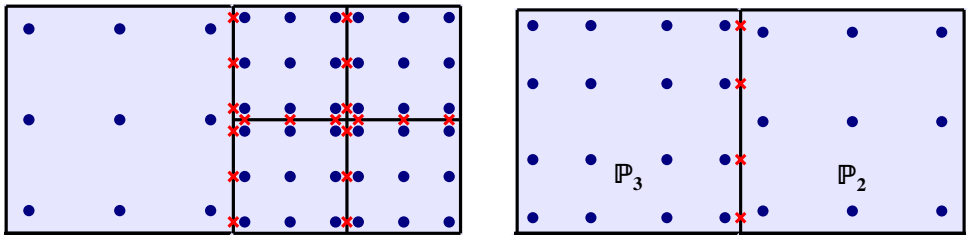
$$\pi(e_i) = \begin{cases} 1, & \text{if } e_i > \theta, \\ 0, & \text{else,} \end{cases} \tag{12}$$

where θ here is some arbitrary global error threshold, and relative threshold-based methods, i.e.,

$$\pi(e_i) = \begin{cases} 1, & \text{if } \frac{e_{\max} - e_i}{e_{\max} - e_{\min}} > \theta, \\ 0, & \text{else,} \end{cases} \tag{13}$$

where $\theta \in [0, 1]$ here is some arbitrary relative error threshold and $e_{\max}$ and $e_{\min}$ are the global maximum and minimum error indicator values, respectively.

`REFINE`. With the output of the `MARK` step, the appropriate refinement and de-refinement actions can be then be applied to the mesh. There exist a variety of mesh refinement methods for AMR, with the most ubiquitous approach being h-refinement. For this method, show on the left-hand side of Fig. 1, refining an element of choice subdivides it into a set of sub-elements, typically referred to as the *child elements*, whereas de-refining a group of sub-elements coalesces them into a singular element, typically referred to as the *parent element*. The number of successive refinement actions performed between a child element and its native parent element is referred to as the *refinement depth*. The use of h-refinement typically results in a non-conforming mesh for which the numerical method of choice must appropriately handle (e.g., the use of constrained degrees of freedom in FEM).



(a) h-refinement (b) p-refinement

Figure 1: Schematic of one-level h-refinement (left) and p-refinement on a two-dimensional quadrilateral discontinuous Galerkin finite element mesh with equal number of quadrature nodes and basis functions. Blue circles represent volume quadrature nodes, red crosses represent surface quadrature nodes.

Due to the element-local polynomial approximation used in finite element methods, another, albeit not as common, AMR approach can be achieved with p-refinement, shown on the right-hand side of Fig. 1. For this method, the order of the polynomial basis within an element is increased or decreased appropriately in accordance with the output of the marking policy, such that effective order of accuracy and number of degrees of freedom within elements can vary. Similarly to h-refinement, this adaptation technique results in a degree of non-conformity, although in the basis instead of in the mesh.

Additional refinement methods exist separately from h-refinement and p-refinement, although they are not considered in this work. For FEM, the joint use of h- and p-refinement, dubbed hp-refinement, may be yield superior performance for certain applications as it is typically assumed that h-refinement is better suited for regions with discontinuous features whereas p-refinement is better suited for regions with smooth features. Furthermore, mesh deformation, dubbed r-refinement, may be used in applications where dynamically varying features and geometries are encountered.

2.4. Reinforcement learning

Reinforcement learning enables an agent or a team of agents to solve sequential decision-making problems in an unknown environment by interacting with the environment and receiving rewards over many episodes [35]. In a

single-agent scenario, the environment is formalized as a stationary Markov decision process (MDP) [36], a discrete-time stochastic process $\{\mathcal{S}, \mathcal{A}, P, R, \gamma\}$ with the following components:

- A state space $\mathcal{S}$.
- An action space $\mathcal{A}$.
- A transition function $P: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ that defines the distribution $P(s_{\tau+1}|s_\tau, a_\tau)$ of next states $s_{\tau+1} \in \mathcal{S}$ given an action $a_\tau \in \mathcal{A}$ taken by the agent at state $s_\tau \in \mathcal{S}$, and an initial state distribution $P_0: \mathcal{S} \to [0, 1]$, from which the initial state of each episode is sampled.
- A reward function $R: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ which includes a temporal discount factor $\gamma \in (0, 1)$ that determines the value of future rewards.

We use $\tau = 0, 1, \ldots$ to denote the discrete time index. An agent's goal is to find a policy $\pi: \mathcal{S} \times \mathcal{A} \to [0, 1]$, which determines the probability distribution of actions $\pi(a_\tau|s_\tau)$ to take at each state, to maximize the expected cumulative discounted reward $J(\pi) := \mathbb{E}[\sum_{\tau=0}^{\infty} \gamma^\tau R(s_\tau, a_\tau)]$, where the expectation is taken over the environment transition and policy. In practice, the transition function contains a termination criterion, formalized by an absorbing state with zero reward, that triggers a reset of the environment for a new episode, so that each episode has a finite time horizon.

An alternative RL approach is the multi-agent viewpoint, which is the method of choice in this work. The environment is formalized as a decentralized partially observable Markov decision process (Dec-POMDP) [37], denoted by $\{\mathcal{S}, \mathcal{O}, \mathcal{A}, P, R, O, \gamma\}$, which extends an MDP in the following ways: 1) multiple agents, indexed by $i = 1, \ldots, N$, interact with the shared environment; 2) each agent i has a limited partial observation $o^i_\tau = O(s_\tau) \in \mathcal{O}$, produced by an observation function O from the true state; 3) actions are decentralized, meaning that each agent i acts according to its own policy $\pi^i(a^i_\tau|o^i_\tau)$; and 4) the environment transition $P(s_{\tau+1}|s_\tau, a^1_\tau, \ldots, a^N_\tau)$ depends on the actions of all agents. For the purposes of this work, the Dec-POMDP definition is specialized in the following two ways. First, agents are homogeneous, meaning that they have the same observation and action spaces. Second, all agents share the same definition of reward function R, but each agent receives its own individual reward value $R(s, a^i, i)$ which depends on its own action and situation within the state. Each agent i learns a policy $\pi^i(a^i|o^i)$ to maximize the objective

$$J(\pi^i) := \mathbb{E}_{s_0 \sim P_0, \mathbf{a}_\tau \sim \boldsymbol{\pi}(\cdot|s_\tau), s_{\tau+1} \sim P(\cdot|s_\tau, \mathbf{a}_\tau)} \left[\sum_{\tau=0}^{\infty} \gamma^\tau R(s_\tau, a^i_\tau, i) \right], \tag{14}$$

where $\mathbf{a} = (a^1, \ldots, a^N)$ denotes the collection of all agents' actions and $\boldsymbol{\pi}(\mathbf{a}|s) = \prod_{i=1}^{N} \pi(a^i|o^i)$ denotes the joint policy.

3. Methodology

Given the drawbacks of standard AMR techniques for the approximation of complex hyperbolic conservation laws, the objective of this work is to explore a reinforcement learning approach for anticipatory mesh refinement strategies. As reinforcement learning is quite general, a wide variety of approaches are feasible for integrating AMR within the RL framework. As such, to have a robust and reliable policy for AMR, certain desirable attributes and features would ideally be satisfied:

1. The policy should be able to generalize to arbitrary hyperbolic conservation laws without requiring substantial domain specific knowledge, including complex nonlinear systems of equations which do not possess closed-form expressions for the error (i.e., analytic solutions).
2. The user should be able to control the relative degree of refinement (i.e., error and cost targets) of the policy *at evaluation time*.
3. The resulting policy should be insensitive to or, ideally, completely invariant to mesh/problem scale, choice of error estimator, mesh resolution, simulation time, and remeshing time interval.

With these characteristics in mind, we present the proposed DynAMO approach, a *Dynamic Anticipatory Mesh Optimization* method for hyperbolic conservation laws using multi-agent reinforcement learning. The goal of DynAMO is to learn more optimal long-term refinement strategies which consider the underlying physics of the problem, allowing for more efficient refinement decisions which account for the spatio-temporal evolution of the solution, longer

remeshing time intervals, and the preemptive refinement of mesh regions prior to the introduction of discretization error. Therefore, novel formulations for the observation and reward functions are introduced to best achieve these target goals and features. We remark here that the primary purpose of this work is to evaluate the ability of RL and the proposed observation and reward formulations as methods of achieving more efficient anticipatory mesh refinement policies for complex nonlinear systems of equations. As such, the focus of this work will be on discovering policies for one-level h- and p-refinement in finite element methods on structured (but non-conforming), two-dimensional, periodic meshes. While the extension to unstructured meshes and curved mesh elements, boundary conditions and complex geometries, and multi-level refinement poses a necessary task for the proposed approach, these difficulties are orthogonal to the study in this work and have been addressed by the formulations presented by the authors in Yang et al. [29] and by Foucart et al. [31] and Freymuth et al. [30] in the context of simpler linear problems. Furthermore, extensions to alternate refinement actions (such as conforming splitting in unstructured meshes) and mesh movement in r-refinement may be achieved through increasing the size of the action space or changing the action space to a continuous output, respectively.

3.1. Dynamic mesh optimization as a Dec-POMDP

DynAMO is instantiated as a Dec-POMDP model of dynamic mesh optimization with the following components: 1) a set of agents corresponding to each element in the mesh; 2) an observation space for each agent consisting of a local spatial window centered on the agent's element that observes derived quantities of the solution in the surrounding elements at the given time; 3) an action space corresponding to refinement and de-refinement operations; 4) a transition function which applies these actions and advances the PDE until the next remesh time; and 5) a reward function which encodes the efficacy of the refinement/de-refinment actions on the local error.

3.1.1. Agent

We consider a multi-agent reinforcement learning approach to AMR, where each element Ω_i is represented by an agent, denoted by an identifier $i \in \{1, \dots, N\}$. Crucially, this viewpoint enables each individual element to make its own refinement or coarsening decision based on its own local observation and independently from other agents. Furthermore, all agents are allowed to act simultaneously at each discrete remeshing step in contrast to single-agent approaches [28, 31], which is critical for maintaining exactly the same mode of execution between training and deployment. For p-refinement, N is constant throughout the simulation, such that the number of agents remains constant. However, for h-refinement, the number of elements varies according to mesh refinement and de-refinement. To avoid the problem of agent creation and deletion, posthumous credit assignment in multi-agent reinforcement learning, and agent co-operation (for coarsening actions in h-refinement), we instead take an alternate approach where the agents are assigned to the initial coarse elements, and any additional elements stemming from subsequent refinement actions on these initial coarse elements also remain under the scope of the original agent. This allows for a constant number of agents throughout the simulation even in the h-refinement case, avoiding the difficulties of agent creation and deletion. Note that while the number of agents is the same during a single episode, it does not have to be the same in different episodes, such that different mesh resolution may be used for training and evaluation.

3.1.2. Observation space

The choice of the agent observation plays arguably the most important role in the design of an anticipatory mesh refinement approach. For time-dependent PDEs such as hyperbolic conservation laws, the observation must have two components: an effective and robust approach for approximating the error distribution within the domain and a general method for predicting the spatio-temporal evolution of the solution and, by extension, the error. These components, which will be described momentarily, form the quantities that are observed by each individual agent.

Given the local domain of influence of hyperbolic conservation laws, it is natural to assume that a local observation window is sufficient to adequately predict the evolution of the underlying physics. As such, we utilize a formulation where each agent i has an individual observation $o^{ij} \in \mathbb{R}^{k_x \times k_y \times n}$ which consists of a $k_x \times k_y$ spatial window with one channel for each of the n observable quantities. The spatial window is centered on the element of agent i and contains the values of $k_x = 2n_x + 1$ and $k_y = 2n_y + 1$ elements, where n_x and n_y specify the number of neighbor elements along each direction, such that $1 \leq j \leq k_x k_y$. An example schematic for this approach is shown in Fig. 2. For h-refinement, these numbers correspond to the initial coarse elements (i.e., agents). This spatial extent is fixed at training time, which constrains the extent of the local domain of influence. As will be later presented, this gives an effective bound maximum remesh time T for the approach.

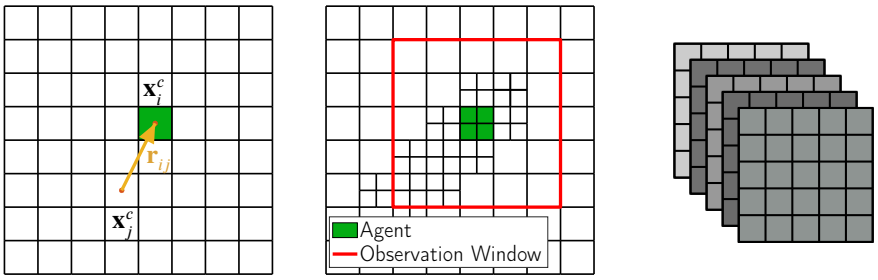


Figure 2: Schematic for an example h-refinement observation setup with an initial mesh of $N = 8^2$ agents/elements, showing the mesh and example displacement vector (left), the observation window of an agent on a refined mesh (center), and the observation channels of an agent with 5 observable quantities (right).

The first observable quantity is an error indicator metric for each element in the observation window. For brevity, we present the observation for an arbitrary agent i with an arbitrary neighbor index j, where j corresponds to any one of the $k_x k_y$ agent/element indices which reside in the spatial window of the observation o^i. As most governing equations do not possess an analytic solution, it is unreasonable to expect that an analytic expression exists for the error in practical applications. As such, we utilize an *error estimator* to approximate the distribution of error within the domain. While the details of the particular error estimators used in this work are left to Section 4, we assume that the error estimator returns a set of quantities $e_\tau^i \; \forall \, i \in \{1, \dots, N\}$, where e_τ^i represents an element-wise constant approximation of the error in element Ω_i at some time index τ. However, observing the raw output of an error estimator poses some drawbacks as this quantity is not scale-invariant and, in a sense, not invariant to the mesh resolution. Instead, we observe a normalized error estimate as

$$o_{\tau,0}^{ij} := -\frac{\log_{10}(e_\tau^j)}{\log_{10}(e_{\tau,\max})}, \tag{15}$$

where a scaled maximum error is defined as

$$e_{\tau,\max} := \alpha |\mathbf{e}_\tau|_\infty. \tag{16}$$

This normalized error observation includes a user-defined parameter α, which can be considered as a form of a relative error threshold that will be used in the reward function to indicate which elements are to be refined or coarsened. A key feature of the proposed approach here is that *α can be varied at evaluation time*, such that the normalized error distribution in the observation can be adjusted by the user. It will be later shown that this allows for the user to control the relative degree of refinement at evaluation time, one of the previously presented critical features of an effective AMR policy. We note here that, in theory, one may observe multiple error components for systems of conservation laws, but we only consider one component in this work which will be discussed in Section 4.

While the normalized error observation is critical to the AMR policy, it does not contain any of the required information to predict its spatio-temporal evolution. Therefore, it is necessary to include some information about the local state in the observation. In Yang et al. [28], the policy observed the high-order FEM solution on an equispaced grid and the physics of the problem were ostensibly learned through the RL approach. However, this method has some notable disadvantages, with the main drawback being that the RL policy has to learn the physics of the underlying PDE essentially from scratch. Furthermore, this approach is not invariant to mesh/problem scale or remesh time. We instead propose an alternate approach which relies on feature engineering based on some useful properties of hyperbolic systems to form a non-dimensional observable quantity which possesses significantly better generalization properties.

We introduce this approach in the form of an example for the linear advection equation, where the advection velocity is given by a constant vector $\mathbf{c}$. We define a displacement vector $\mathbf{r}_{ij}$ as

$$\mathbf{r}_{ij} = \mathbf{x}_i^c - \mathbf{x}_j^c, \tag{17}$$

where $\mathbf{x}_i^c$ is the centroid of the element Ω_i. The propagation velocity along this displacement vector can be simply computed as $\mathbf{c} \cdot \mathbf{r}_{ij} / \|\mathbf{r}_{ij}\|_2$. If this quantity is again normalized by $\|\mathbf{r}_{ij}\|_2$, one yields an estimate for the (inverse) time required for a feature to propagate from $\mathbf{x}_j^c$ to $\mathbf{x}_i^c$. By then introducing the remesh time T as

$$\frac{\mathbf{c} \cdot \mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|_2^2} T,$$

we recover a *non-dimensional* quantity which estimates the likelihood, although not in a probability sense, that element Ω_i will encounter a feature presently in Ω_j in the span of the remesh time T, i.e., in the range $[t_\tau, t_\tau + T]$. This novel observation formulation possesses some very useful properties, namely that it is invariant to problem/mesh scale and remesh time and that the quantity is effectively re-normalized around a critical value of unity.

With this example in mind, we introduce a general form of this novel observable quantity for arbitrary hyperbolic conservation laws. Given m solution variables in the PDE, the next m observable quantities are defined as

$$o_{\tau,k}^{ij} = \left\langle \mathbf{A}_{kl} \frac{\mathbf{u}_k}{\mathbf{u}_l} \right\rangle_j \cdot \frac{\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|_2^2} T \quad \forall\, k \in \{1, \dots, m\}, \tag{18}$$

where $\mathbf{A}_{kl}$ is the k-th component of the flux Jacobian, defined in Section 2, with respect to the chosen solution variable index l at time t_τ. The notation $\langle \cdot \rangle_j$ denotes the average of the quantity $\cdot$ across the element Ω_j. Similarly to the error observation, one may observe multiple components, but we only consider one component in this work which coincides with the solution component used for the error observation. Furthermore, this component is assumed to be non-zero, such that the quantity in Eq. (18) is always well-defined (e.g., density and total energy in the Euler equations). It must be noted that for nonlinear conservation laws, this non-dimensional likelihood quantity is no longer exact (i.e., a value of unity does not necessarily mean that a feature will propagate across the given distance), but it instead approximates a linearization of the exact quantity around $(\mathbf{x}, t) = (\mathbf{x}_i^c, t_\tau)$.

Finally, while this non-dimensional likelihood quantity gives an essential representation of the spatio-temporal evolution of the error, the linearized formulation may not present enough information to adequately predict strongly nonlinear interactions (e.g., shocks in the Euler equations). For such problems for the Euler equations that do exhibit strong discontinuities, we also append the element-wise averaged conserved ($[\rho, \mathbf{m}, E]$) and primitive variables ($[\rho, \mathbf{v}, P]$) to the observation as this was found to yield moderately better results.

Remark (Aggregation for h-refinement)**.** *For h-refinement, as the agents are defined on the coarse elements, it is necessary to aggregate the observables from any child elements up to the coarse parent level (i.e., the agent level). For the error field, this is performed as an L^2 norm across the child elements, whereas for the other derived quantities such as the flux Jacobian and solution, a simple area-weighted average was performed. Any error normalization and element-wise averaging was performed prior to the aggregation.*

3.1.3. Action space

Each agent i then chooses its individual action a^i from the same discrete action space $\mathcal{A}$. In the proposed approach, we utilize an absolute action space for refinement/de-refinement, where the action corresponds to an output state of the mesh independently of the input mesh. As such, for the one-level refinement considered in this work, the action space is simply taken as a binary output,

$$\mathcal{A} := \{0, 1\} = \{\texttt{coarse}, \texttt{fine}\}, \tag{19}$$

which dictates the refinement level for the chosen agent. For the `coarse` action, the agent will set that given element's refinement level to the coarse state. Likewise for the `fine` action, the agent will set that given element's refinement level to the fine state. For p-refinement, these actions simply correspond to setting the approximation order for the element, where for some base approximation order p, the `coarse` action sets the element to a $\mathbb{P}_p$ approximation and the `fine` action elevates it to a $\mathbb{P}_{p+1}$ approximation. If the current state of the agent is such that the element approximation is of order $p + 1$ and the `coarse` action is chosen, the polynomial reduction is performed with an L^2 projection. For h-refinement, since the agents are defined over the coarse mesh, the `coarse` action sets the refinement level of that element to the base level while the `fine` action sets the refinement level to one higher than the base level, subdividing the initial coarse element into four congruent child elements. Similarly as with p-refinement, the reduction from the fine child elements to the coarse parent element is performed using an L^2 projection. An example of these refinement actions for both h- and p-refinement is shown in Fig. 1.

3.1.4. Transition function

The initial state of each episode is defined by the initial coarse mesh and the initial conditions that are sampled from a class of parameterized functions, which are defined in Section 5. Given the agents' actions $\mathbf{a}_\tau$, the environment transitions from current state s_τ to next state $s_{\tau+1}$ according to the conditional distribution $P(s_{\tau+1}|s_\tau, \mathbf{a}_\tau)$ via the following steps:

1. Apply refinement to each element whose agent's action is `refine` and current refinement level is coarse and de-refinement to each element whose agent's action is `coarse` and current refinement level is fine.
2. Advance the PDE in simulation time from t_τ to $t_{\tau+1} = t_\tau + T$.
3. If final time is reached ($t = t_f$), reset the environment.

The transition is deterministic for the PDEs in this work, but the overall methodology applies equally to stochastic transitions in the case of stochastic PDEs.

3.1.5. Reward

After an environment transitions with actions $\mathbf{a}_\tau$, each agent i receives an individual reward $r^i_{\tau+1} \in \mathbb{R}$. Similarly to the observation, the reward function must be carefully crafted to allow for generalizability between different problems, meshes, and target error/cost. We utilize a formulation in which the agents are penalized for doing the "wrong" actions, such that the optimal reward obtainable by the policy is zero. In brief, we penalize the agents if they choose the `coarse` action when refinement is more desirable (i.e., when errors are high) and vice versa. The reward function is explicitly defined as

$$r^i_{\tau+1}(s_\tau, \mathbf{a}_\tau) = \begin{cases} -p_{\text{ur}} \left| \log_{10}\left(\hat{e}^i_{\tau+1} / e_{\tau,\max} \right) \right|, & \text{if } \hat{e}^i_{\tau+1} > e_{\tau,\max} \text{ and } a^i_\tau = \texttt{coarse}, \\ -p_{\text{or}} \left| \log_{10}(\hat{e}^i_{\tau+1} / e_{\tau,\min}) \right|, & \text{if } \hat{e}^i_{\tau+1} < e_{\tau,\min} \text{ and } a^i_\tau = \texttt{fine}, \\ 0, & \text{else}, \end{cases} \tag{20}$$

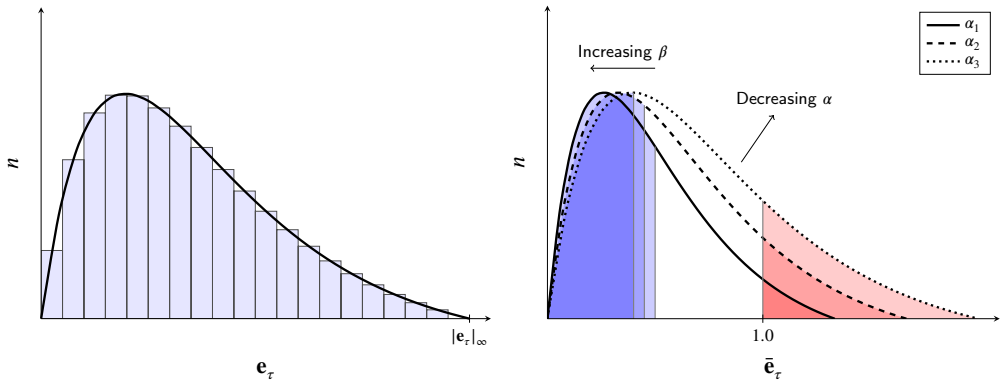
where p_{ur} and p_{or} are strictly positive hyperparameters representing the penalty factors for under-refinement and over-refinement, respectively, and $e_{\tau,\max}$ and $e_{\tau,\min}$ are maximum and minimum error thresholds, respectively, computed at time t_τ. The use of threshold values at t_τ rather than at $t_{\tau+1}$ ensures that the reward depends only on the normalization applied to the observations that the policy used to produce action a^i_τ. Crucially, this reward formulation is time-independent and does not depend on any global spatial information, which means policies trained with this reward can in principle generalize to any simulation duration and mesh size (retaining the same observation normalization) at evaluation time.

For the thresholds in Eq. (20), the maximum error threshold $e_{\tau,\max}$ is computed identically to Eq. (16) while the minimum error threshold is computed as

$$e_{\tau,\min} = e^\beta_{\tau,\max}, \tag{21}$$

where $\beta > 1$ is a hyperparameter representing the error threshold hysteresis. We remark that this formulation is under the assumption that $e_{\tau,\max} < 1$, such that $e_{\tau,\min} < e_{\tau,\max}$, which is typically fair to assume if the problem setup is appropriately normalized. These hyperparameters are chosen purely for the purpose of training the policy and, unlike the user-defined parameter α in the observation, cannot affect the policy at evaluation time. In this work, the parameter β is fixed.

An example of how this error normalization and reward functional operate is shown in Fig. 3. Given a raw error distribution, the error is re-normalized around a unit value per Eq. (16) based on the threshold parameter α, much like in the observation. From the reward functional, the policy is penalized for applying any de-refinement actions on elements with normalized errors above this critical value, shown by the red region in the figure. Furthermore, the policy is also penalized for applying refinement actions to elements with normalized error values below the minimum error threshold computed by Eq. (21), shown by the blue region in the figure. Therefore, the policy is effectively incentivized to maintain the element error distributions within the middle (white) region. The key feature of the observation and reward functional is that the raw error represented by the white region *is controlled by the threshold parameter* α. As a result, by varying the parameter α, the user can control the raw target error at evaluation time even though the reward is not used then since the policy is optimized with respect to a normalized error in the observation which itself can be varied at evaluation time.



(a) Raw error distribution

(b) Normalized error distribution

Figure 3: Schematic for an example raw error distribution (left) transformed to a normalized error distribution (right) as a function of the hyperparameters α and β. Red region denotes normalized error values for which the policy penalizes de-refinement, blue region denotes normalized error values for which the policy penalizes refinement.

Note that the reward is computed with respect to a modified error estimate $\hat{e}^i_{\tau+1}$ instead of the instantaneous error estimate $e^i_{\tau+1}$. We claim that the instantaneous error at time $t_{\tau+1}$ is not necessarily a good indicator of the appropriateness of the action at time t_τ, nor is any combination of the instantaneous error estimates e^i_τ and $e^i_{\tau+1}$. A simple demonstration of the failure mode introduced by using the instantaneous error in the reward can be exemplified through the advection of some compactly-supported characteristic feature of interest, where the feature itself on the order of the element scale. Over the remesh time T, this feature may move through multiple elements (recall that longer remesh times are a desirable feature of the AMR policy), such that it is possible that elements which have encountered the feature in that time interval will still have near-zero error estimates since the feature is no longer currently in that element. As such, a reward based on the instantaneous error may penalize agents which correctly refine elements that experience complex features in the time interval $[t_\tau, t_{\tau+1}]$. To mitigate this failure mode, we utilize a modified error estimate $\hat{e}^i_{\tau+1}$ in the reward, where the modified error estimate is computed as

$$\hat{e}^i_{\tau+1} = \max_{t\in[t_\tau,t_{\tau+1}]} \hat{e}^i(t), \tag{22}$$

i.e., the maximum error estimate for the element across the time range $[t_\tau, t_{\tau+1}]$. This maxima is computed discretely across the discrete time steps of the temporal integration method, which are sufficiently small such that $T >> \Delta t$. Much like with the observation, the modified error field for the h-refinement case is aggregated to the coarsest (agent) level.

An algorithmic overview of the DynAMO approach which combines these individual components is shown in Algorithm 3.1.

Algorithm 3.1 AMR environment step subroutine in DynAMO.

Input: $\mathbf{a}_\tau$: actions of all agents

1: Update mesh with actions $\mathbf{a}_\tau$.
2: Advance PDE in simulation time on new mesh from t_τ to $t_{\tau+1} = t_\tau + T$, computing the maximum element-wise error $\hat{\mathbf{e}}$ over the time interval.
3: Compute new element errors $\mathbf{e}_{\tau+1}$.
4: Compute reward $\mathbf{r}_{\tau+1}$ from $\mathbf{e}_{\tau+1}$ and thresholds $\bar{e}_{\tau,\max}, \bar{e}_{\tau,\min}$ using Eq. (16), Eq. (21), and Eq. (20).
5: Compute new error thresholds $\bar{e}_{\tau+1,\max}$ and $\bar{e}_{\tau+1,\min}$ from $\mathbf{e}_\tau$ using Eq. (21) and Eq. (16).
6: Compute next observations $\mathbf{o}_{\tau+1}$ from $\mathbf{e}_{\tau+1}$ and new error thresholds using Eq. (15) and Eq. (18).

Output: $\mathbf{r}_{\tau+1}, \mathbf{o}_{\tau+1}, \texttt{done}$

3.2. Independent Proximal Policy Optimization algorithm

We employ Independent PPO, a simple yet effective method for fully-decentralized multi-agent reinforcement learning based on Proximal Policy Optimization (PPO) [38]. Each agent i learns a policy $\pi^i_\mu(a^i|o^i)$, parameterized by a neural network with trainable weights μ, to maximize the objective in Eq. (14). A value function $V_\phi(o^i_\tau)$, parameterized by another neural network with trainable weights ϕ, is used to estimate the long-term value of each observation to update the policy at each training step. To improve efficiency, we employ *parameter-sharing* and *experience-sharing* for all agents, which means individual policies $\pi^i_\mu, \forall\, i \in \{1, \dots, N\}$ are represented by one and the same policy network π_μ that is trained using experiences from all agents. This enables faster learning and convergence than completely independent learning using individual experiences [39]. Crucially, note that each agent still acts differently based on its own local information, since each action sample $a^i_\tau \sim \pi_\mu(\cdot|o^i_\tau)$ for agent i is conditioned on its own individual observation.

Although independent learning faces challenges such as the lack of centralized global information and environment non-stationary from each agent's perspective due to the changing policies of other agents, it has significant benefits in the context of dynamic mesh adaptation. As previously mentioned, the spatial domain of dependency for each element is bounded over the time interval T, which means it is sufficient for each element's refinement decision to be based on an adequately large local window instead of the full global state. Furthermore, the use of individual rewards means each agent receives unambiguous feedback on the optimality of its own action, which circumvents the multi-agent credit assignment problem that arises in the case of a team reward [40]. A pseudo-code overview of the DynAMO training routine using Independent PPO is shown in Algorithm 3.2.

Algorithm 3.2 DynAMO training routine.

for each training iteration **do**
  $\mathbf{o}_0 \leftarrow$ env.reset() ▷ Reset environment
  for time step $\tau = 1, \dots, n_t$ **do**
    $\mathbf{a}_\tau \leftarrow \boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{o}_\tau)$ ▷ Get actions for all agents
    $\mathbf{r}_{\tau+1}, \mathbf{o}_{\tau+1}, \texttt{done} \leftarrow$ env.step($\mathbf{a}_\tau$) ▷ Algorithm 3.1
    Store transition $(\mathbf{o}_\tau, \mathbf{a}_\tau, \mathbf{r}_{\tau+1}, \texttt{done})$ into trajectory buffer $\mathcal{B}$
    if `done = True` **then**
      $\mathbf{o}_{\tau+1} \leftarrow$ `env.reset()` ▷ Reset environment
    end if
  end for
  Train on minibatches of transitions from $\mathcal{B}$ by PPO
end for

4. Implementation

With the DynAMO approach set forth, we present here an in-depth overview of the implementation details of the approach, including numerical frameworks and experimental setups. A schematic of the proposed approach is shown in Fig. 4.

4.1. Numerical solver

The numerical solver component of the proposed approach was implemented using the modular open-source C++ FEM library MFEM [41] and its Python interface PyMFEM [42]. The FEM approach of choice was a nodal DG [32] approximation with Gauss–Lobatto solution nodes and the Rusanov approximate Riemann solver [33]. The Rusanov approach was chosen primarily due to its robustness, although better results may be obtained using less dissipative Riemann solvers. Temporal integration was performed using an explicit fourth-order, four-stage Runge–Kutta scheme with a time step based on the Courant–Friedrichs–Lewy (CFL) condition. The CFL number was generally set to CFL $= 0.5$ using the standard RKDG estimate of Cockburn and Shu [43]. The base approximation order was set to $\mathbb{P}_2$ except for problems in the Euler equations which exhibit discontinuities where a $\mathbb{P}_1$ approximation with the Barth–Jespersen limiter [44] was used, with the limiting performed with respect to the density field.

For a given analytic initial condition, the solution was initialized by interpolating the initial conditions to the solution nodes. To compute the estimated error for the observation and reward, we utilized a separate error estimator

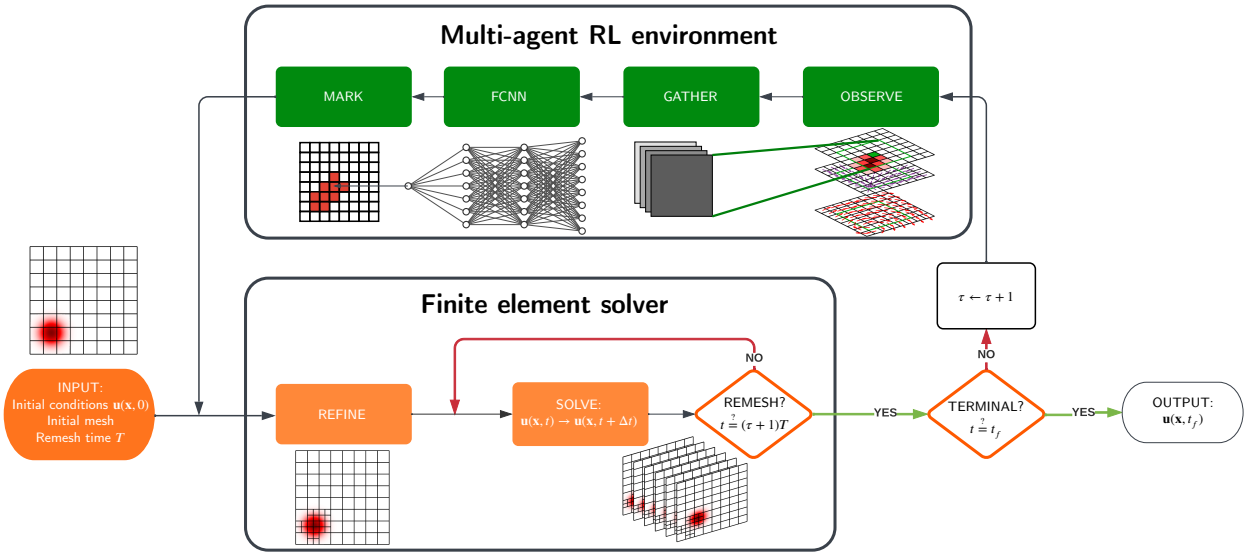


Figure 4: Schematic of the DynAMO approach.

for h- and p-refinement. We remark here that these estimators may not satisfy the efficiency and the reliability condition of traditional *a posteriori* error estimators [45], but they were found to be sufficient to yield highly-performant policies. These estimators primarily target discretization error, although more sophisticated estimators, such as ones based on the solution residual, may achieve more accurate results for hyperbolic conservation laws. However, the particular choice of error estimator is separate from the main aim of this work, and a fair comparison can only be made against baseline approaches using the same error estimators. For p-refinement, the error was estimated through a measure of the difference between the polynomial approximation and itself projected to polynomial space of one degree lower, i.e.,

$$e^i = \|\mathbf{u}_h^i - \Pi_{p-1}\mathbf{u}_h^i\|_{L^2(\Omega_i)},$$

where Π_{p-1} is an L^2 projection operator onto the polynomial space of degree $p-1$. Note that p is the current order of the element, which in the p-refinement case may not be the base order. For h-refinement, the error was estimated by an interface solution jump-based error estimator using a bulk L^2-projection similar to the Zienkiewicz–Zhu error estimator [46, 47]. For an arbitrary element Ω_i, let $\mathcal{N}_e = \{\Omega_+, \Omega_-\}$ be the neighboring elements of an interior edge e and R be the smallest rectangle that contains $\mathcal{N}_e$. We define a polynomial reconstruction operator $\mathcal{R}_e : \mathbb{P}_{p_+}(\Omega_+) \times \mathbb{P}_{p_-}(\Omega_-) \rightarrow \mathbb{P}_r(R)$, where $r = \max\{p_+, p_-\}$ and

$$\mathcal{R}_e(\mathbf{u}_h) := \underset{\mathbf{v} \in \mathbb{P}_r(R)}{\operatorname{argmin}} \|\mathbf{v} - \mathbf{u}_h\|_{L^2(\mathcal{N}_e)}.$$

Then, the error estimate e^i is defined as

$$e^i = \left(\frac{1}{N_{e,i}} \sum_{e \subset \partial\Omega_i \setminus \partial\Omega} \|\mathbf{u}_h - \mathcal{R}_e(\mathbf{u}_h)\|^2_{L^2(\Omega_i)} \right)^{1/2},$$

where $N_{e,i}$ is the number of interior edges contained in the boundary of Ω_i.

While these error estimates were used for the observation and reward, the efficacy of the policies (and the comparison to baseline approaches) was evaluated with respect to the "true" error. For the advection equation, this was simply the analytic solution, whereas for the Euler equations, the true error was approximated using a reference simulation with resolution equivalent to one refinement level higher than the maximum refinement level allowable (e.g., for p-refinement, the reference simulation was performed at $\mathbb{P}_{p+2}$).

4.2. Reinforcement learning framework

The reinforcement learning framework was implemented using RLLib [48]. The PPO approach was implemented using the default hyperparameter values in RLLib except the following: a learning rate of 10^{-4}, a rollout fragment

length of 20, and a batch size of 1000 for each epoch of stochastic gradient descent with minibatches of size 50. The network architecture was a fully-connected neural network with two hidden layers of 256 neurons using a hyperbolic tangent activation function. Purely fully-connected layers were used instead of convolutional neural network (CNN) layers (or a mixture thereof) to maintain the directional structure in the input and to allow for straightforward extension to unstructured observations through approaches such as graph neural networks [29]. For each policy, training was performed across up to 16 CPUs over the span of 24 hours, which covers approximately $10^4 - 10^5$ training episodes. The most performant policy (i.e., the policy with the highest batch-averaged mean reward) over the training period was used for evaluation.

The DynAMO-specific hyperparameters were fixed across the various policies trained. The error threshold value at training time was set to $\alpha_{\text{train}} = 0.1$ and the error threshold hysteresis was set to $\beta = 1.2$. Furthermore, the under-refinement and over-refinement penalty factors were set as $p_{ur} = 10$ and $p_{or} = 5$, respectively. Unless otherwise stated, the error threshold value α at evaluation time was set to its training value. For the Euler equations, the solution component for the observed quantities in Eq. (15) and Eq. (18) was the total energy, which was also the component used for evaluating the error against the reference solution. The positivity of total energy in the Euler equations ensures that the quantity in Eq. (18) remains well-defined. For the observation, the element neighborhood size was set to $n_x = n_y = 8$, such that the size of the observation window was $k_x = k_y = 17$.

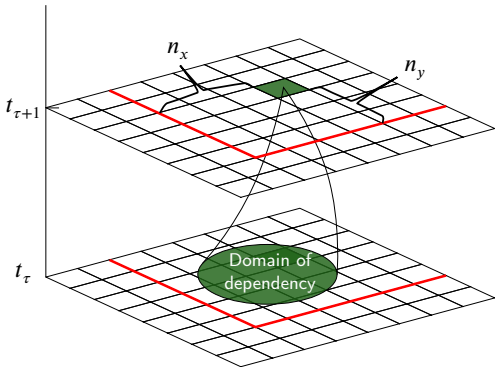


Figure 5: The domain of dependency and observation window for an element over a remeshing time interval $T = t_{\tau+1} - t_\tau$.

To ensure that domain of influence of a given element does not exceed the observable region of the policy, as exemplified in Fig. 5, this observation window size effectively constrained the maximum remeshing time interval T via a CFL-like condition as

$$T \leq \min\left[\frac{n_x \Delta x}{\lambda_{\max}}, \frac{n_y \Delta y}{\lambda_{\max}}\right],$$

where Δx and Δy are the corresponding characteristic mesh spacings along the x and y directions, respectively. The specific remesh time for the experiments was chosen close to this limit during training – note that this inequality does not necessarily always have to be satisfied, but the performance of the approach would likely suffer in scenarios where it is not. As such, since the policy relies on a fixed observation size (due to the architecture of the network), the remesh time can be modified accordingly if needed to ensure that this condition is met at evaluation time. For the purposes of training, we utilize a fixed episode length of 4 RL steps, corresponding to 4 remesh time intervals, such that the final simulation time was set as $t_f = 4T$. However, this episode length and simulation time was freely varied during evaluation. We remark here that the proposed approach is primarily tailored towards solvers utilizing explicit time stepping, where the observation window can encompass the domain of influence over many solver time steps. Extensions to implicit time schemes, which allow for significantly larger time steps, can be readily performed with the caveat that the observation window must be sufficiently large to account for time steps which themselves may be larger than a standard remesh time interval used for explicit time schemes. Although the assumption of a local domain of influence in hyperbolic conservation laws does not necessarily hold for an implicit time scheme, it may still offer a suitable approximation for the local solution behavior.

As the purpose of the proposed approach is to increase the efficiency of AMR, it is important to understand the computational overhead of the RL framework at evaluation time to ensure that its cost do not outweigh its benefits. We remark that this work primarily pertains to evaluating the capabilities of RL for AMR, and due to the use of the

RLLib framework on top of the FEM solver, the current implementation of the proposed approach is not necessarily tailored towards peak computational efficiency. However, even in its current form, the computational cost overhead of the RL framework was a fraction of the cost of the FEM solve (e.g., for the Euler equations on a $N = 64^2$ mesh, the cost of inference was less than 20% of the cost of the FEM solve). Furthermore, due to the use of an independent multi-agent approach which allows for a highly-parallelizable implementation, the cost of the proposed method can still be drastically decreased with parallelization, a more computationally efficient implementation of the network, and the use of GPU-accelerated computing.

5. Results

The proposed DynAMO approach was evaluated on a series of numerical experiments showcasing applications to linear and nonlinear hyperbolic conservation laws with *h*- and *p*-refinement. While the linear advection equation may not be a rigorous test for the approach, it possesses some useful features such as anisotropic solution and error evolution and an analytic solution that allows for the calculation of an exact error. As such, it serves as a suitable preliminary example to present certain features of DynAMO and an initial efficacy comparison to baseline methods. Afterward, the approach is more rigorously evaluated on the compressible Euler equations for both smooth and discontinuous problems, showcasing the viability of DynAMO for complex nonlinear systems. For each set of numerical experiments, a separate policy is trained, such that the results of four policies are shown (one each for advection with *p*-refinement, advection with *h*-refinement, Euler with *p*-refinement, and Euler with *h*-refinement). The goal is to train the policy on smaller representative problems and show that this approach can effectively generalize to larger and unseen problems.

The results are presented for a series of problems, including *in-distribution* examples, where the problem conditions are similar (but not necessarily exactly identical) to episodes encountered during training, and *out-of-distribution* examples, where the approach is applied to unseen problems, including different initial conditions (which are not representative of the initial conditions used during training), longer simulation time/episode lengths, longer remesh time intervals, and different initial mesh resolutions. To compare the proposed approach to baseline methods, we consider a normalized metric for the computational cost $\bar{c}$ and error $\bar{e}$, defined as

$$\bar{c} = \frac{c - c_{\text{coarse}}}{c_{\text{fine}} - c_{\text{coarse}}} \quad \text{and} \quad \bar{e} = \frac{e - e_{\text{fine}}}{e_{\text{coarse}} - e_{\text{fine}}}, \tag{23}$$

where the subscripts *coarse* and *fine* refer to the results of simulations performed on completely unrefined and completely refined meshes, respectively. The computational cost metric c is computed as the cumulative total degrees of freedom in the mesh summed between remesh steps. From these quantities, we define an *efficiency* metric $0 \leq \epsilon \leq 1$ as

$$\epsilon = 1 - \sqrt{\bar{c}^2 + \bar{e}^2}, \tag{24}$$

where a higher efficiency represents an AMR policy that can achieve a higher accuracy for a given computational cost or, conversely, a given accuracy for a lower computational cost. This efficiency metric is the primary point of comparison between DynAMO and baseline methods. For this comparison, we utilize the absolute threshold-based method described in Section 2.3 as the baseline, which for the Euler equations was applied to the total energy error estimate for consistency with the DynAMO approach.

5.1. Advection with *p*-refinement

The DynAMO approach was first applied to the task of *p*-refinement for the advection equation, which allows for an evaluation of the approach for AMR on simple linear PDEs with a constant number of mesh elements. We consider the advection of smooth features for which *p*-refinement is particularly well-suited for. For this task, the initial conditions for the training episodes were sampled from a set of "ring"-like shapes, defined as

$$u(x, y, 0) = 1 + \exp\left(-w\left(\sqrt{(x - x_0)^2 + (y - y_0)^2} - r_0\right)^2\right), \tag{25}$$

where the parameters x_0, y_0, r_0, and w were uniformly distributed across the ranges $x_0 \in [0.3, 0.7]$, $y_0 \in [0.3, 0.7]$, $r_0 \in [0.1, 0.3]$, and $w \in [50, 150]$. This initial condition was chosen to give an extra degree of complexity to the error distribution, adding a region of relatively low error surrounded by a region of high error. Furthermore, the domain was

set to $\Omega = [0, 1]^2$, the remesh time was set as $T = 0.3$ with a total number of 4 RL steps, the initial mesh resolution was set as $N = 24^2$, and the propagation velocity magnitude was sampled across the range $\|\mathbf{c}\|_2 \in [0.7, 1]$, with the direction of propagation uniformly distributed across the azimuthal direction. The efficacy of the agent over the training period is shown through the training reward curve in Fig. 6.

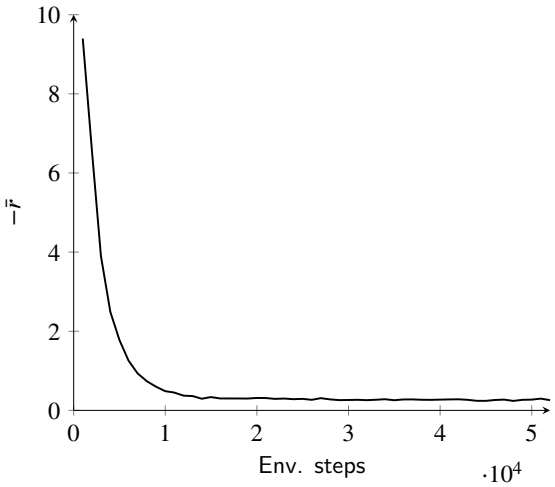
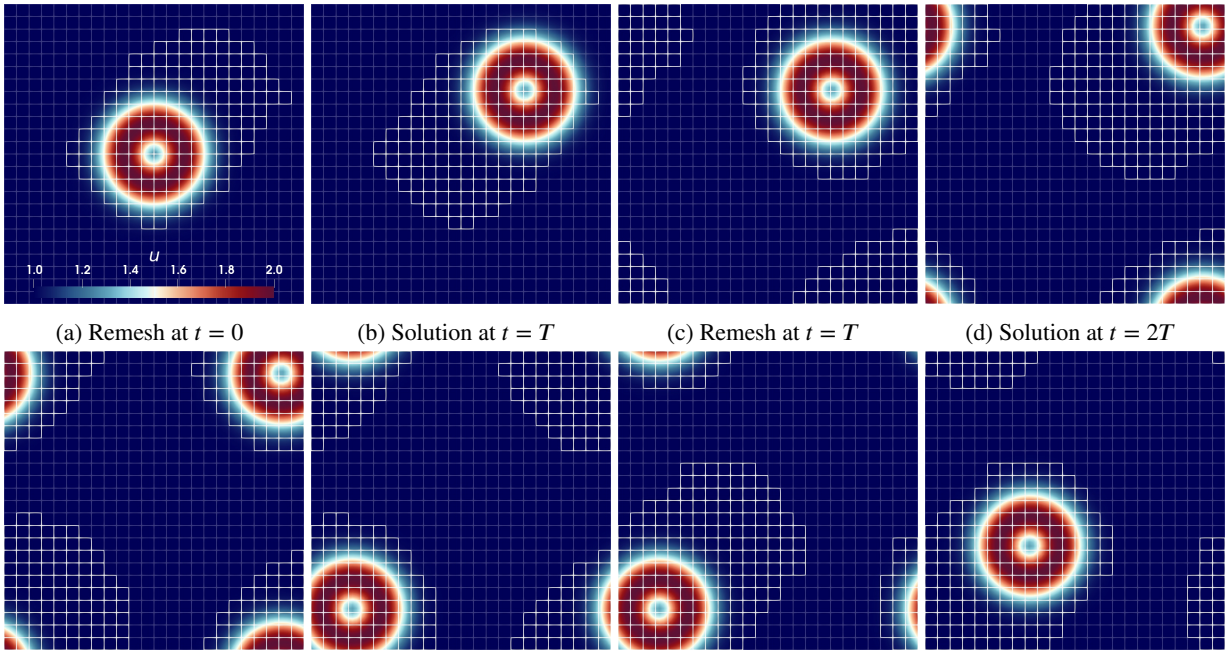


Figure 6: Batch-averaged (negative) reward with respect to number of environment steps during the training process for p-refinement on the advection equations.



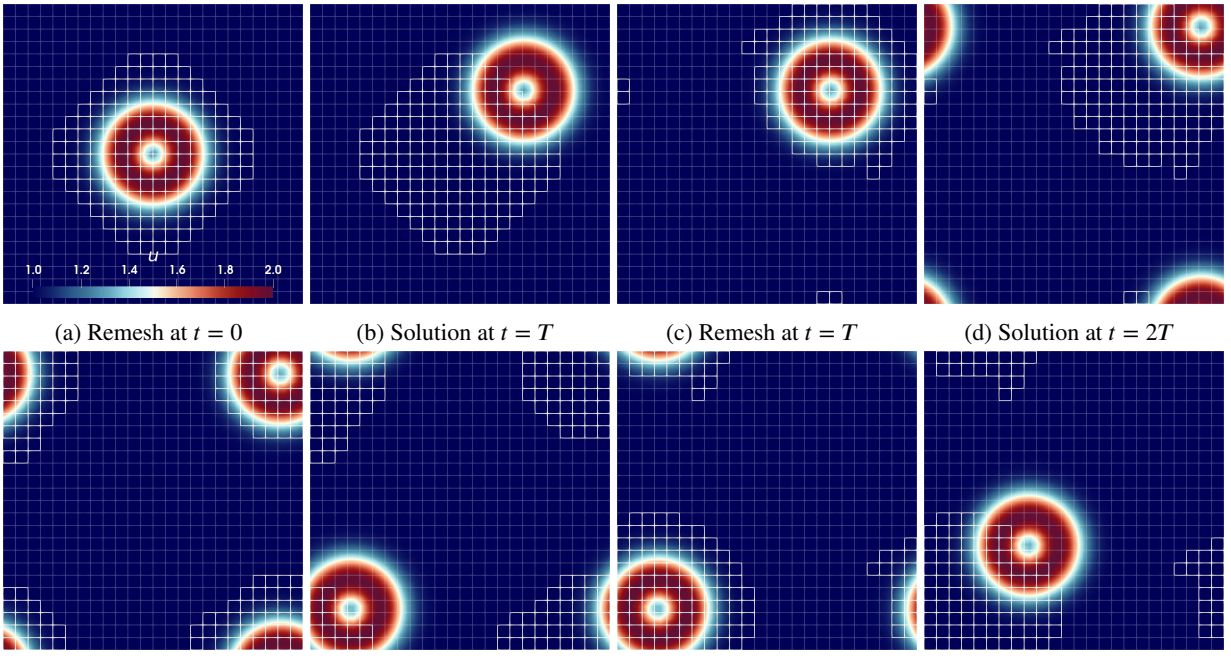
(a) Remesh at $t = 0$ (b) Solution at $t = T$ (c) Remesh at $t = T$ (d) Solution at $t = 2T$

(e) Remesh at $t = 2T$ (f) Solution at $t = 3T$ (g) Remesh at $t = 3T$ (h) Solution at $t = 4T$

Figure 7: Solution contours overlaid with p-adapted mesh at varying remesh intervals using DynAMO for the *in-distribution* advecting ring case. Highlighted elements represent p-refinement.

5.1.1. Sample problem

To showcase the features of the DynAMO approach and present an initial comparison to baseline methods, we first present results for a single *in-distribution* test case, where the parameters were set as $x_0 = y_0 = 0.5$, $r_0 = 0.1$, and $\|\mathbf{c}\|_2 = 1$ with a propagation angle of $\pi/4$. The evolution of the solution and p-adapted mesh across 4 remesh times as computed by the DynAMO approach and the threshold policy is shown in Fig. 7 and Fig. 8, respectively. The error threshold for the DynAMO policy was set identical to the training value, i.e., $\alpha = \alpha_{\text{train}}$, while the error threshold for the threshold policy was set to $\theta = 10^{-3}$ as this will later be shown to be the value at which this policy achieves the highest efficiency. It can immediately be seen that at $t = 0$, the threshold policy based on the instantaneous error estimate results in an isotropic refinement pattern centered about the ring, which cannot account for the propagation of the ring until the next remesh time. Consequently, the majority of the ring exits the refinement region by $t = T$, causing an increase in error. In contrast, the DynAMO policy shows a highly anisotropic refinement pattern, with the refinement region stretched in the direction of the propagation velocity. As such, it is able to preemptively refine the region through which the ring will propagate between the remesh intervals. In fact, the refinement decisions by the DynAMO policy qualitatively seem to cover *exactly* the region necessary to account for the propagation distance between the remesh intervals without excessively refining ahead of the ring. This behavior persisted throughout the entire simulation time, with the DynAMO policy showing excellent preemptive refinement while the threshold policy was unable to adequately account for the spatio-temporal evolution of the solution and subsequent error. As a result, the DynAMO policy was able to use much longer remesh intervals without sacrificing accuracy in comparison to standard AMR approaches based on instantaneous error estimates.



(a) Remesh at $t = 0$ (b) Solution at $t = T$ (c) Remesh at $t = T$ (d) Solution at $t = 2T$

(e) Remesh at $t = 2T$ (f) Solution at $t = 3T$ (g) Remesh at $t = 3T$ (h) Solution at $t = 4T$

Figure 8: Solution contours overlaid with p-adapted mesh at varying remesh intervals using the threshold policy ($\theta = 10^{-3}$) for the *in-distribution* advecting ring case. Highlighted elements represent p-refinement.

A more quantitative comparison between the two approaches was performed by analyzing the error, cost, and efficiency for the given test case. A Pareto plot showing the normalized cost vs. error distributions at varying threshold values for the DynAMO and threshold policies is shown in Fig. 9. The threshold policy shows a clear Pareto front, with the optimal efficiency achieved at $\theta = 10^{-3}$. As expected, the results of the DynAMO policy sit well within the Pareto front of the threshold policy, showing higher efficiency through a much lower error as a result of the policy's ability to achieve preemptive refinement. This much lower error, represented by the green marker in the figure, was achieved using an identical error threshold at evaluation time as with training time. Furthermore, a key feature of the proposed DynAMO approach is that this error threshold α can be varied at evaluation time to achieve different error

and cost targets. The efficacy of this feature was evaluated by changing the error threshold at evaluation time, with the values of α varied between 0.1 and 0.8. It can be seen that this results in a similar Pareto front to the threshold policy, where the user can dictate the trade-off between error and cost. However, the key difference is that the DynAMO policy *unlocks regions of efficiency that are unattainable by conventional AMR approaches*. The result is that the DynAMO policy can achieve significantly lower error for a similar computational cost in comparison to these standard AMR approaches. We remark here that the value of the error threshold α used at training time may not necessarily be the value that obtains the highest efficiency at evaluation time, which is exemplified in this case where the optimal efficiency was achieved at $\alpha = 0.5$ whereas the training value was set as $\alpha_{\text{train}} = 0.1$. This is simply a result of the trade-off between error and cost, where policies that prioritize one over the other will achieve lower efficiency than ones that balance them equivalently and is the expected behavior for a policy that allows users to control error or cost targets. The results from the policy using the training value α_{train} will just be one point along that Pareto front.

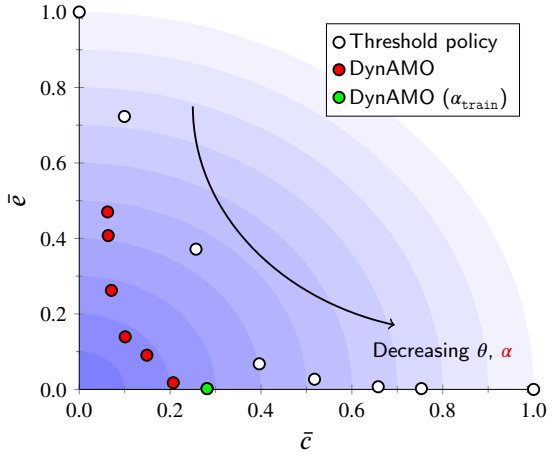


Figure 9: Pareto plot of normalized cost vs. error for p-refinement on the advection equations with DynAMO and the threshold policy for the *in-distribution* advecting ring case. Contours of efficiency shown on the background. Red markers represent a sweep of $\alpha \in [0.1, 0.8]$ at evaluation time, green marker represents α set to the training value of $\alpha_{\text{train}} = 0.1$. Peak efficiency achieved at $\alpha = 0.5$.

5.1.2. In-distribution experiments

While this single example showcases the features and benefits of the DynAMO approach, it is of more interest to observe how its efficacy holds up over a large sample of operating conditions. To this end, a comparison was performed between the DynAMO policy and the threshold policy over 100 *in-distribution* samples of the advecting rings problem, where the operating conditions were uniformly sampled over the training range. The efficiency, normalized error, and normalized cost for the threshold policy at varying values of the absolute threshold θ is compared to the DynAMO policy (evaluated at α_{train}) in Table 1. At its peak, the threshold policy only achieves a mean efficiency of roughly 0.4, with normalized mean error and cost values of 0.204 and 0.465, respectively. In comparison, the DynAMO policy at $\alpha = \alpha_{\text{train}}$, which is not necessarily even the value at which the policy achieves peak efficiency, showed a noticeable increase in mean efficiency from 0.409 to 0.539 (31.8%). The extent of the computational benefits provided by the DynAMO approach in comparison to the most efficient threshold policy is best seen in the relative error and cost reduction. For effectively the same mean computational cost, the use of the DynAMO policy resulted in 93.1% less mean error than the threshold policy, such that the DynAMO policy could achieve essentially the accuracy of a fully-refined mesh with less than half of the cost. Achieving this level of accuracy with the threshold policy would require over double the computational cost.

5.1.3. Generalization experiments

The generalization capabilities of the proposed DynAMO approach were then evaluated by applying the policy to *out-of-distribution* problems, ones with characteristics that were not encountered during training. Three experiments were conducted, testing the ability of the approach to generalize to finer meshes, different shapes for the initial conditions, and longer simulation times. The mean efficiency for the threshold policy at varying values of the absolute

Method	Efficiency	Normalized error	Normalized cost
Threshold ($\theta = 10^{-2}$)	0.076 (0.946)	0.850 (1.008)	0.087 (0.060)
Threshold ($\theta = 10^{-3}$)	**0.409 (0.313)**	**0.204 (0.416)**	**0.465 (0.130)**
Threshold ($\theta = 10^{-4}$)	0.233 (0.302)	0.099 (0.370)	0.715 (0.151)
Threshold ($\theta = 10^{-5}$)	0.141 (0.185)	0.057 (0.234)	0.838 (0.114)
Threshold ($\theta = 10^{-6}$)	0.083 (0.115)	0.039 (0.168)	0.904 (0.081)
Threshold ($\theta = 10^{-7}$)	0.053 (0.063)	0.019 (0.090)	0.943 (0.058)
DynAMO	**0.539 (0.124)**	**0.014 (0.052)**	**0.457 (0.126)**
DynAMO/Optimal θ	+31.8%	-93.1%	-1.7%

Table 1: Comparison of the mean efficiency, normalized error, and normalized cost for p-refinement on the advection equation with DynAMO and the threshold policy for the advecting rings over 100 *in-distribution* runs using uniform random initial conditions. Standard deviation is shown in parentheses.

threshold θ and the DynAMO policy evaluated at α_{train} is shown in Table 2 for the three experiments over 100 samples. For the generalization to finer meshes, the initial mesh resolution was increased from $N = 24^2$ to $N = 96^2$, such that the base finite element space consisted of nearly 10^5 degrees of freedom. It can be seen that while the increase in mesh resolution increased the peak efficiency of the threshold policy, the efficiency of the DynAMO policy also increased appropriately, such that the relative efficiency benefits of DynAMO stayed approximately the same. This positive result can be attributed to the highly generalizable nature of the novel observation proposed in this work, which is insensitive to mesh resolution and problem scale.

Method	In-distribution	Finer mesh	Different shapes	Longer sim. time
Threshold ($\theta = 10^{-2}$)	0.076 (0.946)	0.113 (0.186)	0.000 (0.002)	0.121 (0.185)
Threshold ($\theta = 10^{-3}$)	**0.409 (0.313)**	0.337 (0.255)	0.121 (0.120)	**0.429 (0.396)**
Threshold ($\theta = 10^{-4}$)	0.233 (0.302)	**0.567 (0.244)**	**0.483 (0.260)**	0.281 (0.171)
Threshold ($\theta = 10^{-5}$)	0.141 (0.185)	0.487 (0.192)	0.435 (0.282)	0.160 (0.120)
Threshold ($\theta = 10^{-6}$)	0.083 (0.115)	0.291 (0.190)	0.307 (0.149)	0.090 (0.080)
Threshold ($\theta = 10^{-7}$)	0.053 (0.063)	0.184 (0.172)	0.193 (0.112)	0.050 (0.054)
DynAMO	**0.539 (0.124)**	**0.774 (0.135)**	**0.568 (0.107)**	**0.840 (0.116)**
DynAMO/Optimal θ	+31.8%	+36.5%	+17.6%	+95.8%

Table 2: Comparison of the mean efficiency for p-refinement on the advection equation with DynAMO and the threshold policy for the advecting rings over 100 *out-of-distribution* runs using uniform random initial conditions with finer mesh resolution, different advecting shapes, and longer simulation time. Standard deviation shown in parentheses. In-distribution results from Table 1 shown for comparison.

To show that the trained policy is not overfitting to the solution features observed during training, a second generalization experiment was conducted by changing the class of initial conditions from "ring"-type shapes to "bump"-type shapes of varying size and width, characterized by the Gaussian function

$$u(x, y, 0) = h \exp\left(-w\left((x - x_0)^2 + (y - y_0)^2\right)\right), \tag{26}$$

where the parameters x_0, y_0, h, and w were uniformly distributed across the ranges $x_0 \in [0.3, 0.7]$, $y_0 \in [0.3, 0.7]$, $h \in [0.2, 1]$, and $w \in [25, 100]$. The efficiency comparison in Table 2 shows that the efficacy of the DynAMO policy does not deteriorate with changing initial conditions, but in fact marginally increases. However, due to the simple nature of the bump shape, the efficiency of the threshold policy also increased, such that the efficiency benefits of DynAMO were not as pronounced.

Finally, the ability to generalize to longer simulation times was tested by increasing the simulation length (and total number of remeshing cycles) by a factor of eight. We note here that for simple linear problems such as advection, these experiments may not necessarily show generalization to out-of-distribution problems as the observed features do not vary much over time, but they act as a verification that the fixed episode length used for training does not affect evaluation for longer episode lengths and showcase the type of problems where anticipatory mesh refinement

is expected to yield the most benefits. For this experiment, it is expected that an anticipatory refinement approach would yield far superior performance as it would be able to mitigate the introduction of discretization error much more effectively, error which would continuously accumulate over the course of the simulation. As can be seen from Table 2, this was indeed the case, with the DynAMO policy achieving nearly double the efficiency of the standard threshold policy.

5.2. Advection with h-refinement

With the successful application of the DynAMO approach to the advection equation with p-refinement, the more complex case of h-refinement, where the number of elements varies throughout the simulation, was attempted. For consistency with the results of the p-refinement experiments, we utilize effectively the same experimental setup, although the results are expected to vary due to differences in the choice of error estimators and the efficacy of h-refinement versus p-refinement for smooth solutions. The purpose of these experiments is primarily to validate the proposed methodology for h-refinement where mesh inhomogeneity is addressed from the coarse agent level. The training reward curve is presented in Fig. 10, showing nearly identical behavior to the p-refinement case.

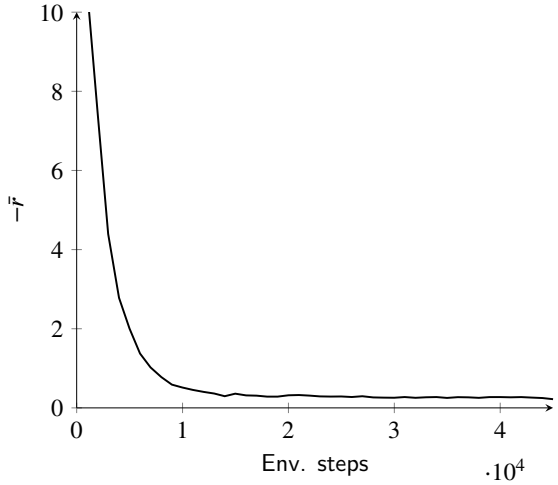
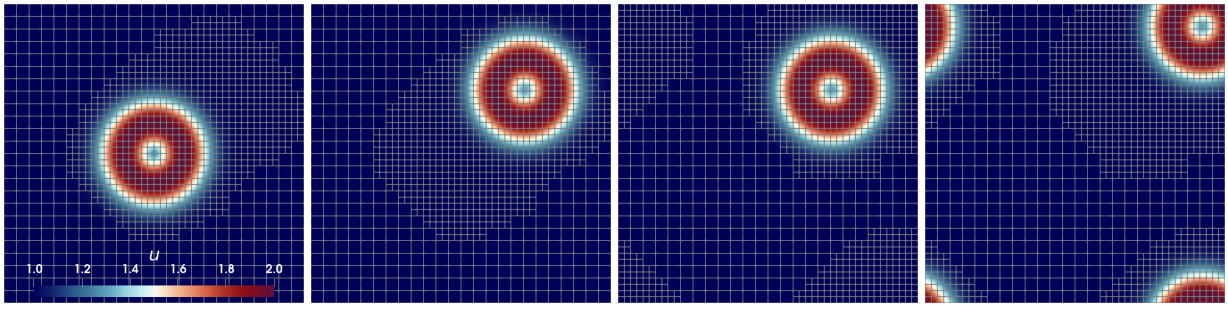


Figure 10: Batch-averaged (negative) reward with respect to number of environment steps during the training process for h-refinement on the advection equations.

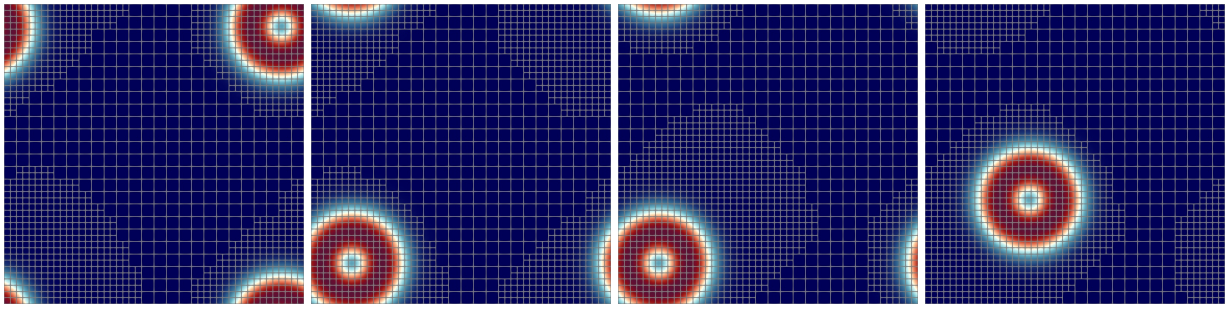
5.2.1. Sample problem

The training procedure was repeated identically to the p-refinement experiments and the resulting approach was compared to the threshold policy for the same advecting ring case, with the initial conditions given by Eq. (25). The h-adapted mesh and solution evolution across 4 remesh times as computed by the DynAMO policy ($\alpha = \alpha_{\text{train}}$) and the threshold policy ($\theta = 10^{-3}$) is shown in Fig. 11 and Fig. 12, respectively. Much like with the p-refinement case, it can be seen that the threshold policy based on the instantaneous error estimator results in an isotropic refinement pattern centered on the ring, which does not adequately account for the movement of the ring between remesh intervals. This resulting introduction of discretization error cause irregular refinement patterns later in the simulation. In contrast, the h-refinement DynAMO policy showed similar behavior to the p-refinement DynAMO policy, with an anistropic refinement pattern stretched in the direction of propagation. This predictive refinement persisted across the entire simulation time, such that the feature of interest remained within the refined region throughout. Furthermore, it was observed that the DynAMO policy tended to produce contiguous regions of refined elements, which would be the proper refinement pattern for linearly advecting compactly-supported features.

A quantitative comparison of the error, cost, and efficiency between the DynAMO policy and the threshold policy for the advecting ring case is shown in the Pareto plot in Fig. 13. By varying the error threshold parameter, both the threshold policy and the DynAMO policy showed the typical Pareto front. However, much like with the p-refinement case, the DynAMO policy achieved noticeably higher efficiency than the threshold policy. For a given computational cost, the DynAMO policy was able to achieve a significantly lower error. Furthermore, by varying the error threshold

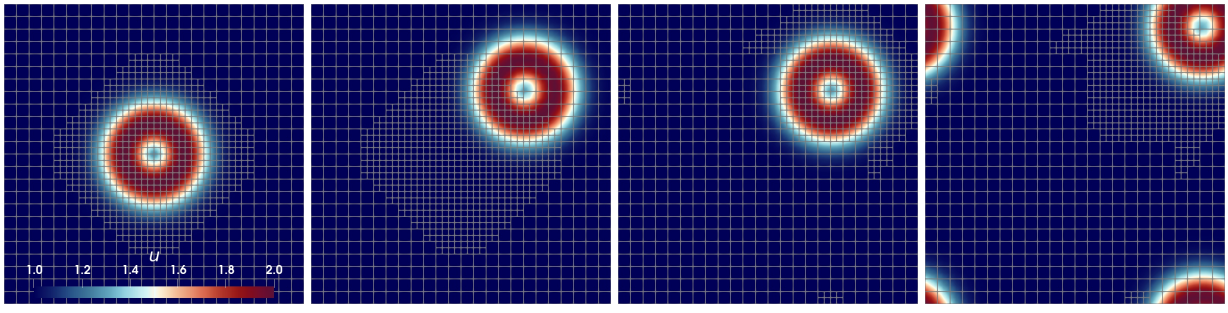


(a) Remesh at $t = 0$ (b) Solution at $t = T$ (c) Remesh at $t = T$ (d) Solution at $t = 2T$

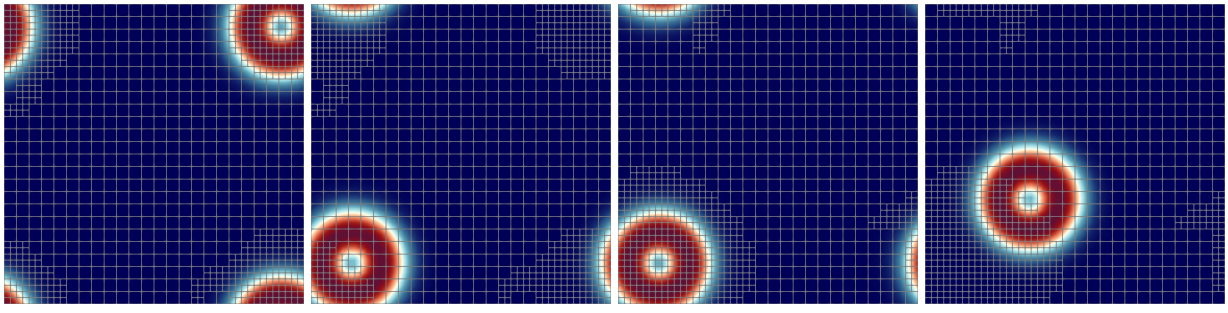


(e) Remesh at $t = 2T$ (f) Solution at $t = 3T$ (g) Remesh at $t = 3T$ (h) Solution at $t = 4T$

Figure 11: Solution contours overlaid with h-adapted mesh at varying remesh intervals using DynAMO for the *in-distribution* advecting ring case.



(a) Remesh at $t = 0$ (b) Solution at $t = T$ (c) Remesh at $t = T$ (d) Solution at $t = 2T$



(e) Remesh at $t = 2T$ (f) Solution at $t = 3T$ (g) Remesh at $t = 3T$ (h) Solution at $t = 4T$

Figure 12: Solution contours overlaid with h-adapted mesh at varying remesh intervals using the threshold policy ($\theta = 10^{-3}$) for the *in-distribution* advecting ring case.

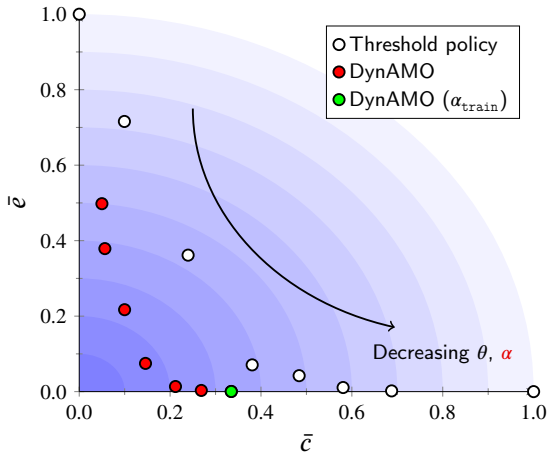


Figure 13: Pareto plot of normalized cost vs. error for p-refinement on the advection equations with DynAMO and the threshold policy for the *in-distribution* advecting ring case. Contours of efficiency shown on background. Red markers represent a sweep of $\alpha \in [0.1, 2.0]$ at evaluation time, green marker represents α set to the training value of $\alpha_{\texttt{train}} = 0.1$. Peak efficiency achieved at $\alpha = 0.8$.

parameter α at evaluation time, various points on that Pareto front could be achieved, such that error and cost targets could be specified by the user. We remark here that again the DynAMO policy did not necessarily achieve peak efficiency for the error threshold parameter used at training time, which resulted in refinement decisions that were biased more towards error reduction than cost reduction. Nevertheless, the efficiency of the DynAMO policy evaluated over a wide range of values of the error threshold parameter was noticeably higher than the threshold policy at its peak efficiency. These results, which show a marked consistency with the results of the p-refinement policy, indicate that the proposed multi-agent reinforcement learning approach for h-refinement using coarse level agents is highly effective and adequately sidesteps the agent creation/deletion problem without sacrificing performance.

5.2.2. In-distribution experiments

To verify that these observations extend over many sampled initial conditions, the mean efficiency, normalized error, and normalized cost over 100 randomly sampled *in-distribution* conditions was calculated, the results of which are presented in Table 3 for the threshold policy at varying values of the error threshold θ and the DynAMO policy at the training error threshold value $\alpha = \alpha_{\text{train}}$. It can be seen that the DynAMO approach still retains efficiency benefits over the baseline threshold policy, with nearly a 20% increase in the mean efficiency. Furthermore, the error-to-cost ratio of the DynAMO approach was significantly lower, showing a nearly 99% decrease in the error for approximately 30% higher computational cost. This accuracy was on par with the accuracy of a fully-refined mesh, which would require nearly double the computational cost. We remark here that the efficiency increase of DynAMO in the h-refinement case was lower than the p-refinement case, but this was primarily due to the choice of error threshold α. At the training value $\alpha = \alpha_{\text{train}}$, the results of the DynAMO policy for h-refinement were farther along the Pareto front, which is evidenced by Fig. 13 and the relative error in Table 3. Much like with the p-refinement policy, even higher efficiencies could be obtained by modifying the value of α as the taken training value biases towards higher error reduction instead of peak efficiency.

5.2.3. Generalization experiments

Finally, the generalization experiments were repeated for *out-of-distribution* problems, such as finer meshes, different classes of initial conditions, and longer simulation times. A comparison of the threshold policy at varying values of the error threshold θ and the DynAMO policy at the training error threshold value $\alpha = \alpha_{\text{train}}$ performed over 100 samples is shown in Table 4 for the three experiments. When the initial mesh resolution was increased from $N = 24^2$ to $N = 96^2$, the DynAMO approach showed a notably higher efficiency (67.1%) than the threshold policy. This efficiency gain was significantly higher than with the similar experiment on p-refinement, but this was primarily as a result of the threshold policy showing poorer performance on the fine mesh with h-refinement than with p-refinement. For the generalization to different initial condition shapes, the DynAMO approach retained the efficiency benefits,

Method	Efficiency	Normalized error	Normalized cost
Threshold ($\theta = 10^{-2}$)	0.174 (0.135)	0.815 (0.157)	0.084 (0.056)
Threshold ($\theta = 10^{-3}$)	**0.407 (0.193)**	**0.417 (0.212)**	**0.401 (0.098)**
Threshold ($\theta = 10^{-4}$)	0.299 (0.159)	0.121 (0.166)	0.675 (0.140)
Threshold ($\theta = 10^{-5}$)	0.191 (0.147)	0.056 (0.144)	0.797 (0.128)
Threshold ($\theta = 10^{-6}$)	0.121 (0.116)	0.029 (0.126)	0.872 (0.098)
Threshold ($\theta = 10^{-7}$)	0.079 (0.072)	0.012 (0.048)	0.920 (0.071)
DynAMO	**0.480 (0.106)**	**0.006 (0.044)**	**0.519 (0.108)**
DynAMO/Optimal θ	+17.9%	-98.6%	+29.4%

Table 3: Comparison of the mean efficiency, normalized error, and normalized cost for h-refinement on the advection equation with DynAMO and the threshold policy for the advecting rings over 100 *in-distribution* runs using uniform random initial conditions. Standard deviation shown in parentheses.

Method	In-distribution	Finer mesh	Different shapes	Longer sim. time
Threshold ($\theta = 10^{-2}$)	0.174 (0.135)	0.092 (0.153)	0.000 (0.002)	0.098 (0.137)
Threshold ($\theta = 10^{-3}$)	**0.407 (0.193)**	0.283 (0.199)	0.040 (0.145)	**0.399 (0.197)**
Threshold ($\theta = 10^{-4}$)	0.299 (0.159)	**0.483 (0.210)**	**0.484 (0.163)**	0.328 (0.142)
Threshold ($\theta = 10^{-5}$)	0.191 (0.147)	0.475 (0.154)	0.479 (0.140)	0.206 (0.132)
Threshold ($\theta = 10^{-6}$)	0.121 (0.116)	0.291 (0.160)	0.351 (0.143)	0.128 (0.099)
Threshold ($\theta = 10^{-7}$)	0.079 (0.072)	0.196 (0.145)	0.241 (0.128)	0.076 (0.068)
DynAMO	**0.480 (0.106)**	**0.807 (0.129)**	**0.540 (0.110)**	**0.719 (0.081)**
DynAMO/Optimal θ	+17.9%	+67.1%	+11.5%	+80.2%

Table 4: Comparison of the mean efficiency for h-refinement on the advection equation with DynAMO and the threshold policy for the advecting rings over 100 *out-of-distribution* runs using uniform random initial conditions with finer mesh resolution, different advecting shapes, and longer simulation time. Standard deviation shown in parentheses. In-distribution results from Table 3 shown for comparison.

resulting in a similar increase in efficiency between DynAMO and the optimal threshold policy. Extensions to longer simulation times showed the most drastic results, with approximately 80% higher efficiency with DynAMO compared to the optimal threshold policy due to superior error mitigation from preemptive refinement. These results are notably similar to the results of the p-refinement experiments, indicating that the proposed method for coarse-level observations in h-refinement does not degrade the accuracy and generalizability of the DynAMO approach.

5.3. Euler equations with p-refinement

With the preliminary experiments for the linear advection equation showing promising results for both p- and h-refinement, the proposed DynAMO approach was extended to the much more complex task of AMR for the compressible Euler equations. Due to the inefficacy of p-refinement for discontinuous features, the p-refinement policy and experiments for the Euler equations focus strictly on smooth solutions while discontinuous solutions are later considered in the h-refinement experiments. For this class of solutions, common representative cases are convection-dominated and acoustic-type problems, where nonlinear flow physics are introduced through bulk propagation and pressure-wave interactions. To this end, the DynAMO policy for p-refinement on the Euler equations was trained on a class of solutions containing multiple convecting pressure pulses/waves, where the initial conditions were

$$\rho(x, y, 0) = 1, \tag{27a}$$

$$u(x, y, 0) = u_0, \tag{27b}$$

$$v(x, y, 0) = v_0, \tag{27c}$$

$$P(x, y, 0) = \sum_{i=1}^{n_p} h_i \exp\left(-w_i \left((x - x_{i,0})^2 + (y - y_{i,0})^2\right)\right), \tag{27d}$$

where the parameters u_0, v_0, h, w, x_0, and y_0 were uniformly distributed across the ranges $u_0 \in [0.0, 3.0]$, $v_0 \in [0.0, 3.0]$, $n_p \in \{1, 2, 3\}$, $h \in [0.05, 0.2]$, $w \in [200, 700]$, $x_0 \in [0.25, 1.25]$, and $y_0 \in [0.25, 1.25]$. The number of pressure pulses n_p was chosen randomly, and the characteristic parameters of each pulse were independently sampled from their respective distributions. The domain size was set to $\Omega = [0, 1.5]^2$, the remesh time was set as $T = 0.05$ with a total number of 4 RL steps, and the initial mesh distribution was set as $N = 48^2$. The training reward curve over the training period is shown in Fig. 14.

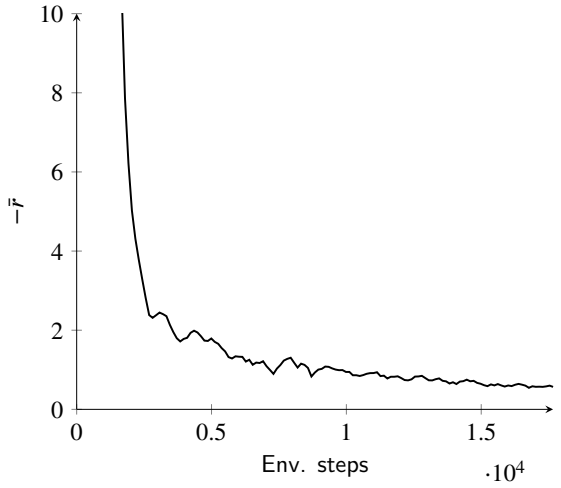


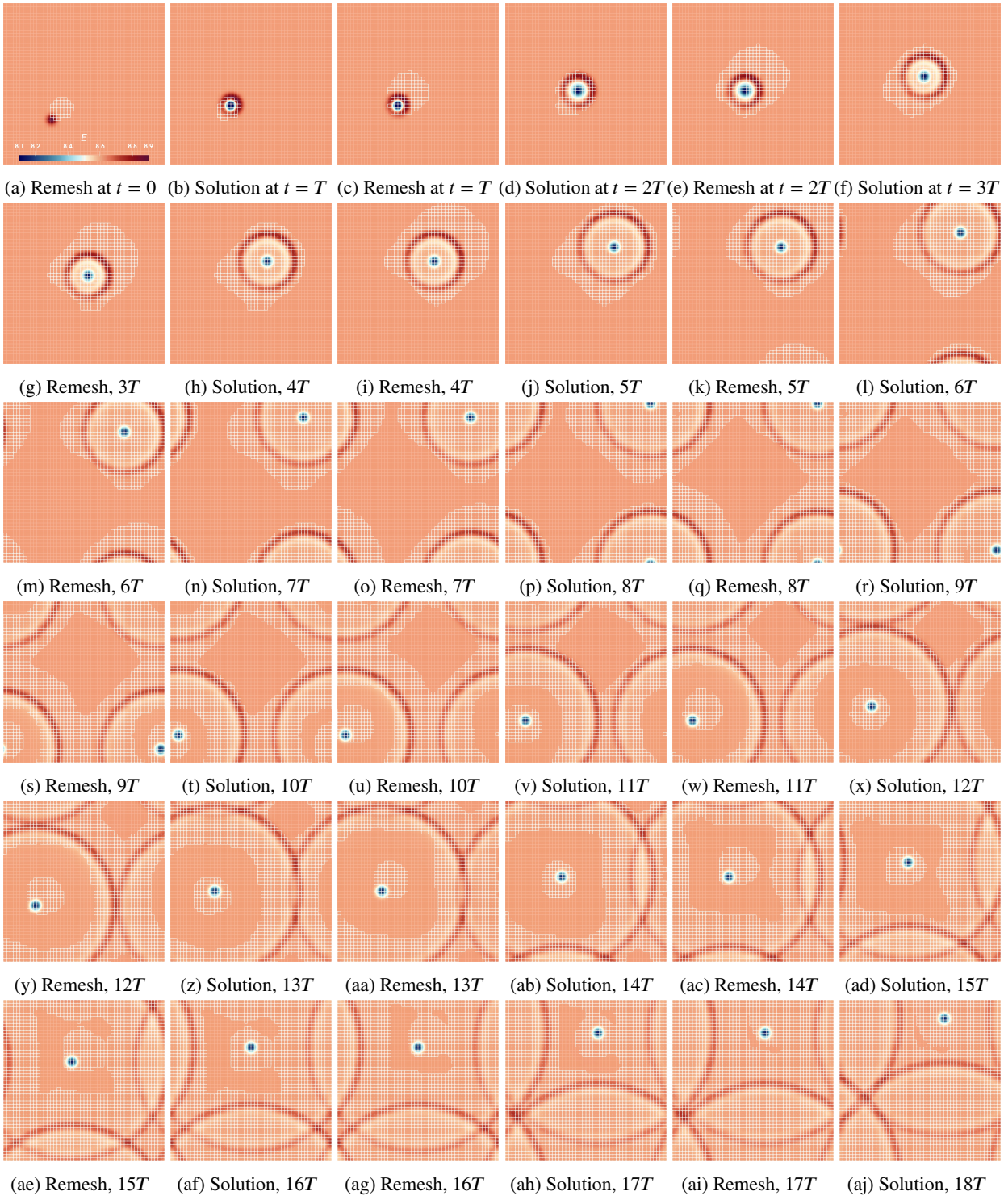
Figure 14: Batch-averaged (negative) reward with respect to number of environment steps during the training process for p-refinement on the Euler equations.

5.3.1. Sample problem

For these flow conditions, pressure sources, initialized as smooth Gaussian bumps, cause outward-running acoustic waves centered over a compact low-density/low-energy region. The combination of this acoustic propagation with a background convecting velocity yields interesting flow behavior, particular in the energy field where features with steep flow gradients are separated by relatively flat regions, presenting an ideal case for AMR. We first present an example of the DynAMO approach applied to this class of problems for a randomly selected *in-distribution* set of initial conditions with the parameters equal to $u_0 = 2.25$, $v_0 = 2.67$, $h = 0.12$, $w = 580$, $x_0 = 0.3$, and $y_0 = 0.53$. The evolution of the total energy and p-adapted mesh as computed by the DynAMO approach (with $\alpha = \alpha_{\text{train}}$) across 18 remesh time intervals, much longer than the 4 remesh time intervals used for training, is shown in Fig. 15. The spatio-temporal evolution of the flow for this class of problems is notably more complex than the convecting density pulse case as a result of the nonlinear acoustic wave interactions. However, even with the significantly longer simulation time, the DynAMO approach was able to preemptively refine regions of the prior to features of interest appearing. At earlier times, the evolution of the acoustic wave was dominated by the background convective velocity, such that the refinement pattern was reminiscent of advection-type problems, with an elongated refinement region stretched in the direction of the propagation velocity. However, as the pressure gradients drove the flow outwards from the pulse center, the preemptive refinement capability of DynAMO was able to accurately account for both the wave propagation and convection. At later times, the acoustic wave propagated far enough from its origin such that a region of low variation in the solution existed between the low energy center and the wave front. The DynAMO policy appropriately de-refined this region without erroneously de-refining the regions across which the wave front propagated. Furthermore, the collisions of the acoustic wave fronts were preemptively predicted in the refinement patterns, indicating that the proposed approach can effectively account for nonlinear flow physics in which the characteristic direction of propagation can strongly vary across elements (e.g., colliding wave fronts).

5.3.2. In-distribution experiments

To quantify the capability of the DynAMO approach for this case and present a comparison to baseline methods, the normalized cost vs. error distributions are shown in the Pareto plot in Fig. 16 for the DynAMO policy and the



(a) Remesh at $t = 0$ (b) Solution at $t = T$ (c) Remesh at $t = T$ (d) Solution at $t = 2T$ (e) Remesh at $t = 2T$ (f) Solution at $t = 3T$

(g) Remesh, $3T$ (h) Solution, $4T$ (i) Remesh, $4T$ (j) Solution, $5T$ (k) Remesh, $5T$ (l) Solution, $6T$

(m) Remesh, $6T$ (n) Solution, $7T$ (o) Remesh, $7T$ (p) Solution, $8T$ (q) Remesh, $8T$ (r) Solution, $9T$

(s) Remesh, $9T$ (t) Solution, $10T$ (u) Remesh, $10T$ (v) Solution, $11T$ (w) Remesh, $11T$ (x) Solution, $12T$

(y) Remesh, $12T$ (z) Solution, $13T$ (aa) Remesh, $13T$ (ab) Solution, $14T$ (ac) Remesh, $14T$ (ad) Solution, $15T$

(ae) Remesh, $15T$ (af) Solution, $16T$ (ag) Remesh, $16T$ (ah) Solution, $17T$ (ai) Remesh, $17T$ (aj) Solution, $18T$

Figure 15: Contours of total energy overlaid with p-adapted mesh at varying remesh intervals using DynAMO for the convecting pressure pulse problem. Highlighted elements represent p-refinement.

standard threshold policy. Both policies were evaluated over a range of threshold parameters to observe the effects on error, cost, and efficiency. For the threshold policy, where the threshold parameter was varied $\theta = 10^{-2}$ to 10^{-11}, a linear relation was observed in the cost to error profile, with the most optimal value yielding a peak efficiency of roughly 0.65. In contrast, the DynAMO policy was able to produce a far more efficient AMR approach. At the training threshold value $\alpha_{\text{train}} = 0.1$, the policy was able to achieve an efficiency of approximately 0.9 with almost zero normalized error, yielding results of similar accuracy to a fully refined mesh. In contrast, the threshold policy could only achieve a normalized error of approximately 0.5 for the same computational cost. Similarly, to achieve the same normalized error, the threshold policy required roughly quadruple the computational cost. When the error threshold for the DynAMO policy was varied at evaluation time from $\alpha = 10^{-6}$ to 0.6, the quintessential convex Pareto curve was recovered with the sampled results much more closely distributed near the origin. Most importantly, all of the sampled points along that curve were well within the bounds of the threshold policy results, showcasing that the proposed approach can unlock efficiency levels that are out of reach with conventional AMR techniques.

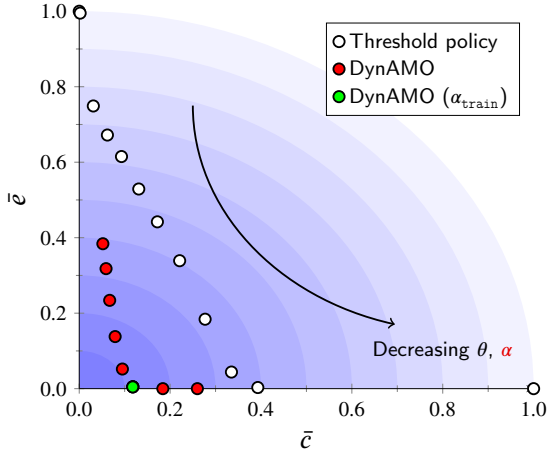


Figure 16: Pareto plot of normalized cost vs. error for p-refinement on the Euler equations with DynAMO and the threshold policy for the in-distribution convecting pressure pulse case. Contours of efficiency shown on background. Red markers represent a sweep of $\alpha \in [10^{-6}, 0.6]$ at evaluation time, green marker represents α set to the training value of $\alpha_{\text{train}} = 0.1$. Peak efficiency achieved at $\alpha = 0.2$.

To verify that the increased efficiency of the proposed DynAMO approach persists across a wide distribution of test cases, a comparison of the DynAMO approach ($\alpha = \alpha_{\text{train}}$) and the threshold policy was performed across 100 randomly sampled *in-distribution* initial conditions. The results of the comparison are shown in Table 5 for varying threshold values for the threshold policy. At its peak, the threshold policy achieves a mean efficiency of 0.664, with normalized mean error and cost values of 0.184 and 0.277, respectively. In comparison, the DynAMO policy achieved a noticeably higher mean efficiency, 0.882, a relative increase of 32.8%. The benefits of the DynAMO approach were more pronounced when observing the relative decrease in mean error and computational cost. The DynAMO policy achieved a relative mean error of 0.005, which effectively results in a level of accuracy identical to a fully-refined mesh. This error was 97.3% less than the mean error provided by the most efficient threshold policy. Furthermore, the DynAMO policy achieved this level of error with a relative mean computational cost of 0.118, which is 57.4% less than the computational cost of the most efficient threshold policy that results in a mean error that is nearly 40 times higher. To achieve the level of error provided by the DynAMO policy, the threshold policy would essentially have to utilize a fully-refined mesh, such that the mean computational cost would be almost 10 times higher. Conversely, for the same mean computational cost as the DynAMO policy, the use of the threshold policy would result in more than two orders of magnitude increase in error.

5.3.3. Generalization experiments

The efficacy and generalization capabilities of the DynAMO approach were further evaluated through experiments for *out-of-distribution* problems, where the threshold parameter of the threshold policy was varied and the threshold parameter for the DynAMO policy was set to its training value. For brevity, certain results for the threshold policy are

Method	Efficiency	Normalized error	Normalized cost
Threshold ($\theta = 10^{-2}$)	0.000 (0.000)	1.000 (0.000)	0.000 (0.000)
Threshold ($\theta = 10^{-3}$)	0.005 (0.020)	0.995 (0.020)	0.002 (0.001)
Threshold ($\theta = 10^{-4}$)	0.250 (0.048)	0.749 (0.049)	0.031 (0.006)
Threshold ($\theta = 10^{-5}$)	0.325 (0.023)	0.672 (0.023)	0.062 (0.006)
Threshold ($\theta = 10^{-6}$)	0.377 (0.033)	0.615 (0.034)	0.093 (0.008)
Threshold ($\theta = 10^{-7}$)	0.455 (0.032)	0.529 (0.034)	0.131 (0.009)
Threshold ($\theta = 10^{-8}$)	0.525 (0.038)	0.442 (0.045)	0.172 (0.010)
Threshold ($\theta = 10^{-9}$)	0.593 (0.040)	0.339 (0.055)	0.221 (0.011)
Threshold ($\theta = 10^{-10}$)	**0.664 (0.029)**	**0.184 (0.057)**	**0.277 (0.011)**
Threshold ($\theta = 10^{-11}$)	0.661 (0.011)	0.044 (0.026)	0.335 (0.011)
DynAMO	**0.882 (0.022)**	**0.005 (0.008)**	**0.118 (0.023)**
DynAMO/Optimal θ	+32.8%	-97.3%	-57.4%

Table 5: Comparison of the mean efficiency, normalized error, and normalized cost for p-refinement on the Euler equations with DynAMO and the threshold policy for the convecting pressure pulse problem over 100 *in-distribution* runs using uniform random initial conditions. Standard deviation shown in parentheses.

omitted as the recovered efficiency levels were not significant.

The first generalization experiment was performed with respect to different flow physics through the convecting density pulse problem. This problem is given by the initial conditions

$$\rho(x, y, 0) = 1 + h \exp\left(-w\left((x - x_0)^2 + (y - y_0)^2\right)\right), \tag{28a}$$

$$u(x, y, 0) = u_0, \tag{28b}$$

$$v(x, y, 0) = v_0, \tag{28c}$$

$$P(x, y, 0) = 1, \tag{28d}$$

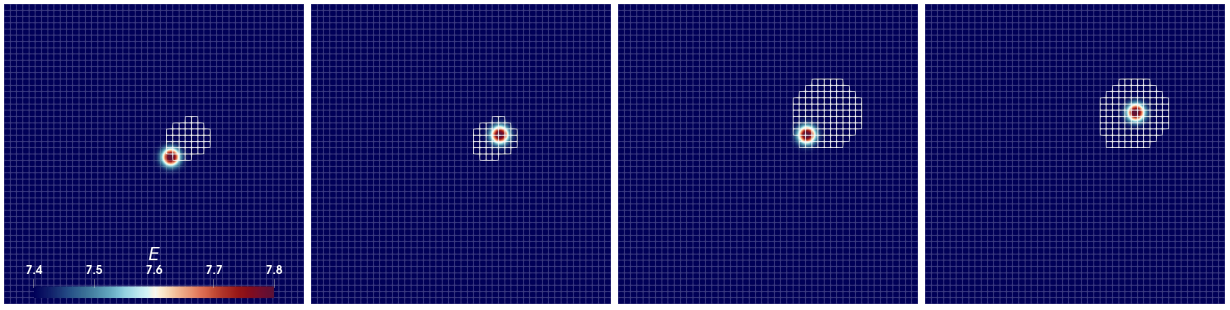
where the parameters u_0, v_0, h, w, x_0, and y_0 were uniformly distributed across the same ranges as Eq. (27a). For this setup, a perturbation in the density field convects at a constant velocity and direction, much like in the advection problems. Besides showing the generalizability of the proposed observation and reward functions, the purpose of this example is to show a direct extension of the DynAMO approach from the advection equation to the Euler equations for a problem which effectively exhibits the same physics but achieves it from a more complex set of governing equations. Recall that while the evolution of the density field is linear with respect to the solution, the evolution of the total energy field, the solution component which is actually being observed by the policy, is not.

While the flow physics may seem relatively simple, this particular class of problems presents a very interesting test for the proposed policy as the initial distribution of the total energy, the observable quantity, is effectively identical between these cases and the pressure pulses that the policy was trained on, but the resulting flow fields are radically different. As such, it presents a validation of the ability of the proposed approach in learning the physics of the underlying governing equations as opposed to simply extrapolating from previously encountered flow problems. A comparison between the threshold policy and DynAMO policy over 100 randomly sampled initial conditions for the problem is shown in Table 6. Due to the compactly supported nature of the convecting density problem, the threshold policy was able to achieve a high mean efficiency, 0.799. However, the DynAMO policy was able to achieve even higher, 0.955, almost reaching the theoretical limit of unity. The relative efficiency benefits of the DynAMO approach over the optimal threshold policy remained similar to the *in-distribution* case, such that it can be surmised that the approach may have the ability to effectively generalize to flow physics not encountered during the training process.

To present an visualization of the approach for the convecting density pulse problem, a random initial condition was chosen from the distribution with parameters equal to $u_0 = 2.46$, $v_0 = 2.99$, $h = 0.08$, $w = 267$, $x_0 = 0.3$, and $y_0 = 0.53$. The evolution of the total energy and p-adapted mesh as obtained by the DynAMO policy trained on the pressure pulse cases is shown in Fig. 17. For brevity, we present only the first four remesh intervals, although this behavior was essentially identical at later simulation times. Furthermore, the results are shown with respect to the total energy, which is the observable field, instead of the density, but these distributions are nearly identical for a constant background velocity and pressure. It can be seen that similar behavior as with the policies trained on the advection

Method	In-distribution	Convecting density	Finer mesh	Longer remesh time	Longer sim. time
Threshold ($\theta = 10^{-2}$)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
Threshold ($\theta = 10^{-4}$)	0.250 (0.048)	0.015 (0.003)	0.007 (0.006)	0.011 (0.005)	0.020 (0.011)
Threshold ($\theta = 10^{-6}$)	0.377 (0.033)	0.036 (0.007)	0.200 (0.062)	0.025 (0.022)	0.114 (0.046)
Threshold ($\theta = 10^{-8}$)	0.525 (0.038)	0.073 (0.008)	0.637 (0.092)	0.075 (0.053)	0.495 (0.025)
Threshold ($\theta = 10^{-9}$)	0.593 (0.040)	0.099 (0.009)	0.738 (0.076)	0.124 (0.076)	**0.516 (0.022)**
Threshold ($\theta = 10^{-10}$)	**0.664 (0.029)**	0.128 (0.011)	0.824 (0.032)	0.224 (0.084)	0.507 (0.014)
Threshold ($\theta = 10^{-11}$)	0.661 (0.011)	0.163 (0.012)	**0.843 (0.007)**	0.308 (0.038)	0.456 (0.015)
Threshold ($\theta = 10^{-12}$)	0.607 (0.010)	**0.799 (0.012)**	0.820 (0.008)	0.348 (0.020)	0.394 (0.014)
Threshold ($\theta = 10^{-14}$)	0.493 (0.010)	0.707 (0.013)	0.757 (0.010)	0.429 (0.040)	0.284 (0.011)
Threshold ($\theta = 10^{-15}$)	0.342 (0.020)	0.459 (0.034)	0.478 (0.047)	**0.470 (0.024)**	0.183 (0.010)
DynAMO	**0.882 (0.022)**	**0.955 (0.014)**	**0.910 (0.012)**	**0.568 (0.074)**	**0.726 (0.026)**
DynAMO/Optimal θ	+32.8%	+19.5%	+7.9%	+20.8%	+40.1%

Table 6: Comparison of the mean efficiency for p-refinement on the Euler equations with DynAMO and the threshold policy for the two-dimensional convecting pressure-pulse over 100 *out-of-distribution* runs using uniform random initial conditions with finer mesh resolution, longer remesh time, and longer simulation time. Standard deviation shown in parentheses. In-distribution results from Table 5 shown for comparison. All sampled thresholds are not shown for brevity.



(a) Remesh at $t = 0$ (b) Solution at $t = T$ (c) Remesh at $t = T$ (d) Solution at $t = 2T$

(e) Remesh at $t = 2T$ (f) Solution at $t = 3T$ (g) Remesh at $t = 3T$ (h) Solution at $t = 4T$

Figure 17: Contours of total energy overlaid with p-adapted mesh at varying remesh intervals using DynAMO for the convecting density pulse problem. Highlighted elements represent p-refinement.

equation is recovered, even with the more complex underlying governing equations and observations. The DynAMO policy predicts an elongated refinement region at the initial time, which adequately covers the propagation path of the density pulse. This preemptive refinement capability is maintained throughout the entire simulation time, such that the density pulse always remains within the refined region of the mesh. While the physics of the problem are simple, these results showcase a direct connection between linear and nonlinear equations for the proposed formulation.

The next generalization experiment was performed with respect to the mesh resolution. The base mesh resolution was increased from $N = 48^2$ to $N = 96^2$, such that the total number of degrees of freedom was over $3 \cdot 10^5$ at the coarsest level. The mean efficiency across 100 sampled initial conditions on the finer mesh level is shown in Table 6

for both the DynAMO policy and the threshold policy. It can be seen that both policies recover a higher efficiency on the finer mesh which is consistent with previous results. For the threshold policy, this efficiency increased from 0.664 to 0.843, whereas for the DynAMO policy, this efficiency increased from 0.882 to 0.910. As a result, the relative benefit of DynAMO was lower, only 7.9%, but this is largely attributed to the marginal gains possible at the highest efficiency levels. However, the results indicate that the DynAMO policy can very effectively generalize to different mesh resolutions for the Euler equations.

The third generalization experiment was performed with respect to the remesh time T, which was doubled from its training value. With longer remesh time intervals, it is expected that the efficiency of policies based on instantaneous error estimators would degrade. This behavior was indeed observed for the threshold policy, also shown in Table 6, where the highest mean efficiency was 0.470 over 100 randomly sampled initial conditions, noticeably less than the in-distribution result of 0.664. Furthermore, this efficiency was only recovered at machine zero values of the threshold parameter, meaning that the policy refines any element with a non-constant total energy distribution. A notably more efficient approach was recovered with the DynAMO policy, with a mean efficiency of 0.568. These relative efficiency benefits were on par with the increase observed for the *in-distribution* experiments, indicating that the approach can also effectively generalize to longer remesh times.

The final generalization experiment was performed with respect to the total simulation time, which was also doubled from its training value. The comparison of the threshold policy at varying values of the threshold parameter to the DynAMO approach over 100 randomly sampled initial conditions is shown in Table 6. It can be seen that the peak efficiency of the threshold approach decreased from 0.664 to 0.516, but the mean efficiency of the DynAMO approach decreased as well, from 0.882 to 0.726. This decrease in efficiency is somewhat expected at longer simulation times for flows whose complexity increases over time as the most appropriate refinement decision tends toward uniform refinement, which is seen in Fig. 15, such that the increase in normalized cost decreases the efficiency. However, we observe that the efficiency benefits of the DynAMO approach do not deteriorate for longer simulation times but, in fact, noticeably appreciate. This further supports the generalization abilities of the proposed approach.

5.4. Euler equations with h-refinement

Following the initial evaluation of the proposed DynAMO approach for the Euler equations on smooth problems, the approach was then extended to significantly more complex and highly nonlinear flow physics featuring shocks, rarefaction waves, and discontinuities. The discontinuous, compact nature of many of these features paired with their nonlinear evolution presents an ideal example for problems where anticipatory mesh refinement would excel while methods based on instantaneous estimators would be unsuccessful. Due to the lack of smoothness of these features, h-refinement was used for the policies as it is typically better suited to resolve such features.

When relaxing the constraint of smoothness in the flow, a wide variety of problems with different flow physics become available for the Euler equations. To evaluate the efficacy of the approach for a sufficiently diverse set of flow physics, we consider the quintessential example of a two-dimensional Riemann problem [49], which possesses complex nonlinear flow behavior including shock waves, contact discontinuities, rarefaction waves, vortical flow, Kelvin–Helmholtz and Richtmyer–Meshkov instabilities, and shock-vortex interactions. This problem is initialized by subdividing the domain into four respective quadrants, shown in Fig. 18, each with their own unique initial state. The partition between these states is representative of a physical diaphragm separating individual gases which is instantaneously broken, allowing for the interaction of these gases. Unless otherwise stated, we utilize a unit domain $\Omega = [0, 1]^2$ with the diaphragms placed at $x_0 = y_0 = 0.5$, respectively. Furthermore, the domain is set as periodic, such that the numerical setup is representative of an infinite array of two-dimensional Riemann problems. While the enforcement of periodic boundary conditions is not standard for two-dimensional Riemann problems, this helps to increase the efficiency of the training process by increasing the overall complexity and variation of features encountered per episode (i.e., a single two-dimensional Riemann problem turns into four unique Riemann problems). For brevity, we use the notation tl, tr, bl, and br to denote the top-left, top-right, bottom-left, and bottom-right quadrants, respectively.

The DynAMO policy was trained on a variety of two-dimensional Riemann problems generated by uniformly sampling the individual quadrant states from the distributions $\rho \in [0.2, 2]$, $u \in [-0.5, 0.5]$, $v \in [-0.5, 0.5]$, and $P \in [0.2, 2]$. The remesh time was set as $T = 0.05$ with a total number of 4 RL steps, and the initial mesh distribution was set as $N = 32^2$. Furthermore, to add more variation to the flow problems during training, we vary the placement of the diaphragms within the ranges $x_0 \in [0.3, 0.7]$ and $y_0 \in [0.3, 0.7]$, but at evaluation time, these values were fixed at $x_0 = y_0 = 0.5$. The reward curve for the agent during training is shown in Fig. 19.

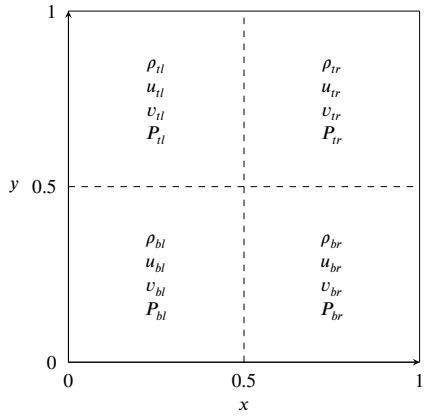


Figure 18: Schematic of the initial conditions and domain for the 2D Riemann problem.

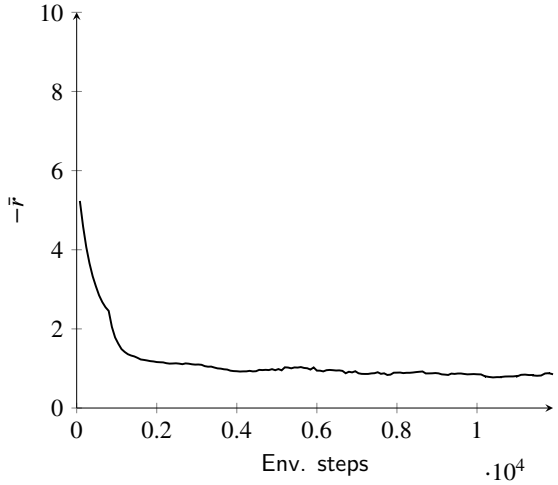


Figure 19: Batch-averaged (negative) reward with respect to number of environment steps during the training process for h-refinement on the Euler equations.

5.4.1. In-distribution experiments

Method	Efficiency	Normalized error	Normalized cost
Threshold ($\theta = 10^{-1}$)	0.215 (0.083)	0.763 (0.088)	0.177 (0.049)
Threshold ($\theta = 10^{-2}$)	**0.251 (0.056)**	**0.439 (0.111)**	**0.596 (0.061)**
Threshold ($\theta = 10^{-3}$)	0.180 (0.031)	0.291 (0.099)	0.760 (0.024)
Threshold ($\theta = 10^{-4}$)	0.162 (0.027)	0.272 (0.093)	0.787 (0.009)
Threshold ($\theta = 10^{-5}$)	0.157 (0.027)	0.266 (0.090)	0.795 (0.002)
Threshold ($\theta = 10^{-6}$)	0.157 (0.027)	0.265 (0.090)	0.796 (0.000)
DynAMO	**0.486 (0.073)**	**0.142 (0.092)**	**0.482 (0.090)**
DynAMO/Optimal θ	+93.6%	-67.7%	-19.1%

Table 7: Comparison of the mean efficiency, normalized error, and normalized cost for h-refinement on the Euler equations with DynAMO and the threshold policy for the two-dimensional Riemann problem over 100 *in-distribution* runs using uniform random initial conditions. Standard deviation shown in parentheses.

An evaluation of the DynAMO policy with $\alpha = \alpha_{\text{train}}$ and a comparison to the baseline threshold policy at varying

values of the threshold parameter was first performed across 100 uniformly sampled *in-distribution* initial conditions. The normalized error, cost, and efficiency for the two approaches is shown in Table 7. Due to the increased complexity of the flow physics, the optimal threshold policy only achieved a mean efficiency of 0.251 with relatively large normalized error and cost values of 0.439 and 0.596, respectively. Furthermore, it can be seen that the threshold policy effectively stagnates around a minimum mean error of 0.265 even with a decreasing threshold value. This behavior is as a result of the inability of instantaneous error estimators to account for the spatio-temporal evolution of the error. For this particular set of problems, this largely manifests at the first time step, where the estimator evaluated on the initially discontinuous state can, at best, predict a non-zero error only along (or one element adjacent to) the discontinuities. As such, the adapted mesh based on the instantaneous error estimator quickly becomes suboptimal, introducing a large degree of discretization error as the discontinuities evolve away from their initial positions. This effect is also encountered at later times as the discontinuities propagate across the mesh, although to a lesser extent. In contrast, the DynAMO approach showed a significantly better mean efficiency of 0.486, a 93.6% increase over the optimal threshold policy. This increased efficiency also came with a significantly lower mean error, 0.142, a 67.7% decrease in comparison to the optimal threshold policy. Most importantly, this error was much lower than the minimum error achievable with the threshold policy, showing that preemptive refinement decisions provided by the DynAMO policy can achieve levels of accuracy that standard AMR approaches may not be able to. Additionally, this increased accuracy came with a 19.1% decrease in the computational cost as a result of the more efficient refinement decisions of the DynAMO policy.

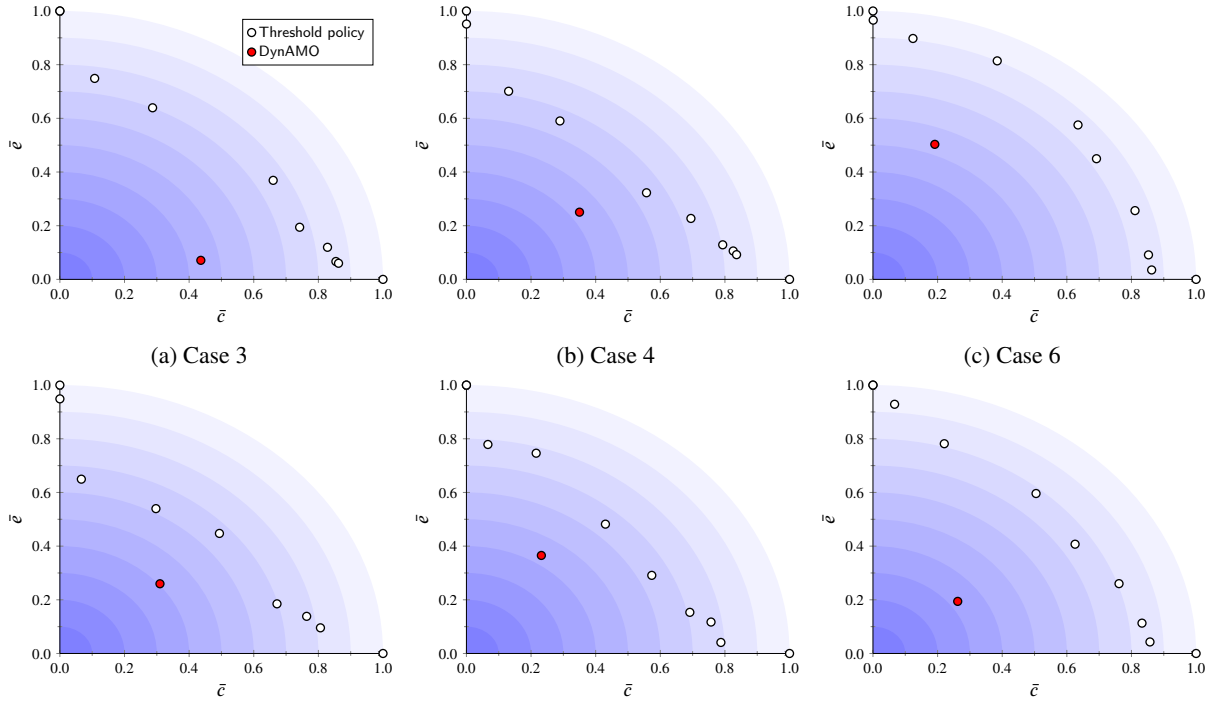
5.4.2. Sample problems

To further validate the DynAMO approach, we consider canonical instances of two-dimensional Riemann problems that are used as example cases for evaluating numerical methods. In particular, we consider the six Riemann problems used in the work of Liska and Wendroff [50] which are derived from the work of Lax and Liu [51]. The initial conditions as well as the final simulation time for these six Riemann problems, denoted by case numbers 3, 4, 6, 12, 15, and 17, are shown in Table 8. For these problems, the standard implementation requires Neumann boundary conditions to yield the expected flow patterns. As the current implementation of the DynAMO approach utilizes periodic meshes, we replicate the effects of the required boundary conditions by extending the domain by a factor of two to $\Omega = [0, 2]^2$ and considering only the interior subdomain $\Omega' = [0.5, 1.5]^2$ with the diaphragm placed at $x_0 = y_0 = 1$. Due to the finite rate of propagation of flow perturbations as well as the self-similarity of Riemann problems with respect to the similarity parameter x/t, this approach yields identical results to applying Neumann boundary conditions on the subdomain Ω' over the time period of the problem. The characteristic mesh scale was kept identical by doubling the resolution to $N = 64^2$, and analysis of the cost and error was performed with respect to only the subdomain Ω'. We remark here that these conditions are effectively *out-of-distribution*, not only due to the increased domain size and mesh resolution but also due to the many of the quadrant states residing outside of the training distribution.

Case	ρ_l	u_l	v_l	P_l	ρ_r	u_r	v_r	P_r	t_f
	0.5323	1.206	0.0	0.3	1.5	0.0	0.0	1.5	
3	0.138	1.206	1.206	0.029	0.5323	0.0	1.206	0.3	0.3
	0.5065	0.8939	0.0	0.35	1.1	0.0	0.0	1.1	
4	1.1	0.8939	0.8939	1.1	0.5065	0.0	0.8939	0.35	0.25
	2.0	0.75	0.5	1.0	1.0	0.75	-0.5	1.0	
6	1.0	-0.75	0.5	1.0	3.0	-0.75	-0.5	1.0	0.3
	1.0	0.7276	0.0	1.0	0.5313	0.0	0.0	0.4	
12	0.8	0.0	0.0	1.0	1.0	0.0	0.7276	1.0	0.25
	0.5197	-0.6259	-0.3	0.4	1.0	0.1	-0.3	1.0	
15	0.8	0.1	-0.3	0.4	0.5313	0.1	0.4276	0.4	0.2
	2.0	0.0	-0.3	1.0	1.0	0.0	-0.4	1.0	
17	1.0625	0.0	0.2145	0.4	0.5197	0.0	-1.1259	0.4	0.3

Table 8: Initial conditions for the two-dimensional Riemann problem tests from Liska and Wendroff [50]. Top and bottom quadrant states shown in upper and bottom rows, respectively.

A comparison of the normalized cost vs. error between the threshold policy and the DynAMO approach for these six two-dimensional Riemann problem cases is shown through Pareto plots in Fig. 20. Due to the previously mentioned inability of instantaneous error-based policies to appropriately refine the mesh at the initial state, the Pareto curve for the threshold policy stagnates at above a fixed error and does not show as well-behaved of a curve as with the smooth problems for p-refinement. As a result, the threshold policies did not achieve an efficiency of more than 0.4 for any of the cases. Of the six cases, the threshold policy showed the worst performance for Case 6 and the best performance for Case 4 and Case 12. In comparison, the DynAMO policy showed significantly better performance, generally achieving an efficiency of approximately 0.6 with minor variation across the cases. This increase in efficiency resulted in the policy achieving a similar error to the threshold policy with significantly lower computational cost, owing to its ability to preemptively refine mesh regions prior to features appearing. These results provide an initial quantitative evaluation of the proposed approach for providing more efficient adaptive mesh refinement strategies for complex problems of practical interest.



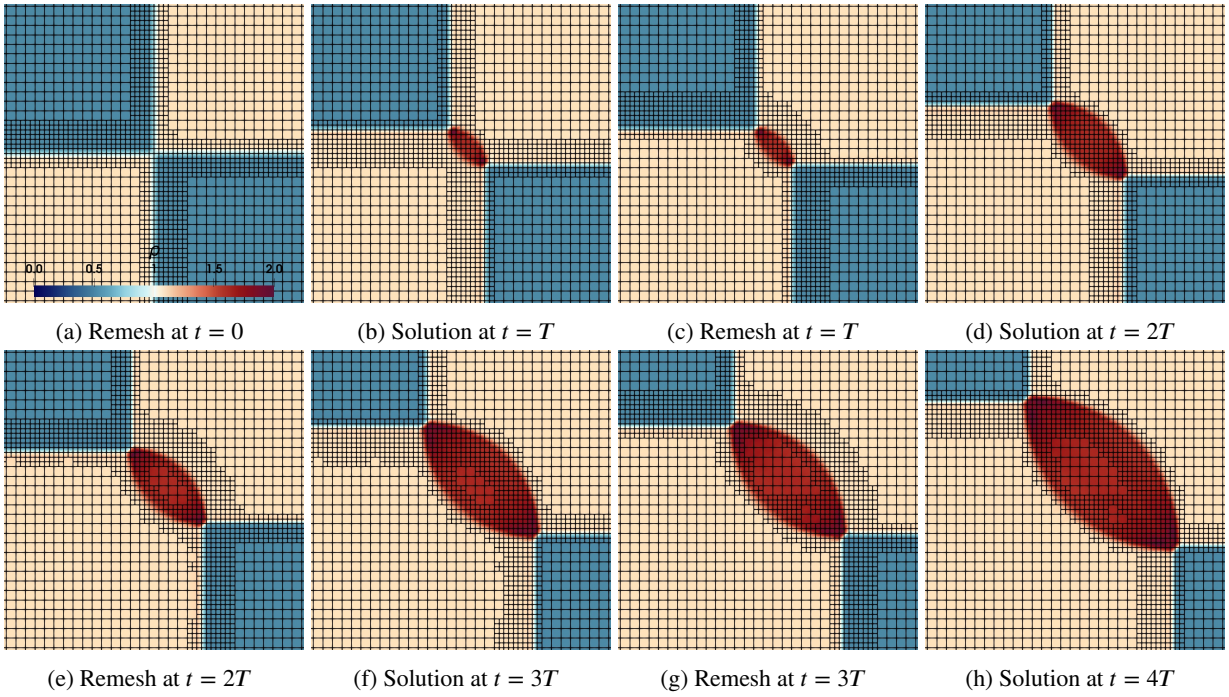
(a) Case 3 (b) Case 4 (c) Case 6

(d) Case 12 (e) Case 15 (f) Case 17

Figure 20: Pareto plot of normalized cost vs. error for h-refinement on the Euler equations with DynAMO and the threshold policy for the two-dimensional Riemann problem tests from Liska and Wendroff [50]. Contours of efficiency shown on background.

The DynAMO policy and threshold policy was further qualitatively compared for these six two-dimensional Riemann problems by observing the mesh refinement decisions. A comparison was made for Case 4 and Case 12 as these are the cases for which the threshold policy showed the best performance. The threshold parameter value was set to $\theta = 10^{-2}$ which corresponds to the points of highest efficiency along the Pareto curve for both cases. The evolution of the density and h-adapted mesh for Case 4 as obtained by the DynAMO policy and threshold policy is shown in Fig. 21 and Fig. 22, respectively. Note that the density is shown instead of the total energy, the observed quantity, as this is the typical component used for comparison in the literature. It can be seen that at the initial time, the DynAMO policy refines several elements ahead of the discontinuities. This refinement region covered the entire expansion wave as well as the propagation of the discontinuities across the remesh interval. This behavior persisted at further times, showcasing the preemptive refinement capabilities of DynAMO. Furthermore, as the expansion wave region became sufficiently large, the DynAMO policy started to de-refine its interior which contains features with relatively low variation and, by extent, error. In comparison, the threshold policy could only refine the elements along the discontinuities

and their immediate neighbors at the initial time. Therefore, by the next remesh time, the discontinuities propagated outside of the refinement region. This behavior became more egregious at the next remesh interval, where both the discontinuities as well as the expansion wave front moved outside of the refinement region, introducing a significant degree of discretization error. This error resulted in some irregular refinement patterns at later times which contributed to the cost of threshold policy without any benefits in the accuracy.



(a) Remesh at $t = 0$ (b) Solution at $t = T$ (c) Remesh at $t = T$ (d) Solution at $t = 2T$

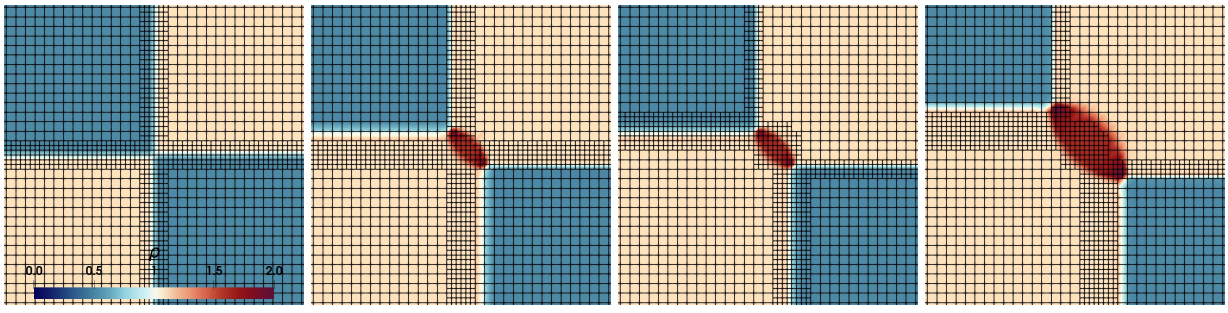
(e) Remesh at $t = 2T$ (f) Solution at $t = 3T$ (g) Remesh at $t = 3T$ (h) Solution at $t = 4T$

Figure 21: Contours of density overlaid with h-adapted mesh at varying remesh intervals using DynAMO for the Case 4 two-dimensional Riemann problem from Liska and Wendroff [50].

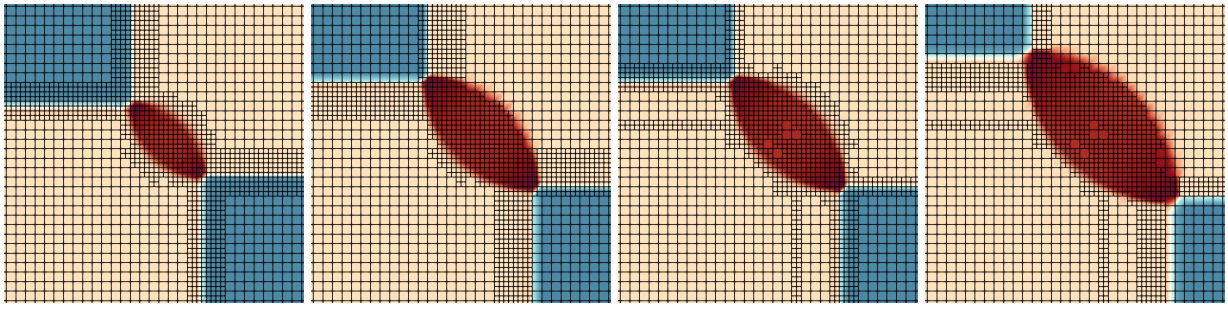
The comparison between the DynAMO policy and threshold policy for Case 12 is also shown through the evolution of the density and h-adapted mesh in Fig. 23 and Fig. 24, respectively. Much like with Case 4, the DynAMO policy was able to preemptively refine the mesh at the initial time, such that the evolution of the discontinuities and expansion wave was contained within the refined region. However, it must be noted that since the contact discontinuities bordering the bottom-left quadrant are stationary, the DynAMO approach over-refined around these discontinuities. This behavior was later corrected, such that the refined region consisted of only a single element-wide region. The DynAMO policy did correctly predict the necessary refinement regions to cover the evolution of the discontinuities in the top-right quadrant. At later times, when the expansion wave was sufficiently developed, the DynAMO policy de-refined the interior region which could be sufficiently resolved on the coarse mesh level. For the threshold policy, an identical initial refinement pattern was observed with Case 12 as with Case 4 due to the similarity of the initial states. However, this refinement pattern also did not adequately cover the evolution of the discontinuities and expansion wave, such that a significant amount of discretization error was introduced around the discontinuity front. This behavior persisted throughout the simulation, with the threshold policy unable to adequately account for the propagation of the features.

5.4.3. Generalization experiments

The generalization capabilities of the DynAMO approach were further evaluated through *out-of-distribution* experiments for problems on finer meshes, with longer remesh times, and with longer simulation times. The first generalization experiment was performed by increasing the base mesh resolution from $N = 32^2$ to 64^2. A comparison between the mean efficiency of the DynAMO approach and the threshold policy on this finer base mesh is shown in Table 9 averaged over 100 runs. Both approaches showed an increase in mean efficiency, which is consist with previous generalization experiments, with the optimal threshold policy increasing from 0.251 to 0.328 and the DynAMO policy increasing from 0.486 to 0.545. This resulted in a relative efficiency increase of 66.2% with the DynAMO policy, such

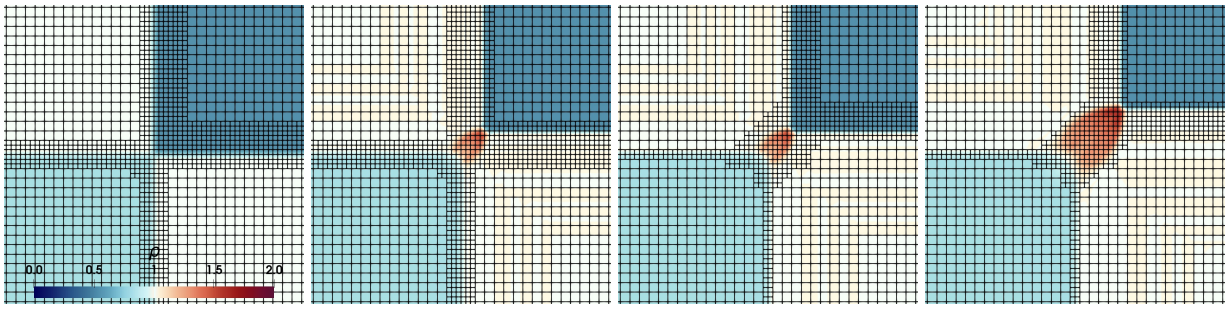


(a) Remesh at $t = 0$ (b) Solution at $t = T$ (c) Remesh at $t = T$ (d) Solution at $t = 2T$

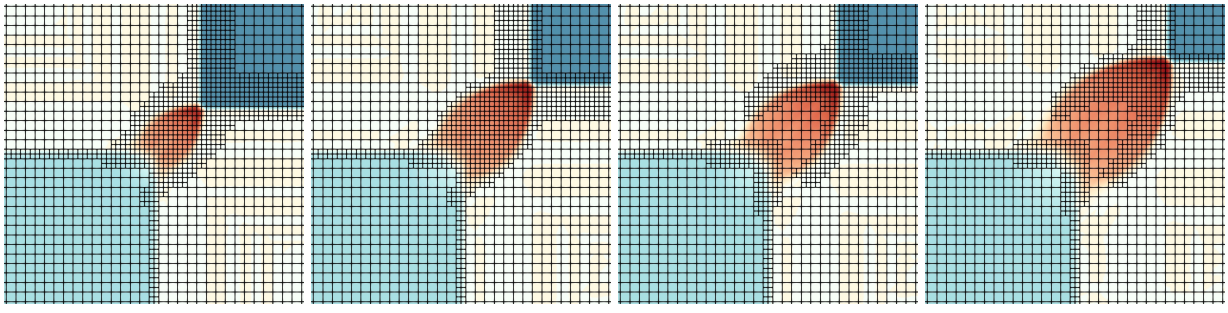


(e) Remesh at $t = 2T$ (f) Solution at $t = 3T$ (g) Remesh at $t = 3T$ (h) Solution at $t = 4T$

Figure 22: Contours of density overlaid with h-adapted mesh at varying remesh intervals using the threshold policy ($\theta = 10^{-2}$) for the Case 4 two-dimensional Riemann problem from Liska and Wendroff [50].

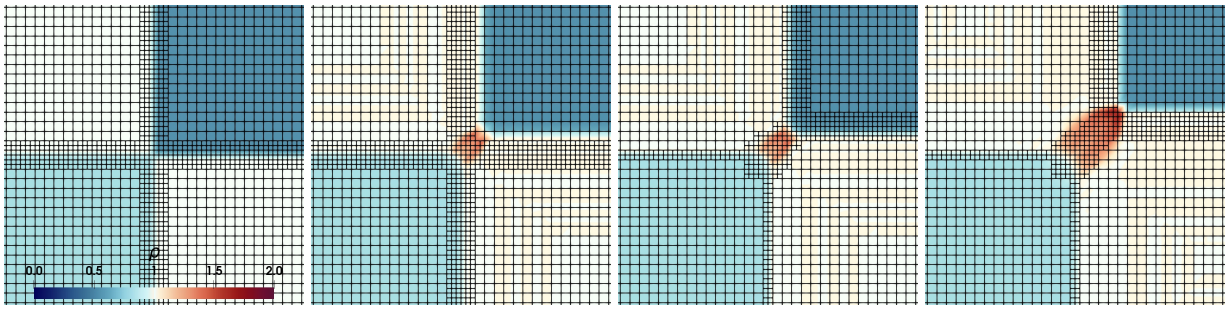


(a) Remesh at $t = 0$ (b) Solution at $t = T$ (c) Remesh at $t = T$ (d) Solution at $t = 2T$



(e) Remesh at $t = 2T$ (f) Solution at $t = 3T$ (g) Remesh at $t = 3T$ (h) Solution at $t = 4T$

Figure 23: Contours of density overlaid with h-adapted mesh at varying remesh intervals using DynAMO for the Case 12 two-dimensional Riemann problem from Liska and Wendroff [50].



(a) Remesh at $t = 0$ (b) Solution at $t = T$ (c) Remesh at $t = T$ (d) Solution at $t = 2T$

(e) Remesh at $t = 2T$ (f) Solution at $t = 3T$ (g) Remesh at $t = 3T$ (h) Solution at $t = 4T$

Figure 24: Contours of density overlaid with h-adapted mesh at varying remesh intervals using the threshold policy ($\theta = 10^{-2}$) for the Case 12 two-dimensional Riemann problem from Liska and Wendroff [50].

that it significantly outperformed the threshold policy even on the finer mesh.

Method	In-distribution	Finer mesh	Longer remesh time	Longer sim. time
Threshold ($\theta = 10^{-1}$)	0.215 (0.083)	0.280 (0.102)	0.223 (0.082)	**0.226 (0.096)**
Threshold ($\theta = 10^{-2}$)	**0.251 (0.056)**	**0.328 (0.082)**	**0.315 (0.052)**	0.202 (0.065)
Threshold ($\theta = 10^{-3}$)	0.180 (0.031)	0.245 (0.051)	0.299 (0.044)	0.084 (0.022)
Threshold ($\theta = 10^{-4}$)	0.162 (0.027)	0.212 (0.041)	0.298 (0.042)	0.075 (0.021)
Threshold ($\theta = 10^{-5}$)	0.157 (0.027)	0.193 (0.035)	0.298 (0.041)	0.073 (0.021)
Threshold ($\theta = 10^{-6}$)	0.157 (0.027)	0.180 (0.030)	0.298 (0.041)	0.073 (0.021)
DynAMO	**0.486 (0.073)**	**0.545 (0.069)**	**0.611 (0.048)**	**0.423 (0.113)**
DynAMO/Optimal θ	+93.6%	+66.2%	+93.9%	+87.2%

Table 9: Comparison of the mean efficiency for h-refinement on the Euler equations with DynAMO and the threshold policy for the two-dimensional Riemann problem over 100 *out-of-distribution* runs using uniform random initial conditions with finer mesh resolution, longer remesh time, and longer simulation time. Standard deviation shown in parentheses. In-distribution results from Table 7 shown for comparison.

The second generalization experiment was performed with respect to the remesh time T, where the remesh time was doubled from its training value. A comparison between the mean efficiency of the DynAMO approach and the threshold policy averaged over 100 runs for this longer remesh time is shown in Table 9. The optimal threshold policy showed a marginal increase in efficiency in comparison to the *in-distribution* results, from 0.251 to 0.328. The DynAMO policy also showed a similar increase, from 0.486 to 0.545. As such, the relative efficiency increase of the DynAMO policy over the optimal threshold policy stayed almost identical to the *in-distribution* value, 93.9%, showcasing the ability of the approach to generalize to arbitrary remesh times.

The third generalization experiment was performed with respect to the simulation time t_f, where the simulation time was doubled from its training value. Table 9 shows a comparison of the mean efficiency between the DynAMO approach and the threshold policy over 100 runs for this longer simulation time. For both approaches, the mean

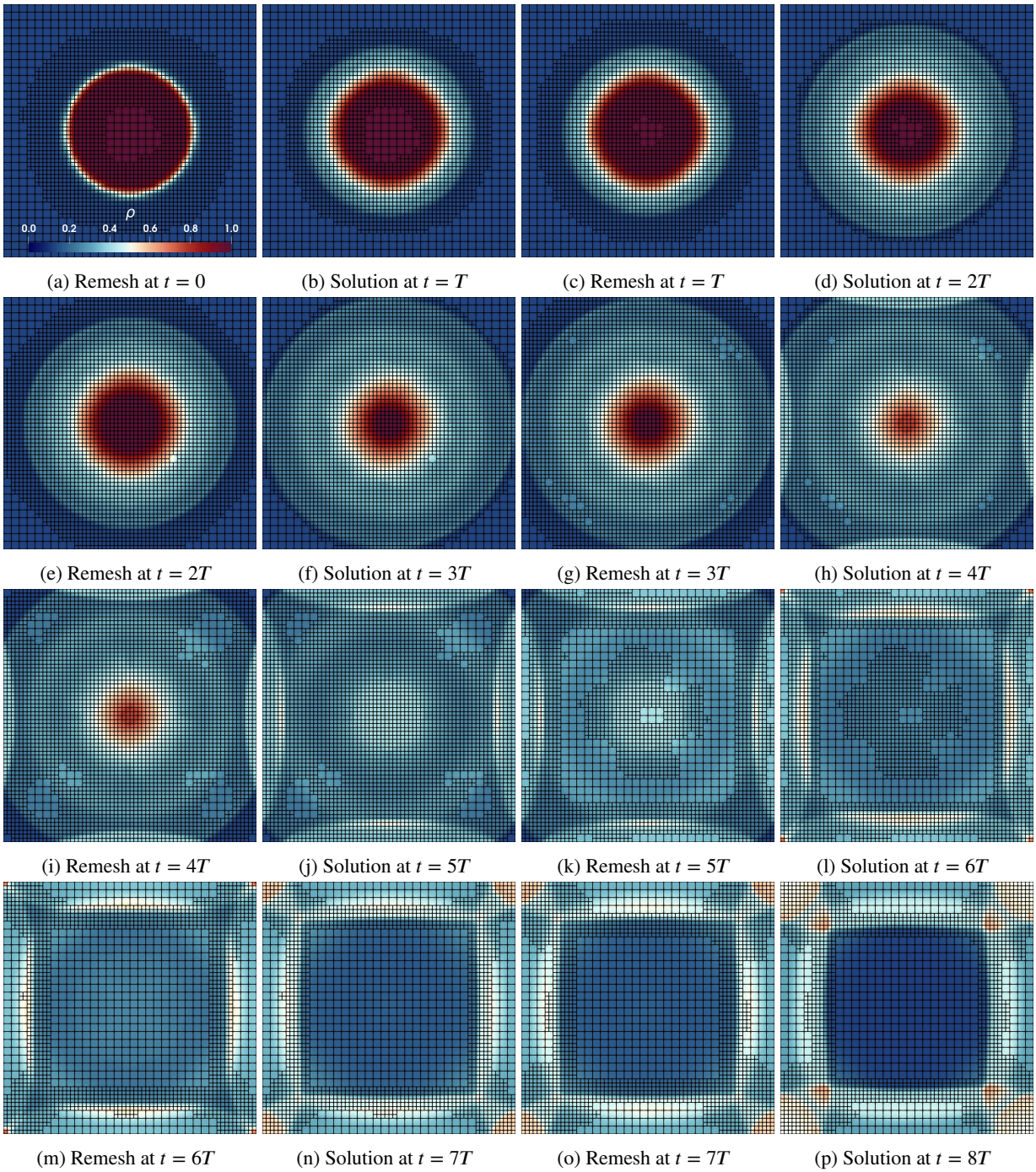
efficiency marginally decreased, likely as a result of the chaotic nature of the flow at later times for which the policies would tend towards more refinement. For the optimal threshold policy, this resulted in a mean efficiency decrease from 0.251 to 0.226, whereas for the DynAMO policy, this resulted in a mean efficiency decrease from 0.486 to 0.423. However, the DynAMO policy still showed a large relative efficiency increase over the threshold policy, 87.2%, which is on par with the *in-distribution* results. These results indicate that the proposed approach can effectively generalize to longer simulation times and, by extent, flow physics which were not encountered during the training process.

As a final validation of the DynAMO approach, we apply the DynAMO policy trained on two-dimensional Riemann problems to a particular *out-of-distribution* problem: the two-dimensional radial Sod shock tube. This case is a more complex form of the quintessential example of the shock tube which exhibits the three main features of the Riemann problem, namely shock waves, contact discontinuities, and rarefaction waves. Furthermore, in its periodic form, it also exhibits shock-shock and shock-contact interactions at longer simulation times. The problem is solved on the domain $\Omega = [-0.5, 0.5]^2$, and the initial conditions are given as

$$[\rho, u, v, P]^T = \begin{cases} [\rho_l, 0, 0, P_l]^T, & \text{if } \sqrt{x^2 + y^2} \leq 0.25, \\ [\rho_r, 0, 0, P_r]^T, & \text{else,} \end{cases} \tag{29}$$

where $\rho_l = 1.0$, $P_l = 1.0$, $\rho_r = 0.125$, and $P_r = 0.1$. The base mesh resolution was set as $N = 32^2$ and the remesh time was set as $T = 0.05$. The evolution of the density and h-adapted mesh as obtained by the DynAMO policy is shown in Fig. 25 over 8 remesh intervals. At the initial time, the DynAMO policy preemptively refines the region both inside and outside of the initial discontinuity. This refinement region covers the propagation of the out-running shock front and the in-running expansion wave front, although with a slight overprediction in the refinement region necessary to cover the propagation distance of the shock. Furthermore, the interior region of the mesh was correctly left de-refined as the expansion wave front did not reach this region over the remesh interval, although with some minor asymmetry in the refinement patterns. Similar behavior was observed at the next few remesh intervals, with the DynAMO policy accurately predicting the preemptive refinement decisions necessary to account for the evolution of the shock front. As the shock front propagated towards the domain boundaries, certain spurious de-refinement decisions were chosen by the policy which were not consistent with the decisions at other similar locations in mesh. However, these decisions primarily acted to introduce some minor asymmetry in the refinement patterns without appreciably degrading the accuracy of the approach. After the shock fronts propagated through the periodic boundaries, the shock-shock and shock-contact interactions were correctly refined by the DynAMO policy. Furthermore, the interior region of the mesh, where the solution variation was low, was appropriately de-refined, such that the approach effectively balanced computational cost and accuracy. These results further showcase the ability of the proposed approach to be an effective method of performing anticipatory mesh refinement for complex nonlinear flows and generalizing to flow physics not encountered during training.

The two-dimensional Sod shock tube problem was also used to evaluate the computational cost of the proposed approach. The cost, shown in absolute CPU wall clock time, was computed for the problem across four major categories: 1) advancing the unsteady solver between remesh intervals; 2) computing the observation (i.e., evaluating the error estimates, computing the observable quantities and metrics, etc.); 3) performing inference on the neural network; and 4) adapting the mesh and restructuring data. The absolute and relative computational costs of these four categories are shown in Fig. 26. It can be seen that the unsteady solver was the predominant factor in the overall computational cost, accounting for 93.1% of the total compute time. In contrast, the cost of the proposed approach was only 6.6% of the overall compute time, with the formation of the observation and network inference contributing approximately equally to the cost, and the cost of the mesh refinement process was essentially negligible. These results indicate that even with an RL framework that was not necessarily tailored for peak computational efficiency, the computational cost of the proposed approach did not significantly increase the overall cost of the solver for a complex system of equations.



(a) Remesh at $t = 0$ (b) Solution at $t = T$ (c) Remesh at $t = T$ (d) Solution at $t = 2T$

(e) Remesh at $t = 2T$ (f) Solution at $t = 3T$ (g) Remesh at $t = 3T$ (h) Solution at $t = 4T$

(i) Remesh at $t = 4T$ (j) Solution at $t = 5T$ (k) Remesh at $t = 5T$ (l) Solution at $t = 6T$

(m) Remesh at $t = 6T$ (n) Solution at $t = 7T$ (o) Remesh at $t = 7T$ (p) Solution at $t = 8T$

Figure 25: Contours of density overlaid with h-adapted mesh at varying remesh intervals using DynAMO for the two-dimensional Sod shock tube problem.

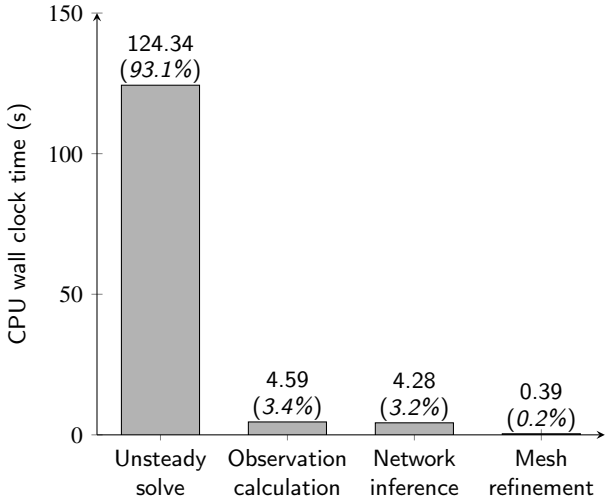


Figure 26: Computational cost distribution for the two-dimensional Sod shock tube problem in terms of unsteady solve, observation calculation, network inference, and mesh refinement. Labels represent absolute wall clock time (CPU-seconds) with relative cost shown in parentheses.

6. Conclusions

In this work, we presented DynAMO, a novel reinforcement learning paradigm for guiding anticipatory mesh refinement strategies for complex time-dependent PDEs. In comparison to standard adaptive mesh refinement approaches for time-dependent problems, which rely on instantaneous error indicators that typically cannot account for the spatio-temporal evolution of the error, the goal of the proposed approach is to predict future error propagation and preemptively refine the mesh to achieve superior accuracy and efficiency. To this end, we considered a multi-agent reinforcement learning approach to a decentralized partially observable Markov decision process model of mesh optimization, with mesh elements corresponding to independent agents that observe local surrounding information. To maximize the generalizability and robustness of the approach, we introduced novel observation and reward functions that utilized a user-adjustable error estimate normalization and a non-dimensionalized measure of information propagation. These proposed formulations enabled the extension of the approach to arbitrary nonlinear hyperbolic conservation laws, ensured that the approach was invariant to problem scale, mesh resolution, simulation time, and remeshing time interval, and allowed for the user to control error/cost targets at evaluation time.

The DynAMO approach was applied to both h- and p-refinement for discontinuous Galerkin approximations of the linear advection and compressible Euler equations. We showed that due to its anticipatory refinement capability, the proposed approach could achieve significantly higher efficiency than conventional AMR approaches, such as threshold-based methods, yielding more accurate simulations at lower computational costs. Furthermore, the use of DynAMO allowed for longer remesh intervals without sacrificing accuracy by producing meshes that were able to preemptively account for the underlying dynamics of the system. These benefits were observed over a wide variety of problems, ranging from simple linear transport to complex nonlinear shock interactions, including problems and numerical setups on which the policies were not trained. These results indicate that the proposed approach can effectively anticipate the spatiotemporal evolution of the error for complex nonlinear PDEs, generalize to unseen problems at evaluation time, and robustly extend to different meshes, simulation times, and remesh time intervals.

Future developments for this approach will focus on mitigating the limitations present in its current form. In particular, this work showcases the method for one-level refinement on periodic meshes using a structured observation window. To extend this approach to higher refinement levels, problems with boundary conditions, and arbitrary unstructured meshes, the formulation can be modified to utilize a graph neural network as explored in the works of Yang et al. [29] and Foucart et al. [31]. In fact, the proposed observation, which utilizes a non-dimensionalized measure of information propagation that is dependent on the mesh displacement vector, is highly amenable to graph-type implementations. Additionally, further work can be performed on exploring more complex mesh refinement techniques such as hp-refinement, for which optimal strategies are far from fully understood.

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE–AC52–07NA27344 and the LLNL-LDRD Program under Project tracking No. 21–SI–001. Release number LLNL–JRNL–855285.

References

[1] Tomohiro Takaki, Toshimichi Fukuoka, and Yoshihiro Tomita. Phase-field simulation during directional solidification of a binary alloy using adaptive finite element method. *Journal of Crystal Growth*, 283(1):263–278, 2005. doi: https://doi.org/10.1016/j.jcrysgro.2005.05.064.

[2] Lorenz Berger, Rafel Bordas, David Kay, and Simon Tavener. A stabilized finite element method for finite-strain three-field poroelasticity. *Computational Mechanics*, 60(1):51–68, 2017. doi: 10.1007/s00466-017-1381-8.

[3] Sarvesh Kumar, Ricardo Oyarzúa, Ricardo Ruiz-Baier, and Ruchi Sandilya. Conservative discontinuous finite volume and mixed schemes for a new four-field formulation in poroelasticity. *ESAIM: Mathematical Modelling and Numerical Analysis*, 54(1):273–299, 2020.

[4] Daniel Kitzmann, Jan Bolte, and A Beate C Patzer. Discontinuous Galerkin finite element methods for radiative transfer in spherical symmetry. *Astronomy & Astrophysics*, 595:A90, 2016.

[5] FN Van de Vosse, J De Hart, CHGA Van Oijen, D Bessems, TWM Gunther, A Segal, BJBM Wolters, JMA Stijnen, and FPT Baaijens. Finite-element-based computational methods for cardiovascular fluid-structure interaction. *Journal of engineering mathematics*, 47:335–368, 2003.

[6] Roland Becker and Rolf Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica*, 10:1–102, May 2001. doi: 10.1017/s0962492901000010.

[7] William C. Tyson, Katarzyna Swirydowicz, Joseph M. Derlaga, Christopher J. Roy, and Eric de Sturler. Improved functional-based error estimation and adaptation without adjoints. In *46th AIAA Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, June 2016. doi: 10.2514/6.2016-3809.

[8] William C. Tyson and Christopher J. Roy. A higher-order error estimation framework for finite-volume cfd. *Journal of Computational Physics*, 394:632–657, October 2019. doi: 10.1016/j.jcp.2019.06.017.

[9] William C. Tyson, Gary K. Yan, Christopher J. Roy, and Carl F. Ollivier-Gooch. Relinearization of the error transport equations for arbitrarily high-order error estimates. *Journal of Computational Physics*, 397:108867, November 2019. doi: 10.1016/j.jcp.2019.108867.

[10] Hongyu Wang and Christopher J. Roy. Error transport equations implementation for discontinuous Galerkin methods. *Journal of Computational Physics*, 474:111760, February 2023. doi: 10.1016/j.jcp.2022.111760.

[11] Azam Moosavi, Răzvan Ştefănescu, and Adrian Sandu. Multivariate predictions of local reduced-order-model errors and dimensions. *International Journal for Numerical Methods in Engineering*, 113(3):512–533, October 2017. doi: 10.1002/nme.5624.

[12] Martin Drohmann and Kevin Carlberg. The ROMES method for statistical modeling of reduced-order-model error. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):116–145, January 2015. doi: 10.1137/140969841.

[13] DN Dyck, DA Lowther, and S McFee. Determining an approximate finite element mesh density using neural network techniques. *IEEE transactions on magnetics*, 28(2):1767–1770, 1992.

[14] Maciej Paszyński, Rafał Grzeszczuk, David Pardo, and Leszek Demkowicz. Deep learning driven self-adaptive *hp* finite element method. In *International Conference on Computational Science*, pages 114–121. Springer, 2021.

[15] Tomasz Służalec, Rafał Grzeszczuk, Sergio Rojas, Witold Dzwinel, and Maciej Paszyński. Quasi-optimal hp-finite element refinements towards singularities via deep neural network prediction. *Computers & Mathematics with Applications*, 142:157–174, July 2023. doi: 10.1016/j.camwa.2023.04.023.

[16] Andrew Gillette, Brendan Keith, and Socratis Petrides. Learning robust marking policies for adaptive mesh refinement. *SIAM Journal on Scientific Computing (to appear)*, 2023.

[17] Jan Bohn and Michael Feischl. Recurrent neural networks as optimal mesh refinement strategies. *Computers & Mathematics with Applications*, 97:61–76, 2021.

[18] Guodong Chen and Krzysztof Fidkowski. Output-based error estimation and mesh adaptation using convolutional neural networks: Application to a scalar advection-diffusion problem. In *AIAA Scitech 2020 Forum*, page 1143, 2020.

[19] Guodong Chen and Krzysztof J Fidkowski. Output-based adaptive aerodynamic simulations using convolutional neural networks. *Computers & Fluids*, 223:104947, 2021.

[20] Ayan Chakraborty, Thomas Wick, Xiaoying Zhuang, and Timon Rabczuk. Multigoal-oriented dual-weighted-residual error estimation using deep neural networks, 2021.

[21] Julian Roth, Max Schröder, and Thomas Wick. Neural network guided adjoint computations in dual weighted residual error estimation. *SN Applied Sciences*, 4(2):1–17, 2022.

[22] R Chedid and N Najjar. Automatic finite-element mesh generation using artificial neural networks-part i: Prediction of mesh density. *IEEE Transactions on Magnetics*, 32(5):5173–5178, 1996.

[23] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

[24] Keefe Huang, Moritz Krügener, Alistair Brown, Friedrich Menhorn, Hans-Joachim Bungartz, and Dirk Hartmann. Machine learning-based optimal mesh generation in computational fluid dynamics. *arXiv preprint arXiv:2102.12923*, 2021.

[25] Zheyan Zhang, Yongxing Wang, Peter K Jimack, and He Wang. Meshingnet: A new mesh generation method based on deep learning. In *International Conference on Computational Science*, pages 186–198. Springer, 2020.

[26] Wenbin Song, Mingrui Zhang, Joseph G Wallwork, Junpeng Gao, Zheng Tian, Fanglei Sun, Matthew D Piggott, Junqing Chen, Zuoqiang Shi, Xiang Chen, et al. M2N: Mesh movement networks for PDE solvers. *arXiv preprint arXiv:2204.11188*, 2022.

[27] Chiu Ling Chan, Felix Scholz, and Thomas Takacs. Locally refined quad meshing for linear elasticity problems based on convolutional neural networks. *arXiv preprint arXiv:2203.07843*, 2022.

[28] Jiachen Yang, Tarik Dzanic, Brenden Petersen, Jun Kudo, Ketan Mittal, Vladimir Tomov, Jean-Sylvain Camier, Tuo Zhao, Hongyuan Zha, Tzanio Kolev, Robert Anderson, and Daniel Faissol. Reinforcement learning for adaptive mesh refinement. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 5997–6014. PMLR, 25–27 Apr 2023.

[29] Jiachen Yang, Ketan Mittal, Tarik Dzanic, Socratis Petrides, Brendan Keith, Brenden Petersen, Daniel Faissol, and Robert Anderson. Multi-agent reinforcement learning for adaptive mesh refinement. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, page 14–22, 2023.

[30] Niklas Freymuth, Philipp Dahlinger, Tobias Würth, Luise Kärger, and Gerhard Neumann. Swarm reinforcement learning for adaptive mesh refinement, 2023.

[31] Corbin Foucart, Aaron Charous, and Pierre F.J. Lermusiaux. Deep reinforcement learning for adaptive mesh refinement. *Journal of Computational Physics*, 491:112381, 2023.

[32] Jan S. Hesthaven and Tim Warburton. *Nodal Discontinuous Galerkin Methods*. Springer New York, 2008. doi: 10.1007/978-0-387-72067-8.

[33] V.V Rusanov. The calculation of the interaction of non-stationary shock waves and obstacles. *USSR Computational Mathematics and Mathematical Physics*, 1(2):304–320, January 1962. doi: 10.1016/0041-5553(62)90062-9.

[34] S. F. Davis. Simplified second-order Godunov-type methods. *SIAM Journal on Scientific and Statistical Computing*, 9(3):445–473, May 1988. doi: 10.1137/0909030.

[35] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[36] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[37] Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[39] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on*

machine learning, pages 330–337, 1993.

[40] Yu-Han Chang, Tracey Ho, and Leslie P Kaelbling. All learning is local: Multi-agent learning in global reward games. In *Advances in neural information processing systems*, pages 807–814, 2004.

[41] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. Cerveny V. Dobrev, Y. Dudouit, A. Fisher, Tz. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, and S. Zampini. MFEM: A modular finite element methods library. *Computers & Mathematics with Applications*, 81:42–74, 2021. doi: 10.1016/j.camwa.2020.06.009.

[42] pymfem. PyMFEM: Modular Finite Element Methods [Software]. https://github.com/mfem/PyMFEM/, 2022.

[43] Bernardo Cockburn and Chi-Wang Shu. Runge–Kutta discontinuous Galerkin methods for convection-dominated problems. *Journal of Scientific Computing*, 16(3):173–261, 2001. doi: 10.1023/a:1012873910884.

[44] Timothy Barth and Dennis Jespersen. The design and application of upwind schemes on unstructured meshes. In *27th Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, January 1989. doi: 10.2514/6.1989-366.

[45] Carsten Carstensen, Michael Feischl, Marcus Page, and Dirk Praetorius. Axioms of adaptivity. *Computers & Mathematics with Applications*, 67(6):1195–1253, 2014.

[46] Olgierd Cecil Zienkiewicz and Jian Zhong Zhu. The superconvergent patch recovery and a posteriori error estimates. Part 1: The recovery technique. *International Journal for Numerical Methods in Engineering*, 33(7):1331–1364, 1992.

[47] Olgierd Cecil Zienkiewicz and Jian Zhong Zhu. The superconvergent patch recovery and a posteriori error estimates. Part 2: Error estimates and adaptivity. *International Journal for Numerical Methods in Engineering*, 33(7):1365–1382, 1992.

[48] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062. PMLR, 2018.

[49] Carsten W. Schulz-Rinne, James P. Collins, and Harland M. Glaz. Numerical solution of the Riemann problem for two-dimensional gas dynamics. *SIAM Journal on Scientific Computing*, 14(6):1394–1414, November 1993. doi: 10.1137/0914082.

[50] Richard Liska and Burton Wendroff. Comparison of several difference schemes on 1D and 2D test problems for the Euler equations. *SIAM Journal on Scientific Computing*, 25(3):995–1017, January 2003. doi: 10.1137/s1064827502402120.

[51] Peter D. Lax and Xu-Dong Liu. Solution of two-dimensional Riemann problems of gas dynamics by positive schemes. *SIAM Journal on Scientific Computing*, 19(2):319–340, January 1998. doi: 10.1137/s1064827595291819.

A. Euler flux Jacobian

Recall the solution and flux of the compressible Euler equations,

$$\mathbf{u} = \begin{bmatrix} \rho \\ \mathbf{m} \\ E \end{bmatrix} \quad \text{and} \quad \mathbf{F} = \begin{bmatrix} \mathbf{m}^T \\ \mathbf{m} \otimes \mathbf{v} + P\mathbf{I} \\ (E+P)\mathbf{v}^T \end{bmatrix} = \begin{bmatrix} \mathbf{m}^T \\ \mathbf{m} \otimes \mathbf{m}/\rho + P\mathbf{I} \\ \mathbf{m}^T(E+P)/\rho \end{bmatrix},$$

where $P = (\gamma - 1)(E - \frac{1}{2}\|\mathbf{m}\|^2/\rho)$. The following relation is convenient to derive the flux Jacobian

$$\frac{\partial P}{\partial \mathbf{u}} = (\gamma - 1) \begin{bmatrix} \frac{1}{2\rho^2}\|\mathbf{m}\|^2 \\ -\frac{1}{\rho}\mathbf{m} \\ 1 \end{bmatrix}.$$

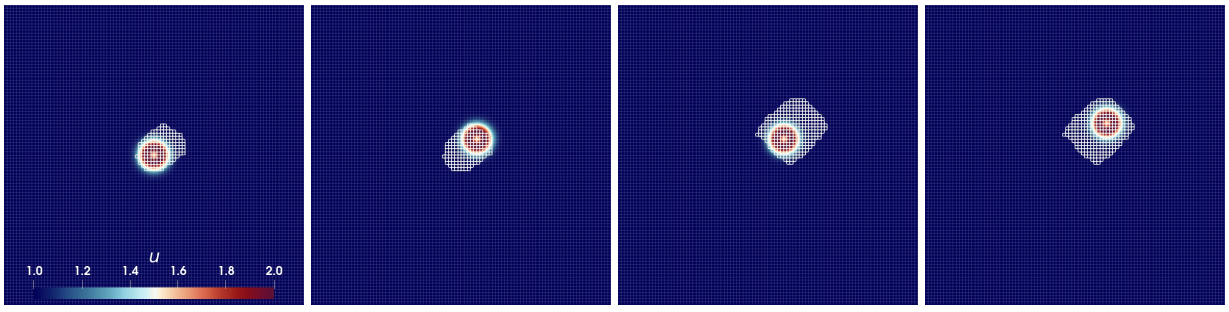
Let $\mathbf{F}_i$ be the i-th column of $\mathbf{F}(u)$. We can then calculate the flux Jacobian as

$$\frac{\partial \mathbf{F}_i}{\partial \mathbf{u}} = \begin{bmatrix} 0 & \mathbf{d}_i^T & 0 \\ -\frac{m_i}{\rho^2}\mathbf{m} + \frac{1}{2\rho^2}\|\mathbf{m}\|^2\mathbf{d}_i & \frac{1}{\rho}(m_i\mathbf{I} + \mathbf{m} \otimes \mathbf{d}_i) - (\gamma-1)\frac{1}{\rho}\mathbf{d}_i \otimes \mathbf{m} & (\gamma-1)\mathbf{d}_i \\ -\frac{m_i}{\rho^2}(E+P) + (\gamma-1)\frac{m_i}{2\rho^3}\|\mathbf{m}\|^2 & \frac{1}{\rho}(E+P)\mathbf{d}_i^T - (\gamma-1)\frac{m_i}{\rho^2}\mathbf{m}^T & \gamma\frac{m_i}{\rho} \end{bmatrix},$$

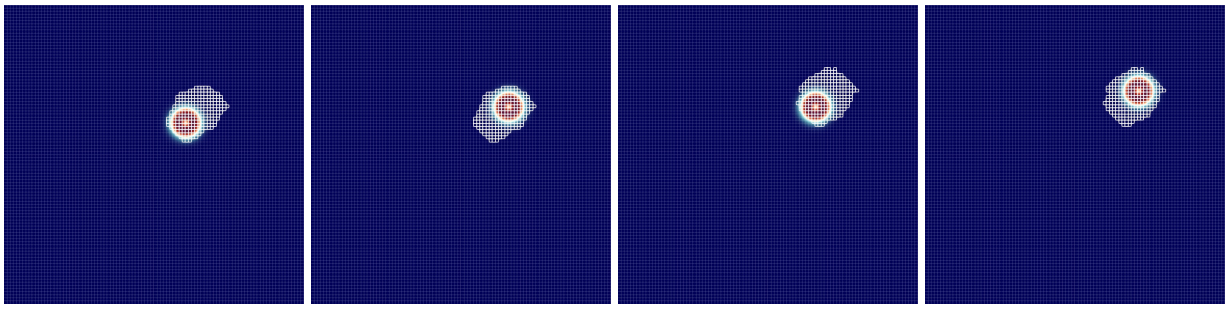
where $\mathbf{d}_i$ is the standard basis in $\mathbb{R}^d$ with $(\mathbf{d}_i)_j = \delta_{ij}$ and m_i is the i-th component of the momentum $\mathbf{m}$.

B. Sample generalization experiments

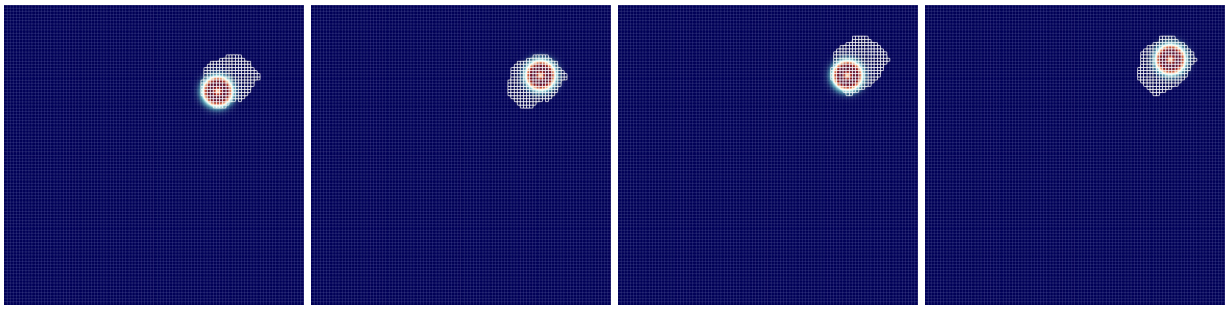
Some mesh refinement examples from the generalization experiments are presented in this section. Fig. 27 showcases the adapted meshes for p-refinement on the advection equation. This setup utilizes a finer mesh (with 16 times more elements) and a longer simulation time. Fig. 28 shows the adapted meshes for h-refinement on the advection equation, with a finer mesh (with 8 times more elements), different solution profiles (multiple Gaussian bumps of varying width and height), and a longer simulation time.



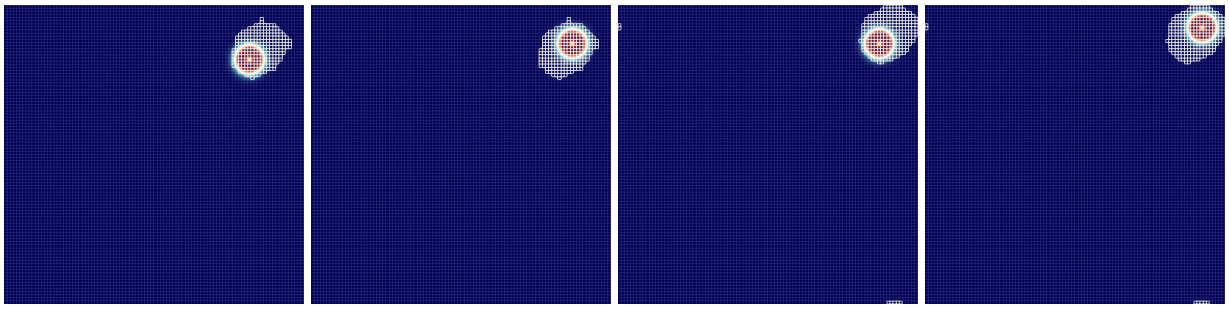
(a) Remesh at $t = 0$ (b) Solution at $t = T$ (c) Remesh at $t = T$ (d) Solution at $t = 2T$



(e) Remesh at $t = 2T$ (f) Solution at $t = 3T$ (g) Remesh, $3T$ (h) Solution, $4T$

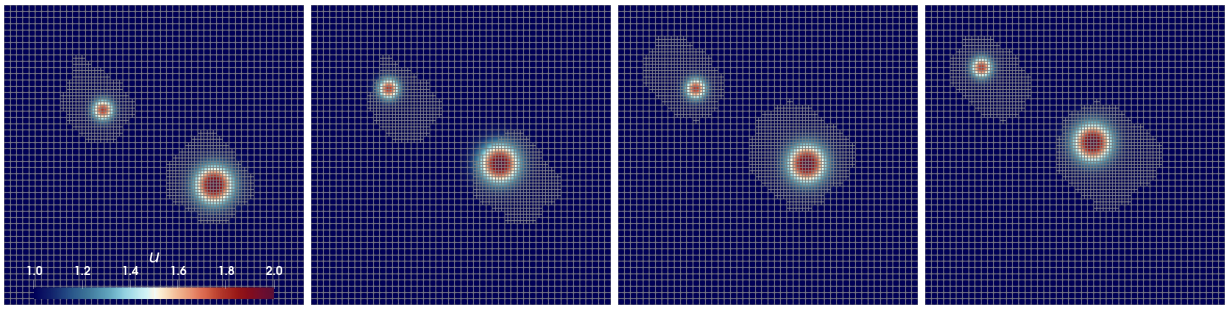


(i) Remesh, $4T$ (j) Solution, $5T$ (k) Remesh, $5T$ (l) Solution, $6T$

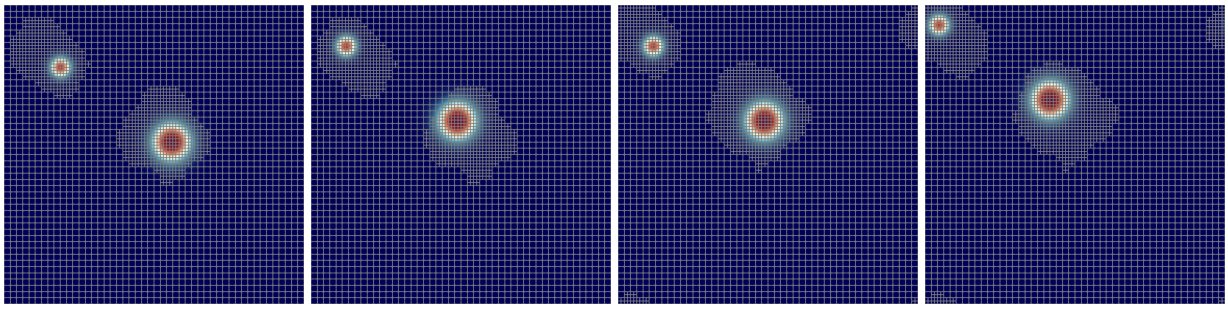


(m) Remesh, $6T$ (n) Solution, $7T$ (o) Remesh, $7T$ (p) Solution, $8T$

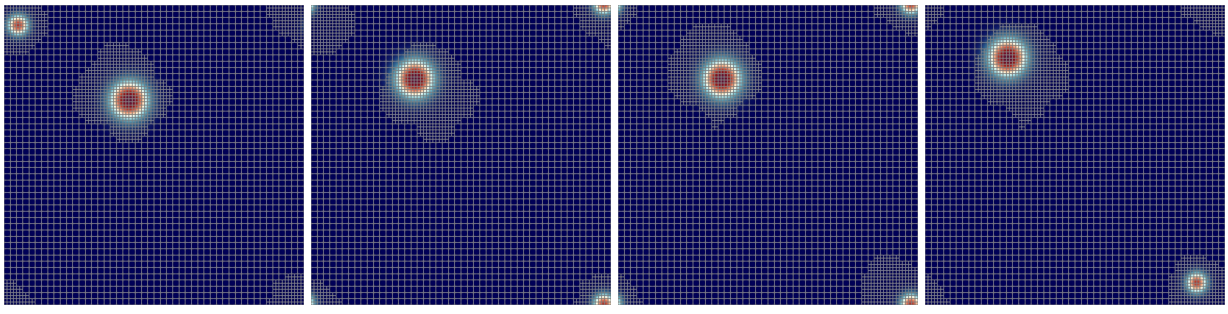
Figure 27: Solution contours overlaid with p-adapted mesh at varying remesh intervals for a generalization experiment on the advection equation. Highlighted elements represent p-refinement.



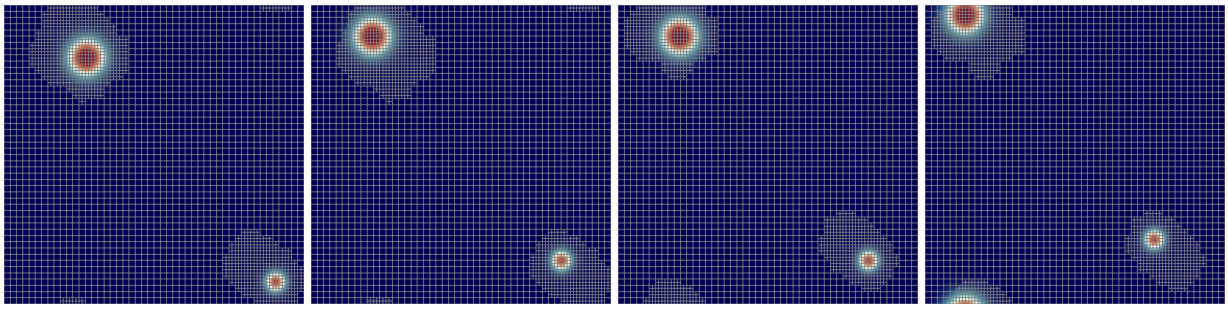
(a) Remesh at $t = 0$ (b) Solution at $t = T$ (c) Remesh at $t = T$ (d) Solution at $t = 2T$



(e) Remesh at $t = 2T$ (f) Solution at $t = 3T$ (g) Remesh, $3T$ (h) Solution, $4T$



(i) Remesh, $4T$ (j) Solution, $5T$ (k) Remesh, $5T$ (l) Solution, $6T$



(m) Remesh, $6T$ (n) Solution, $7T$ (o) Remesh, $7T$ (p) Solution, $8T$

Figure 28: Solution contours overlaid with h-adapted mesh at varying remesh intervals for a generalization experiment on the advection equation.