SWARM REINFORCEMENT LEARNING FOR ADAPTIVE MESH REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Adaptive Mesh Refinement (AMR) is crucial for mesh-based simulations, as it allows for dynamically adjusting the resolution of a mesh to trade off computational cost with the simulation accuracy. Yet, existing methods for AMR either use task-dependent heuristics, expensive error estimators, or do not scale well to larger meshes or more complex problems. In this paper, we formalize AMR as a Swarm Reinforcement Learning problem, viewing each element of a mesh as part of a collaborative system of simple and homogeneous agents. We combine this problem formulation with a novel agent-wise reward function and Graph Neural Networks, allowing us to learn reliable and scalable refinement strategies on arbitrary systems of equations. We experimentally demonstrate the effectiveness of our approach in improving the accuracy and efficiency of complex simulations. Our results show that we outperform learned baselines and achieve a refinement quality that is on par with a traditional error-based AMR refinement strategy without requiring error indicators during inference.

1 INTRODUCTION

Adaptive Mesh Refinement (AMR) is a powerful technique for complex Finite Element Method (FEM)-based simulations (Reddy, 2019; Sabat & Kundu, 2021) that allows to dynamically allocate resources to relevant parts of the simulated domain to trade off computational speed and simulation accuracy (Plewa et al., 2005; Anderson et al., 2021). Applications of AMR include fluid dynamics (Berger & Colella, 1989; Baker, 1997; Borker et al., 2019), structural mechanics (Ortiz & Quigley Iv, 1991; Provatas et al., 1998; Stein, 2007; Gibert et al., 2019) and astrophysics (Cunningham et al., 2009; Bryan et al., 2014; Guillet et al., 2019). Yet, classical approaches for AMR usually rely on problem-dependent heuristics or error indicators (Mukherjee, 1996; Arnold et al., 2000; Kita & Kamiya, 2001; Bangerth & Rannacher, 2003; Cerveny et al., 2019).

Here, we instead formalize AMR as a Reinforcement Learning (RL) (Sutton & Barto, 2018) problem. Following previous work (Yang et al., 2021; Foucart et al., 2022; Yang et al., 2022) in this direction, we encode the state of the current simulation as local observations that we feed to RL agents. The chosen actions then determine which elements of a mesh to refine. While these existing approaches are trained on refinements of individual elements (Yang et al., 2021; Foucart et al., 2022) or make use of a learned value decomposition of parallel refinements (Yang et al., 2022), we instead view the problem through the lens of Swarm Reinforcement Learning (Šošić et al., 2017; Hüttenrauch et al., 2019) to efficiently train reliable and scalable refinement strategies. For this, our approach uses a novel spatial reward formulation that assigns each element of the mesh an agent and each agent its own reward throughout an episode. We combine this reward with Message Passing Networks (MPNs) (Sanchez-Gonzalez et al., 2020), a class of Graph Neural Networks (GNNs) (Scarselli et al., 2009; Battaglia et al., 2018; Wu et al., 2020) that has proven highly effective for physical simulations (Pfaff et al., 2021; Linkerhägner et al., 2023). The resulting method, Adaptive Swarm Mesh Refinement (ASMR), produces highly refined meshes with thousands of elements while being applicable to arbitrary Partial Differential Equations (PDEs). Experimentally, we show the effectiveness of ASMR on challenging elliptic PDEs, outperforming strong learned baselines (Yang et al., 2021; 2022). Further, our results are on par with a traditional error-based AMR heuristic without requiring expensive error indicators during inference. We additionally conduct a series of ablations to show which parts of the approach make it uniquely effective.

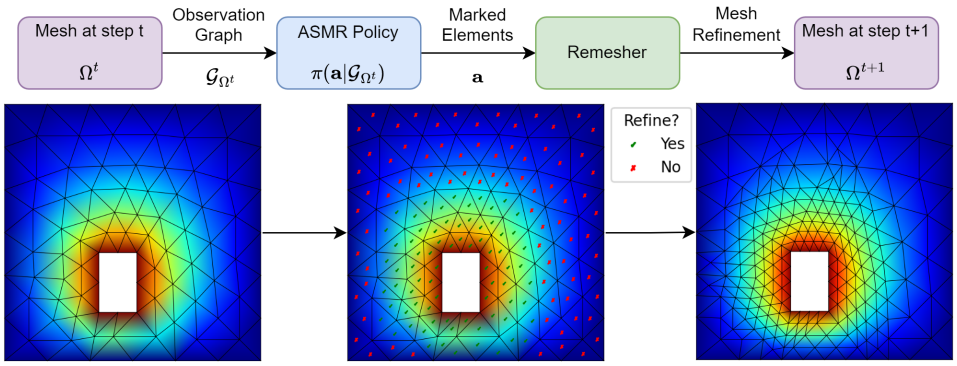


Figure 1: Overview of Adaptive Swarm Mesh Refinement (ASMR). Given a mesh Ω_t, the observation graph $\mathcal{G}_{\Omega^t}$ encodes the elements as graph nodes, their neighborhood as edges, and general information as global features. The graph is given to a learned policy, which marks mesh elements for refinement. A remesher refines these elements, resulting in a refined mesh.

2 Adaptive Swarm Mesh Refinement

Message Passing Networks. Let $\mathcal{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X_V}, \mathbf{X_E}, \mathbf{g})$ be a directed graph with nodes $\mathbf{V}$ and edges $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$. Then $\mathbf{X_V} : \mathbf{V} \to \mathbb{R}^{d_\mathbf{V}}$ and $\mathbf{X_E} : \mathbf{E} \to \mathbb{R}^{d_\mathbf{E}}$ are node and edge features of dimensions $d_\mathbf{V}$ and $d_\mathbf{E}$ respectively, and $\mathbf{g} \in \mathbb{R}^{d_\mathbf{g}}$ are global features. A MPN (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021; Linkerhägner et al., 2023) is a GNN consisting of L *Message Passing Steps*. Each step l receives the output of the previous step and updates the features for all nodes $v \in \mathbf{V}$, edges $e \in \mathbf{E}$, and globals $\mathbf{g}$. Using linear embeddings $\mathbf{x}_v^0$, $\mathbf{x}_e^0$, and $\mathbf{g}^0$ of the initial node, edge, and global features, the step is given as

$$\mathbf{x}_e^{l+1} = f_\mathbf{E}^l(\mathbf{x}_v^l, \mathbf{x}_u^l, \mathbf{x}_e^l, \mathbf{g}^l), \text{ with } e = (u, v),$$
$$\mathbf{x}_v^{l+1} = f_\mathbf{V}^l(\mathbf{x}_v^l, \bigoplus_{e=(v,u)\in\mathbf{E}} \mathbf{x}_e^{l+1}, \mathbf{g}^l), \quad \text{and} \quad \mathbf{g}^{l+1} = f_\mathbf{g}^l(\bigoplus_{v\in\mathbf{V}} \mathbf{x}_v^{l+1}, \bigoplus_{e\in\mathbf{E}} \mathbf{x}_e^{l+1}, \mathbf{g}^l).$$

The operator $\oplus$ is a permutation-invariant aggregation such as a sum, max, or mean operator. Each $f_\cdot^l$ is a learned function that we generally parameterize as a simple Multilayer Perceptron (MLP). The network's final output is a learned representation $\mathbf{x}_v^L$ for each node $v \in \mathbf{V}$. Appendix A.1 provides a schematic overview of the MPN architecture.

Swarm Markov Decision Process We frame AMR as an instance of a Swarm Swarm Markov Decision Process (SwarmDP) (Šošić et al., 2017; Hüttenrauch et al., 2019), which can be seen as a special case of a decentralized partially observable Markov Decision Process (MDP) for swarm systems. For this, we adapt the process to per-agent rewards, a shared observation structure, and varying numbers of agents. Details can be found in Appendix A.2.

Agents and Actions. Given a mesh Ω^t for a domain Ω, we view each element $\Omega_i^t \subseteq \Omega^t$ as an agent. The action space per element consists of a binary decision of whether it should be marked for refinement. A given remesher refines all marked elements i as $\Omega_i^t = \bigcup_j \Omega_{i,j}^{t+1}$, resulting in a finer mesh $\Omega^{t+1} = \bigcup_j \Omega_j^{t+1}$. Note that the remesher may also refine unmarked elements to assert a conforming solution (Arnold et al., 2000).

Observations. We encode an *observation graph* $\mathcal{G}_{\Omega^t}$ by creating a node for each mesh element and (directed) edges between all pairs of neighboring elements. The node, edge, and global features of the resulting graph depend on the considered system of equations. However, we encode positions on the mesh only as relative Euclidean distances in the edge features $\mathbf{X_E}$ to ensure that our observations are equivariant under the Euclidean group (Bronstein et al., 2021; Pfaff et al., 2021).

Reward. A good refinement strategy trades off the accuracy of the mesh Ω^t with its total number of elements $\Omega_i^t \subseteq \Omega^t$. Let $\tilde{\text{err}}(\Omega_i^t)$ be an error estimate of element Ω_i^t, $\text{Area}(\Omega_i^t)$ its total area and $\delta^t(\Omega_i^t) = \{\Omega_j^{t+1} \subseteq \Omega_i^t\}$ the mapping that assigns the element to all elements Ω_j^{t+1} that it refines into. We then define a *reward per element* as

$$R(\Omega_i^t) = \frac{1}{\text{Area}(\Omega_i^t)} \left(\tilde{\text{err}}(\Omega_i^t) - \sum_{\Omega_j^{t+1} \in \delta^t(\Omega_i^t)} \tilde{\text{err}}(\Omega_j^{t+1}) \right) - \alpha \left(|\delta^t(\Omega_i^t)| - 1 \right), \tag{1}$$

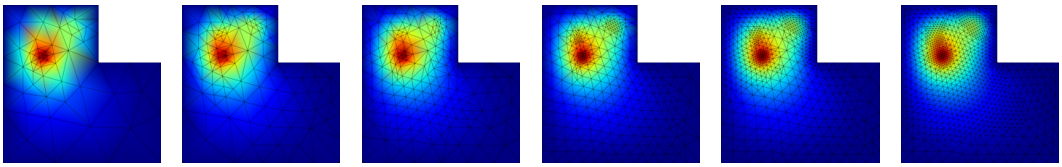


Figure 2: Qualitative results for ASMR on Poisson's Equation on an L-shaped domain. Ranging from high (left) to low (right) element penalties α, our approach produces highly accurate refinements for different numbers of mesh elements. In all cases, the policy focuses on and refines the relevant parts of the underlying system of equations. We list more examples in Appendix E.

where α is a hyperparameter that acts as a penalty for adding new elements. Note that the reference mesh Ω^* is only needed during training and that the trained policy does not depend on it when refining a new mesh during inference. To track the influence of the agents even though they may appear and disappear throughout an episode, we can iteratively assign the mapping $\delta^t(\Omega_i^t)$ for k time steps as $\delta_k^t(\Omega_i^t) = \{\Omega_j^{t+k} \subseteq \Omega_i^t\}$. This mapping allows us to compute a return per element

$$J_i^t = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k \sum_{\Omega_j^{t+k} \in \delta_k^t(\Omega_i^t)} R(\Omega_j^{t+k}) \right].$$

We provide additional details on the construction of the error estimate $\tilde{\text{err}}(\Omega_i^t)$, as well as an intuition on the reward in Appendix A.3.

Adaptive Swarm Mesh Refinement. Our approach trains a policy $\pi(\mathbf{a}|\mathcal{G}_{\Omega^t})$ that computes a joint action vector $\mathbf{a} \in \mathcal{A}^{|\{\Omega_i^t \in \Omega^t\}|}$ for all mesh elements Ω_i^t by passing messages along the observation graph $\mathcal{G}$. Figure 1 gives a schematic overview.

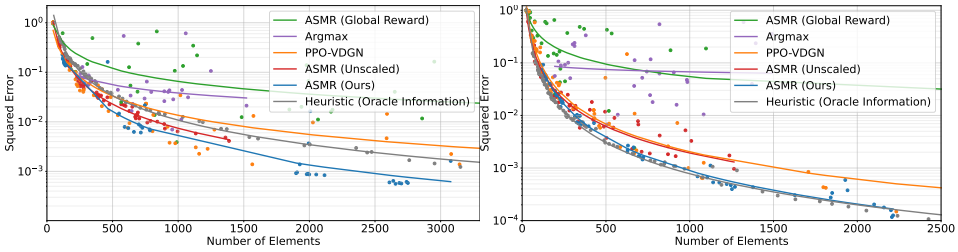
3 Experiments

Systems of Equations and Graph Features. We experiment on two families of elliptical PDEs, namely Laplace's Equation on domains with an inner and an outer boundary with different boundary conditions and Poisson's Equation on L-shaped domains for Gaussian Mixture Model load functions. Both equations are ubiquitous in physics (Strauss, 2007), with applications including heat flow (Zill, 2016), electrodynamics (Griffiths, 2005) and astrophysics (Misner et al., 1973). We refer to Appendix B for further details.

For each node $v \in \mathbf{V}$, we use the area of the corresponding element, the distance of the element midpoint to the closest boundary, and the mean and standard deviation of the solution on the element's vertices as node features $\mathbf{X}_v$. The edge features consist of euclidean distances between the element midpoints, and the global features $\mathbf{g}$ contain the total number of mesh elements, the total number of mesh vertices, and the current environment step. For Laplace's Equation, we add the closest distance to the inner boundary as a node feature. For Poisson's Equation, we instead add the evaluation of the load function f at the respective face midpoint.

Setup. We use Proximal Policy Optimization (PPO) (Schulman et al., 2017) as our RL algorithm for all experiments due to its simplicity and empirical success. We experiment on different 2d domains with triangular meshes and linear elements. The PDEs are implemented using *scikit-fem* (Gustafsson & Mcbain, 2020). All methods are evaluated for different α to produce a wide range of solutions, and we repeat each experiment configuration for 10 different random seeds. We report the average performance on 100 randomly sampled but fixed evaluation PDEs for each seed. Appendix D.1 lists all hyperparameters for the network training and the PPO algorithm.

All environment episodes start with a coarse initial mesh Ω^0 and iteratively refine this mesh $T = 4$ times unless mentioned otherwise. During training, the reference mesh Ω^* is computed by uniformly refining the initial mesh Ω^0 a total of 4 times. We evaluate a given refinement via its normalized numerically integrated squared error compared to a 5 times refined reference mesh to ensure a small numerical error in the evaluation. Details on this metric are given in Appendix C.1. We also provide results for a normalized linear error in Appendix C.2.



(a) Laplace's Equation (b) Poisson's Equation

Figure 3: Normalized squared error plotted against the number of elements for (a) the Laplace Equation and (b) the Poisson Equation. Each point represents the average performance over 100 evaluation environments for one random seed. The lines are a Log-Log linear regression over the aggregated results for each method. For both experiments, all methods can provide refinements for different numbers of elements. However, ASMR is significantly more stable than the other learned methods, producing more consistent and higher-quality refinements, especially for larger meshes.

Baselines. We adapt an *Argmax* (Yang et al., 2021) and a *VDGN*-inspired (Yang et al., 2022) baseline to our setup. For both, we provide a global reward by summing over all local rewards without area scaling and use our MPN for the policy and value functions. For the *Argmax* baseline, each element predicts a continuous action, and in each environment step the element with the highest action is marked. We remove the face penalty and train for up to 100 environment steps since the method refines only one element at a time. For *PPO-VDQN*, we compute a value function for each agent and then sum them up to calculate a value decomposition (Sunehag et al., 2017). We also compare to a traditional error-based AMR baseline (Binev et al., 2004; Bangerth et al., 2012; Foucart et al., 2022). Given a parameter θ, this *Heuristic* iteratively refines all elements Ω_i^t for which $\tilde{\text{err}}(\Omega_i^t) > \max_j \tilde{\text{err}}(\Omega_j^t)$. This baseline uses the reference mesh Ω^* for error calculation as oracle information, which is usually unavailable during inference. Additionally, the heuristic is purely local and thus may be sub-optimal for globally propagating errors, which is a well-known issue for elliptic PDEs (Strauss, 2007; Foucart et al., 2022). Opposed to this, RL methods learn to maximize an expected return, allowing them to find more strategic marking strategies.

We additionally conduct ablations to determine which parts of our approach make it effective. For this, we consider an *Unscaled* variant for our method that drops the area scaling in Equation 1, as well as a *Shared Reward* version that considers the sum of all agent's rewards as a single global reward. Note that the combination of these two changes results in *PPO-VDGN*.

Results. Figure 2 shows qualitative results of ASMR on an exemplary domain and load function for the Poisson problem. For different values of α, our approach provides refinements of different granularity. Quantitatively, we can see from Figure 3 that our approach outperforms other learned methods across both systems of equations, especially as the number of elements increases. We provide additional evaluations for a linear instead of a squared error in Appendix C.2. Interestingly, ASMR also performs on par or better than the error-based Heuristic baseline, suggesting that it learns sophisticated refinements strategies that minimize the error over the course of the full environment episode without access to the error estimates. Appendix C.3 provides further ablation experiments on different observation graphs, while Appendix E lists additional visualizations for all methods.

4 Conclusion

We propose Adaptive Swarm Mesh Refinement (ASMR), a novel method for Adaptive Mesh Refinement that utilizes Swarm Reinforcement Learning to iteratively refine meshes to efficiently solve Partial Differential Equations. By viewing the mesh elements as agents in a swarm system, our method can efficiently train a shared policy for all agents. Combining this approach with Graph Neural Networks and a novel per-agent reward formulation, our method yields stable and efficient mesh refinements for meshes with thousands of elements without requiring access to an error estimate during inference. Experimentally, we show that our method outperforms existing Reinforcement Learning mesh refinement techniques and compares well to an error-based heuristic. In future work, we want to extend our approach to more complex systems of equations, including time-dependent Partial Differential Equations. Other promising directions include extending our approach to quadrilateral meshes and combining the current per-agent reward with a global measure of mesh quality.

REFERENCES

Robert Anderson, Julian Andrej, Andrew Barker, Jamie Bramwell, Jean-Sylvain Camier, Jakub Cerveny, Veselin Dobrev, Yohann Dudouit, Aaron Fisher, Tzanio Kolev, et al. Mfem: A modular finite element methods library. *Computers & Mathematics with Applications*, 81:42–74, 2021.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=nIAxjsniDzg`.

Douglas N Arnold, Arup Mukherjee, and Luc Pouly. Locally adapted tetrahedral meshes using bisection. *SIAM Journal on Scientific Computing*, 22(2):431–448, 2000.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *stat*, 1050:21, 2016.

Timothy J Baker. Mesh adaptation strategies for problems in fluid dynamics. *Finite Elements in Analysis and Design*, 25(3-4):243–273, 1997.

Wolfgang Bangerth and Rolf Rannacher. *Adaptive finite element methods for differential equations*. Springer Science & Business Media, 2003.

Wolfgang Bangerth, Carsten Burstedde, Timo Heister, and Martin Kronbichler. Algorithms and data structures for massively parallel generic adaptive finite element codes. *ACM Transactions on Mathematical Software (TOMS)*, 38(2):1–28, 2012.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL `http://arxiv.org/abs/1806.01261`.

Marsha J Berger and Phillip Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1):64–84, 1989.

Peter Binev, Wolfgang Dahmen, and Ron DeVore. Adaptive finite element methods with convergence rates. *Numerische Mathematik*, 97:219–268, 2004.

Raunak Borker, Daniel Huang, Sebastian Grimberg, Charbel Farhat, Philip Avery, and Jason Rabinovitch. Mesh adaptation framework for embedded boundary methods for computational fluid dynamics and fluid-structure interaction. *International Journal for Numerical Methods in Fluids*, 90(8):389–424, 2019.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Greg L Bryan, Michael L Norman, Brian W O'Shea, Tom Abel, John H Wise, Matthew J Turk, Daniel R Reynolds, David C Collins, Peng Wang, Samuel W Skillman, et al. Enzo: An adaptive mesh refinement code for astrophysics. *The Astrophysical Journal Supplement Series*, 211(2):19, 2014.

Jakub Cerveny, Veselin Dobrev, and Tzanio Kolev. Nonconforming mesh refinement for high-order finite elements. *SIAM Journal on Scientific Computing*, 41(4):C367–C392, 2019.

Andrew J Cunningham, Adam Frank, Peggy Varnière, Sorin Mitran, and Thomas W Jones. Simulating magnetohydrodynamical flow with constrained transport and adaptive mesh refinement: algorithms and tests of the astrobear code. *The Astrophysical Journal Supplement Series*, 182(2): 519, 2009.

Corbin Foucart, Aaron Charous, and Pierre FJ Lermusiaux. Deep reinforcement learning for adaptive mesh refinement. *arXiv preprint arXiv:2209.12351*, 2022.

Gaël Gibert, Benoit Prabel, Anthony Gravouil, and Clémentine Jacquemoud. A 3d automatic mesh refinement x-fem approach for fatigue crack propagation. *Finite Elements in Analysis and Design*, 157:21–37, 2019.

David J Griffiths. Introduction to electrodynamics, 2005.

Thomas Guillet, Rüdiger Pakmor, Volker Springel, Praveen Chandrashekar, and Christian Klingenberg. High-order magnetohydrodynamics for astrophysics with an adaptive mesh refinement discontinuous galerkin scheme. *Monthly Notices of the Royal Astronomical Society*, 485(3):4209–4246, 2019.

Tom Gustafsson and Geordie Drummond Mcbain. scikit-fem: A python package for finite element assembly. *Journal of Open Source Software*, 5(52):2369, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Maximilian Hüttenrauch, Šošić Adrian, Gerhard Neumann, et al. Deep reinforcement learning for swarm systems. *Journal of Machine Learning Research*, 20(54):1–31, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Eisuke Kita and Norio Kamiya. Error estimation and adaptive mesh refinement in boundary element method, an overview. *Engineering Analysis with Boundary Elements*, 25(7):479–495, 2001.

Jonas Linkerhägner, Niklas Freymuth, Paul Maria Scheikl, Franziska Mathis-Ullrich, and Gerhard Neumann. Grounding graph network simulators using physical sensor observations. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL `https://openreview.net/forum?id=jsZsEd8VEY`.

Charles W Misner, Kip S Thorne, and John Archibald Wheeler. *Gravitation*. Macmillan, 1973.

Arup Mukherjee. *An adaptive finite element code for elliptic boundary value problems in three dimensions with applications in numerical relativity*. The Pennsylvania State University, 1996.

M Ortiz and JJ Quigley Iv. Adaptive mesh refinement in strain localization problems. *Computer Methods in Applied Mechanics and Engineering*, 90(1-3):781–804, 1991.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL `https://arxiv.org/abs/2010.03409`.

Tomasz Plewa, Timur Linde, V Gregory Weirs, et al. Adaptive mesh refinement-theory and applications. 2005.

Nikolas Provatas, Nigel Goldenfeld, and Jonathan Dantzig. Efficient computation of dendritic microstructures using adaptive mesh refinement. *Physical Review Letters*, 80(15):3308, 1998.

Junuthula Narasimha Reddy. *Introduction to the finite element method*. McGraw-Hill Education, 2019.

Lovely Sabat and Chinmay Kumar Kundu. History of finite element method: a review. *Recent Developments in Sustainable Infrastructure*, pp. 395–404, 2021.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

Adrian Šošić, Wasiur R KhudaBukhsh, Abdelhak M Zoubir, and Heinz Koeppl. Inverse reinforcement learning in swarm systems. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 1413–1421, 2017.

Erwin Stein. *Adaptive finite elements in linear and nonlinear solid and structural mechanics*, volume 416. Springer Science & Business Media, 2007.

Walter A Strauss. *Partial differential equations: An introduction*. John Wiley & Sons, 2007.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

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

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Jiachen Yang, Tarik Dzanic, Brenden Petersen, Jun Kudo, Ketan Mittal, Vladimir Tomov, Jean-Sylvain Camier, Tuo Zhao, Hongyuan Zha, Tzanio Kolev, et al. Reinforcement learning for adaptive mesh refinement. *arXiv preprint arXiv:2103.01342*, 2021.

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. *arXiv preprint arXiv:2211.00801*, 2022.

Dennis G Zill. *Differential equations with boundary-value problems*. Cengage Learning, 2016.

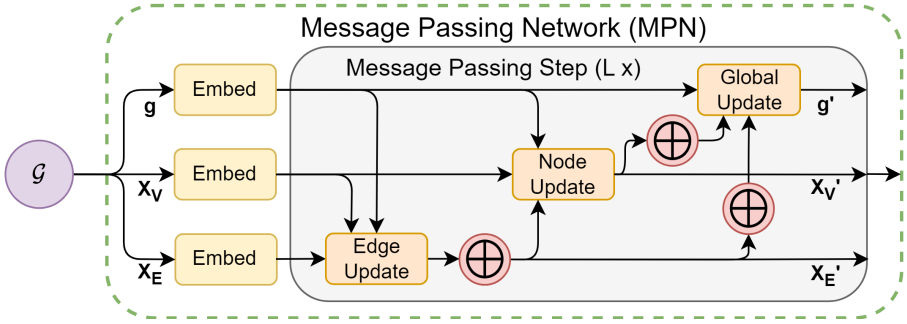


Figure 4: A detailed view of the MPN architecture employed by ASMR. Given a graph $\mathcal{G}$, the global features $\mathbf{g}$, node features $\mathbf{X_V}$ and edge features $\mathbf{X_{E'}}$ are linearly embedded into a latent space. From this latent space, L message passing steps are performed. The resulting latent features per node are interpreted as a local observation encoding and can be given to an RL policy or value function.

A Method Details

A.1 Message Passing Network Architecture

Figure 4 shows the Message Passing Network employed by ASMR.

A.2 Swarm Markov Decision Process

Formally, we define an abstract agent as a tuple $\mathbb{A} := \langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \pi \rangle$. Here, $\mathcal{S}$, $\mathcal{O}$, $\mathcal{A}$ are sets of *local* states, observations and actions, respectively and $\pi : \mathcal{O} \times \mathcal{A} \to [0, 1]$ is the *local* policy of the agent. From this definition, the Swarm MDP is characterized by a tuple $\langle N, \mathbb{A}, T, R, \xi \rangle$. More precisely, given a number of agents N, the system follows a *global* transition model $T : \mathcal{S}^N \times \mathcal{A}^N \to \mathcal{S}^M$, where potentially $M \neq N$ and uses a *per-agent* reward function $R : \mathcal{S}^N \times \mathcal{A}^N \to \mathbb{R}^N$. The observations of the agents are calculated from their states via the observation model $\xi : \mathcal{S}^N \to \mathcal{O}^N$.

In our case, the local states and observations share a lot of structure. Given a mesh Ω^t with N elements $\Omega_i^t \subseteq \Omega^t$, each element can be considered an agent. The agent's state then is its local view on the full mesh, and its observations is a local encoding of the mesh. We make use of this shared structure by encoding the observations as a graph, where each node represents a mesh element and each edge the neighborhood of two elements. This allows us to compute joint actions for all agents by using the whole observation graph as the input for an MPN-based policy.

A.3 Reward Design

Adaptive Mesh Refinement fundamentally trades off the accuracy of the refined mesh with its usage of resources. To estimate the accuracy, we define an *error per element* as the numerically integrated difference in solution of this element when compared to some fine-grained reference mesh Ω^*. For this, we consider the midpoints $p_{\Omega_r^*} \in \Omega_r^*$ of all elements Ω_r^* of the reference mesh. For each element Ω_i^t we then integrate over all midpoints that fall into it, scaling each by the area of its respective element in Ω^*. This results in an error estimate

$$\hat{\text{err}}(\Omega_i^t) \approx \sum_{\Omega_r^* \subseteq \Omega_i^t} \text{Area}(\Omega_r^*) \Big(|u_{\Omega^*}(p_{\Omega_r^*}) - u_{\Omega^t}(p_{\Omega_r^*})| \Big)$$

for the solution u_{Ω^*} on the fine mesh and the solution u_{Ω^t} on the current mesh. To get an error estimate that is consistent across different geometries, we normalize the error with the total error of the initial mesh Ω^0, i.e.,

$$\tilde{\text{err}}(\Omega_i^t) = \frac{\hat{\text{err}}(\Omega_i^t)}{\sum_{\Omega_e^0 \in \Omega^0} \hat{\text{err}}(\Omega_e^0)}.$$

From here, we formulate a *reward per element* as given in Equation 1. This reward intuitively evaluates whether a given refinement decreases the overall error enough to warrant the extra use of resources, and becomes 0 for unrefined faces. The added area scaling term leads to more accurate refinements, as it discourages the refinements of large elements with a relatively low average error. Mathematically, optimizing this reward corresponds to minimizing the error in elements with a high error density while keeping the total usage of elements low. We expect a high reward of the refinement of elements with a high error density, because the error is locally high compared to the fine solution. A refinement thus leads to a more accurate solution in this area. We note that this reward formulation is purely local, allowing the approach to scale to a large number of agents, at the cost of providing less accurate rewards for PDEs with global dependencies.

B Systems of Equations

In its most general form, the FEM is used to approximate the solution $u(x)$ for the set of test functions ψ, which satisfies the weak form $\forall x : \ \forall \psi(x) : \ a(u(x), \psi(x)) = l(\psi(x))$ of the underlying system of equations.

B.1 Laplace's Equation

Let Ω be a domain with an inner boundary $\partial\Omega_0$ and an outer boundary $\partial\Omega_1$. Laplace's Equation is given as

$$\nabla^2 u(x) = 0 \quad \text{in } \Omega.$$

The weak solution to this equation consists of finding $u(x)$ such that

$$\int_\Omega \nabla u(x) \cdot \nabla v(x) \, \mathrm{d}x = 0$$

for all test functions $\psi(x)$. Additionally, the solution has to satisfy the Dirichlet boundary conditions

$$u(x) = 0, x \in \partial\Omega_0, \qquad u(x) = 1, x \in \partial\Omega_1.$$

We use a unit square $(0,1)^2$ for the outer boundary $\partial\Omega_0$ of the domain and add a randomly sampled square hole, whose borders are considered to be the inner boundary $\partial\Omega_1$. The size of the hole is sampled from the uniform distribution $U(0.05, 0.25)^2$, and its mean position is sampled from $U(0.2, 0.8)^2$.

B.2 Poisson's Equation

We additionally consider the Poisson problem

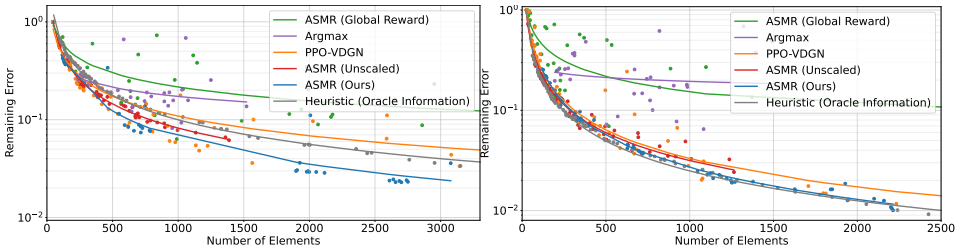
$$\nabla^2 u(x) = f(x) \quad \text{in } \Omega, \qquad u(x) = 0 \quad \text{on } \partial\Omega$$

for a load function $f(x) : \Omega \to \mathbb{R}$. Here, the weak solution consists of finding $u(x)$ such that

$$\int_\Omega \nabla u(x) \cdot \nabla \psi(x) \, \mathrm{d}x = \int_\Omega f(x)\psi(x) \, \mathrm{d}x$$

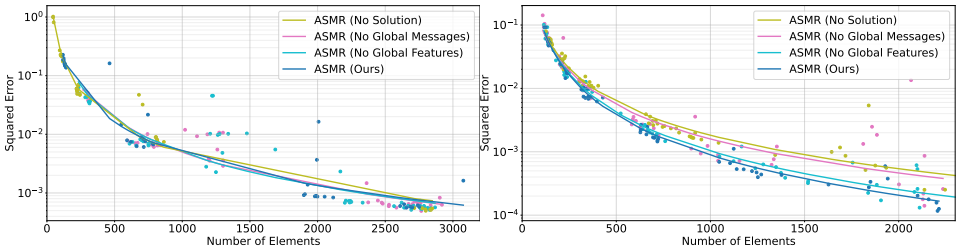
for all test functions $\psi(x)$ while being 0 on the boundary $\partial\Omega$.

We model Poisson's Equation on L-shaped domains, using a rectangular cutoff whose lower left corner is sampled from $p_0 \sim U(0.2, 0.95)^2$, resulting in a domain $\Omega = (0,1)^2 \backslash (p_0 \times (1,1))$. On this domain, we sample a Gaussian Mixture Model with $o = 3$ components. The mean of each component is sampled from $U(0.1, 0.9)^2$, using rejection sampling to ensure that all means lie within the domain. The components' covariances are determined by first drawing diagonal covariances, where each dimension is drawn independently from a log-uniform distribution $\exp(U(\log(0.0003, 0.003)))$. The diagonal covariances are then rotated by a random angle in $U(0, 180)$ to produce Gaussians with a full covariance matrix. The component weights are drawn from the distribution $\exp(N(0,1)) + 1$ and subsequently normalized, where the 1 in the end is used to ensure that all components have relevant weight.



(a) Laplace's Equation (b) Poisson's Equation

Figure 5: Normalized linear error plotted against the number of elements for (a) the Laplace Equation and (b) the Poisson Equation. For the linear error, areas with a high error are weighted less. As a result, methods that produce more regular meshes perform slightly better, because their main source of error is not taken into account as much.



(a) Laplace's Equation (b) Poisson's Equation

Figure 6: Normalized squared error plotted against the number of elements for ablations of ASMR on (a) the Laplace Equation and (b) the Poisson Equation. Finding a good refinement for Laplace's Equation seems to be possible without either global information or information about the solution $u(x)$. However, for Poisson's Equation, the performance degrades when not providing global features. It further decreases for policies without any access to global messages and when not providing solution information per mesh element.

C Experiments and Evaluations

C.1 Evaluation Metric

Given a reference mesh Ω^* and a final refined mesh Ω^T, we compute the normalized squared error by evaluating the squared difference in the reference solution u_{Ω^*} and the solution u_{Ω^T} as

$$\frac{\sum_m \text{Area}(\Omega^*_m)\left(u_{\Omega^*}(p_{\Omega^*_m}) - u_{\Omega^T}(p_{\Omega^*_m})\right)^2}{\sum_m \text{Area}(\Omega^*_m)\left(u_{\Omega^*}(p_{\Omega^*_m}) - u_{\Omega^0}(p_{\Omega^*_m})\right)^2}. \tag{2}$$

Here, we use the squared error during evaluation as it is more sensitive to outliers and thus provides a cleaner measure of the quality of the mesh. We additionally provide results for the linear error, i.e., for Equation 2 without the squares, in Appendix C.2.

C.2 Linear Error Metric

Figure 5 shows results for the linear error. The general trends for the linear errors are consistent with the quadratic errors in Figure 3. However, we find that the linear metrics does not punish meshes with very high local errors as much, resulting in slightly better results for methods that produce less adaptive meshes.

C.3 Additional Ablations

We experiment with a version of ASMR that does not use global features $\mathbf{g}$ (ASMR (No Global Features)) but still relies on a global update to pass messages across the full mesh, and with a

version that removes the global message passing altogether (ASMR (No Global Messages)). In both cases, the global information of the mesh is limited, and for the latter, the policies have to act purely from local information. Additionally, we consider an ablation of our approach that does not use the solution information $u(x)$ as a node feature. For this ablation, the RL algorithm has to refine relevant areas of the domain from just an encoding of the mesh and basic information about the underlying PDE.

Figure 6 shows the results for these ablations. We find that all methods perform roughly equally well on the simpler Laplace Equation. However, the solution information and global features and messages significantly improve performance on the more challenging Poisson problem. Interestingly, the results for ASMR (No Solution) suggest that the RL algorithm is able to calculate relevant regions of the PDE from just an encoding of the domain and the boundary conditions. This is likely achieved by implicitly construction some form of solution within the policy.

D HYPERPARAMETERS

D.1 GENERAL HYPERPARAMETERS

We use the same general hyperparameters across all methods and environments for the sake of simplicity. All experiments are repeated for $n = 10$ random seeds with randomized PDEs and network parameters. During training, a new PDE is sampled for every environment reset. All PDEs are normalized to be in $(0, 1)^2$. We evaluate all approaches for the final policies on 100 randomly sampled PDEs that we keep consistent across random seeds for better comparability.

For PPO, we largely follow the suggestions of Andrychowicz et al. (2021). We train for a total of 500 iterations, drawing 256 samples per iteration and then training for 5 epochs with a batch size of 32. All networks are implemented in PyTorch (Paszke et al., 2019) and trained using the (Kingma & Ba, 2014) ADAM optimizer with a learning rate of $3.0e-4$. The loss of the value function is multiplied with a factor of 0.5. We clip the norm of the gradient during backpropagation to 0.5, and set the clip range for the policy and the value function to 0.2. We normalize the observations with a running mean and standard deviation, use a discount factor of $\gamma = 0.99$ and estimate the advantages using Generalized Advantage Estimate (Schulman et al., 2015) with $\lambda = 0.95$. The policy and value function heads MLPs with 2 hidden layers with a latent dimension of 32 and use a tanh activation function.

We use separate MPNs for the policy and the value function. Each MPN consists of 2 Message Passing Steps, where each function $f_\cdot$ is represented as a 2-layer MLP with a latent dimension of 32. All message aggregations $\bigoplus$ are mean aggregations. Additionally, we apply Layer Normalization (Ba et al., 2016) and Residual Connections (He et al., 2016) independently for the node, edge and global features after each message passing step.

Table 1: Ranges for the refinement hyperparameters for the Laplace Equation. The Argmax baseline considers a different number of timesteps T,while the error-based Heuristic uses different thresholds θ. All other methods make use of an element penalty α. Note that values of α need to be different when scaling/not scaling the reward in Equation 1 with the element area term.

Method	Parameter	
ASMR	α	$\{2e-1, 1e-1, 5e-2, 2e-2, 1e-2, 5e-3\}$
ASMR (Unscaled)	α	$\{1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3, 1e-2\}$
ASMR (Shared Reward)	α	$\{3e-1, 1e-1, 3e-2, 1e-2\}$
PPO-VDGN	α	$\{1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3, 1e-2\}$
Argmax	T	$\{20, 60, 100\}$
Heuristic	θ	$\{0, 0.02, 0.04, ..., 0.98\}$

Table 2: Ranges for the refinement hyperparameters for the Poisson Equation. The Argmax baseline considers a different number of timesteps T, while the error-based Heuristic uses different thresholds θ. All other methods make use of an element penalty α. Note that values of α need to be different when scaling/not scaling the reward in Equation 1 with the element area term.

Method	Parameter	
ASMR	α	$\{2e-1, 1e-1, 5e-2, 2e-2, 1e-2, 5e-3\}$
ASMR (Unscaled)	α	$\{1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3, 1e-2\}$
ASMR (Shared Reward)	α	$\{2e-1, 1e-1, 5e-2, 2e-2, 1e-2, 5e-3\}$
PPO-VDGN	α	$\{1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3, 1e-2\}$
Argmax	T	$\{20, 60, 100\}$
Heuristic	θ	$\{0, 0.02, 0.04, ..., 0.98\}$

D.2 METHOD-SPECIFIC PARAMETERS

All methods control the number of mesh elements that they produce with a parameter. The error-based Heuristic uses the refinement threshold θ, the Argmax baseline a number of timesteps T to refine for, and all other methods make use of an element penalty α. Table 1 lists the different ranges for these parameters for our experiments for Laplace's Equation, and Table 2 for Poisson's Equation.

E ADDITIONAL VISUALIZATIONS

We provide additional visualizations for both systems of equations on randomly selected evaluation environments and algorithm random seeds. For Laplace's Equation, visualizations are shown in Figure 7(ASMR), Figure 8 (PPO-VDGN), Figure 9 (Argmax), and Figure 10 (error-based Heuristic). For Poissons's Equation, visualizations are provided in Figure 11(ASMR), Figure 12 (PPO-VDGN), Figure 13 (Argmax), and Figure 14 (error-based Heuristic). The visualizations show that ASMR is able to provide consistent refinements of high quality for different mesh resolutions. Opposed to this, VDGN sometimes over- or underrefines meshes for the same element penalty α. The Argmax baseline generally performs well, but sometimes focuses on uninteresting regions of the mesh. Finally, the error-based Heuristic greedily refines the elements with the largest errors. This may lead to issues for globally propagating errors (Strauss, 2007).

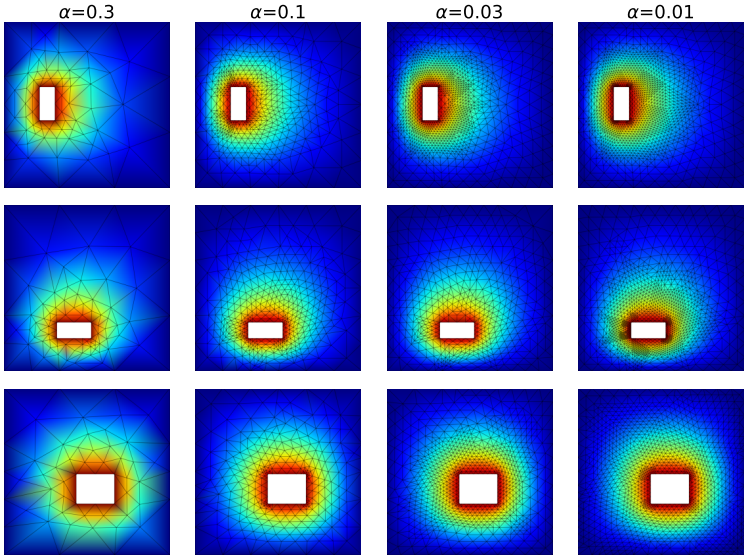


Figure 7: Refinements for Laplace's Equation for refinements of ASMR on randomly sampled PDEs for different values of the element penalty α.

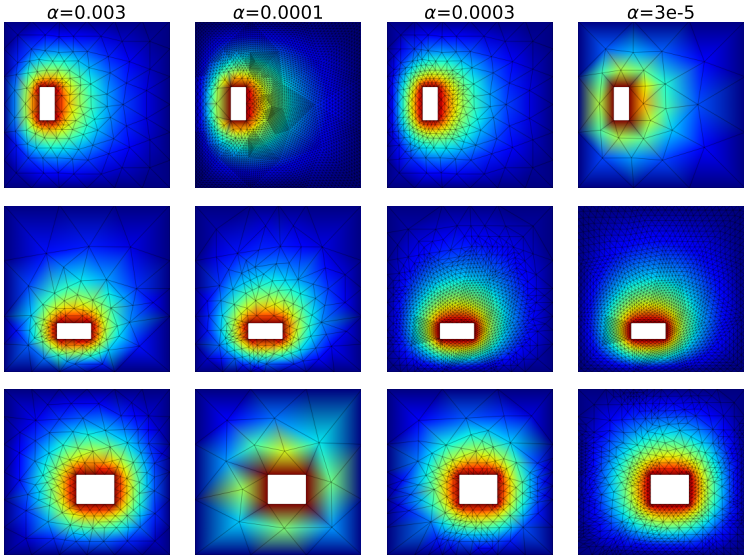


Figure 8: Refinements for Laplace's Equation for refinements of PPO-VDGN on randomly sampled PDEs for different values of the element penalty α.

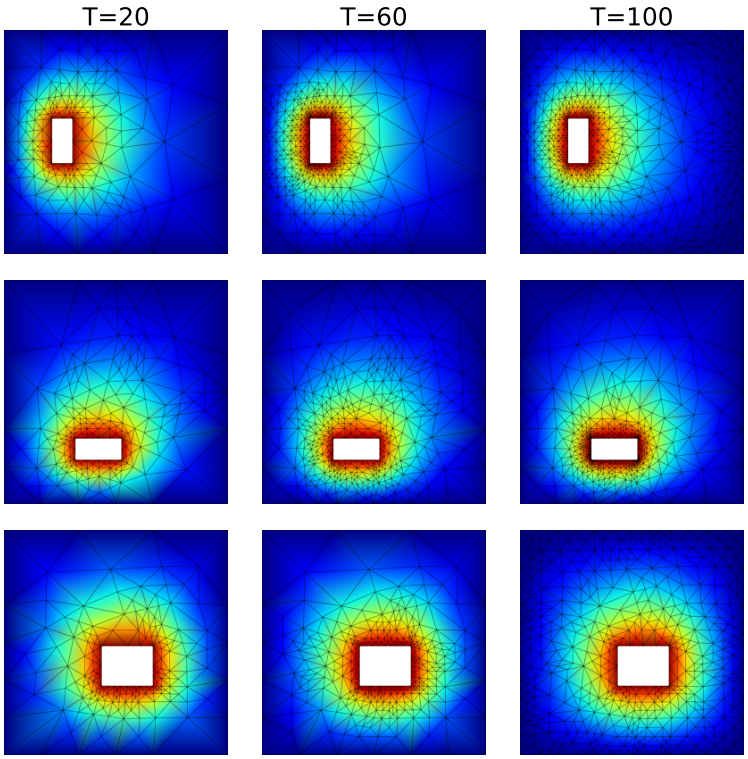


Figure 9: Refinements for Laplace's Equation for refinements of the Argmax baseline on randomly sampled PDEs for different environment rollout lengths T.

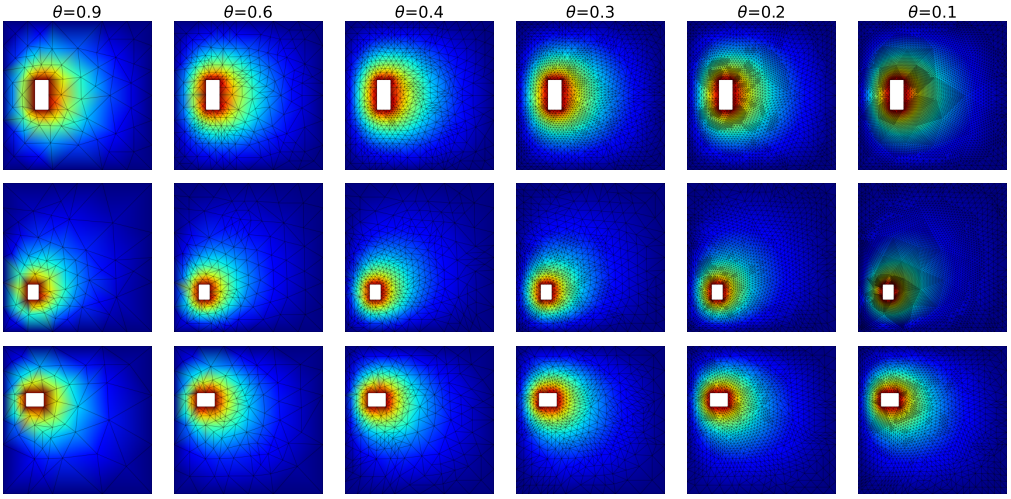


Figure 10: Refinements for Laplace's Equation for refinements of the error-based Heuristic on randomly sampled PDEs for different refinement thresholds θ.

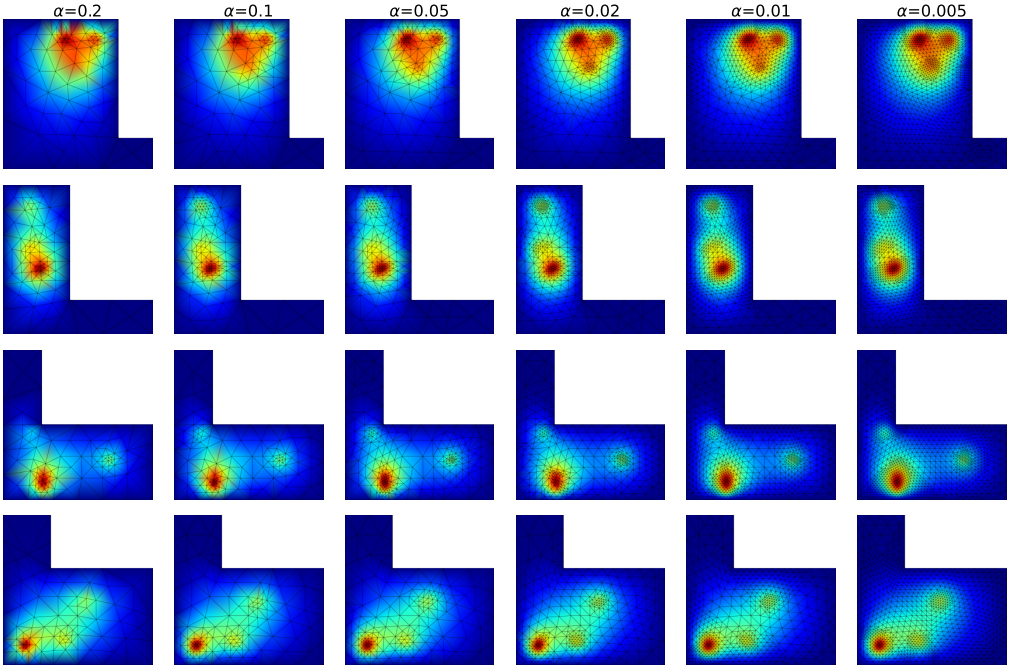


Figure 11: Refinements for Poisson's Equation for refinements of ASMR on randomly sampled PDEs for different values of the element penalty α

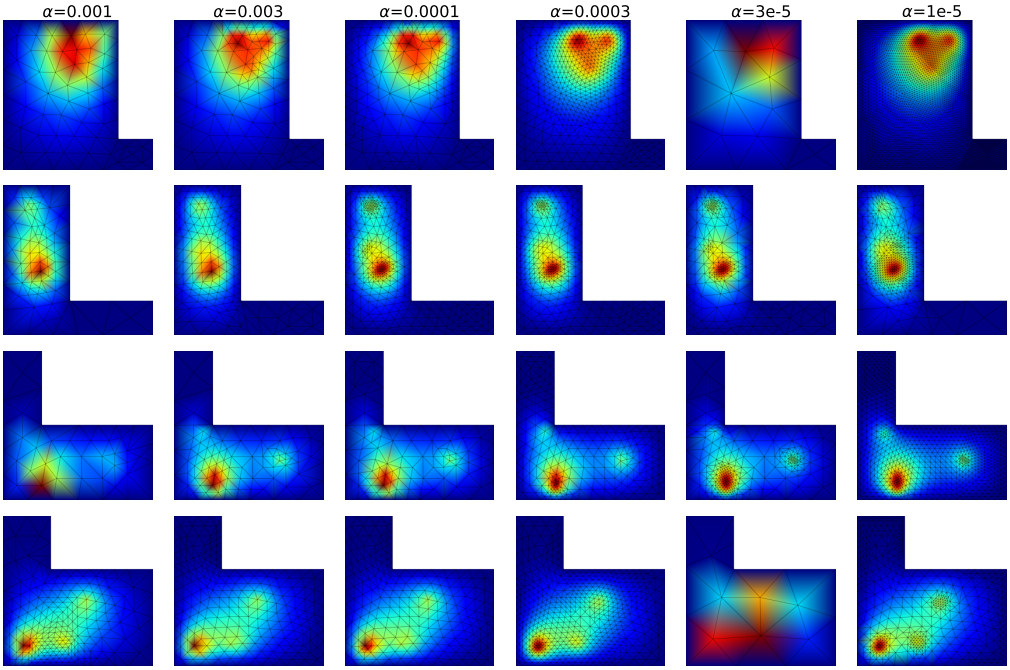


Figure 12: Refinements for Poisson's Equation for refinements of PPO-VDGN on randomly sampled PDEs for different values of the element penalty α

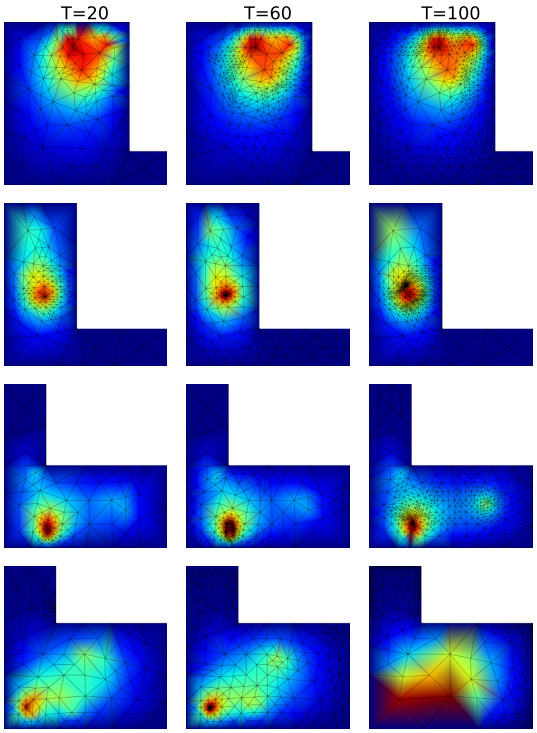


Figure 13: Refinements for Poisson's Equation for refinements of the Argmax baseline on randomly sampled PDEs for different environment rollout lengths T.

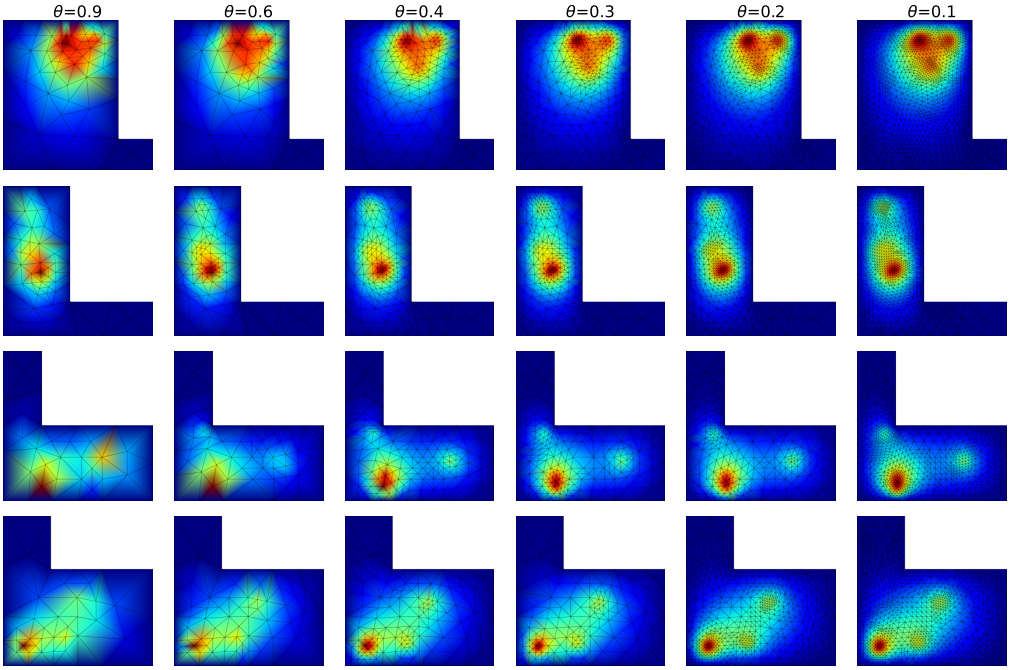


Figure 14: Refinements for Poisson's Equation for refinements of the error-based Heuristic on randomly sampled PDEs for different refinement thresholds θ.