CS2470 Final Project - PINNS + RL-Driven Sampling

Lucas Zheng (lzheng35) Yash Kothari (ykothari)
Kevin Pan (kpan32) Fillip Cutiuba (fcutiuba)

December 2025

1 Introduction

Seismic hypocenter localization - the process of determining the source location of microseismic activity - is a critical task in microseismic monitoring. Human-induced seismic events (from industrial activities like fracking and mining) have been on the rise [1], making rapid, accurate source localization of induced events increasingly necessary to ensure safety and regulatory compliance. Among other use cases, source localization can give early warnings of dangerous collapses and fault slips, potentially preventing accidents and mitigating triggered earthquakes. Given a known seismic velocity model (we work with the synthetic Marmousi2 dataset designed by geophysicists to model complex geology) and generated surface receiver picks, the goal is to accurately predict the source of the seismic activity by solving the underlying wave equation, a regression task. We train a Physics Informed Neural Network (PINN) with both Adaptive Residual Sampling and RL-driven Sampling for collocation points to attempt to solve this problem of source localization.

We decided to pursue this project since it combined a lot of our interests in PINNs and RL. We also were generally interested in earthquakes, and the data was readily available.

2 Related Work

There has been work that sought to address the limitations of uniform and heuristic adaptive sampling in Physics-Informed Neural Networks (PINNs). Notably, Song proposed RL-PINNs [2], which is a framework that reformulates the normal adaptive sampling as a Markov Decision Process (MDP). The normal Residual-Based Adaptive Refinement struggles with high compute costs due to repeated retraining and gradient-based error estimation, but with RL-PINNs, Song utilized a reinforcement learning agent to learn an optimal sampling policy in a single training round, which would take up less compute. The main

innovation in this approach is the use of function variation as a gradient-free reward signal, which also is combined with a delayed reward mechanism. This methodology focuses on high-dimensional PDE problems, and establishes the viability of using RL agents. This can extend to inverse problems and seismic source localization.

3 Data

3.1 Geophysical Environment: The Marmousi Model

In order to validate the method we propose, we used the **Marmousi2 velocity model** [3]. This model provides us complex velocity changes and contrasts, which overall provided a challenging testbed for these types of propagation algorithms, especially compared to homogenous or layered data.

For computational feasibility, we downsampled the original high-resolution model of size 13601×2801 to a grid size of $N_z \times N_x = 681 \times 141$. The physical domain is then also normalized to the unit square [0,1]. Since the Eikonal equation is governed by slowness rather than velocity, the raw velocity field $v(x, z)$ is inverted to produce the slowness map $s(x, z) = 1/v(x, z)$. This static slowness map serves as the environment in which the Physics-Informed Neural Network (PINN) operates.

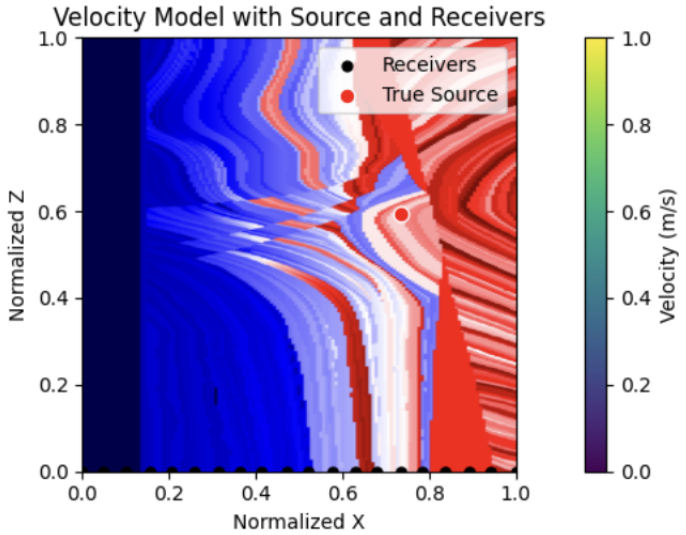
3.2 Synthetic Seismic Data Generation

We did not use actual earthquake data, instead, we utilized the **Fast Marching Method (FMM)** via the eikonalfm package to create a synthetic dataset. We select a position in the domain to act as the "earthquake source", then solve the Eikonal equation with FMM to get the true traveltime field.

The experimental setup emulates a sparse surface seismic survey:

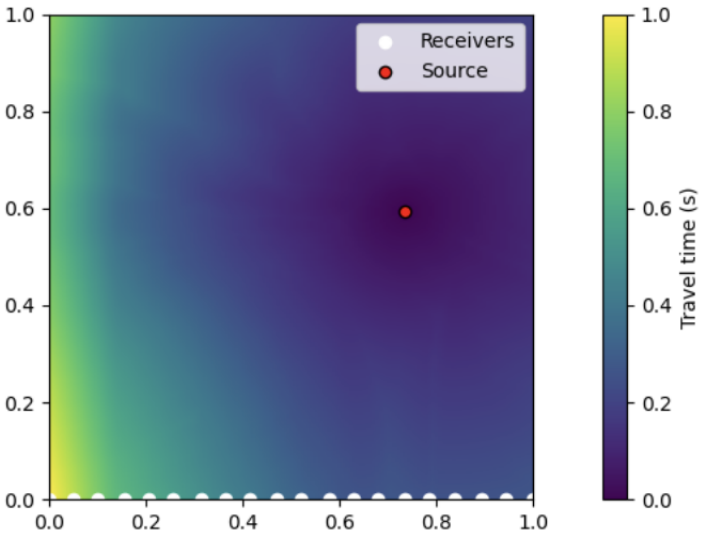
- **Receivers:** A set of 20 sensors is distributed equidistantly along the surface boundary ($z = 0$) to simulate a real life setting. Using the previously computed true traveltime field, we provide the arrival times at these receiver locations.
- **Seismic Velocity Map:** We also provide the seismic velocity map (from Marmousi) to the model via slowness lookups - this is used for PDE loss.

Essentially, we model the real-world scenario where we have a known underground velocity map and a set of receivers placed near the surface, and the task is to identify the source of the earthquake.




```
Velocity model shape: (681, 141)
Source location: (0.736, 0.593)
```

A visualization of the problem setup - we see the downsampled velocity model with the different wave velocities and the 20 receivers placed at the surface (the bottom of the image).



The true travel time field, solved by giving true source location and velocity map to eikonalfm. We take the true travel times at the receivers to simulate true receivers.

3.3 RL Agent Input Features

The data inputs for the Reinforcement Learning agent differ from the PINN. The agent does not see the raw waveforms but rather observes the *state* of the PINN training process. At each decision step, 200 candidate points are sampled. The feature vector has:

1. **Log-Scaled Residuals:** The PDE residual $r = ||\nabla\tau|^2 - s^2|$ is log-transformed ($\log_{10}(r + \epsilon)$) to compress the dynamic range of errors, stabilizing the neural network training.
2. **Relative Spatial Coordinates:** The distance $(\Delta x, \Delta z)$ between each candidate point and the PINN's current estimated source location. This provides the agent with spatial context regarding the wavefront origin.
3. **Global Loss Metrics:** The current scalar values for the PDE, data, and source losses, providing a snapshot of global convergence.

4 Methodology

We used Physics-Informed Neural Networks to solve our problem of seismic source localization. Our model incorporates the Eikonal equation into the loss function in order to have the physical loss. We also implement and compare different adaptive sampling strategies to optimize the selection of collocation points.

The Eikonal equation relates the time field to the medium's (such as rock, for example) slowness:

$$\|\nabla\tau(x)\|^2 = s(x)^2$$

Our PINN approximates the function $\tau(x, z)$ such that it satisfies the differential equation while matching the observed travel times.

Such high-frequency functions are difficult for standard MLPs to understand. We used Fourier Feature Encoding [4] before to the main network in order to fix this. For a given input location (x, z), we apply:

- **Fourier Feature Encoding:** Maps the 2D coordinates to a 512D space.

$$\Phi(x, z) = [\sin(B[x, z]^\top),\ \cos(B[x, z]^\top)] \in R^{512}.$$

- **MLP:** The encoded input is passed through a 3-layer MLP with 256 neurons per layer and Tanh activation functions.
- **Output:** The model outputs a scalar travel time.
- **Source:** The source location is treated as a learnable parameter, initialized randomly and then updated.

The network is trained by minimizing a composite loss function consisting of:

- **PDE Loss:** Enforces physics via the Eikonal equation.

$$L_{pde} = \|\nabla \hat{\tau}_\theta(x_i, z_i)\|^2 - s(x_i, z_i)^2.$$

- **Data Loss:** Matches predicted travel times to observed picks from receivers.

$$L_{data} = \frac{1}{N_r} \sum_{j=1}^{N_r} \left(\hat{\tau}_\theta(x_j, z_j) - \tau_j^{obs}\right)^2.$$

- **Source Loss:** Enforces the boundary condition that the predicted travel time at the source location is 0.

$$L_{src} = \hat{\tau}_\theta(x_{src}, z_{src})^2.$$

We also apply learnable weights w_{pde} and w_{data} to allow the model to balance which parts of loss it should focus on during training. In addition, we have w_{src} increase linearly throughout training from 0.1 to 1. Our heuristic is that the model should focus more on learning the physics and data (receivers) before converging on a proper source prediction. Finally, we also tested different hyperparameters manually for more scaling as seen.

$$L = 100 * w_{pde}\, L_{pde} + 50 * w_{data}\, L_{data} + 0.1 * w_{src}\, L_{src}.$$

Finally, we have adaptive collocation sampling. A key challenge in PINNs is selecting the points where we solve the PDE. We implemented two strategies for this:

A. Residual-Based Probabilistic Sampling This method computes the PDE residual at candidate points across the domain and turns these residuals into a probability distribution. Points where the physics are currently violated (high residual) are more likely to be sampled for the next training batch.

B. Reinforcement Learning (RL) Sampling We formulated the point selection as a Markov Decision Process (MDP) and trained a Proximal Policy Optimization (PPO) agent to select points that maximize training efficiency.

State Space: A 403-dimensional vector containing the PDE residuals at a fixed 20 by 20 grid of candidate points, plus the current scalar loss values.

Action Space: A discrete selection of one grid point to add to the training set.

Reward function: The reduction in total loss between steps, minus a penalty for the distance between the estimated and the true source.

$$R = (L_{t-1} - L_t) - 0.1 \cdot \|src_{est} - src_{true}\|$$

5 Results

5.1 Evaluation Metrics

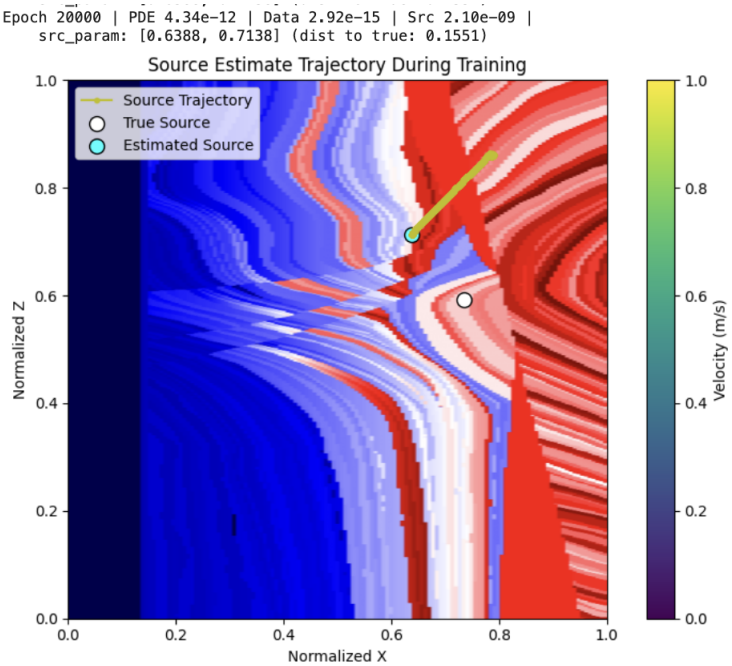
We defined success using two metrics for our project:

- **Source Localization Error:** The Euclidean distance between the estimated source location (x_{est}, z_{est}) and the true ground truth source (x_{true}, z_{true}).
- **PDE Loss Convergence:** The value of the physics residual $(\|\nabla\tau\|^2 - s^2)$ over training iterations. Faster convergence is better.

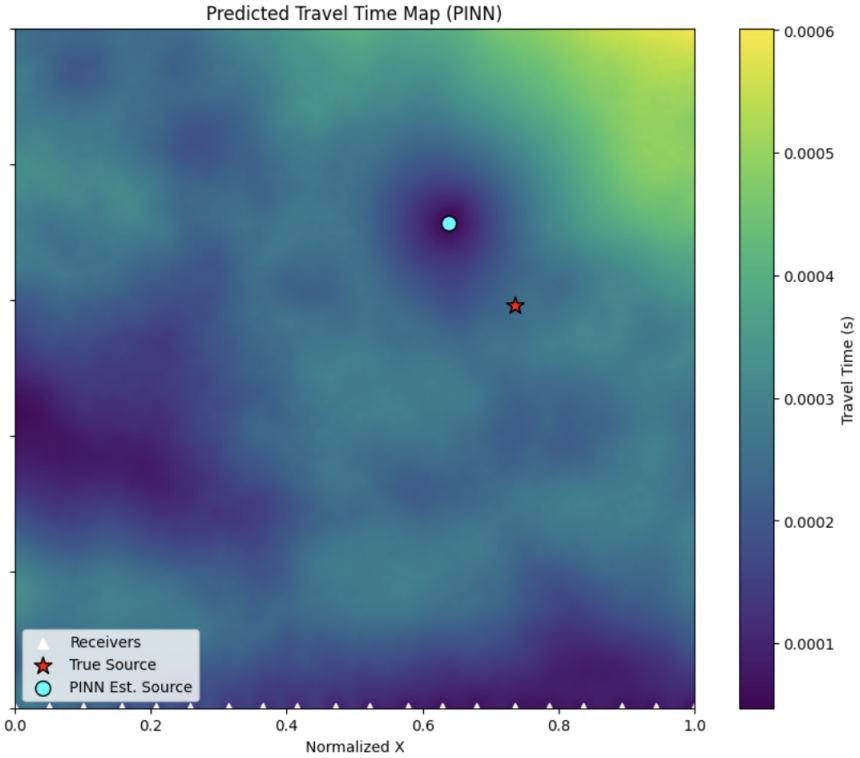
We compared the algorithms for finding collocation points using a fixed seed and over a fixed number of steps (1,000) Our baseline was Random Sampling, and we also had Greedy Sampling, Residual Probabilistic, and our trained RL Agent.

5.2 Source Localization

We trained the PINN + Adaptive Sampling model for 20000 epochs with adaptive sampling 10000 points every 100 epochs and visualized the resulting trajectory of source predictions. This run converged to a distance from true source of 0.1551. While results are not as good as we expect with the model often having a purely "to the surface" trajectory, we believe one of the reasons is due to downsampling, which makes the underlying velocity model harder to work with.



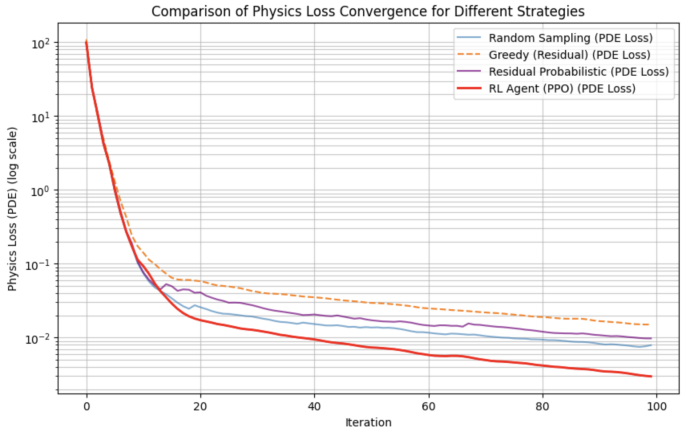
We also visualized the predicted travel time field compared to the true travel time field.



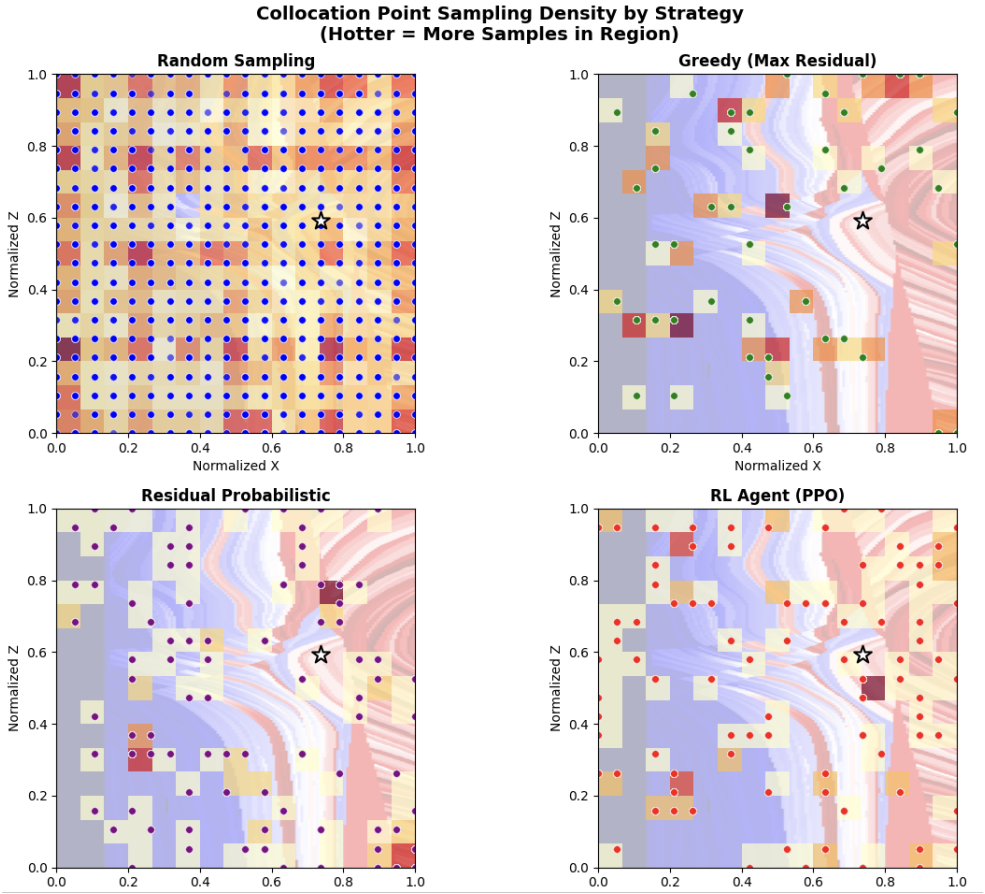
While it does a decent job, it struggles to enforce the radial structure (expanding from the source) structure that is expected. For next steps, we believe having some type of radial loss would help with this.

5.3 PDE Convergence

We looked at the log-scale convergence of the PDE loss (Figure 1 below) to determine which strategy enforced the physical constraints the best. The top performer here was the RL Agent. It had the fastest initial descent, and from iteration 15 onward, it had the lowest loss among all the strategies and in the end achieved the lowest final loss. Surprisingly, Random Sampling performed second best, and was then followed by Residual Probabilistic Sampling. The Greedy Sampling strategy resulted in the highest loss, probably because just looking at the highest-residual points without exploration causes the model to overfit specific regions.



Looking at the Sampling Density and Entropy Analysis (Figure 2 below), our suspicions are confirmed. We analyzed the spatial distribution of the selected collocation points, with a score of 1 being a perfectly uniform distribution.
For Random Sampling, as our baseline for maximum exploration, the entropy was 0.994. The strong performance of the Random Sampling shows that coverage is critical for this problem. Greedy Sampling though, had an entropy of 0.749, which was the lowest entropy and confirms the hypothesis that the algorithm repeatedly over-samples the same high-error regions. Adaptive residual sampling had an entropy of 0.854, higher than greedy and lower than random. The RL Agent had an entropy of 0.882, higher than greedy and adaptive but lower than random. This shows that the agent learned a policy that has the balance between ensuring global coverage and targeting high-error regions - or conceptually a balance between exploitation and exploration which was necessary for this problem. While pure random beat the more exploitative approaches of greedy and adaptive, the RL policy managed to beat pure random - showing that it did find at least some valuable balance in approaches.



6 Ethics

6.0.1 Who are the major "stakeholders" in this problem, and what are the consequences of mistakes made by your algorithm?

The major stakeholders would be public safety agencies, and residents in active earthquake zones. The agencies would use localization for Earthquake Early Warning systems, and residents would rely on those accurate warnings from the agencies to take cover. Accuracy here is crucial because lives are literally at stake. If the PINN fails to converge during a real earthquake, alerts are not sent, and people could die. If the localization error is high, wrong people might be informed and emergency responders could be looking in the wrong place.

6.0.2 Potential Issues with Reasons for Dataset Creation

The dataset for our project was the Marmousi2 velocity model. Our project focuses on source localization for earthquake safety, but the underlying physics and Deep Learning architecture are nearly identical to those ued in seismic exploration for fossil fuels (which is what the velocity model was actually first created for). This model was originally developed by the petroleum industry to test methods for locating oil and gas reservoirs, which improves these methods and increases the efficiency of oil extraction, which was negative implications when it comes to climate change. This just shows that the societal impact depends only on the use of the same exact algorithm.

7 Division of Labor

The project was split up evenly, and everyone took part in developing all parts of the project, including both the code portion as well as the presentation/reflection portions. Everyone contributed to the code portion, since we met up together to talk about the bugs, blockers, and next steps. The same is with the presentation and the reflections, all were done in collaboration, and everyone was present at the time of creating these documents.

8 Reflection

We feel that our project turned out well in terms of learning - we got to explore many cool ideas like physics-informed loss and RL (but definitely there are some next steps to explore to better performance :)). We have achieved our goal of implementing an RL agent in order to find collocation points, and when compared to other methods of finding the points, the RL agent does win out. We also get reasonable results with the PINN.
The model does generally work as we expected it to, although some inconsistencies arise mainly from not having enough time and compute for a longer training of the RL agent, as well as the down sampling of the Marmousi dataset, which definitely caused issues as well. The only theory that some of us had was that the Greedy model would perform better than it did since it picked the points with the highest error, but it ended up not being true because with a complex medium like Marmousi, the Greedy sampling ended up being stuck in local minima and repeatedly sampled the same high-error regions. This is why adaptive residual sampling and the RL sampling performed better as they inherently balanced more exploration.
If we had more time for improvements/had to do this again, we would definitely train the RL agent more, since we feel that because of the time constraints, the agent performed worse than we think it can, so giving it more time and more compute will really help. Another idea would be to have a more complex state representation such as including a history of past residuals, which might yield even stronger policies. Lastly, testing the model on noisy data would test the

robustness of the model and simulate realistic field conditions. We would also use a Multi-Modal Architecture instead of just a standard MLP policy, introducing a CNN to process the residual grid. This would allow the agent to see the spatial heatmaps and make everything more effective.
We learned a lot from this project. The biggest takeaway was that it is not enough to just add a loss term to the model making it "physics-informed", the way data is sampled and fed into the model is also very important as we saw with our testing. This goes into the second lesson which is that the intuitive strategy such as sampling only the "hardest" points can and does backfire when the landscape is complex, hence why a balance between exploration and exploitation is necessary. And lastly we learned that an RL agent can be effectively used as a sort of an optimizer for other neural networks. Overall this project was very fulfilling and we learned a lot while doing it!

9 Citations

References

[1] Gillian R. Foulger, Miles P. Wilson, Jon G. Gluyas, Bruce R. Julian, and Richard J. Davies. Global review of human-induced earthquakes. *Earth-Science Reviews*, 178:438–514, 2018.

[2] Z. Song. Rl-pinns: Reinforcement learning-driven adaptive sampling for efficient training of pinns. *arXiv preprint arXiv:2504.12949*, 2025.

[3] Marmousi2 velocity model dataset. https://zenodo.org/records/14233581. Accessed: 2025-12-10.

[4] Fourier feature mapping enables mlps to learn high-frequency functions. https://medium.com/syncedreview/fourier-feature-mapping-enables-mlps-to-learn-high-frequency-functions-in-low-dimensional-domains-3d7ea03fd1a0. Accessed: 2025-12-10.