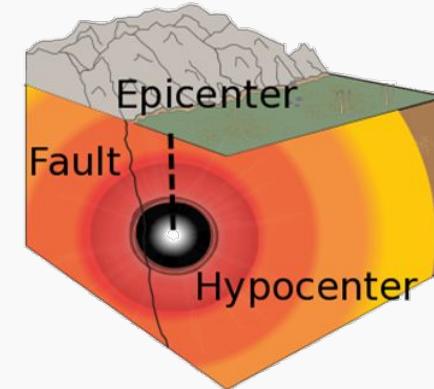


# Physics-Informed Source Localization via RL-driven Adaptive Sampling PINNs

Lucas Zheng, Yash Kothari, Kevin Pan, Fillip Cutiuba



# Table of contents

01

Background &  
Motivation

03

PINNs

02

Dataset &  
Problem Setup

04

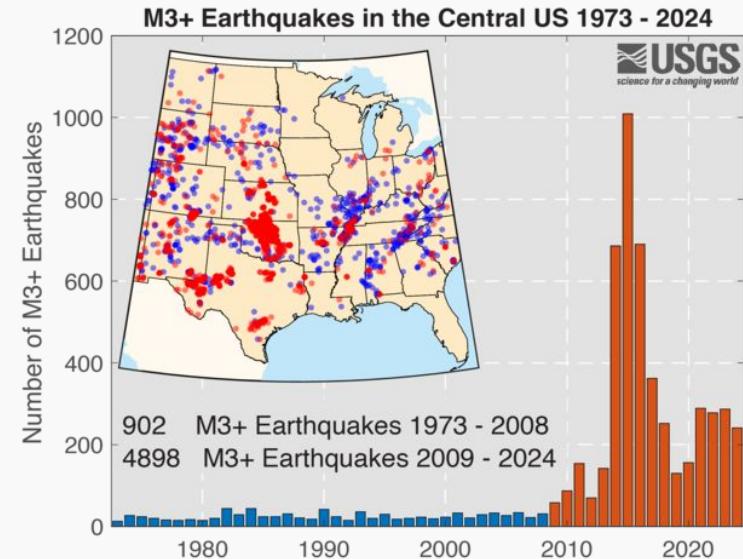
RL + Adaptive  
Sampling

01

# Background & Motivation

# Background

- Significant rise in induced earthquakes across U.S. since 2009
- Correlates with a rise in fluid injection, fracking, or geothermal operations
- Smaller seismic events (< M4 earthquakes) still pose significant regulatory and operational risks
- Determining source location of microseismic activity is critical for safety
  - Used to prevent nearby induced activity
  - Early detection of stronger seismic activity
- **Physics-Informed Neural Networks (PINNs) offer a solution to solving the source localization problem**

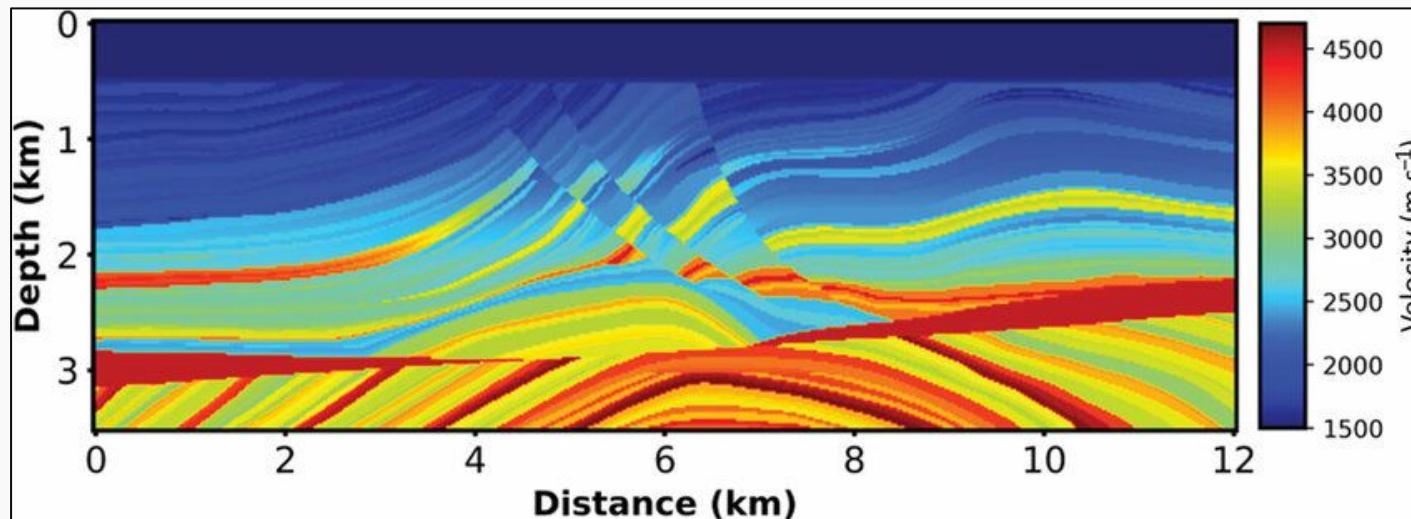


02

# Dataset & Problem Setup

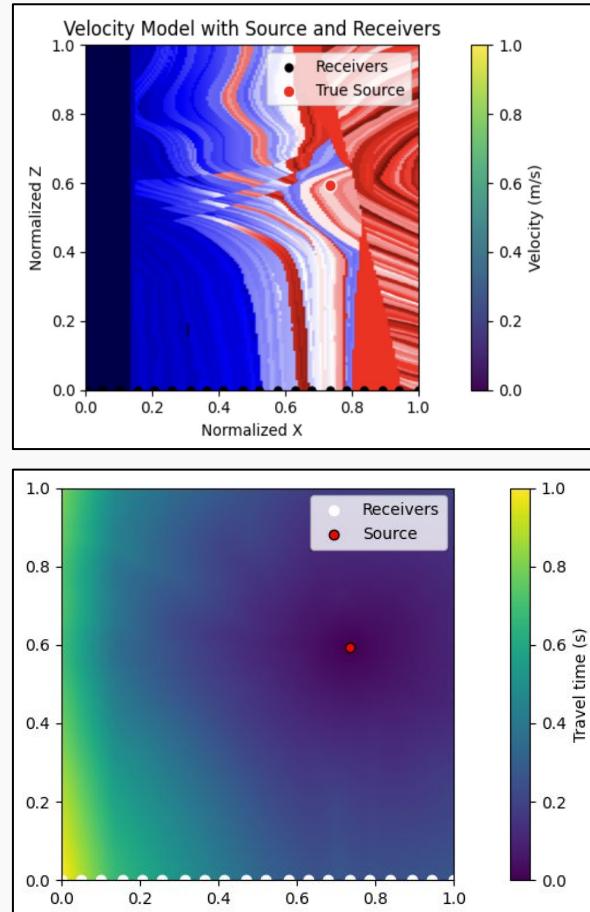
# Dataset & Problem Setup

- **Marmousi2 velocity model:** synthetic 2D map of seismic velocities simulating complex geology, designed by geophysicists
- We downsample (by taking every 20th grid coord) to form a 2D subsurface domain with spatially varying wave speed
- Roughly 38 mil grid points to 96k from downsampling



# Dataset & Problem Setup

- True source location is randomly selected within domain
- 20 receivers are placed uniformly along the surface ( $z=0$ ), ground truth travel times are calculated using numerical methods of solving the Eikonal equation (to simulate actual receivers)
- Typically, Eikonal eq. solvers are given the velocity map and source to solve the traveltime field, here we solve (a harder version) of the inverse problem - given velocity map and surface receiver picks (parts of traveltime field), determine the source
- **Objective: Recover the source locale using only travel times at receivers and the underlying velocity map**



03

# PINN

# PINN Formation

Objectives:

- **Estimate unknown source location ( $x_s, z_s$ )**  
from surface receiver picks and velocity map
- **Predict travel time field  $\tau(x, z)$**  by solving  
the Eikonal equation with a neural network
- **Eikonal equation:**  $|\nabla \tau(x, z)|^2 = s(x, z)^2$ 
  - $\tau(x, z) = \text{traveltime from a source to point } x$
  - $v(x, z) = \text{wave velocity at position } (x, z)$
  - $s(x, z) = 1 / v(x, z) = \text{slowness}$

Let  $\tau(x, z)$  solve

- $|\nabla \tau(x, z)|^2 = s(x, z)^2$  (Eikonal eq.)
- $\tau(x_s, z_s) = 0$  (boundary condition:  
travel time at source should be 0)
- $s(x, z) = 1 / v(x, z)$

where

- $v(x, z)$ : velocity map of waves from  
downsampled Marmousi2 velocity  
model

# PINN Architecture

Model Architecture:

Fourier Feature Encoding (256 mapping size, scale = 10.0)

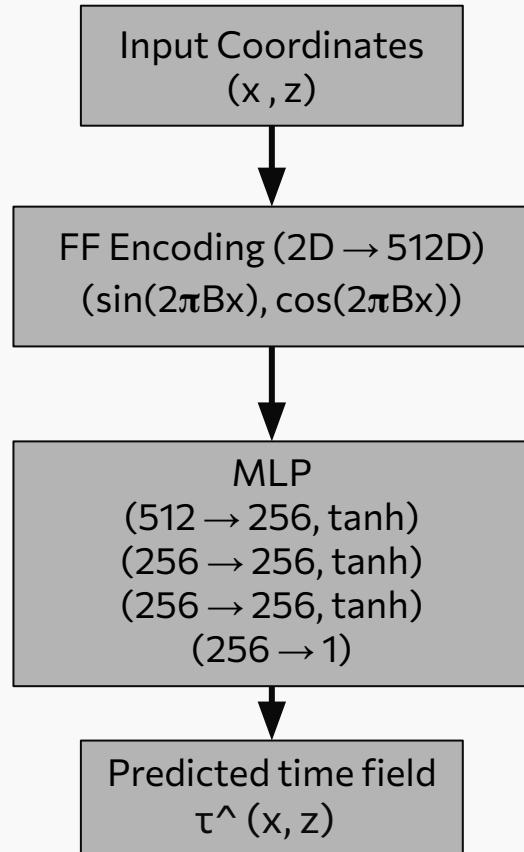
- Transforms 2D spatial inputs ( $x, z$ ) into 512-dimensional high-frequency space using sinusoidal projections
- Helps PINN learn high-frequency spatial variations in the velocity model, in order to model fine-scale wavefront behavior

MLP

- 3 hidden layers, 256 neurons per, Tanh activations

*Inputs: Normalized spatial coordinates ( $x, z$ )*

*Outputs: Predicted traveltime  $\tau^*(x, z)$*



# Loss Components

$$L = w_{pde} L_{pde} + w_{data} L_{data} +$$

$$w_{src} L_{src}$$

where  $w_{pde}$ ,  $w_{data}$  are learnable loss weights to let the model figure out how strongly to enforce PDE and data conditions

- Also apply weight schedule on

$w$  starts lower then increases

## Loss Components

(1) *PDE Loss (enforces physics w/ Eikonal eq.):*

$$\nabla \tau(x, z)^2 - s(x, z)^2 \rightarrow 0$$

(2) *Data Loss (matches predicted traveltimes to observed picks from receivers):*

$$\tau^*(x_i, z_i) \approx t_i^{(obs)}$$

(3) *Source Loss (boundary condition: enforces predicted traveltime at source location to be 0):*

$$\tau^*(x_s, z_s) = 0$$

# Loss Components

(1)

```
# pde loss: how well model time predictions at the collocation point satisfy eikonal equation
tau_c      = model(colloc)
grads = autograd.grad(tau_c.sum(), colloc, create_graph=True)[0]
loss_pde = ((grads.pow(2).sum(1,True) - get_slowness(colloc).pow(2))**2).mean()
```

(2)

```
# data loss: how well model time predictions at receiver positions satisfy true times
tau_r = model(rec_coords).squeeze()
loss_data = (tau_r - rec_tts_t).pow(2).mean()
```

(3)

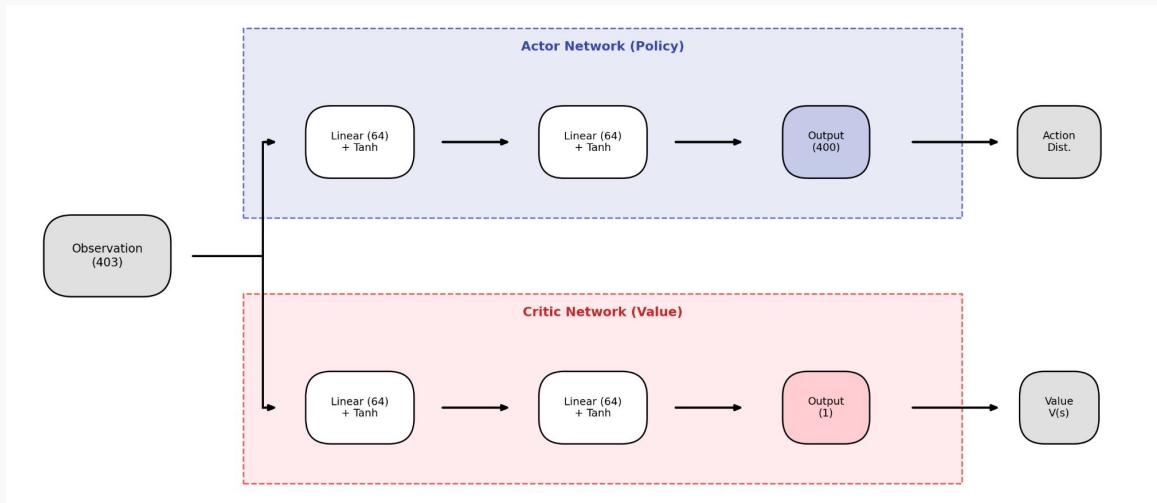
```
# source loss: how well model time prediction at source satisfies the boundary condition time = 0
tau_s = model(src_param.unsqueeze(0))
loss_src = (tau_s**2).mean()
```

03

# RL + Adaptive Residual Sampling

# RL

- Standard Heuristics are Limited
- Random  $\Rightarrow$  Wasted Computation on “easy” areas
- Residual Sampling  $\Rightarrow$  Resamples noisy gradients & singularities inhibiting global convergence
- Goal: Replacing rules with policy for faster long-term convergence



# Residual Sampling vs. RL Sampling

## Adaptive Residual Sampling

- Instead of always randomly sampling points to train, every 100 epochs we choose collocation points based on where PINN violates physics the most
- Turn point-wise PDE residuals into a probability distribution, sample from this distribution to get training points

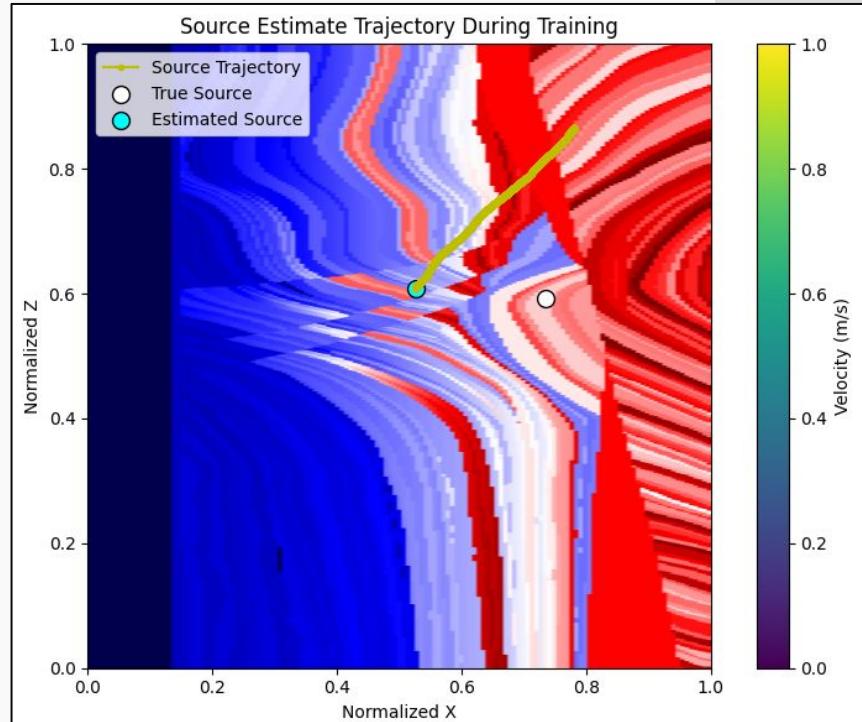
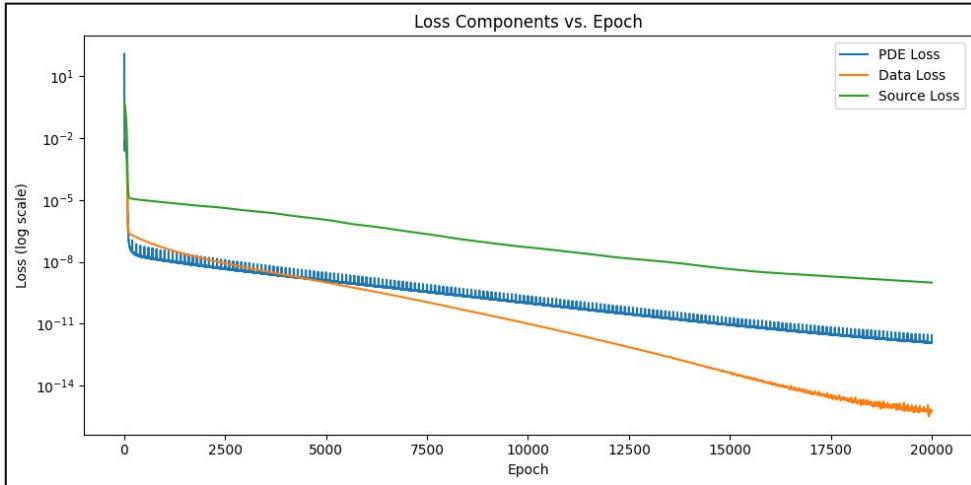
## RL Sampling w/ PPO

- **Framing:** Collocation Point  
Selection as seq. decision making problem
- **State:** PDE Residuals at fixed candidates + current loss values
- **Action:** Selection out of 400 grids (20x20) to add to collocation set
- **Reward:** Loss reduction across steps - localization\_error

04

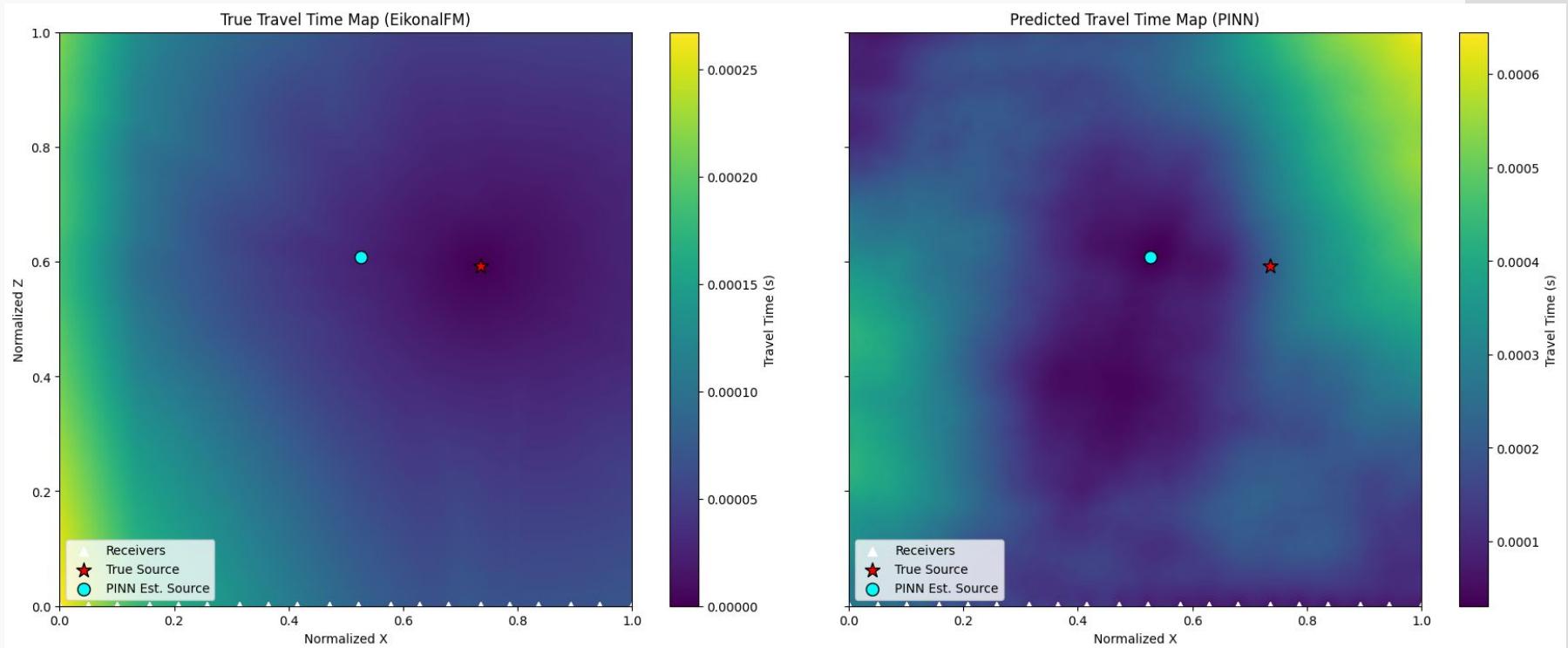
# Results & Conclusion

# PINN + Adaptive Sampling

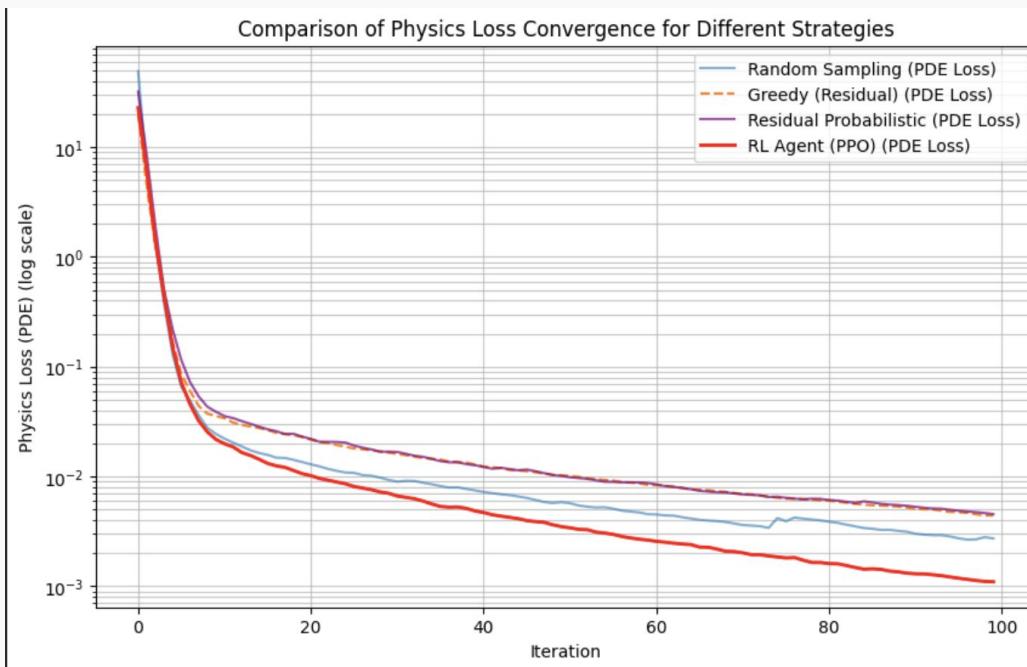


- Spikes from adaptive sampling when sampler samples from training points where physics is violated the most
- Issue we observe is that with this velocity map, predicted source almost always travels “down and to the left”

# PINN + Adaptive Residual Sampling



# RL Loss Convergence Comparison



# Conclusion / Reflection

- **The Core Challenge:** The "Coupled" Optimization Problem.
  - **Standard Methods:** Numerical Eikonal equation solvers require a **fixed, true source** to mathematically define the travel-time field (Source  $\Rightarrow$  Field)
  - Our Bottleneck: Our task attempted to solve for both traveltime field and true source simultaneously with only receiver data
  - Result: This removed the necessary "anchor" for the physics. Because the PINN had to learn the field and the source at the same time, it faced an unstable "moving target" rather than a clear optimization path
- Next steps:
  - Adopt a more traditional approach, optimize the predicted source while relying on eikonal fm approach to calculate traveltime field rather than learning it

