The 1d heat equation with PINNs

elphaim

January 23, 2026

1 Basic problem

The 1d heat equation is a PDE given by

$$u_t - \alpha\, u_{xx} = 0\,, \tag{1}$$

where $u(x,t)$ is the temperature profile, α is the thermal diffusivity of the material, and we should supplement the above with some boundary/initial conditions (to be fixed below).

The forward problem consists of fixing α and solving for $u(x,t)$ using PINNs [1]. This involves using a loss function that is a sum of a loss for the PDE residual and a loss for the boundary conditions. We use **MSE for the losses**.

We also need to specify collocation points in the interior of the space-time domain and at the boudary. These can either be fixed (uniformly or by Latin Hypercube) or adaptively sampled.[1]

The inverse problem is to find α given (possibly sparse/noisy) measurements of u. In this case, the loss function involves the PDE residual loss and the boundary losses together with a term enforcing agreement with the observed data:

$$\mathcal{L}_m = \frac{\lambda_m}{N_m} \sum_{i=1}^{N_m} |u(x_m^i, t_m^i) - u_m|^2\,. \tag{2}$$

Let's now specify the boundary conditions. We will be looking at an isotropic and homogeneous rod of length L so α is constant and the domain of the problem is $\Omega = [0,L]\times[0,T]$. We impose Dirichlet boundary conditions

$$u(0,t) = u(L,t) = 0 \tag{3}$$

and set

$$u(x,0) = \sin(\pi x) \qquad \text{and} \qquad L = 1 \tag{4}$$

so that the problem has an analytic solution:

$$u(x,t) = \sin(\pi x)\, e^{-\alpha\pi^2 t}\,. \tag{5}$$

This will provide a ground truth for the PINN.

[1] We can even work with independent subregions that we glue back together at the end, especially if we expect steep gradient solutions to matter.

2 Concrete implementation

We will simulate measurements by temperature sensors placed at regular intervals of size $L/10$ along the length of the rod and queried 10 times during the experiment. This fixes $N_m = 100$. We add random Gaussian noise to these measurements,

$$u_m = u + \epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{6}$$

Let us assume that $\sigma \approx 1\%$ of the signal amplitude, so as to keep a high Signal-to-Noise Ratio of around 40 dB. The effect of varying noise on the solution is studied below.

We fix $T = 1$ and use 10k collocation points sampled uniformly in the domain $\Omega = [0, 1] \times [0, 1]$. On the boudary we use $N_{\text{bc}} = 100$ for a total of 200 points (100 for each boundary). We also set $N_{\text{ic}} = 200$ for the initial surface at $t = 0$. The dataset is shown in Figure 1

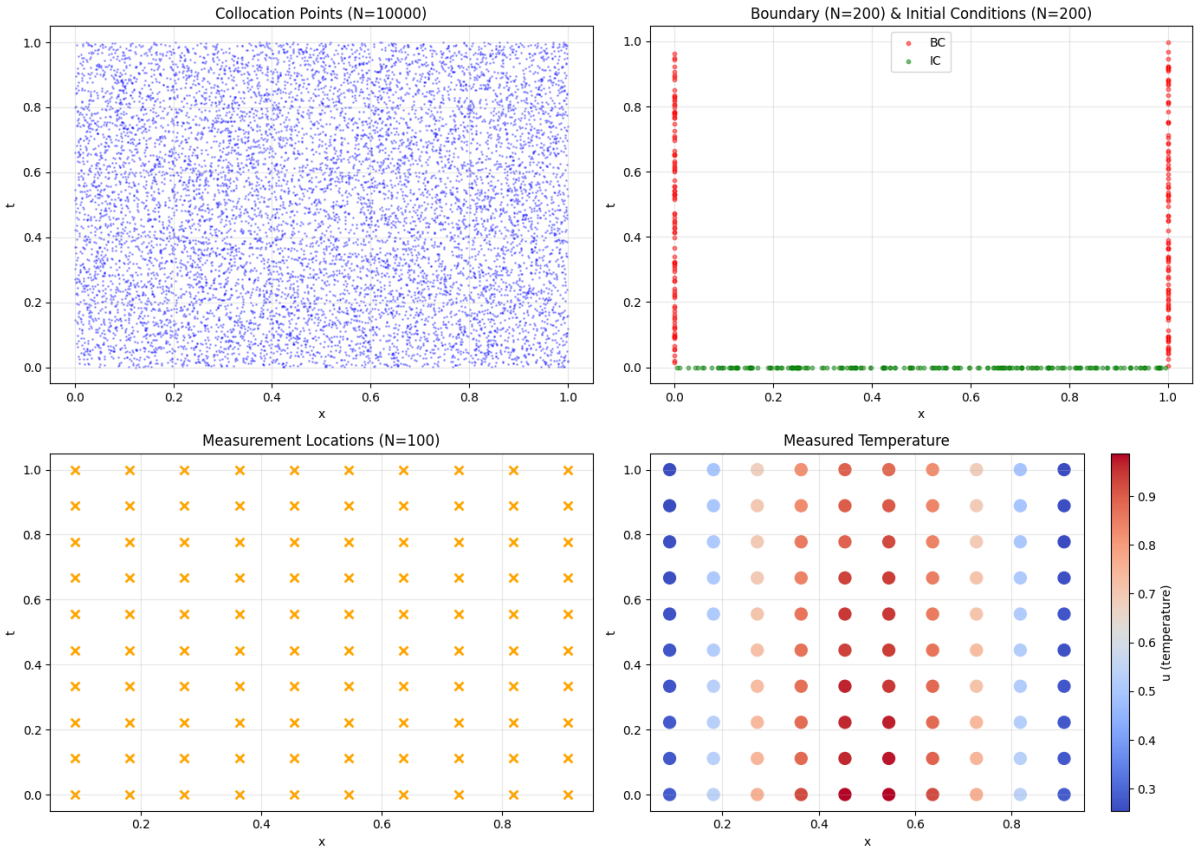

Figure 1: Datapoints for the 1d heat equation

The NN architecture we use is as follows:

- Input: $(x, t) \in \Omega$
- Hidden: 4 layers of 50 neurons
- Activation: tanh (maybe try scaled activation with learnable parameter later)
- Output: $u(x, t)$ for the forward problem, $[u(x, t), \alpha]$ for the inverse problem

Metrics for success will be low L2 relative error for the solution

$$\frac{\|u_{\text{pred}} - u_{\text{true}}\|_2}{\|u_{\text{true}}\|_2} < 1\% \tag{7}$$

and α recovered to within 5% error for the inverse problem

$$\frac{|\alpha_{\text{pred}} - \alpha_{\text{true}}|}{\alpha_{\text{true}}} < 5\% \tag{8}$$

We fix $\alpha_{\text{true}} = 0.01$.

3 Forward problem

3.1 Training without adaptive weights

We fix all loss weights $\lambda_f = \lambda_{\text{bc}} = \lambda_{\text{ic}} = 1$ and train the model for 5000 epochs using the Adam optimizer. The training time is $\approx$**270 seconds** on CPU. The losses are shown in Figure 2.

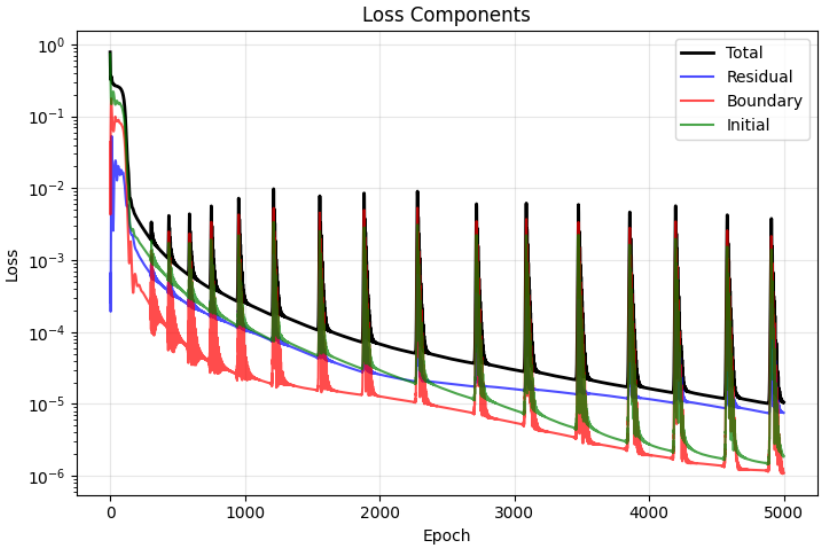

Figure 2: Loss curves for the forward problem (no adaptive weights)

The spikes match the ones seen in the norm of the derivative of the loss functions with respect to the model's parameters $\|\nabla_\theta \mathcal{L}_i\|_2$, as shown in Figure 3 This feature signals the presence

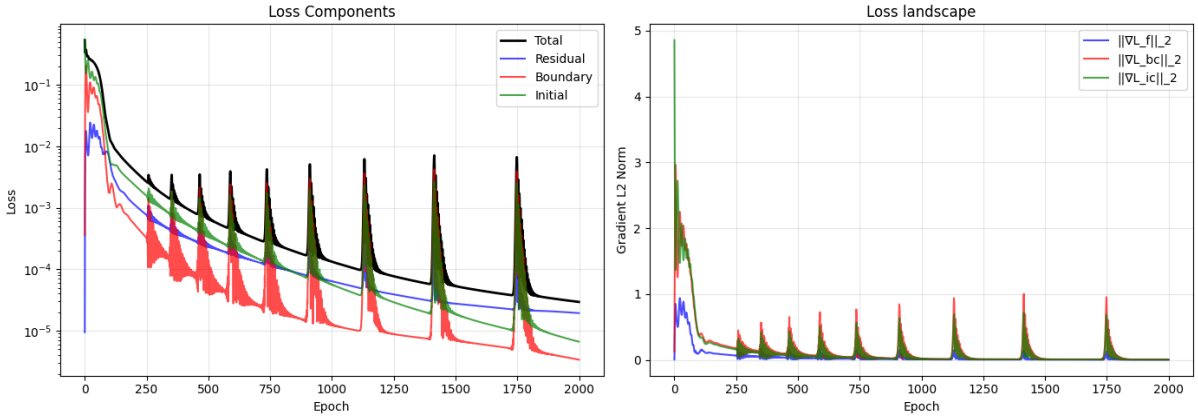

Figure 3: Loss curves and gradient norms (no adaptive weights)

of **high-curvature regions** in the loss landscape. Such regions are primarily associated to boundary/initial conditions, as can be inferred from the fact that their gradient norms dominate

over the residual one. This is a *known feature of PINN optimization*, where balancing various components of the loss function can push the optimizer into steep regions.

The temperature profile $u_{\text{pred}}(x,t)$ predicted from the model is shown in Figure 4, together with the ground truth and the plot of the absolute error $|u_{\text{pred}} - u_{\text{true}}|$.

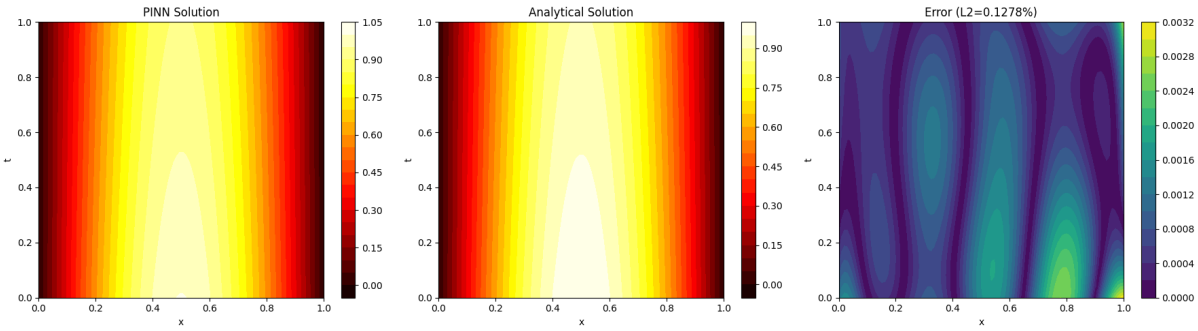

Figure 4: Temperature profile prediction and error (no adaptive weights)

Conclusion: the training dynamics are under control. The temperature profile is accurately predicted by the model with **L2 = 0.13%**, which is well below our 1% threshold.

3.2 A word on optimizers

In the literature, it is common to find PINN training regimen which involve **two optimizers**. One starts the training with Adam to find a basin in the loss landscape, and then finish training with L-BFGS to exploit second-order features around local minima. We implement this in the training loop, where we switch from Adam to L-BFGS once the total loss falls under 5×10^{-4}.

Doing so proves to be *very efficient*. The training only takes **≈60 seconds** on CPU, and the L-BFGS terminal step pushes the loss curves down by more than 2 orders of magnitude, see Figure 5.

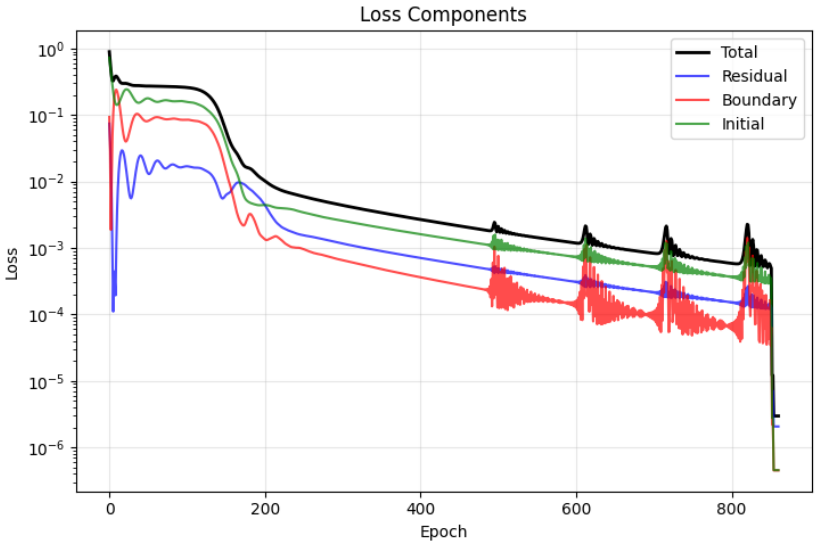

Figure 5: Loss curves with L-BFGS ending (no adaptive weights)

The reconstructed solution after less than 1000 epochs displays remarkable agreement with the ground truth, with a relative L2 error of merely **0.07 %**.

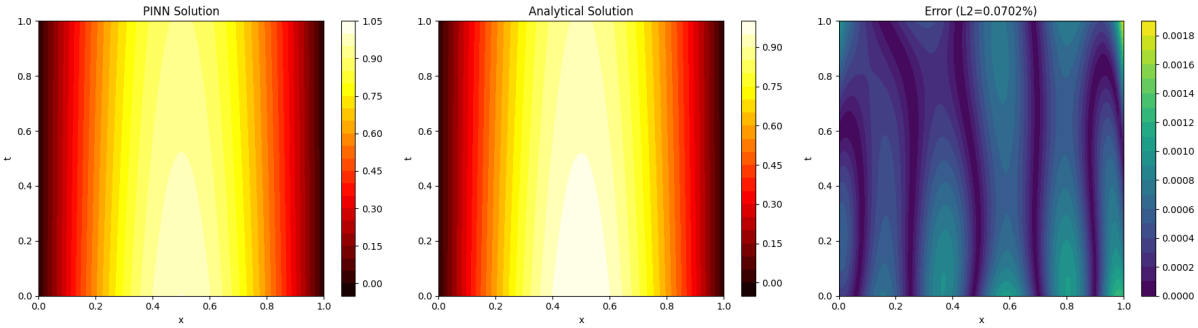

Figure 6: Temperature profile prediction and error L-BFGS ending (no adaptive weights)

Conclusion: L-BFGS can help push the accuracy higher while cutting down on training time.

3.3 Training with adaptive weights based on gradient magnitudes

We now train the model with adaptive weights. We start with $\lambda_f = \lambda_{\text{bc}} = \lambda_{\text{ic}} = 1$ and update their values every 100 epochs according to the following steps:

1. compute the L2 norm of the loss gradients: $\|\nabla_\theta \mathcal{L}_i\|_2 \leftarrow \hat{\lambda}_i$
2. smoothen with EMA: $s\,\hat{\lambda}_i^{\text{old}} + (1-s)\,\hat{\lambda}_i^{\text{current}} \leftarrow \hat{\lambda}_i$
3. update: $\text{mean}_i(\hat{\lambda}_i)/\hat{\lambda}_i \leftarrow \lambda_i$

The EMA smoothing factor is set to $s = 0.9$. This algorithm for adaptive weights is meant as a proxy for the full Neural Tangent Kernel approach put forward in [2].

We train with Adam & L-BFGS. The training takes ≈**40 seconds** on CPU, and the loss curves are shown in Figure 7.

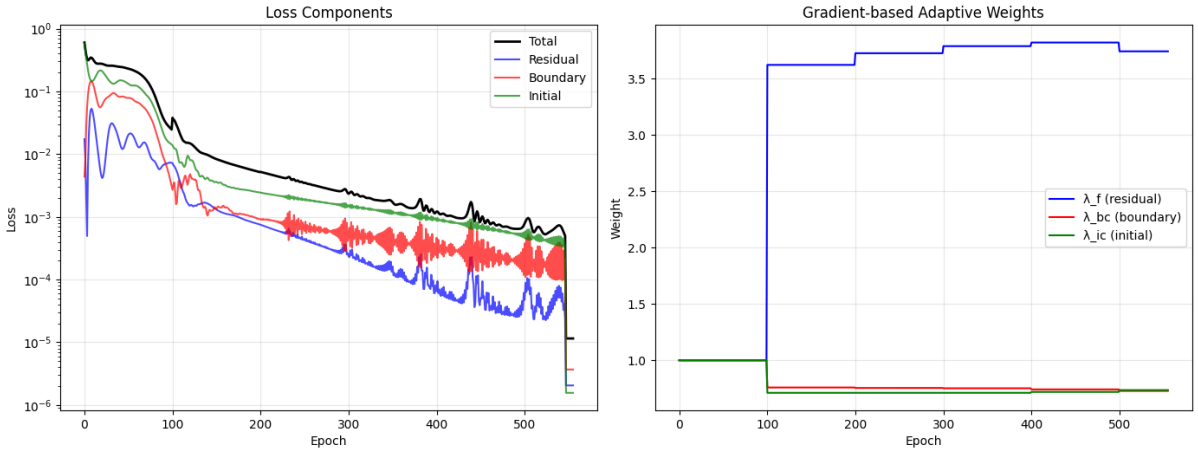

Figure 7: Loss curves for the forward problem (with adaptive weights)

We see that the residual loss is the lowest throughout most of the training, so our adaptive alogrithm increases λ_f while decreasing $\lambda_{\text{ic/bc}}$ in order to maintain the total loss as low as possible. The reconstructed temperature profile is shown in Figure 8.

Conclusion: the L2 relative error has slightly increased to **L2 = 0.11%** (still far below our 1% threshold). The fact that adaptive weights are not very helpful in the present context is likely

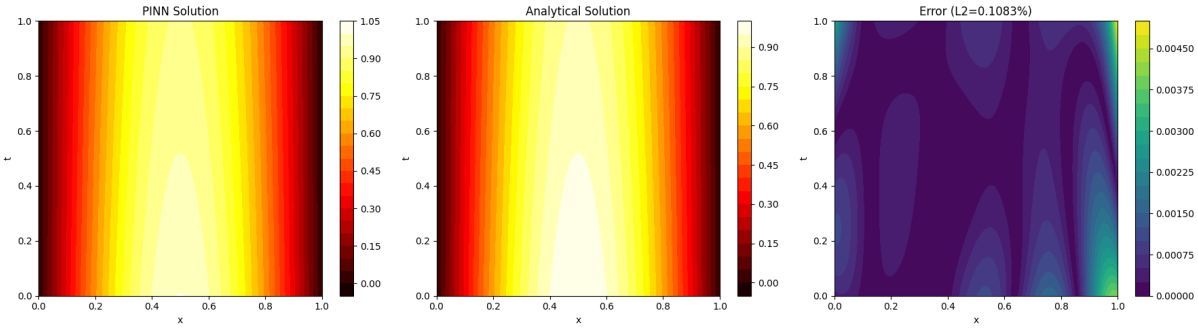

Figure 8: Temperature profile prediction and error (with adaptive weights)

due to our problem having smooth solutions which are easy to recover with a vanilla PINN. We have checked that the adaptive weighting is implemented correctly.

4 Inverse problem

We now study the inverse problem where we try to recover both the temperature profile and the value of $\alpha_{\text{true}} = 0.01$ using temperature measurements. Our starting guess is $\alpha_{\text{guess}} = 0.02$.

4.1 Training without adaptive weights

We train for 5k epochs with Adam & L-BFGS, which takes ≈**60 seconds** on CPU. loss curves in Figure 9.

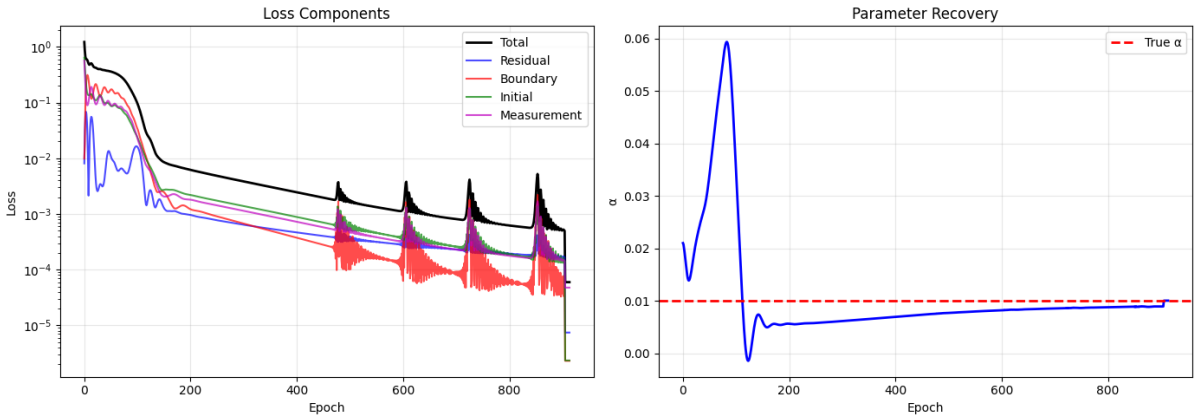

Figure 9: Loss and reconstructed α for the inverse problem (no adaptive weights)

The reconstructed solution is shown in Figure 10.

Conclusion: we recover α **within 0.36%** error, and the net converges quickly.

4.2 Training with adaptive weights

Results are shown in Figures 11 and 12.

Conclusion: adaptive weights help for the inverse problem, we recover α **within 0.05%** error.

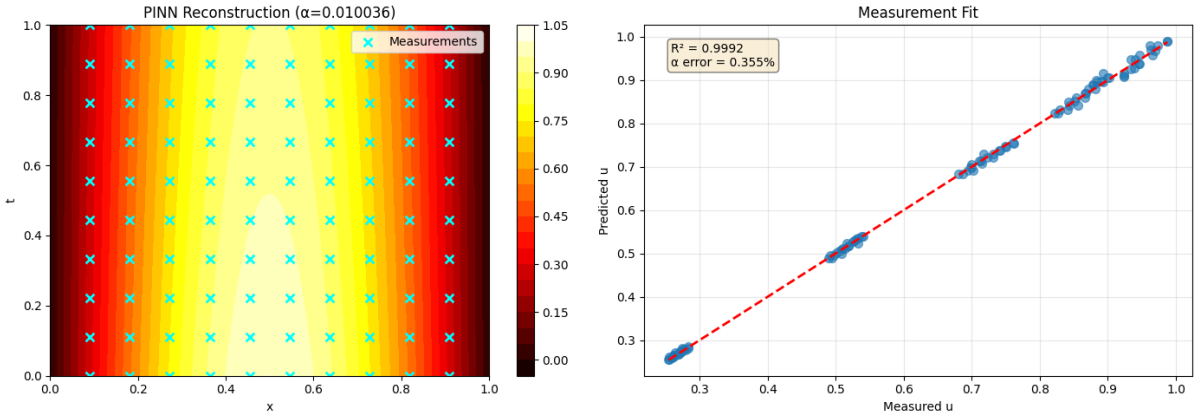

Figure 10: Temperature profile prediction and deviation from measures (no adaptive weights)

5 Experiments

5.1 Effect of noise

We train the neural net for 5k epochs with Adam & L-BFGS on data that has been generated with varying amount of noise

$$\sigma \in \{0.1\%, 1\%, 5\%, 10\%\} \text{ of signal amplitude} \tag{9}$$

We use adaptive weights for the training. It is interesting to note that for high noise levels (>5%) our trainer never switches to L-BFGS since the total loss remains above the 5×10^{-4} threshold.[2] To illustrate this, the loss curves for $\sigma = 10\%$ are shown in Figure 13.

The errors for parameter recovery are shown in Figure 14.

Conclusion: the noise level impacts our ability to recover α, with high noise (10%) pushing us past the 5% error threshold. With noisy data, longer training may be required.

5.2 RNG sensitivity

We train 10 models on the inverse problem, all with different RNG seeds. The resulting mean and std for the recovered parameter is

$$\alpha = 0.010089 \pm 0.000027 \tag{10}$$

Conclusion: RNG has very little effect on parameter recovery.

5.3 Initial guess sensitivity

We train a model on three inverse problems, each with a different value for the initial guess

$$\alpha_{\text{guess}} \in \{0.005, 0.05, 0.5\} \tag{11}$$

The results are shown in Figure 15 **Conclusion**: The initial guess has a *high impact* on parameter recovery.

[2]This suggests to implement a dynamic L-BFGS switch based on e.g. stagnation of the loss.

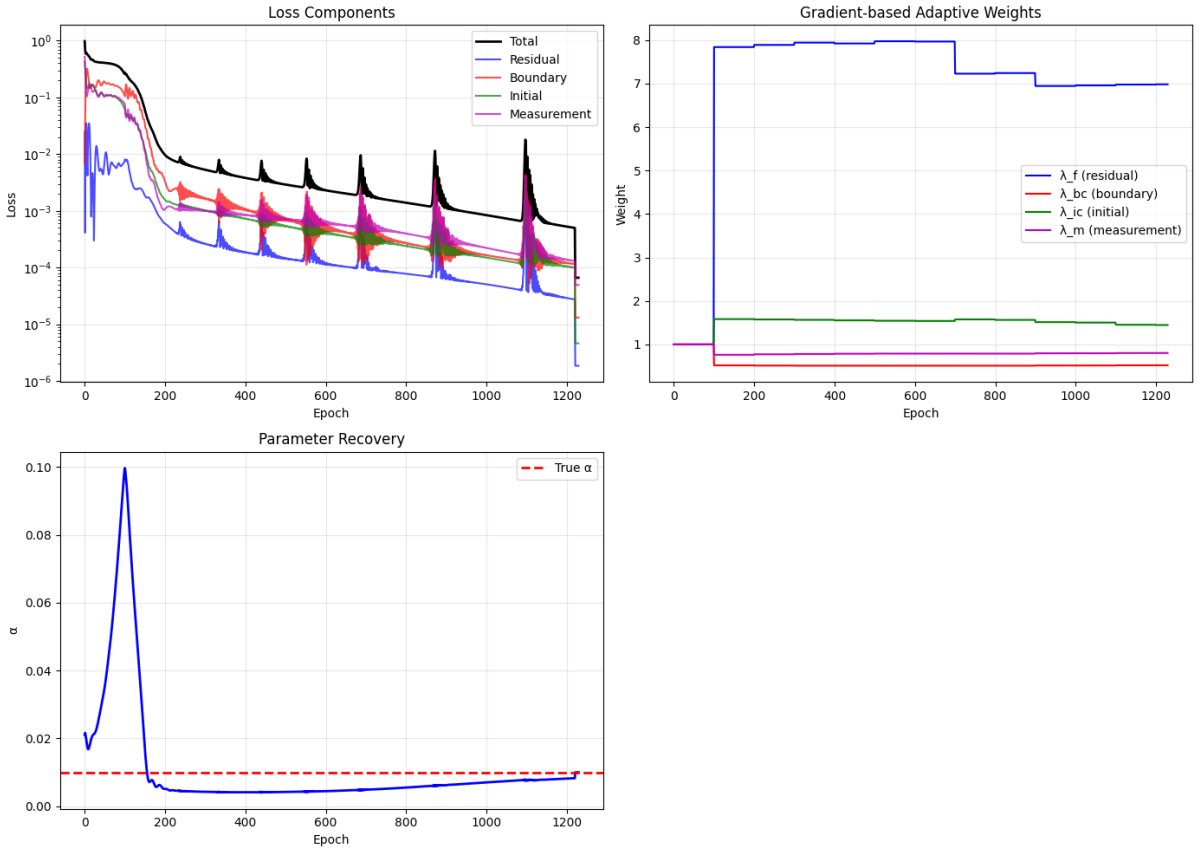

Figure 11: Loss and reconstructed α for the inverse problem (with adaptive weights)

References

[1] M. Raissi, P. Perdikaris, and G. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[2] S. Wang, X. Yu, and P. Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

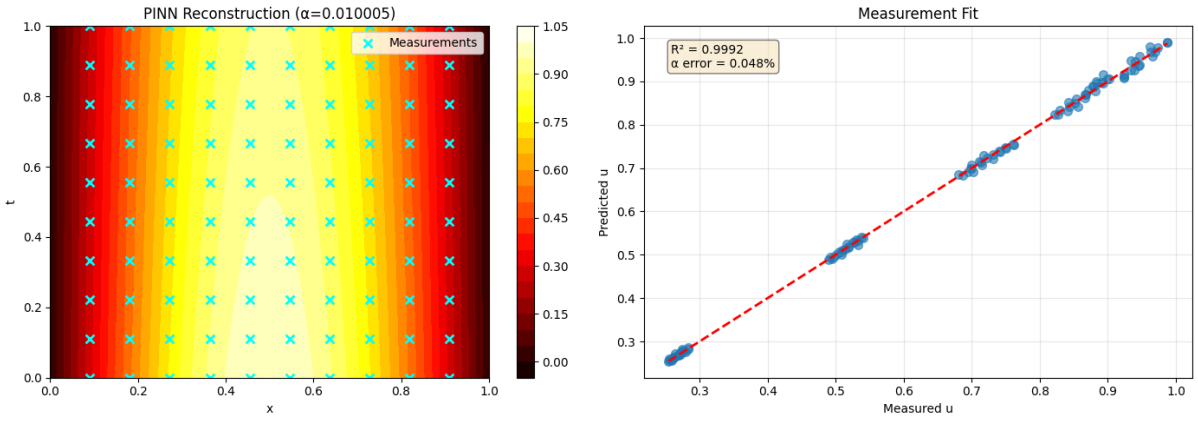

Figure 12: Temperature profile prediction and deviation from measures (with adaptive weights)

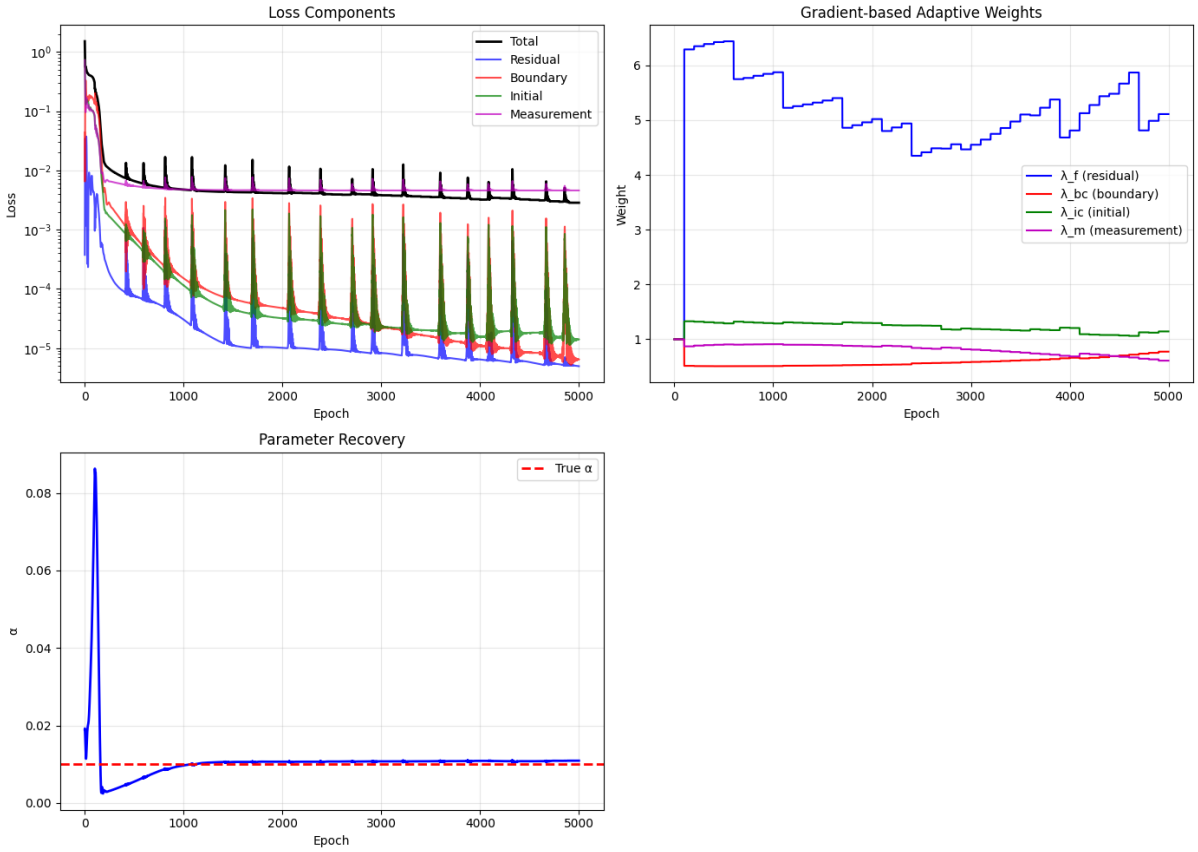

Figure 13: Loss curves for noisy data (with adaptive weights)

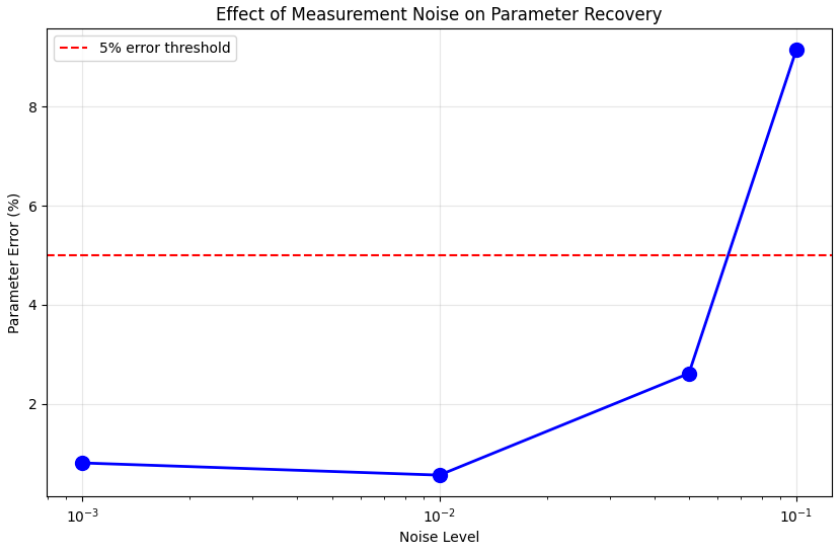

Figure 14: Error on parameter recovery as a function of noise level

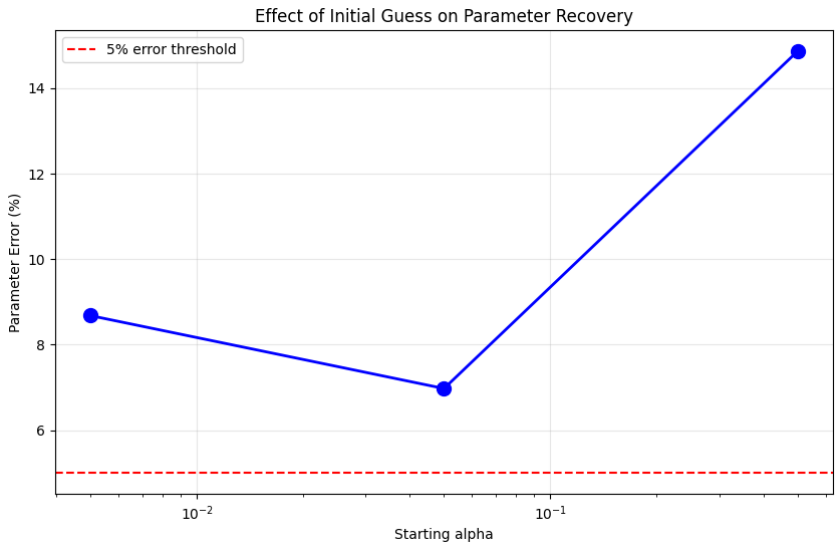

Figure 15: Error on parameter recovery as a function of α_{guess}