Problem sheet

You've got code for a minimal Ritz-method solver for the equation

$$-u''(x) = f(x), \quad x \in (0,1), \quad u(0) = u(1) = 0,$$

where $f(x) = -9\pi^2 \sin(3\pi x)$. The solution uses:

- A neural network with a single hidden layer,
- A smooth activation function (default: tanh),
- Stratified Monte Carlo integration,
- Multiplicative enforcement of Dirichlet boundary conditions.

Goal. In this session, you'll modify the code and investigate how architecture, sampling, and the PDE data affect training and accuracy. This sheet offers *suggested directions*, but you're free to follow your curiosity - the aim is for you to get a feel for how changes in the problem can affect training, challenges in identifying solutions. Several aspects you can adjust are the following - bear in mind that changing one part of the problem. may mean you have to change others to get good solutions.

Architecture

A.1 Initialisation. You can try with basically any function tensorflow can evaluate, the typical ones are

- tanh (baseline),
- sigmoid,
- softplus,
- ELU.

If you try with something like $\exp(x)$ or $\mathrm{ReLU}(x)$, you will likely have big problems, can you see why?

A.2 Initialisation.

- You can change the initialisations using the "black-box" initialisers, or doing things by hand.
- Keep in mind that different initialisations are more appropriate for different activation functions.
- You can plot the initialised NN to see what it's initial behaviour is like, and see what features do/don't arise.

A.3 Size. Experiment with:

- Width (Neurons per layer)
- Depth (number of layers)

Do these help or hinder?

B. Quadrature

B.4 Try alternative quadrature:

- Pure Monte Carlo (uniform points),
- Fixed midpoints,
- Sobol (if you're familiar).
- Deterministic quadrature (in particular, how Deep Ritz and PINNs behave differently)

How does this change the convergence?

B.5 Plot variability across runs. Fix your model but run training several times with different seeds. Plot error curves. How much does training vary?

C. RHS variations

Changing the right-hand side may lead to very different needs when training (architectures, optimisers, etc.) Some more extreme examples, just by changing the right-hand side could be seen with

C.6 Highly oscillatory RHS.

C.7 RHS with discontinuities (solutions wont be C^2).

C.8 Right-hand sides with (light) singularities.

D. Boundary conditions

D.9 In the Deep Ritz Method, you can add natural BCs in the weak formulation, remove the cutoff at $x = 1$, and add another term in your loss like $-(u^*)'(1)u_{NN}(1)$. This will give the same solution, how does it affect training?

D.10 What happens if you replace the cutoff with penalties on the boundary? Or one method at one end, another method at the other.

E. Pushing further

E.11 The code can easily be adapted to PINNs, non-linear PDEs, introducing (synthetic) data or other strange things in the loss.

E.12 You can include optimising over some other variables (design/inverse problems). You can invent a problem like, "what should my coefficients be to maximise the PDE solution at a certain point?", "can I reclaim a parameter given (noisy) data?".

E.13 Extending to higher dimensions is relatively straightforward too - norms on the boundary become integrals, etc.

Wrap-up. Which modifications made the most difference? Any surprises? What would you try next?