Lecture 2b: Variational NNs and the Deep Ritz Method

Chris Budd and Aengus Roberts [1]

[1]University of Bath

CWI, October 2025

Some papers to look at

- Grossman et. al. *Can PINNS beat the FE method?*
- Shin et. al. *On the convergence of PINNS for linear elliptic and parabolic PDEs*
- E and Yu *The Deep Ritz Method*
- Dondl et. al. *Uniform convergence guarantees for the deep Ritz method for nonlinear problems*
- Jiao, Lai, Lo, Wang, Yang, *Error analysis of deep Ritz methods for elliptic equations*
- Müller and Zeinhofer, *Deep Ritz Revisited*

Motivation: Calculus of variations

Seek to solve PDE problems eg. of the form

$$-\epsilon^2 u_{xx} + u = 1, \quad u(0) = u(1) = 0, \quad \Omega = [0,1] \quad (*)$$

In the previous lecture we tried to solve this PDE using a PINN with the PDE residual as the loss function

In this lecture we consider using ideas from the calculus of variations to find a loss function

We will also introduce some ideas which will be needed when we study **Neural Operators** in later lectures.

Functional minimisers

Consider the functional

$$F(u) = \int_\Omega \frac{\epsilon^2 u_x^2}{2} + \frac{u^2}{2} - u \, dx.$$

Claim $F(u)$ is minimised when $u = u^*$ is the (unique) solution of the PDE (*)

Question: Over what space is $F(u)$ minimised.

Answer: Space is $H_0^1(\Omega)$

$$H_0^1(\Omega) = \{u : \int_\Omega u_x^2 \, dx < \infty, \quad u(0) = u(1) = 0.\}.$$

Proof

Set $u = u^* + \phi$ where $\phi \in H_0^1$ is arbitrary.

$$F(u) = F(u^*) + \int_\Omega \epsilon^2 u_x^* \phi_x + u^* \phi - \phi \, dx + \frac{1}{2} \int_\Omega \epsilon^2 \phi_x^2 + \phi^2 \, dx$$

Integrate by parts and use the boundary conditions to give:

$$= F(u^*) + \int_\Omega (-\epsilon^2 u_{xx}^* + u - 1)\phi \, dx + \frac{1}{2} \int_\Omega \epsilon^2 \phi_x^2 + \phi^2 \, dx.$$

$$= F(u^*) + 0 + \mathit{positive}$$

So $F(u)$ has a **global** minimum at $u = u^*$

Some definitions

Strong form of the PDE:

$$-\epsilon^2 u_{xx} + u = 1.$$

Weak form of the PDE

$$\int_\Omega \epsilon^2 u_x \phi_x + u\phi - \phi \, dx = 0 \quad \forall \quad \phi \in H^1$$

Also

$$\langle u, v \rangle = \int_\Omega u_x v_x \, dx, \quad \|u\|_{H^1}^2 = \langle u, u \rangle.$$

With natural extensions to higher dimensions

Approach 1: Finite Element Methodology

Express $u(x)$ as a Galerkin approximation:

$$u(x) \approx U(x) = \sum_{i=0}^{N} U_i(t)\, \phi_i(x)$$

with $\phi_i(x)$ **Not** globally differentiable *locally supported, piece-wise polynomial spline functions*

U is in the linear space $\mathcal{V}$ spanned by the functions $\phi_i(x)$.

- Require U to satisfy the *weak form* of the PDE.
- If the PDE is linear, this leads to a linear system for the coefficients U_i of the approximation
- Solve this system using (for example) a conjugate-gradient method.

Features of the FE method

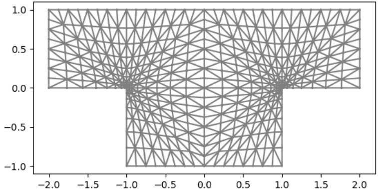
- Unique solution (if the PDE is linear)
- Have guaranteed error estimates, due to **Céa's lemma**, of the form

$$\|u - U\|_{H^1} < C(u)N^{-\alpha}$$

- Can reduce $C(u)$ and increase α using an adaptive approach.
- But .. Awkward in higher dimensions!

Meshes

Traditional PDE computations using Finite Element Methods use a computational mesh τ comprising mesh points and a mesh topology with $\phi_i(x)$ defined over the mesh



Meshes can be tricky to construct!
(Although ML methods can help a lot!)

Approach 2: The Deep Ritz Method [Weinan E and Bing Yu, (2017)]

Lovely Idea!!!

Let $y(x)$ be the output of a parametrised NN

$$y(x) = NN(x)$$

with Y the (nonlinear) set of functions parametrised by θ

Then set

$$y^* = argmin_Y F.$$

Allowing for the boundary conditions

Deep Ritz method for the Poisson equation

The **Deep Ritz Method** (DRM) seeks the solution u satisfying

$$y = \arg\min_{v \in H} \mathcal{I}(v),$$

where H is the set of admissible functions (trial functions) and

$$\mathcal{I}(v) = \int_\Omega \left(\frac{1}{2} |\nabla v(\mathbf{x})|^2 - f(\mathbf{x}) v(\mathbf{x}) \right) d\mathbf{x} + \beta \int_{\partial\Omega} (v(\mathbf{x}) - u_D)^2 d\mathbf{s}$$

The Deep Ritz method is based of the following assumptions:

- $y(\mathbf{x})$ is DNN based approximation of u which is in H^1 eg. ReLU
- A numerical quadrature rule for the functional using chosen quadrature points eg. random, optimal
- An algorithm for solving the optimization problem eg. SGD on random quadrature points

- Assume that $y(\mathbf{x})$ has enough regularity for F to be defined at any point x eg. $y(x) \in H^1$
- $ReLU$ is OK, but may prefer $ReLU^3$.
- Differentiate $y(\mathbf{x})$ once, exactly using the inbuilt chain rule
- Calculate $F(y)$ by using quadrature at quadrature points $\mathbf{X}_i$ (chosen to be uniformly spaced, or random)
- Train the neural net to minimise a loss function combining F and the boundary and (if needed) initial conditions
- Include known point values if available.

DRM vs.Finite Element

The DRM is superficially similar to the Finite Element method but has crucial differences as **the NN approximating subspace is nonlinear**, and the problem of finding the weights is non-convex and badly conditoned.

FE: linear

- Limited expressivity, reduced accuracy
- Adaptive only with effort
- Not equivariant
- Need a complex mesh data structure
- Convex with guarantees of uniqueness for many problems (and direct calculation using linear algebra)
- Work on saddle-point problems (eg. most problems)
- Good (a-priori and a-posteriori) guaranteed error bounds :

  Cea's Lemma: Bounds solution error by interpolation error on the FE space

DRM:nonlinear

- Very expressive (potential high accuracy for a small number of degrees of freedom)
- Self adaptive
- Equivariant
- Don't need a complex mesh data structure
- Don't work on saddle-point problems eg. most problems for example:

$$u_{xx} + u = 1, \quad u_x x + u^3 = 0.$$

- Don't have Céas Lemma

Comparing a DRM to an adaptive finite element method

If $\sigma(z) = ReLU(z)$ then

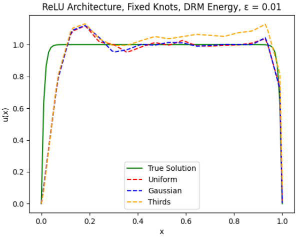
$$y(\mathbf{x}) = \sum_{i=0}^{W-1} c_i (a_i \mathbf{x} + \mathbf{b}_i)_+$$

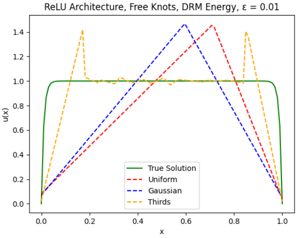
In d-dimensions this is a **piece-wise linear function**
SO .. in principle a ReLU network has the same expressivity as an adaptive Finite Element Method and should deliver the same error estimates if correctly trained.

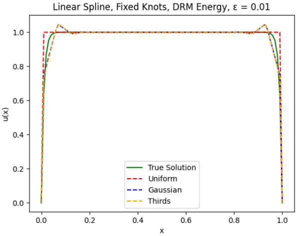
Compare with a traditional linear spline (used in FE) which is often a **piecewise linear Galerkin approximation to a function with a fixed mesh.** Good convergence, but often much slow than an adaptive FE method and hence a well trained PINN

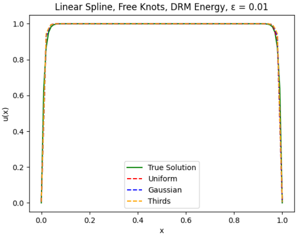
BUT do we ever see this in practice?

Example 1: $-\epsilon u'' + u = 1, \quad u(0) = u(1) = 0.$





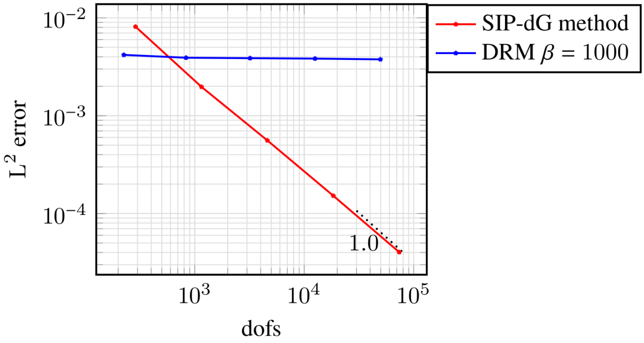




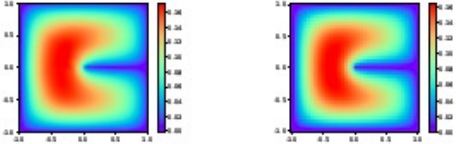
Example 2: Poisson equation in 2D:

DRM method works well for small DOF

dG Finite Element Method is **much** better for more DOF



This convergence pattern is seen in many other examples



(a) Solution of Deep Ritz method, 811 parameters (b) Solution of finite difference method, 1,681 parameters

Figure 2: Solutions computed by two different methods.

Table 1: Error of Deep Ritz method (DRM) and finite difference method (FDM)

Method	Blocks Num	Parameters	relative L_2 error
DRM	3	591	0.0079
	4	811	0.0072
	5	1031	0.00647
	6	1251	0.0057
FDM		625	0.0125
		2401	0.0063

Start of a convergence theory for DRMs

[Jiao, Lai, Lo, Wang, Yang],[MÏler and Zeinhofer]

- DRM error is a combination of approximation error, trainin error and optimization error
- Show that a DRM (depth L width W) can be constructed with low approximation error which reduces as the complexity of the DRM increases.
- Show that the training error reduces as the number of quadrature points increases (random sample)
- Use Gamma-convergence to prove convergence of the minimisers
- Hope that the optimization error can be reduced to acceptable levels.
- Try things out on simple problems

BUT AS IN PINNS

- **Non convex (no guarantee of uniqueness or convergence of training)**
- DRMS are nonlinear function approximators. No equivalent of Cea's lemma giving a bound on the solution error.
- Solutions can sometimes have no connection to reality!
- Location of the quadrature points can matter a lot.

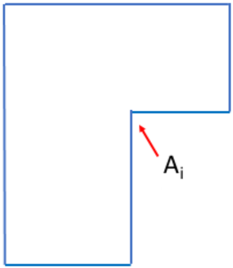
Preconditioning is essential to avoid these problems!!

When they work, DRMs work well!

Example (again) Poisson Problem on an L-shaped domain

Problem to solve:

$$\boxed{\begin{aligned} -\Delta u &= f \text{ in } \Omega \\ u &= u_D \text{ on } \Gamma_D \\ \nabla u \cdot \vec{n}_\Omega &= g \text{ on } \Gamma_N . \end{aligned}}$$



Singular solution

- Solution $u(\vec{x})$ has a gradient singularity at the interior corner A_i
- If the interior angle is ω and the distance from the corner is r then

$$u(r,\theta) \sim r^{\alpha} f(\theta), \quad \alpha = \frac{\pi}{\omega}$$

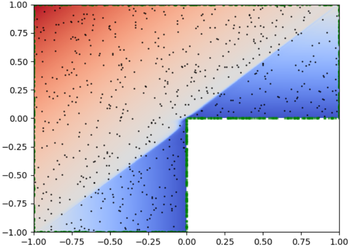
where $f(\theta)$ is a regular function of θ

- Corner problem

$$u(r,\theta) \sim r^{2/3}, \quad r \to 0.$$

Numerical results: **random** quadrature points

Solve $\Delta u(x) = 0$ on $\Omega_L \quad u(r,\theta) = r^{2/3} sin(2\theta/3)$ on $\Gamma = \partial\Omega_L$



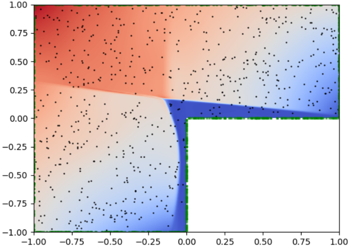


Figure: Left: PINN Right: DRM

Can we improve the accuracy by a better choice of collocation/quadrature points?

Optimal collocation points for the L-shaped domain

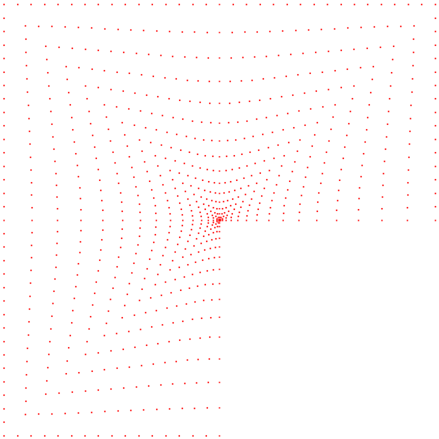


Figure: Optimal points for interpolating $u(r, \theta) \sim r^{2/3}$

Optimal points and PINN/Deep Ritz

Solutions with Optimal quadrature points

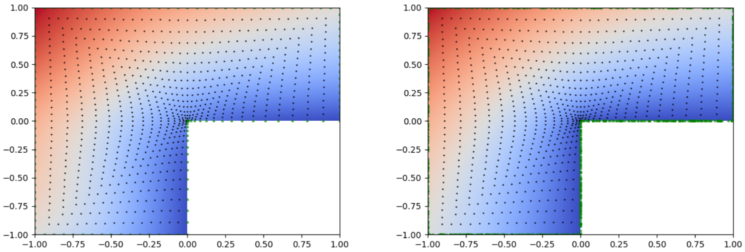


Figure: L^2 error - randomly sampled points: 0.468 | Optimal: 0.0639

Left: PINN, Right: Deep Ritz

Good choice of quadrature points makes a big difference, but still problems with pre-conditioning

Summary

- PINNS and DRMS both show promise as a quick way of solving PDEs but have only really been tested on quite simple problems so far
- PINS not (yet) competitive with FE in like-for-like comparisons
- PINNs need careful meta-parameter tuning and preconditioning to work well
- Long way to go before we understand PINNS orn DRMS completey and have a satisfactory convergence theory for them in the general case.
- Lots of great stuff to do!