r-adaptivity, deep learning and the deep Ritz method

Simone Appella[1], **Chris Budd[1]**, Tristan Pryer[1]

[1]University of Bath

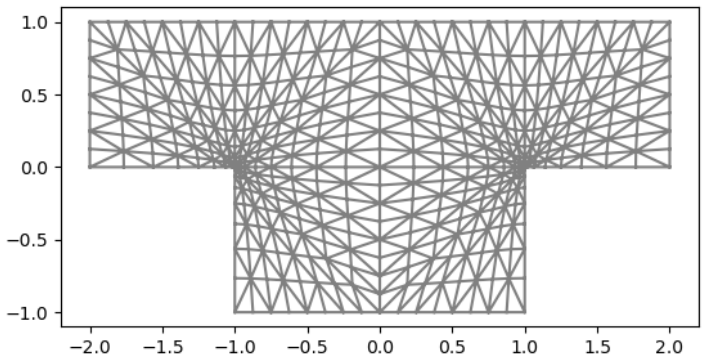
Limerick, March, 2022

Motivation: Solving PDEs

Seek to solve PDE problems of the form

$$u_t = F(x, t, u, \nabla u, \nabla^2 u; \lambda)$$

Traditional PDE computations using Finite Element Methods use a computational mesh τ comprising mesh points and a mesh topology:



Mesh choice

Accuracy of the computation depends crucially on the choice and shape of the mesh

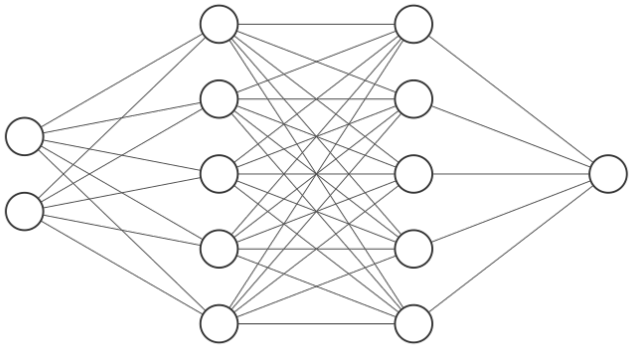
Mesh needs to be

- Fine Enough to capture (evolving) small scales/singular behaviour
- Coarse Enough to allow practical computations
- Able to resolve local geometry eg. re-entrant corners in non-convex domains

PINNS

Physics Informed Neural Networks for solving PDEs: "Mesh free methods". Use a Deep Neural Net to give a functional approximation to $u(x,t)$ with inputs x and t.

$$U(x,t) = DNN(x,t)$$



$U(x,t)$ is constructed via a combination of linear transformations and nonlinear/semi-linear activation functions.

Example: Shallow neural net

$$U(x,t) = \sum_{i=0}^{N-1} c_i(t)\sigma(a_i(t)x + b_i(t))$$

Can take

$$\sigma(z) = \text{ReLu}(z) \equiv z_+,$$

Then

$$U(x,t) = \sum_{i=0}^{N-1} c_i(t)(a_i(t)x + b_i(t))_+$$

Which is **piece-wise linear interpolation** with **free knots** at $x_i(t) = -b_i(t)/a_i(t)$.

I: Operation of a 'traditional' PINN

- Assume that $U(X,t)$ has strong regularity eg. C^2
- Differentiate $U(x,t)$ exactly using the chain rule
- Evaluate the PDE residual at collocation points X_i, t_j (chosen to be uniformly spaced, or random)
- Train the neural net to minimise a **loss function** L combining the PDE residual and boundary and initial conditions

Eg. Solution of two-point BVPs by PINNs

Consider the two-point BVP with Dirichlet boundary conditions:

$$-u_{xx} = f(x, u, u_x),\ x \in [0,1] \quad u(0) = a,\ u(1) = b.$$

Define output of the PINN by U and residual $r(x) := U_{xx} + f(x, U, U_x)$.
The PINN is trained by minimising the loss function

$$L = \frac{1}{N_r} \sum_i^{N_r} |r(X_i^r)|^2 + \frac{1}{2} \left(|U(0) - a|^2 + |U(1) - b|^2 \right),$$

where $\{X_i^r\}_i^{N_r}$ are the collocation points placed in $(0,1)$.

Numerical results for: $-u'' = \pi^2 sin(\pi x)$.

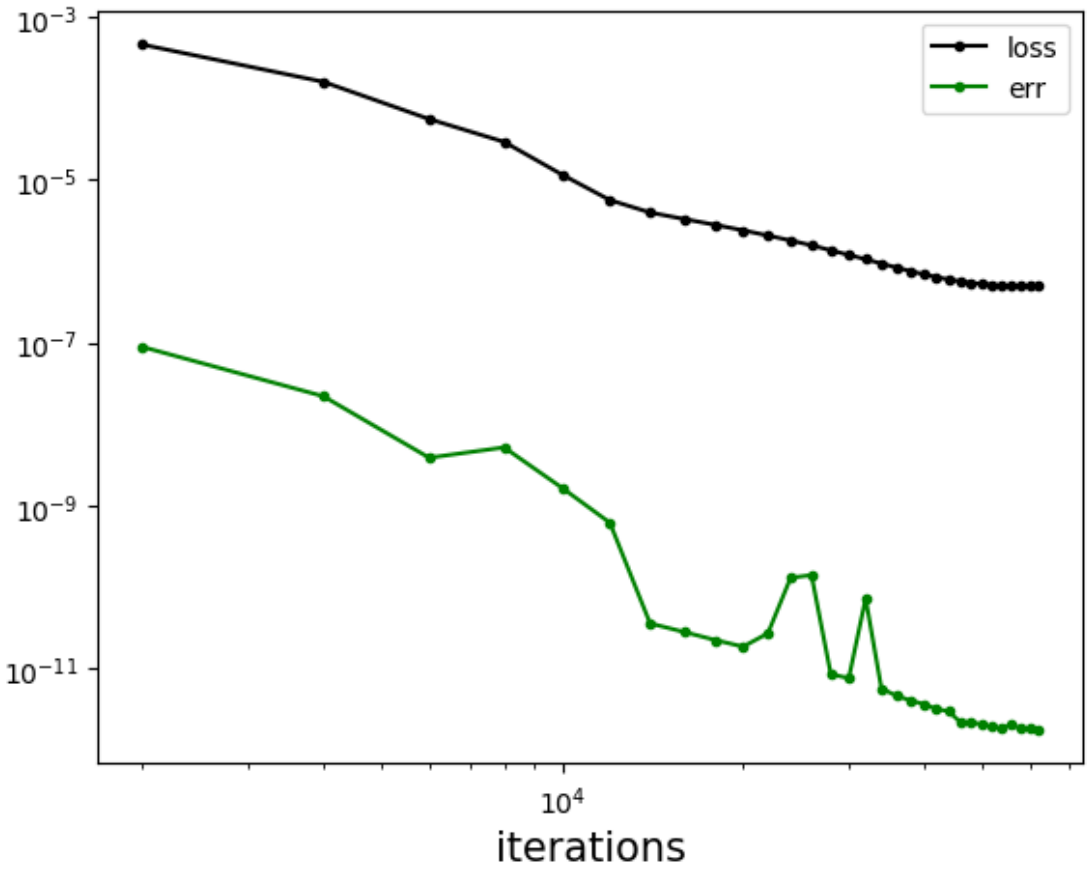


Figure: Loss and L^2 error for linear interpolant of the PINN solution for $N_r = 100$

Numerical results: $u(x) = sin(\pi x)$

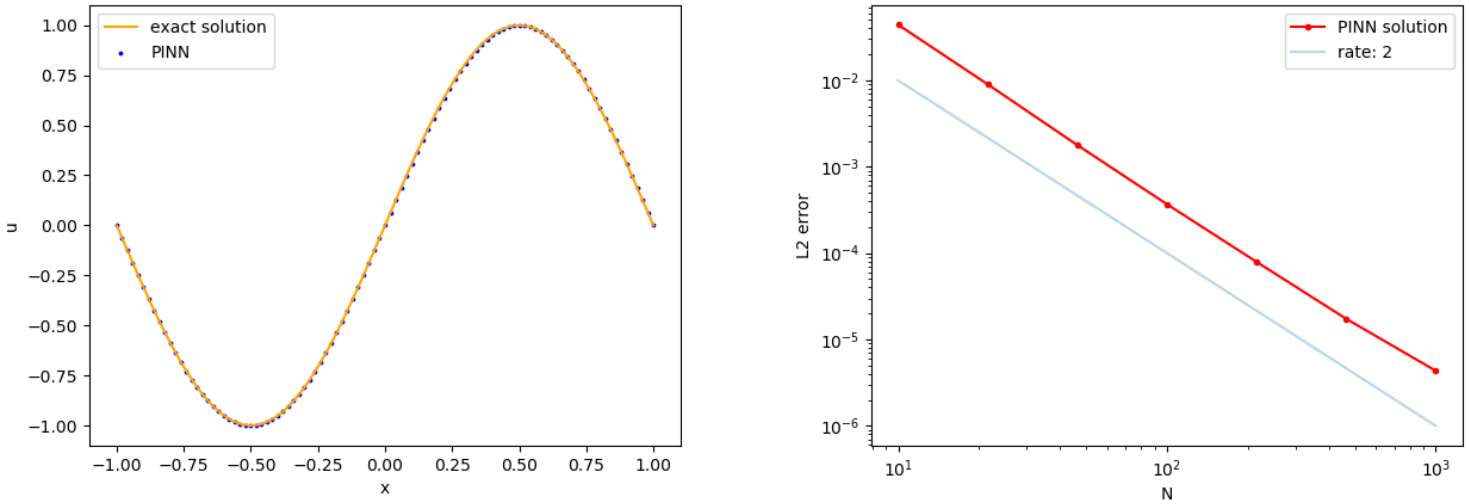


Figure: Left: PINN with 2 hidden layers and 30 hidden nodes
$N_r = 100$ uniformly distributed
activaction function: Tanh | optimizer: Adam with $lr = 1e-3$
Right: Convergence rate for 1st order interpolant

II Operation of a 'variational' PINN

- Assume that $U(X,t)$ has weak regularity eg. H^1
- Differentiate $U(x,t)$ exactly using the chain rule
- Construct an appropriate weak form of the PDE (typically involving an integral)
- Evaluate the weak form by using quadrature at quadrature points X_i, t_j (chosen to be uniformly spaced, or random)
- Train the neural net to minimise a loss function combining the weak form and boundary and initial conditions

Questions for the seminar

Questions for the seminar

1. When do and don't PINNS work
2. Can the performance of the PINN be improved by a 'good' choice of collocation points
3. Can we learn where to place the collocation points?
4. How well do PINNS and 'traditional' numerical analysis fit together?

'Classical' r-adaptivity for PDEs

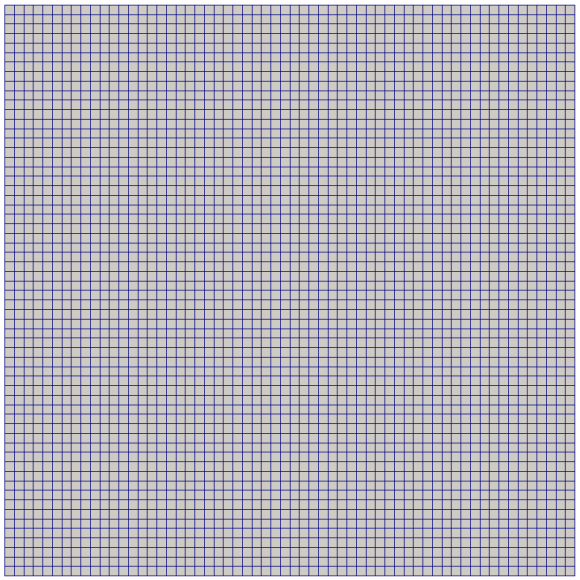
Want to construct a suitable mesh τ to solve a PDE.
Using a FE method i.e find (learn) the

"best possible mesh" to give the "best possible solution"

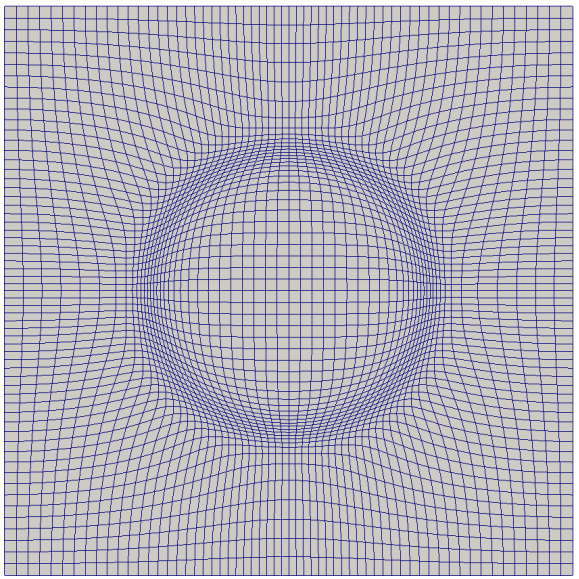
- r-adaptivity: relocate mesh vertices X_i, preserving mesh connectivity/topology.
- When done dynamically $X_i(t)$, "moving mesh" method.
- Some advantages/constraints
  - Avoid sudden changes in mesh resolution
  - Maintain control over global mesh regularity
  - Keep the mesh topology unchanged (if needed)
  - Avoid load-balancing issue when used in parallel

r-adaptivity as a map

- Ω_C – "computational" domain eg. plane, sphere (often with uniform mesh) mapped to:
- $\Omega_P = \vec{F}(\Omega_C)$ – "physical" domain
- $\vec{\xi}$ – coordinate in computational domain
- $\vec{x} = \vec{F}(\vec{\xi})$ – coordinate in physical domain



Ω_C, $\vec{\xi}$



Ω_P, $\vec{x}$

Equidistribution

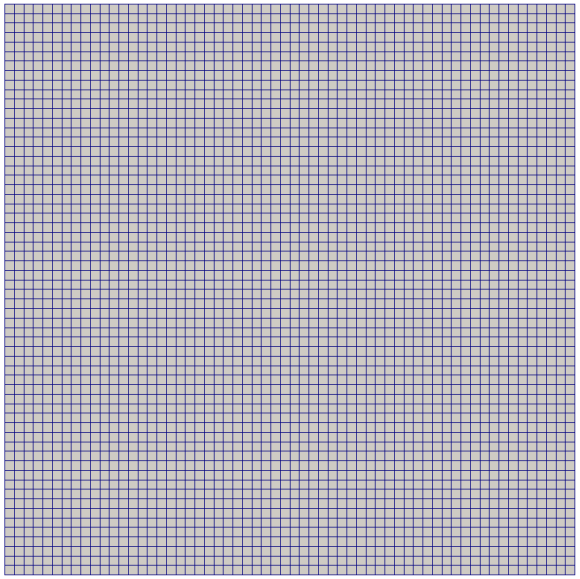
Mesh density controlled by monitor function $m(\vec{x}, t) > 0$, through **equidistribution**:

$$m(\vec{x}, t) \times \text{cell area} = \text{const}$$

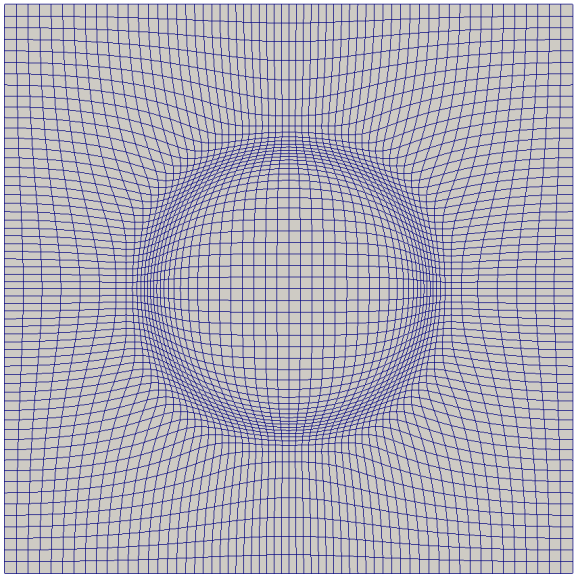
Monitor m is derived from current simulation state

- A-priori error estimates (e.g. of interpolation error): $m \propto \|\nabla u\|_{L^p}, \|\nabla^2 u\|_{L^p}$, etc., so that 'the function $u(\vec{x})$ is represented as well as possible'
- m based on diagnostic derived from physical principles, e.g. vorticity
- A-posteriori error estimates
- ?? learned ??

r-adaptivity



Ω_C, $\vec{\xi}$



Ω_P, $\vec{x}$

Adapted mesh is defined by a map $\vec{x}(\vec{\xi}, t)$

The Jacobian of this map is J, $J_{ij} := \frac{\partial x_i}{\partial \xi_j}$

Equidistribution requirement:

$$m(\vec{x}, t) \det J = \text{const} =: \theta$$

Optimal *r*-adaptivity

Equidistribution requirement becomes:

$$m(\vec{x}, t) \det J = \theta \tag{1}$$

In 1D, this defines the mesh (almost) uniquely.

In 2D/3D, additional regularisation constraints are needed.

Budd & Williams (2006): subject to (1), pick $\vec{x}(\vec{\xi})$ **minimising the Wasserstein distance**

$$\int_{\Omega_C} |\vec{x}(\vec{\xi}) - \vec{\xi}|^2 \, \mathrm{d}\vec{\xi}.$$

Prevents tangling and reduces skewness.

r-adaptivity in 1D

In 1D we have moving mesh points

$$X_{i-1}(t) < X_i(t) < X_{i+1}(t), \quad X_i(t) \equiv x(i/N, t),$$

$$h_i = X_{i+1} - X_i, \quad \text{mesh size}$$

Equidistribution gives

$$mx_\xi = \theta(constant) \implies (mx_\xi)_\xi = 0.$$

$$M_{i+1/2}(t)(X_{i+1} - X_i) - M_{i-1/2}(t)(X_i - X_{i-1}) = 0.$$

$$M_{i+1/2} = (m(X_i) + m(X_{i+1}))/2$$

MMPDE1: Solve this to find $X_i(t)$: Prone to instability

Moving mesh PDES

Can overcome instability via a **moving mesh PDE** [Huang and Russell], [B]

$$\tau\ m\ x_t = (m x_\xi)_\xi. \qquad (MMPDE5)$$

Solve this to find $X_i(t) = x(i/N, t)$.

Solving the time evolving PDE 1

'Traditional methods to solve the PDE'

Simultaneous:

Solve MMPDE5 together with the PDE in a Lagrangian frame:

$$\tau\, m(U)\,\frac{dX}{dt} = d\left(m(U)\frac{dX}{d\xi}\right)/d\xi$$

$$\frac{dU(\xi)}{dt} - \frac{dX(\xi)}{dt}U_x(\xi) = f(x(\xi), t, U_x(\xi), U_{xx}(\xi))$$

This avoids interpolation problems.
But leads to very stiff ODEs with stability problems.

Solving the time evolving PDE 2

Rezoning:

Alternate between

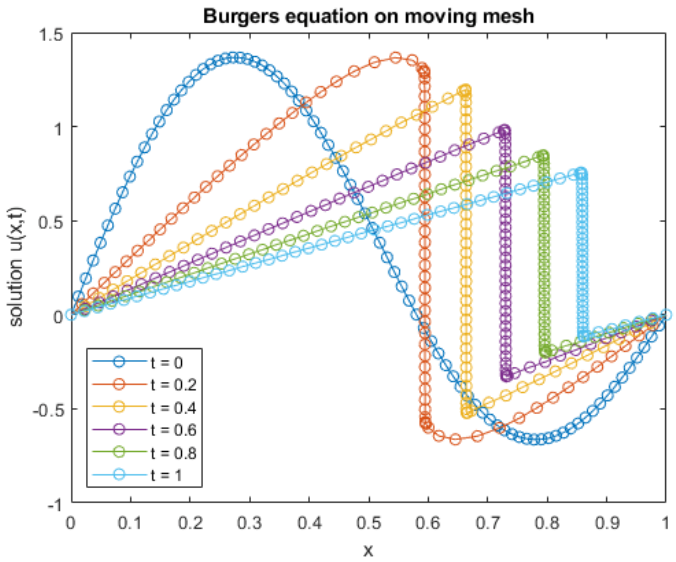
1. Solving the PDE in a Euclidean or a semi-Lagrangian frame
2. Solving MMPDE5 for a short time to find a new mesh
3. Interpolating the solution onto the new mesh (Euclidean frame)

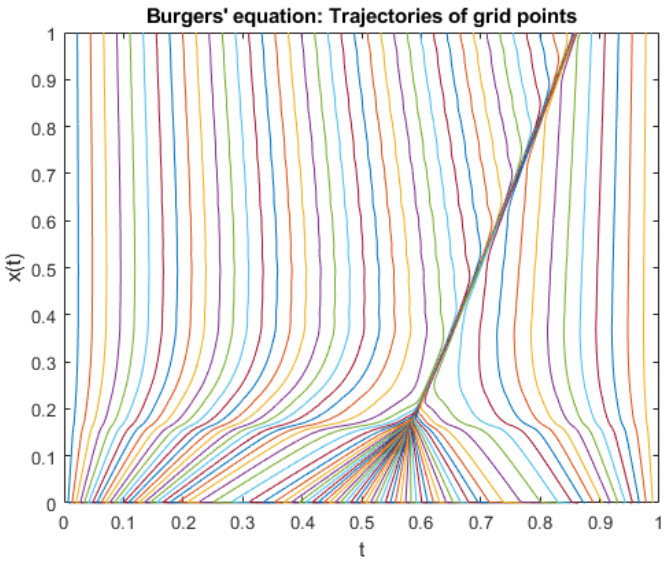
- Much more stable, less stiff system
- Requires interpolation
- Can be used in a PINNs framework

Burgers' equation

As an example consider Burgers' equation

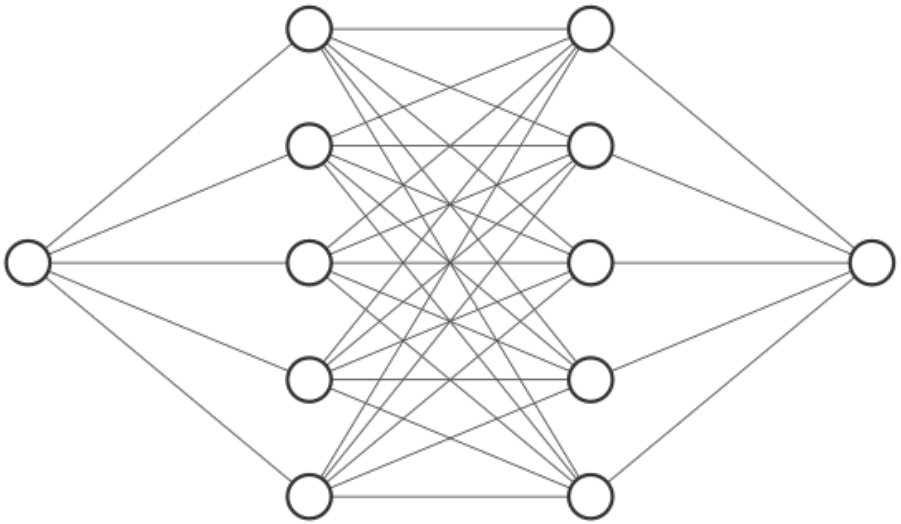
$$u_t = uu_x + \epsilon u_{xx}, \quad m(x,t) = \sqrt{1+u_x^2}$$





Deep Neural Network for r-adaptivity in 1D

A feed-forward Deep Neural Network (DNN) can be 'in principle' trained to approximate a function



$$f(x) = f_L \circ f_{L-1} \circ \cdots \circ f_0$$

$$f_i = \sigma(W_i f_{i-1} + b_i)\ i = 1, \cdots, L \quad f_0 = \xi$$

Direct Learned Function Approximation

For a function $u(x)$ solve the following

$$\min_{\mathbf{z}} \; L(\mathbf{z}) \equiv \sum_{k=1}^{N} |f(x_k) - u(x_k)|^2$$

where we use the shallow ReLU network:

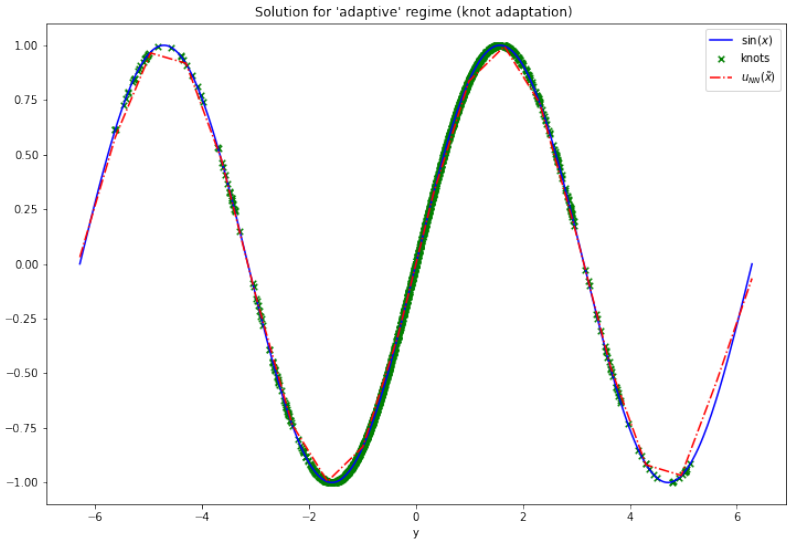
$$f(x) = \sum_{j=1}^{M} c_j [a_j x - b_j]_+, \quad \mathbf{z} = [\mathbf{a}, \mathbf{b}, \mathbf{c}].$$

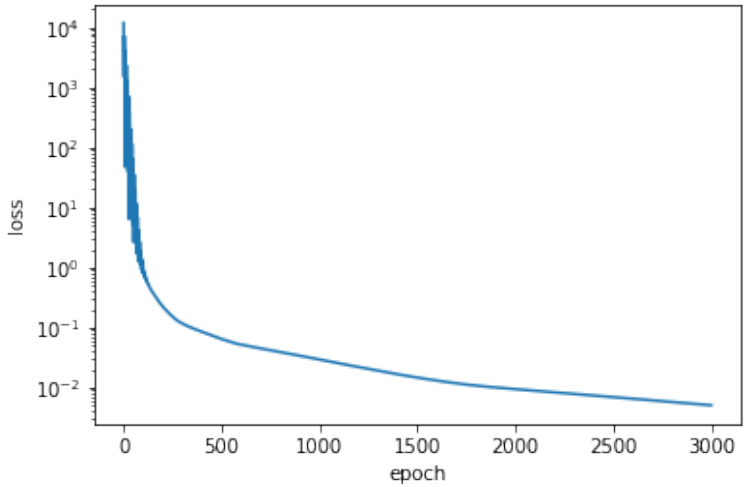
and x_k are the *quadrature points*.

Use an ADAM SGD (over the quadrature points) optimiser

Approximation of: $u(x) = \sin(x)$

Uniform quadrature points:

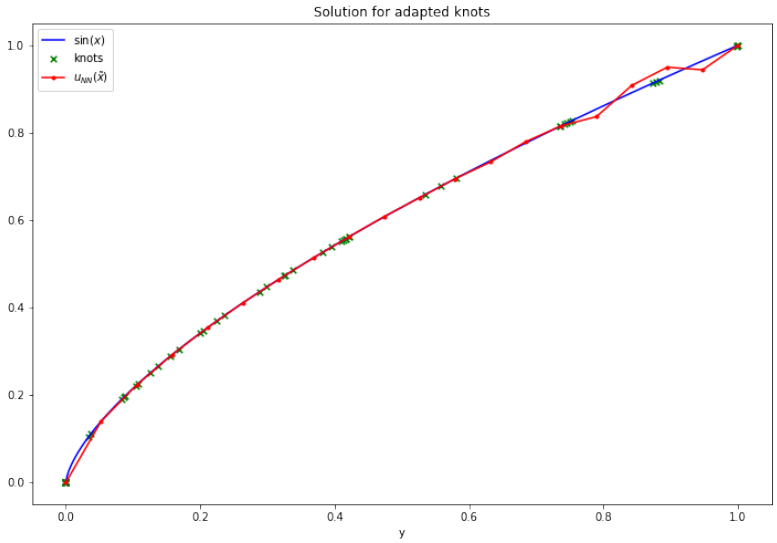


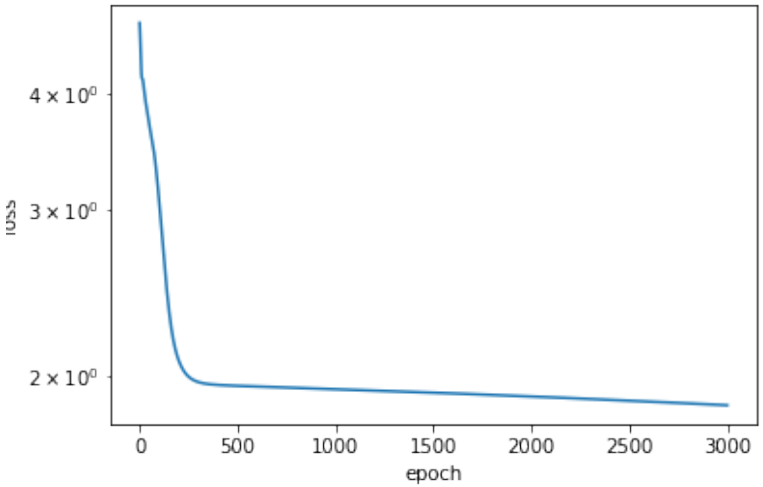


Results poor. Depend crucially on the starting values. Even then poor.

Approximation of: $u(x) = x^{2/3}$

Uniform quadrature points:





Results still poor! Depend crucially on the starting values. Even then poor.

Learned Function Approximation: Using Optimal Equidistribution

Instead take a loss function of the DNN which seeks to minimize the L^2 error of the piecewise linear interpolant of the function $u(x)$ given by:

$$||u - \Pi u||_{L^2}^2 \leq C \sum_{i}^{N} \left(h_i m_{i+1/2}\right)^5 \equiv L$$

where $m(x) = (1 + u_{xx}^2)^{1/5}$ and L is the loss function.

Network architecture and training parameters:

- Input: Computational variable ξ | Output: Physical variable x
- Network architecture: 100 hidden nodes in 3 layers
- Optimizer: Adam with learning rate 10^{-3}
- Epochs: 50000

Numerical Result: $u(x) = x^{2/3}$ (singularity at $x = 0$)

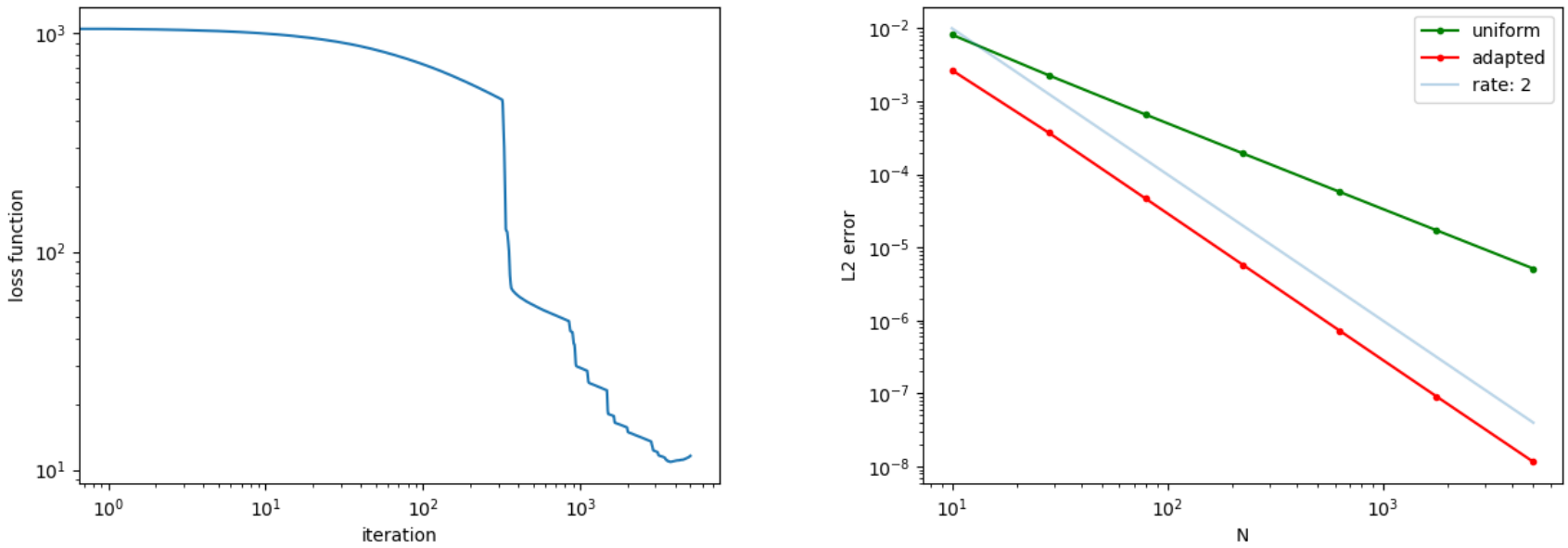


Figure: Left: Loss function for $N = 300$ decreases abruptly after 200 iterations. Right: L^2 error of $h(x)$ interpolated at the equidistributed points. The convergence rate is optimal even when N is greater than the training sample size.

Numerical Result: $u(x) = x^{2/3}$ (singularity at $x = 0$)

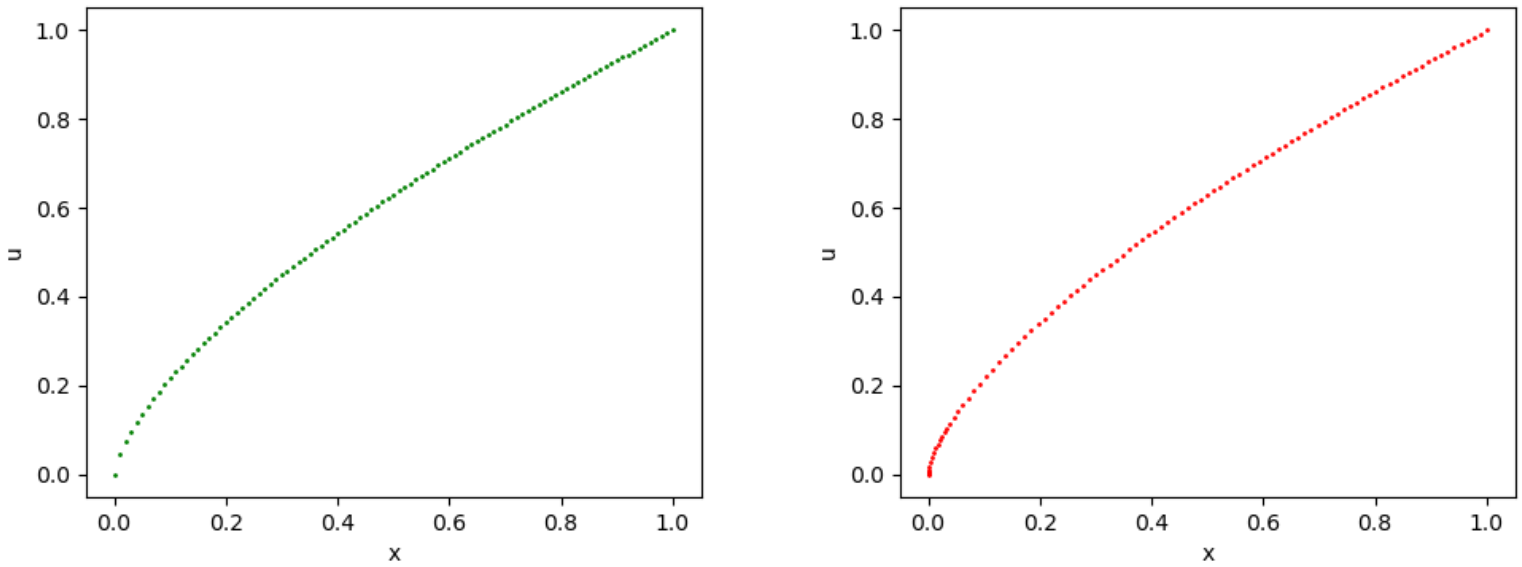


Figure: Comparison between uniform and adapted mesh for $N = 100$. Note that the equidistributed mesh clusters towards $x = 0$, where the solution exhibits a singular behaviour.

MMPDE5

The result of the DNN can be compared with other numerical methods. MMPDE5: Find the steady-state solution of the PDE

$$\partial_t x = \frac{1}{\tau m}\partial_\xi(m(x)\partial_\xi x);$$

The spatial derivative is discretized using central finite difference and the time derivative with the implicit Backward Euler:

$$\frac{X_i^{n+1}-X_i^n}{\Delta t} = \frac{1}{\tau m_i^n \Delta\xi^2}\left(m_{i+1/2}^n(X_{i+1}^{n+1}-X_i^{n+1}) - m_{i-1/2}^n(X_i^{n+1}-X_{i-1}^{n+1})\right)$$

De Boor's algorithm

For $\Omega = [a, b]$, define an initial uniform mesh $\tau^0 = \{X_i^0\}_{i=1}^N$. We then approximate $m(x)$ as a piece-wise constant function and apply iteratively for a maximum number of iterations/prescribed tolerance

$$X_i^{n+1} = X_{k-1}^n + \frac{2(\xi_i P(b) - P(X_{k-1}^n))}{m(X_{k-1}^n) + m(X_k^n)}, \quad \xi_i = \frac{i-1}{N-1}$$

where $k - 1 < i \leq k$, $P(X_k) = \int_a^{X_k} m(x)dx$.

The mesh sequence converges to a limit mesh τ provided that m is sufficiently smooth and N is sufficiently large. The larger N, the faster the iteration converges.

Convergence measure-1D

Following from the desired equidistribution condition

$$h_i m_{i+1/2} = \frac{\sigma_h}{N-1} \; i = 1, \cdots, N \; ,$$

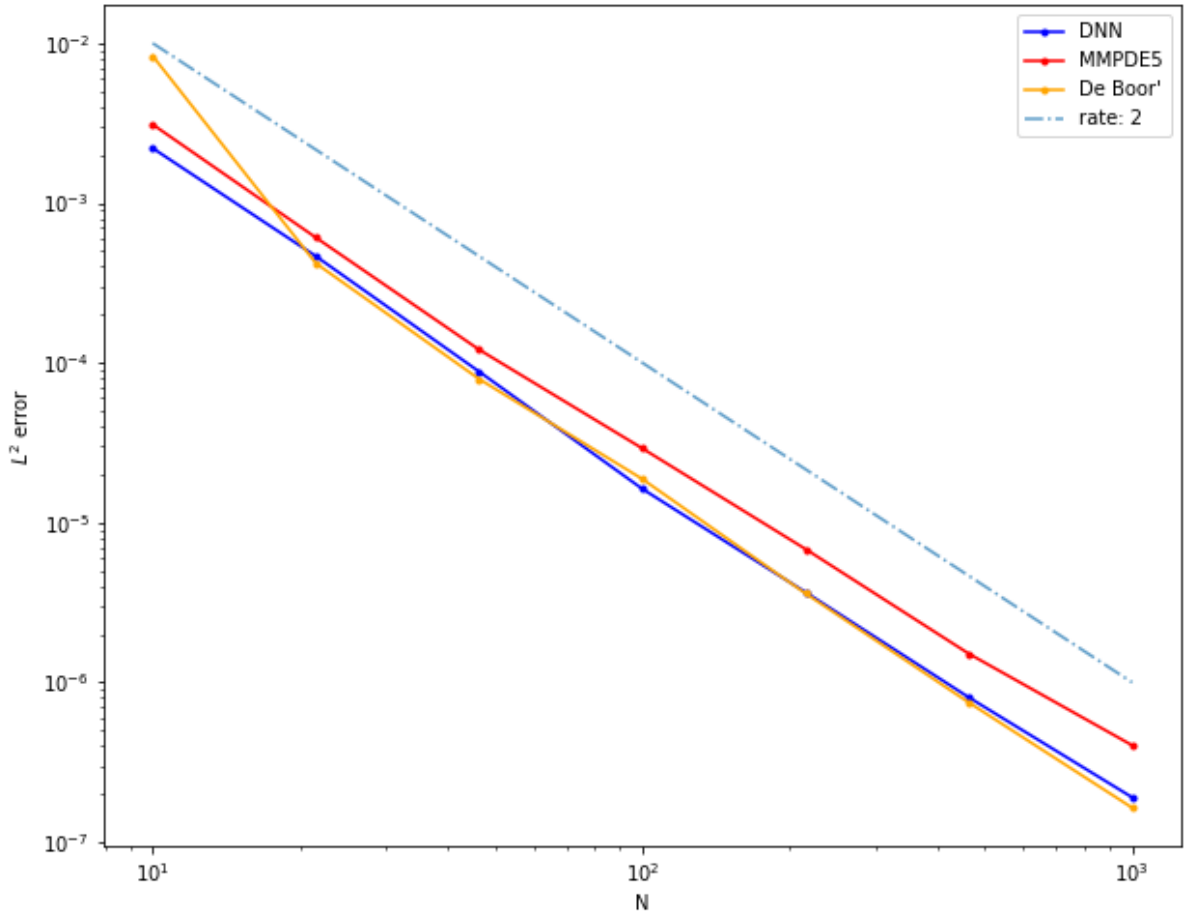
the **convergence measure** for an equidistributed mesh satisfies

$$Q_{eq} = \frac{(N-1) h_i m_{i+1/2}}{\sigma_h} \leq \kappa_{eq},$$

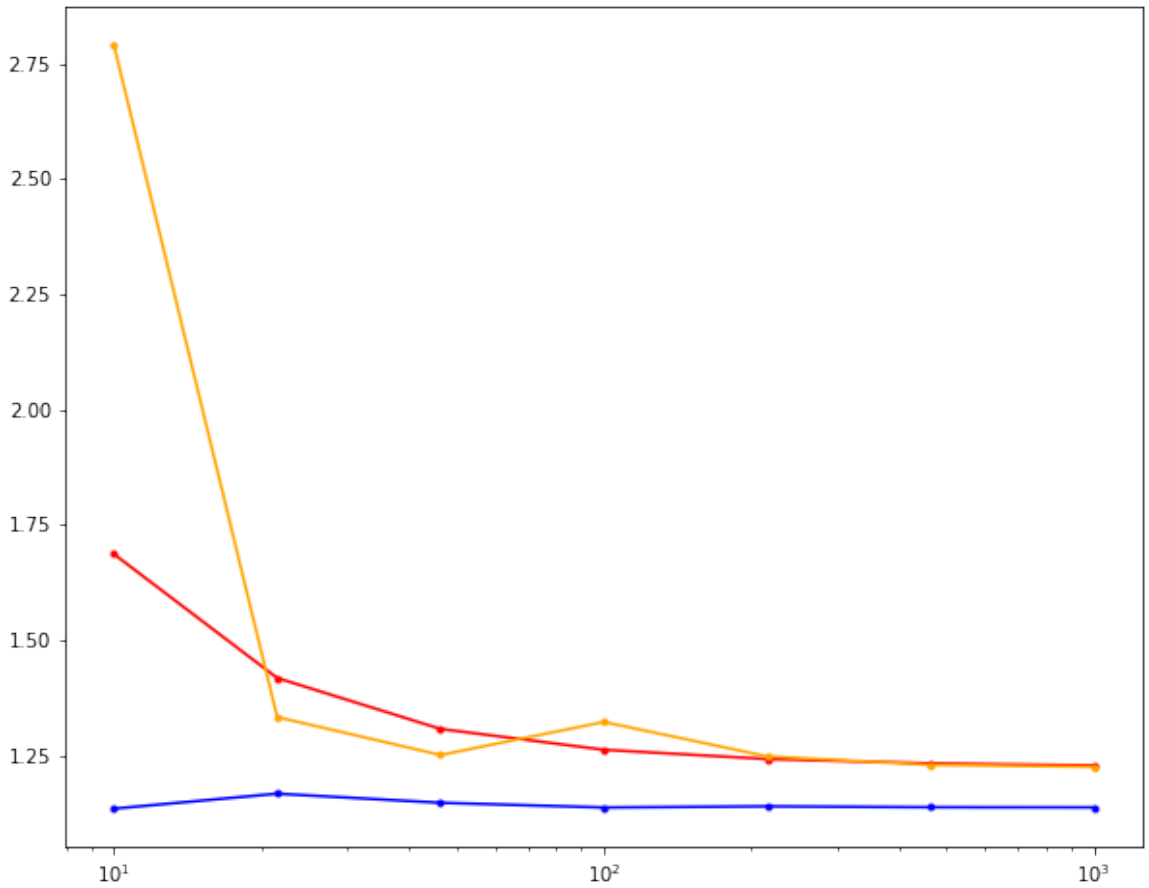
where $\kappa_{eq} \geq 1$ is independent of i and N. Here $\sigma_h = \sum_i h_i m_{i+1/2}$.

Note that when $\kappa_{eq} = 1$ the mesh is **perfectly equidistributed**.

DNN vs standard approaches - convergence rate



DNN vs standard approaches - convergence measure



Solving PDEs with Deep Learning

Now apply these ideas in the context of PINNs

Rezoning method

Apply alternatively

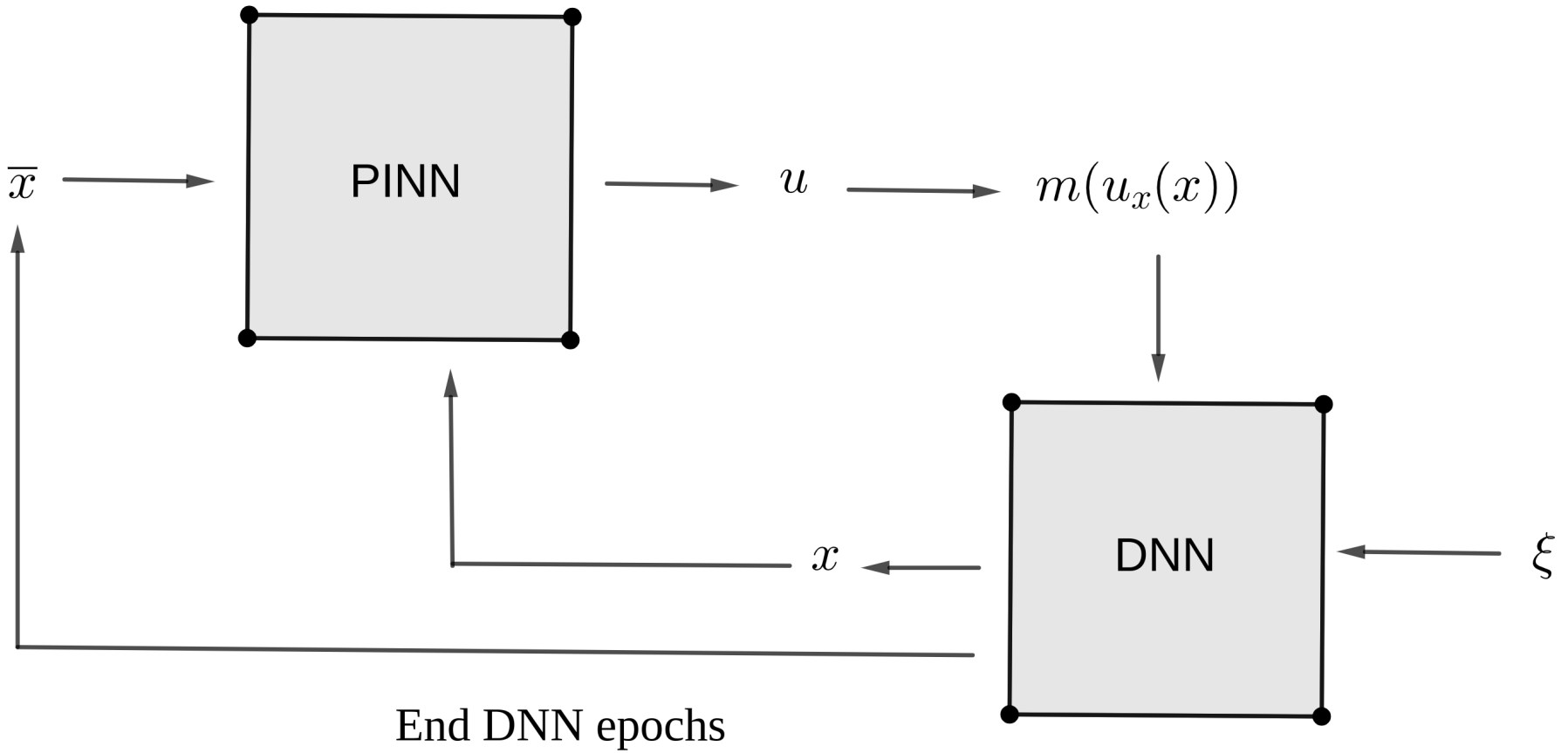
- Train PINN to solving PDE on given collocation points $(X_i, t_j) \to U(X, t)$
- Use DNN and $U(X, t)$ to equidistribute the meshes τ^{t_j} and find the new points (X_i, t_j)

Semi-Langrangian framework

Train a single PINN to learn simultaneously both the mesh τ and solve the PDE on that mesh [Pardo, David et. al. (in progress)].

Will consider only time independent BVPs for the rest of this seminar

Rezoning Method for BVPs



Good news: Reaction-Diffusion Equation

Solve $-\varepsilon^2 u_{xx} + u = 1 - x$ on $[0,1] \quad u(0) = u(1) = 0$

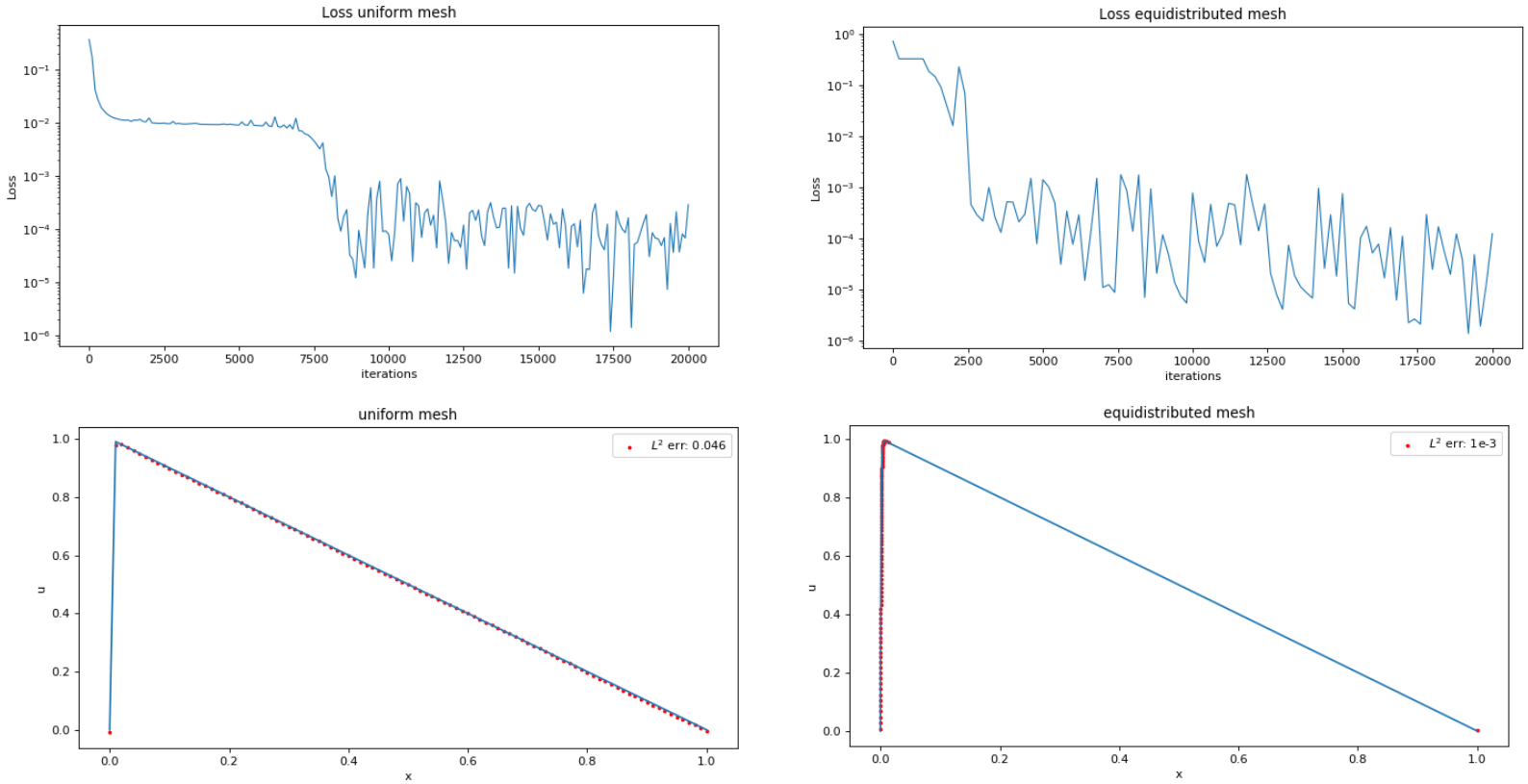


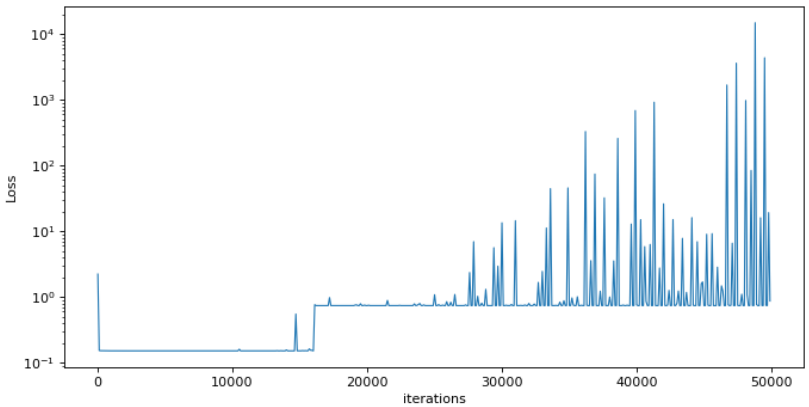
Figure: PINN trained for 20000 epochs, $N_r = 101$, Adam optimizer with $lr = 1e-3$.

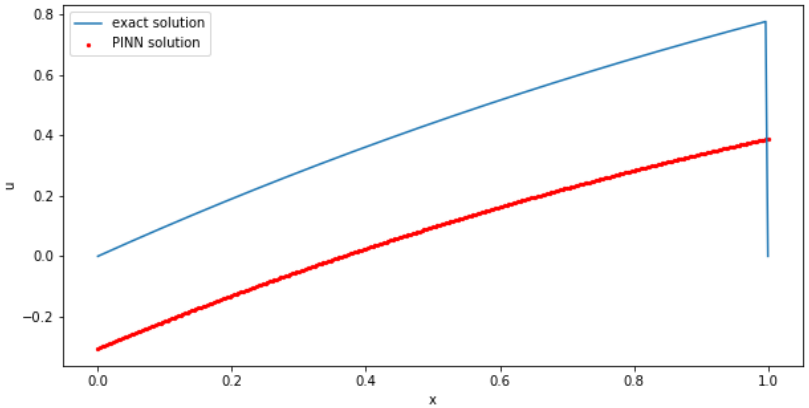
Bad news: Convection-dominated equation

PINNs fail to train when the solution of the BVP exhibits singular behaviour [Krishnapriyan, Aditi et. al., (2021)]:

$$-\varepsilon u_{xx} + \left(1 - \frac{\varepsilon}{2}\right) u_x + \frac{1}{4}\left(1 - \frac{1}{4}\varepsilon\right) u = e^{-x/4} \text{ on } [0,1] \quad u(0) = u(1) = 0$$

$$u(x) = exp^{\frac{-x}{4}} \left(x - \frac{exp^{-\frac{1-x}{\varepsilon}} - exp^{-\frac{1}{\varepsilon}}}{1 - exp^{-\frac{1}{\varepsilon}}} \right)$$





Homotopy method

The **homotopy** method can be used to obtain the solution by reducing logarithmically ε at each iteration.

Given an initial uniform mesh of $2N$ points, after a fixed time of iterations N are uniformly relocated on $[0, 1-2\varepsilon]$, while the remaining N are uniformly distrisbuted on $[1-2\varepsilon, 1]$.

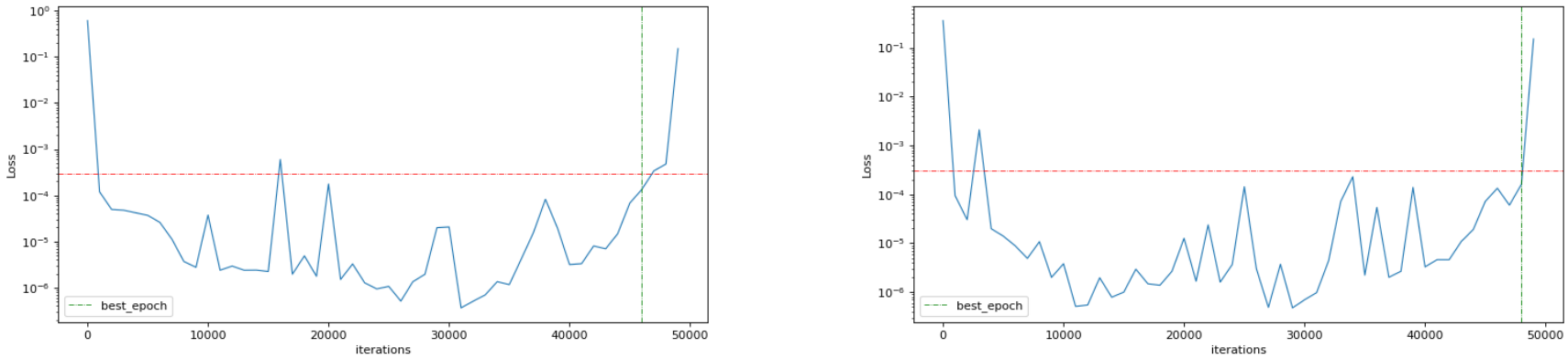


Figure: Loss function for uniform and iteratively adapted mesh.

Numerical results: Convection Equation

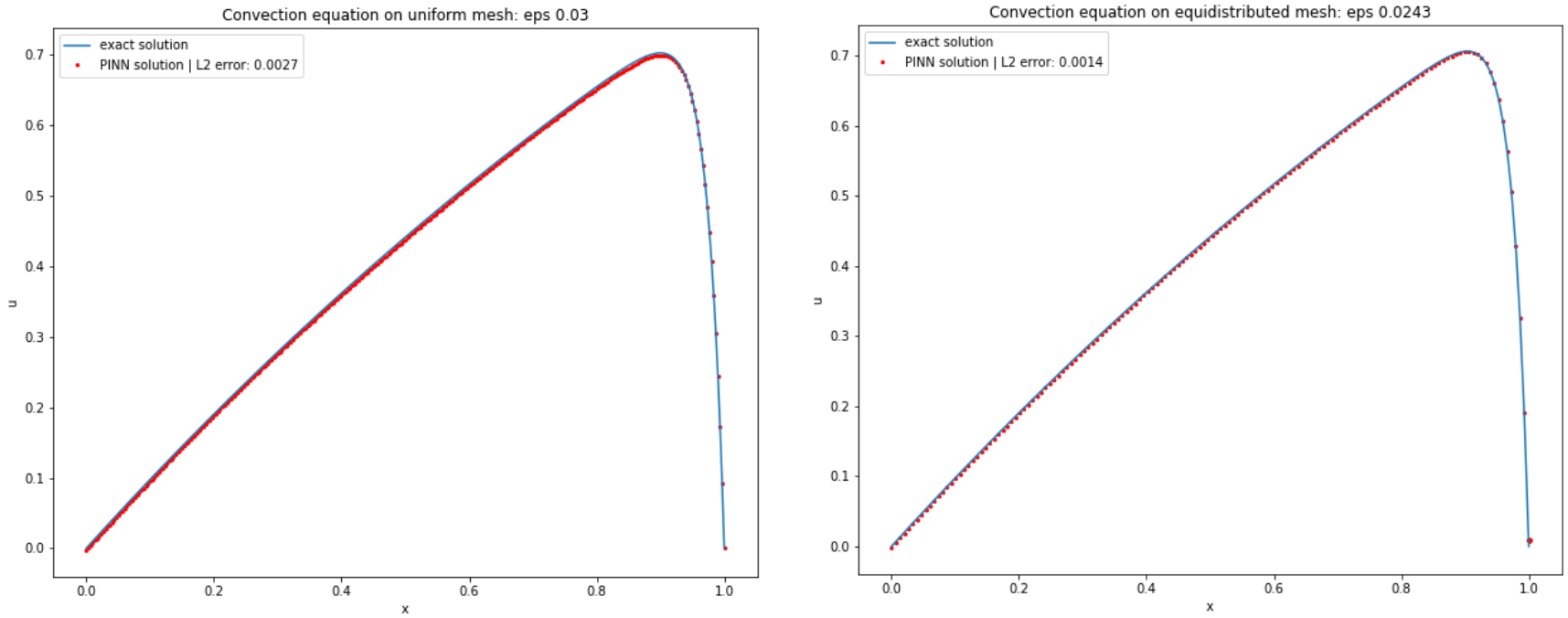
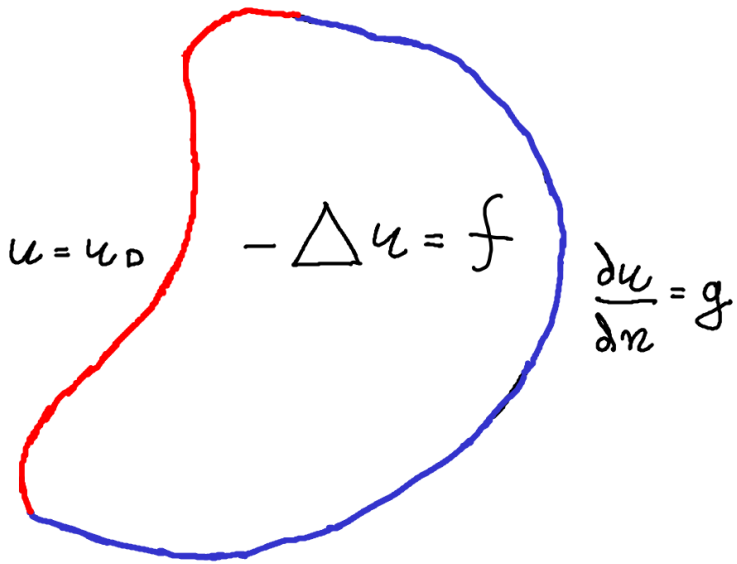


Figure: PINN trained for 50000 epochs with Adam optimizer ($lr = 1e-3$).
Left: uniform $N = 300$ | Right: equidistributed $N = 150$

Better news: Poisson equation in 2D

For higher dimensional BVPs more robust network architectures can be employed:



Now consider a variational-PINN for this PDE problem.

Deep Galerkin Method for the Poisson equation

The **Deep Galerkin Method** (DGM) mimics the action of a PINN

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

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

$$\mathcal{I}(u) = \int_\Omega (\Delta u(\vec{x}) + f(\vec{x}))^2 \, d\vec{x} + \beta \int_{\partial\Omega} (u(\vec{x}) - u_D)^2 d\vec{x}$$

- DNN based approximation of u which is in C^2
- A numerical quadrature rule for the functional using chosen quadrature points
- An algorithm for solving the optimization problem

Deep Ritz method for the Poisson equation

The **Deep Ritz Method** (DRM) [Weinan E and Bing Yu, (2017)] seeks the solution u satisfying

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

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

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

The Deep Ritz method is based of the following assumptions:

- DNN based approximation of u which is in H^1
- A numerical quadrature rule for the functional using chosen quadrature points
- An algorithm for solving the optimization problem

Network Structure

$$f_{i+1}(x) = \sigma(W_{i,2} \circ \sigma(W_{i,1} f_i(x) + b_{i,1}) + b_{i,2}) + f_i(x). \qquad (2)$$

The final output is $U(x) = f_{L+1}(x) = W_L f_L(x) + b_L$, where $W_L \in \mathbb{R}^{n \times d}$ and $b_L \in \mathbb{R}^n$.

For this type of architecture [E] suggests the activation function $RELU^3 = \max(0, x^3) \in C^2$. Other possible choices in C^2 are:

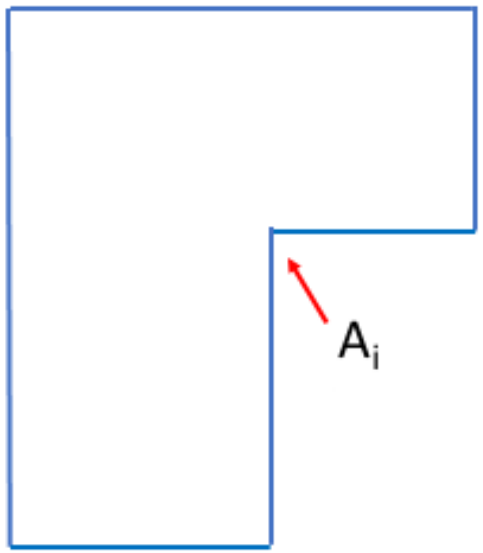
- $sigmoid(x) = \frac{1}{1+\exp(-x)}$
- $swish(x) = \frac{x}{1+\exp(-x)}$
- $\tanh(x)$
- $\sigma_{sin}(x) = (\sin x)^3$

ADAM optimiser.

Poisson Problem on an L-shaped domain

Problem to solve:

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



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

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

Solution error

The L_2 error is computed by evaluating the approximate solution on a Delaunay mesh.

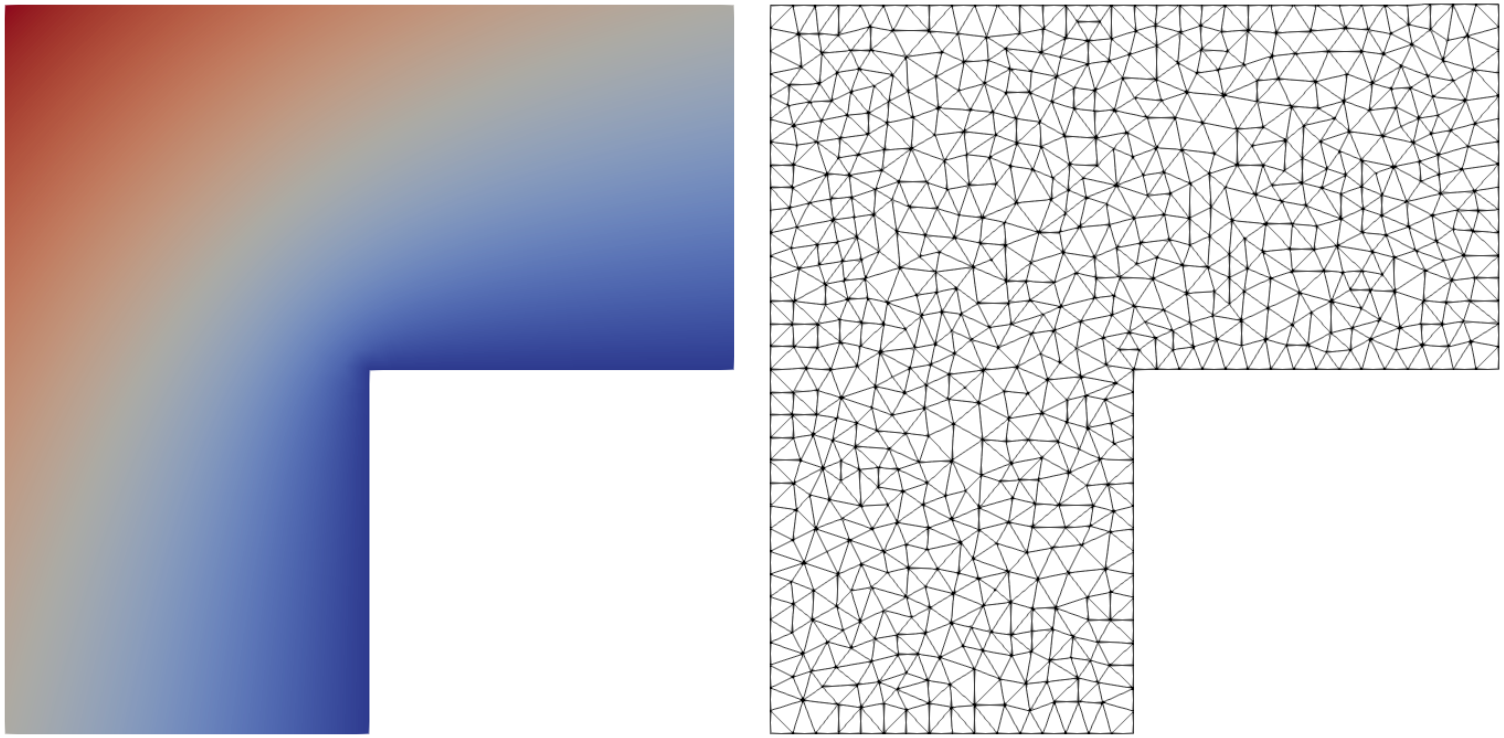


Figure: Left: exact solution. Right: Delaunay mesh with $N = 833$ on which the L_2 error is computed.

Numerical results: random quadrature points

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

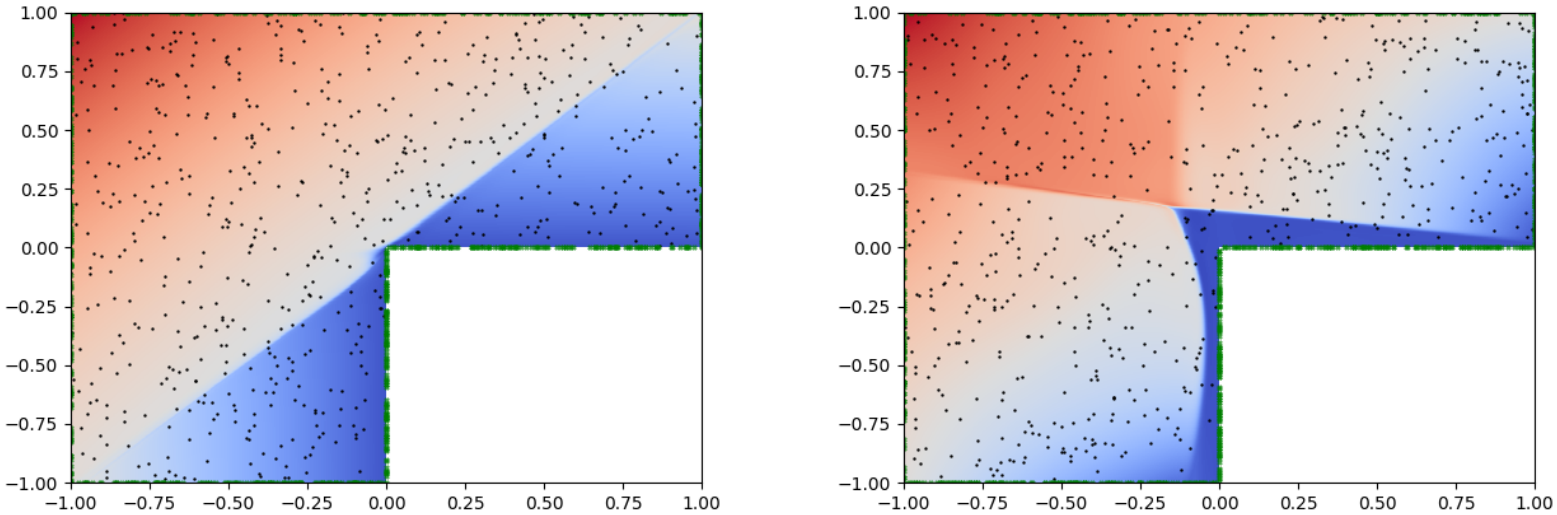


Figure: Left: DGM Right: DRM

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

OT Based r-adaptivity

Can do r-adaptivity in $\mathbb{R}^n$ using optimal transport, giving a close link to machine learning.

Idea Think of m as a *measure*, and minimise **Wasserstein distance**

$$\min_{\vec{\mathbf{x}}} \int |\vec{\mathbf{X}} - \vec{\xi}|^2 d\mu$$

Such that

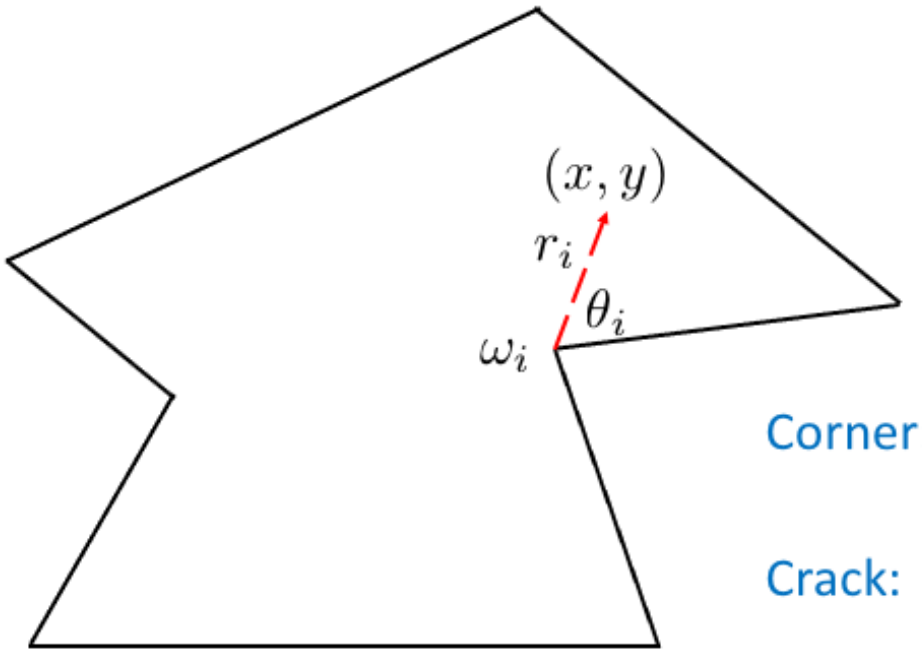
$$m(\vec{\mathbf{X}}, t)|d\vec{\mathbf{X}}| = \theta|d\vec{\xi}|.$$

Find $\vec{\mathbf{X}}$

- Directly eg. Using the Sinkhorn algorithm
- Indirectly eg. Solving the Monge-Ampére equation [B], [PICANNS, Singh et. al. 21]

OT Adaptive method

- Solve the Monge-Ampére equation locally at each interior corner (semi-analytically)
- Locally redistribute the mesh according to the solution
- Use monitor function m based on a-priori interpolation error estimates in L_∞ or in L_2



Corner: $\omega = \frac{3\pi}{2}$

Crack: $\omega = 2\pi$

OT mesh for the L-shaped domain

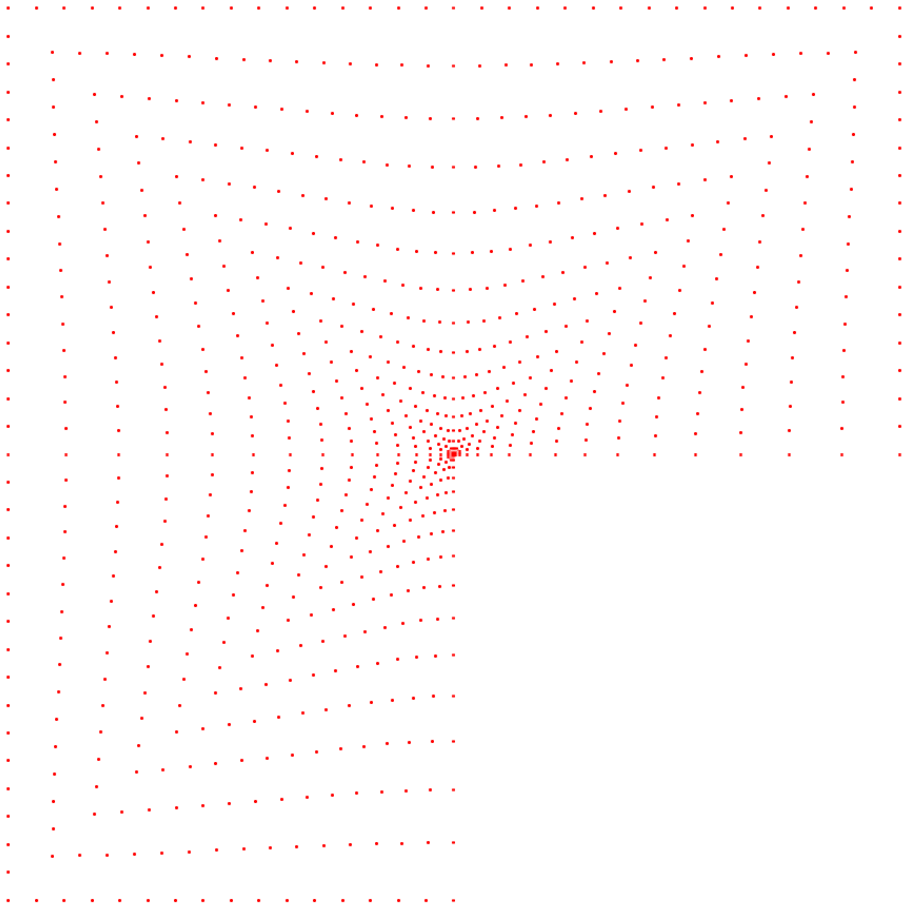


Figure: OT Mesh for solving Poisson's eq. in a L-shaped domain $u(r,\theta) \sim r^{2/3}$

OT and Deep Galerkin/Ritz

Solutions with OT quadrature points

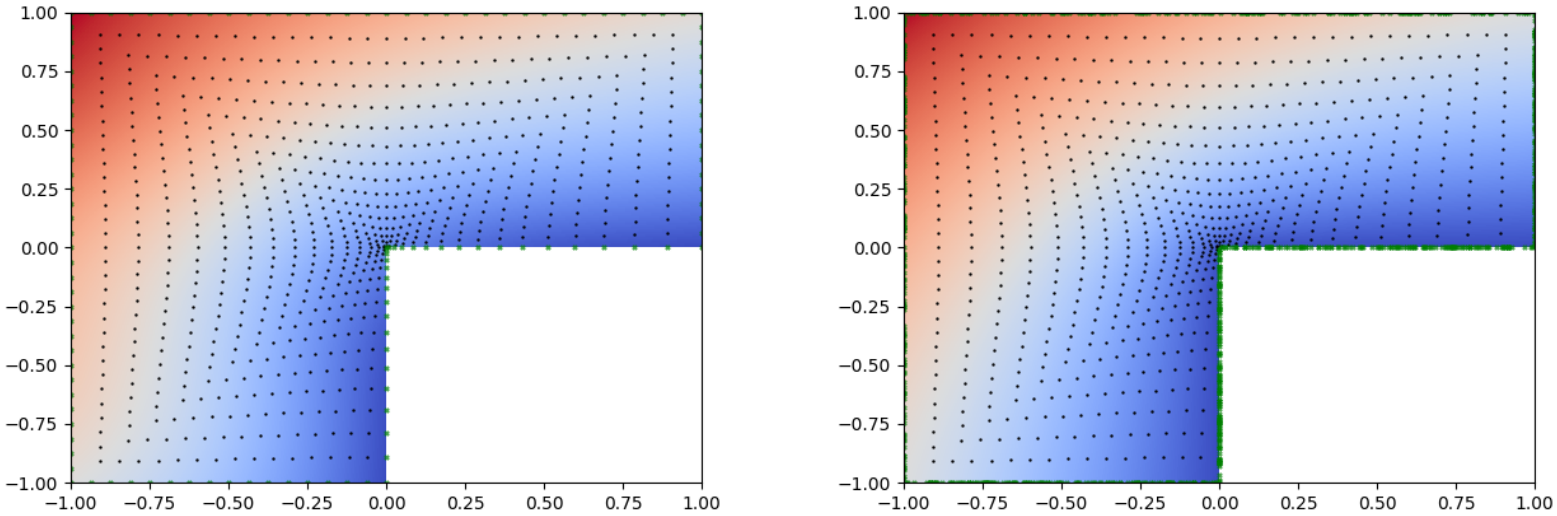
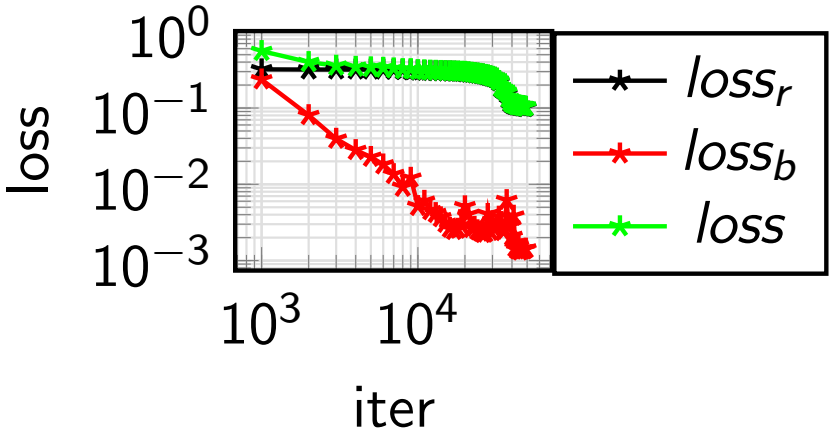


Figure: L^2 error - randomly sampled points: 0.468 | OT: 0.0639

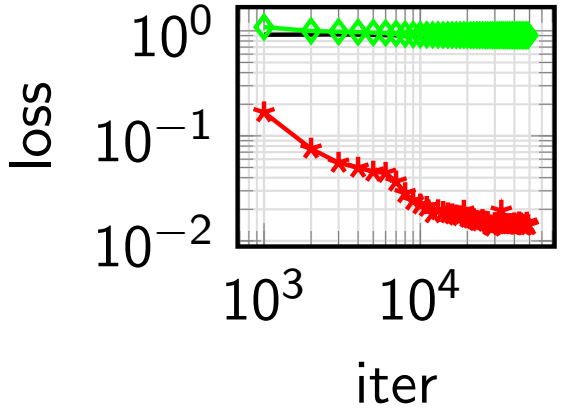
Left: Deep Galerkin, Right: Deep Ritz

Good choice of quadrature points makes a big difference

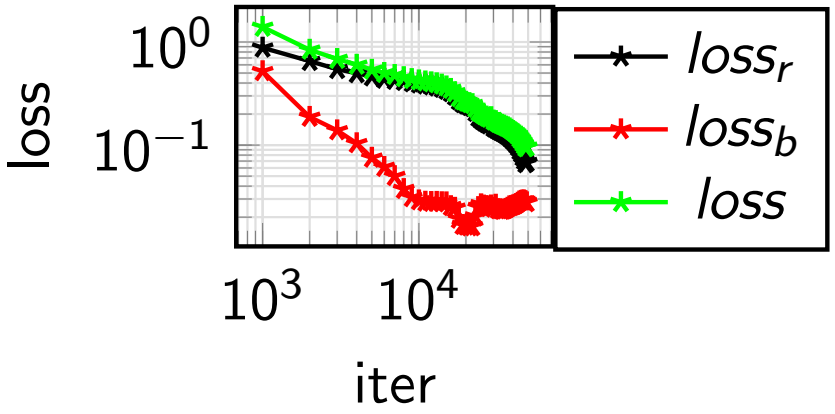
Loss function



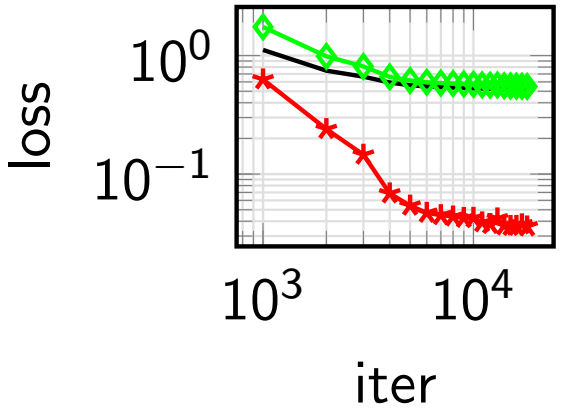
(a) DRM: Uniform random points



(b) DRM: OT-based points

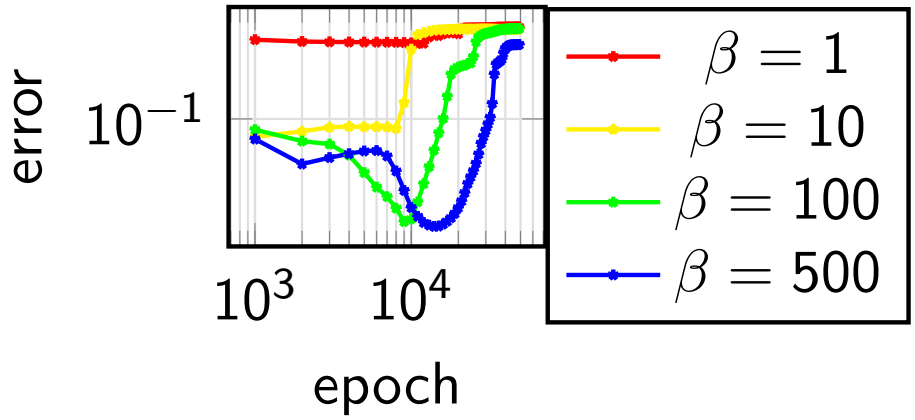


(c) DGM: Uniform random points

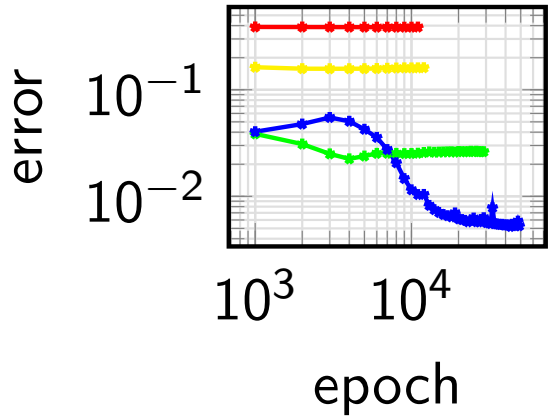


(d) DGM: OT-based points

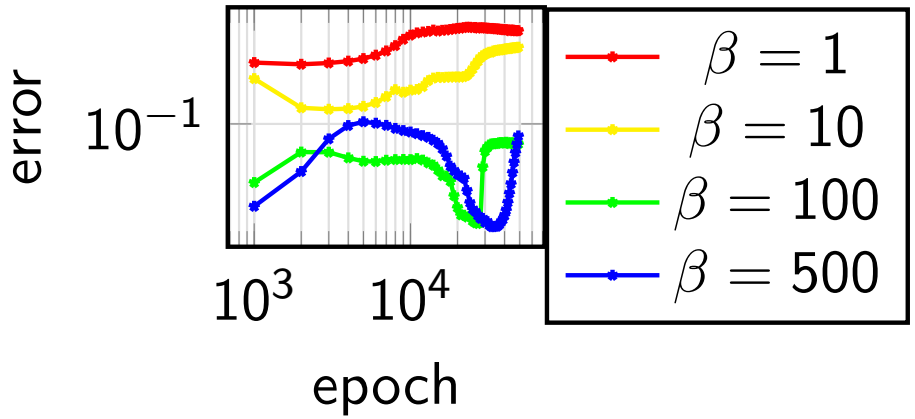
Accuracy I - Relative L^2 error (N = 833)



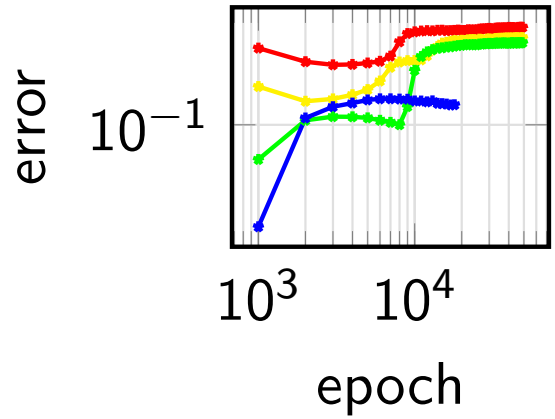
(e) DRM: uniform random points



(f) DRM: OT-based points

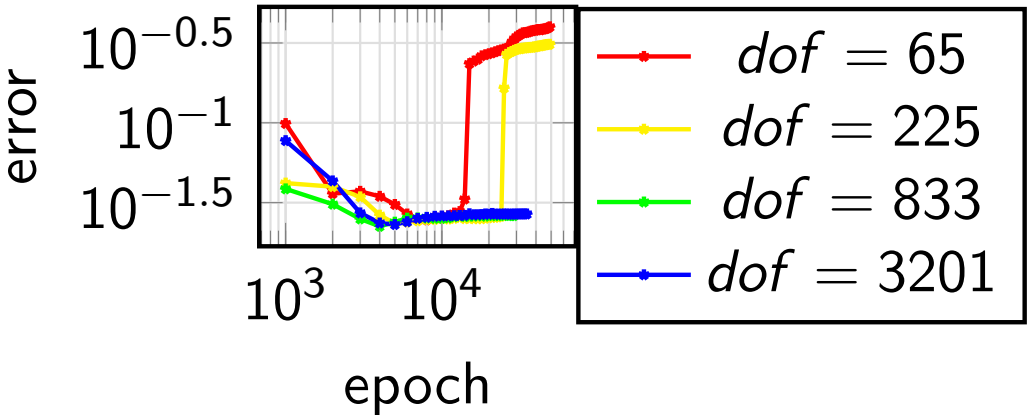


(g) DGM: random uniform points

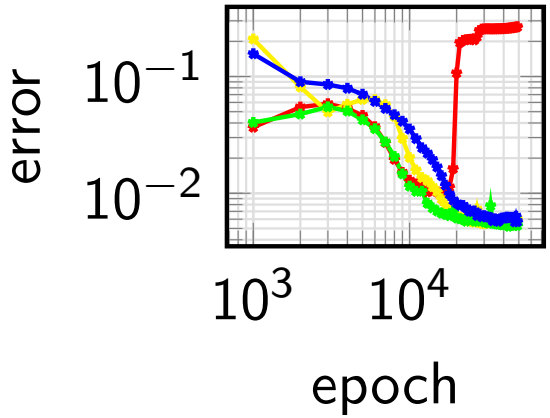


(h) DGM: OT-based points

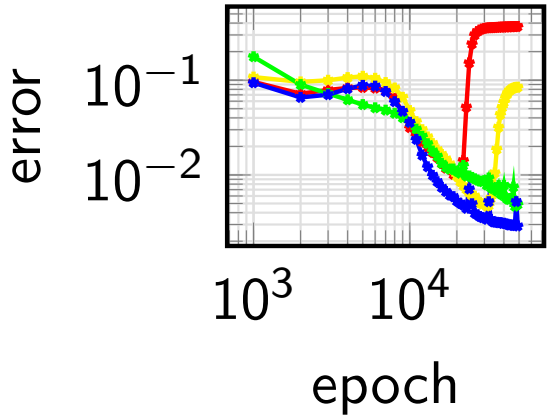
Accuracy II - relative L^2 error on OT collocation points



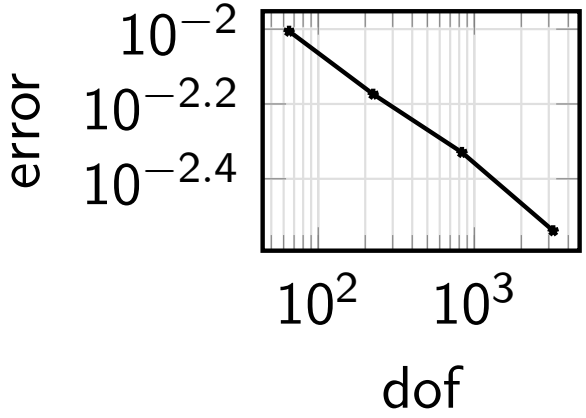
(i) $\beta = 100$



(j) $\beta = 500$



(k) $\beta = 1000$



(l) Error rate: $\beta = 1000$

Summary

- PINNS work best when combined with good numerical analysis methods
- The DNN can be trained to learn the equidistribution process, and outperforms other standard numerical methods
- Makes a big difference for elliptic two-point BVPs
- Smaller difference for convective problems, which need homotopy methods to work at all
- OT based r-adaptivity is very effective for 2D problems using the Deep Ritz method

- Next Goal: Implement the Rezoning approach for adapting the mesh and solving the PDE, maybe with a learned monitor function
- Proper convergence theory and proper test comparisons

References

- Andrew T. T. McRae, Colin J. Cotter, B. (2017). *Optimal-transport-based mesh adaptivity on the plane and sphere using finite elements*, SIAM SISC
- Budd, C., Huang, W., Russell, R. (2009). *Adaptivity with moving grids*, Acta Numerica
- Simone Appella, Chris Budd and Tristan Pryer (2021). *Adaptive meshes in non-convex domain using h-adaptive and Optimal Transport methos*, in preparation
- Weinan E and Bing Yu (2017). *The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems*, Communications in Mathematics and Statistics
- Krishnapriyan, Aditi Zhe, Shandian Kirby, Robert Mahoney, Michael (2021). *Characterizing possible failure modes in physics-informed neural networks.*
- Amanpreet Singh and Martin Bauer and Sarang Joshi (2021). *Physics Informed Convex Artificial Neural Networks (PICANNs) for Optimal Transport based Density Estimation*
- Williams F. et. al. (2019) *Gradient dynamics of shallow univariate ReLU Networks*, arXiv