Deep Learning and Computations of PDEs

Siddhartha Mishra

Seminar for Applied Mathematics (SAM),
D-MATH, ETH Zürich, Switzerland.
Abstract form of PDEs:

\[ D(u) = f, \quad u = u(x, t), \quad (t, x) \in \mathbb{R}_+ \times D \subset \mathbb{R}^d. \]

\( D \) is the Differential operator + Initial (Boundary) conditions.

Goals of computation:

- Fields: \( u(x, t) \).
- Observables: \( \mathcal{L} = \int_{\mathbb{R}_+} \int_D \psi(x, t)g(u(x, t)) \, dx \, dt \).

Variety of successful numerical methods: Finite Difference, Finite Element, Finite Volume, Spectral

Too expensive or infeasible in High Dimensions: \( d \geq 4 \)
PDEs in High-dimensions I

- PDEs of **Parametric** form:

\[ \mathcal{D}(u(x, t, y)) = f(y), \quad (t, x) \in \mathbb{R}_+ \times \mathbb{R}^d, \quad y \in Y \subset \mathbb{R}^{\tilde{d}}, \quad \tilde{d} >> 1. \]

- **UQ**: Y parametrizes probability space.

- **Optimal control and design**: Y parametrizes design space.

- Need to evaluate fields \( u(t, x, y) \) or observables \( L(y) = \int_{\mathbb{R}_+} \int_{\mathcal{D}} \Psi(x, t) g(u(x, t, y)) \, dx \, dt \) large number of times.

- Multiple calls to PDE solvers is very expensive.

- Approximate solution Fields \( u^*(t, x, y) \) or observables \( L^*(y) \) with deep neural networks \( u^*(t, x, y), L^*(y) \).
Example: Uncertain flow past airfoils

- Governing PDEs: Compressible *Euler* or *Navier-Stokes*
- Observables: *Lift*, *Drag*
- Uncertain parameters: Incident Mach Number, Angle of Attack, Pressure, Shape defects.
- Design parameters: Shape via *Hicks-Henne* functions.
Supervised learning with Deep Neural networks

\[ x = z^0 \rightarrow \text{Act fn} \rightarrow z^1 \rightarrow \text{Act fn} \rightarrow z^2 \rightarrow \text{Act fn} \rightarrow z^3 \rightarrow \text{Out fn} \rightarrow \hat{y} \]

- \( \mathcal{L}^*(y) = \sigma \circ C_K \circ \sigma \circ C_{K-1} \ldots \circ \sigma \circ C_2 \circ \sigma \circ C_1(y) \).
- At the k-th Hidden layer: \( y^{k+1} := \sigma(C_k y^k) = \sigma(W^k y^k + B^k) \).
- Parameters: \( \theta = \{W_k, B_k\} \in \Theta \), \( \sigma \): scalar Activation function.
- Training set: \( \mathcal{S} = \{y_i\}, \) for \( 1 \leq j \leq N \) with \( y_i \in Y \).
- Use Stochastic gradient descent (ADAM) to find,

\[
\theta^* := \arg \min_{\theta \in \Theta} \left[ J(\theta) + \lambda \mathcal{R} (\|W\|_q) \right], \quad J(\theta) := \sum_{i=1}^{N} |\mathcal{L}(y_i) - \mathcal{L}^*_\theta(y_i)|^p,
\]

- Trained Neural network \( \mathcal{L}^* = \mathcal{L}^*_\theta^* \) approximates \( \mathcal{L} \)
Supervised learning for observables

- Generate training data \( \mathcal{L}^\Delta(y_i) \) from simulation.
- Can ensure that \( \|\mathcal{L} - \mathcal{L}^\Delta\| \sim O(\epsilon) \).
- Can we find Trained DNN such that \( \|\mathcal{L}^* - \mathcal{L}^\Delta\| \sim O(\epsilon) \) ?
- Thrm (Yarotsky, Petersen et al.): If \( \mathcal{L} \in W^{s,p} \), \( \exists \) DNN \( \hat{\mathcal{L}} \) with ReLU activation, \( O(\epsilon^{-\frac{d}{s}}) \) parameters: \( \|\mathcal{L} - \hat{\mathcal{L}}\|_p \sim \epsilon \).
- Observables or Fields may not be very Regular
- If \( \mathcal{L} \in W^{1,\infty} \), \( d = 6 \): 1% error, need network of size \( 10^{12} \) !
- Approximation theory: No information about behavior of Trained Networks.
On Generalization Error of Trained Networks

- **Generalization Error**: \( \mathcal{E}_G := \int \| \mathcal{L}(y) - \mathcal{L}^*(y) \| d\bar{\mu}(y) \).
- Choose **Training set** \( \{y_i\} \) as i.i.d samples from \( \mu \in \text{Prob}(Y) \).
- We compute **Training error**: \( \mathcal{E}_T := \frac{1}{N} \sum_{i=1}^{N} |\mathcal{L}(y_i) - \mathcal{L}^*(y_i)| \).
- If \( \mathcal{L}^*(y_i) \) were independent, we bound:
  \[
  \mathcal{E}_G \sim \mathcal{E}_T + C \frac{\text{std}(\mathcal{L}) + \text{std}(\mathcal{L}^*)}{\sqrt{N}}.
  \]
- \( \mathcal{L}^*(y_i) \) could be **Highly correlated** \( \Rightarrow \) Very hard to bound \( \mathcal{E}_G \)!!
- Use **Statistical learning theory tools**: Bias-Variance decomposition, VC dimension, Rademacher complexity ....
A pragmatic bound on the generalization gap

- Introduce the **Validation Set**: \( \{z_j\}, 1 \leq j \leq N \) i.i.d from \( \mu \).
- Define **Validation gap**: 
  \[
  \mathcal{E}_{TV} = \left| \mathcal{E}_T - \frac{1}{N} \sum_{j=1}^{N} |\mathcal{L}(z_j) - \mathcal{L}^*(z_j)| \right|
  \]
- From Lye, SM, Molinaro 2020, we have:
  \[
  \mathcal{E}_G \sim \mathcal{E}_T + \mathcal{E}_{TV} + 2\sqrt{2} \frac{\text{std}(\mathcal{L}) + \text{std}(\mathcal{L}^*)}{\sqrt{N}}.
  \]
- In practice, one always has a **Validation Set**, but with fewer samples than the training set.
Sharpness of the bound on Generalization error

- **Projectile motion** with $d = 7$. 

![Graph showing the relationship between the horizontal range and the coordinate values.](image1)

![Graph illustrating the mean absolute error over training samples.](image2)
Assume we can find DNN such that $\mathcal{E}_T, \mathcal{E}_{TV} \ll 1$

Still **Generalization Error** behaves as

$$\mathcal{E}_G \sim \frac{\text{std}(\mathcal{L}) + \text{std}(\mathcal{L}^*)}{\sqrt{N}}.$$

If $\text{std}(\mathcal{L}) \sim \mathcal{O}(1)$, error of 1% requires $10^4$ training samples !!

**Challenge:** learn maps of low regularity in a data poor regime

Contrast with **Big Data** successes of machine learning.
Use Low discrepancy sequences \( \{y_i\}_{i=1}^{N} \in Y \) as Training Set

These sequences are Equidistributed (better spread out).

Examples: Sobol, Halton, Owen, Niederreiter ++

Basis of Quasi-Monte Carlo (QMC) integration.
Recall Generalization Error: $\mathcal{E}_G := \int \mathcal{L}(y) - \mathcal{L}^*(y) \, dy$.

Training error: $\mathcal{E}_T := \frac{1}{N} \sum_{i=1}^{N} |\mathcal{L}(y_i) - \mathcal{L}^*(y_i)|$

$\mathcal{E}_T$ is QMC quadrature for $\mathcal{E}_G$ with,

$$\mathcal{E}_G \leq \mathcal{E}_T + C \left( V_{HK}(\mathcal{L}), V_{HK}(\mathcal{L}^*) \right) \frac{(\log N)^{d}}{N},$$

With Hardy-Krause Variation $V_{HK}(f) \sim \int_{\mathcal{Y}} \left| \frac{\partial^d f(y)}{\partial y_1 \partial y_2 \ldots \partial y_d} \right| \, dy$

ReLU may not work, need smoother activations: Sigmoid, tanh
Parameterize airfoil shape in terms of Hicks-Henne functions:

\[ S = S_{ref} + \sum_{i=1}^{d} a_i \sin^{t_i} \left( \pi x^{\frac{\ln(0.5)}{\ln(x_{Mi})}} \right). \]
Drag Prediction for airfoils

Varying $d$

Sobol vs MC for $d = 20$
Higher Order QMC

- Train on Extrapolated Lattice points or Interlaced Lattice points (SM, Longo, Schwab, Rusch, 2020):
- For a class of Holomorphic maps $\mathcal{L}$:
  - Generalization gap decays as $N^{-2}$.
  - No curse of dimensionality !!
- Elliptic parametric PDE with affine parameters:
Applications I: Forward UQ

- Speedup over (Quasi-)Monte Carlo is guaranteed !!

Sample  | Lift PDF | Drag PDF
--- | --- | ---

<table>
<thead>
<tr>
<th>Observable</th>
<th>Speedup (MC)</th>
<th>Speedup (QMC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lift</td>
<td>246.02</td>
<td>6.64</td>
</tr>
<tr>
<td>Drag</td>
<td>179.54</td>
<td>8.56</td>
</tr>
</tbody>
</table>
Find \( y^* = \arg \min_{y \in Y} G(\mathcal{L}(y)) \), with PDE observable \( \mathcal{L} \).

- **DNNOpt**: Use standard optimizer for \( \arg \min_{y \in Y} G(\mathcal{L}^*(y)) \).

- Even for **Convex** cost functions + other hypotheses:
  - Slow decay of error + High variance: \( O \left( (KN)^{-\frac{1}{d}} \right) \).

- **ismo** *(Lye, SM, Ray, Praveen, 2020)*.

- An active learning variant of DNNOpt.

- Learner (DNN) queries Teacher (BFGS) for iteratively improving training sets.

- Fast decay of error + Low variance: \( O \left( (N)^{-\frac{K}{d}} \right) \).
Change airfoil shape to **Minimize Drag for constant Lift**.

- > **50% Drag reduction** at near constant lift.
What about the whole field?

- Find $\mathbf{DNN} \; \mathbf{u}^*(x, t, y) \approx \mathbf{u}(x, t, y)$, solution field of
- Training set is a grid in $D_T$ and Random (Sobol) points in $Y$.
- Analogous estimates on Generalization error ($SM, Molinaro$).
- Large networks might be needed.
- Long training times.
- Example: 2-D Incompressible Euler equations.
- $256^2$ spatial grid + 128 sobol points $\Rightarrow 2^{21}$ data points.
- Randomly choose $2^{14}$ data points.
- Approx 1.5% error but with networks with $50K$ parameters.
- PINNs could be a way out!
Results

Ground Truth

DNN
Many PDEs have explicit high dimensionality i.e., $d \geq 4$.

- **Kinetic Equations**: Boltzmann ($d = 6$)
- **Radiative transport Equation**: ($d \geq 5$)
- **Computational Finance** Black-Scholes ($d \gg 1$)
- **Computational Chemistry** Schrödinger ($d \gg 1$).

Using **Supervised learning** does not directly work in this context.

**PINNs** could be a way out.
Physics Informed Neural Networks (PINNs)

▶ Reintroduced and extensively developed by Karniadakis and collaborators, 2017-till date.
▶ > 50 articles on PINNs in 2 years !!
▶ Used for both forward and inverse problems for PDEs.
▶ Very little theoretical justification.
Let $X, Y$ be Banach spaces with $Y = L^p(D; \mathbb{R}^m)$, with $1 \leq p < \infty$.

- $X^* \subset X$, $Y^* \subset Y$.
- $D = D$ or $D = D \times (0, T)$, with $D \subset \mathbb{R}^d$.
- Abstract PDE is

$$D(u) = f,$$

- $D : X^* \mapsto Y^*$ is the **Differential operator**.
- $f \in Y^*$ is the input.
- **Boundary** (Initial) conditions are implicit.
PINNs for the PDE forward problem

- For Parameters $\theta \in \Theta$, $u_\theta : \mathbb{D} \mapsto \mathbb{R}^m$ is a DNN, with $u_\theta \in X^*$
- Aim: Find $\theta \in \Theta$ such that $u_\theta \approx u$ (in suitable sense).
- Define PDE Residual:
  \[ \mathcal{R} := \mathcal{R}_\theta(y) = \mathcal{D}(u_\theta(y)) - f(y), \ y \in \mathbb{D} \ \mathcal{R}_\theta \in Y^*, \ \forall \theta \in \Theta \]
- PINNs are minimizers of $\|\mathcal{R}_\theta\|_Y \sim \int_{\mathbb{D}} |\mathcal{R}_\theta(y)|^p \ dy$
- Replace Integral by Quadrature!
- Let $S = \{y_i\}_{1 \leq i \leq N}$ be quadrature points in $\mathbb{D}$, with weights $w_i$
- Could be Random, Sobol, Grid points (Gauss rules)
- PINN for approximating PDE is defined as $u^* = u_{\theta^*}$ such that
  \[ \theta^* = \arg\min_{\theta \in \Theta} \sum_{i=1}^{N} w_i |\mathcal{R}_\theta(y_i)|^p \]
On generalization error for PINNs

- Generalization error is error of network on Unseen data
- For PINNs, it is Approximation error: $\mathcal{E}_G := \|u - u^*\|_X$
- For estimates on $\mathcal{E}_G$, we need assumptions:
- Assumptions on underlying Quadrature rule:
- For any function $f \in Y^*$,

$$\left| \int_D f(y) dy - \sum_{i=1}^{N} w_i f(y_i) \right| \leq C_{quad} N^{-\alpha},$$

- for some $\alpha > 0$ and $C_{quad} = C_{quad} (d, \|f\|_{Y^*})$
Assumptions on PDE $\mathcal{D}(u) = f$, $\mathcal{D}(\bar{u}) = \bar{f}$

- Solutions of the forward problem for PDE are **Stable**:
  \[
  \|u - \bar{u}\|_X \leq C_{pde} \|\mathcal{D}(u) - \mathcal{D}(\bar{u})\|_Y, \quad \forall u, \bar{u} \in X^*
  \]
  with constant $C_{pde} = C_{pde} (\|u\|_Z, \|\bar{u}\|_Z)$ for some $Z \subset X$.

- For **Linear** PDEs: a **Bounded inverse** $\mathcal{D}^{-1}$ is sufficient.

- For **Nonlinear** PDEs: a **Bounded inverse** $\mathcal{D}_{u,\bar{u}}^{-1}$ of **Linearized** operator is sufficient:
  \[
  \mathcal{D}(u) - \mathcal{D}(\bar{u}) = \mathcal{D}_{u,\bar{u}} (u - \bar{u}).
  \]
Estimate on PINN generalization error for $D(u) = f$

- Shown in SM, Molinaro 2020, based on observations,

$$R^* := R_{\theta^*} = D(u^*) - f = D(u^*) - D(u),$$

$$\|R^*\|_Y^p \leq E_T^p + C_{quad} N^{-\alpha}, \text{ quadrature error},$$

- With computable Training error $E_T = \left( \sum_{i=1}^{N} w_i |R_{\theta^*}(y_i)|^p \right)^{1/p}$,

$$E_G = \|u - u^*\|_X$$

$$\leq C_{pde} \left( \|D(u^*) - D(u)\|_Y \right) \quad \text{(PDE stability)}$$

$$\leq C_{pde} \|R^*\|_Y$$

$$\Rightarrow E_G \leq C_{pde} \left( E_T + C_{quad}^p N^{-\alpha/p} \right)$$
A Lax Equivalence theorem for PINNs

- Let \( u^*_N \) be a PINN with training set of size \( N \).
- Stability: of the PDE + Regularity: \( \|u\|_Z < +\infty \)
- Stability of PINNs: \( \|u^*_N\|_Z \leq C \) (independent of \( N \))
- Stability of Residuals: \( \|R^*_N\|_{Y^*} \leq C \) (independent of \( N \)).
- Consistency of training: \( \lim_{N \to \infty} \mathcal{E}_{T,N} = 0 \)
- Stability + Consistency \( \Rightarrow \) Convergence:
  \[
  \lim_{N \to 0} \|u - u^*_N\|_X = 0.
  \]
- Estimate for Random training points:
  \[
  \bar{\mathcal{E}}_G \leq C_{pde} \left( \bar{\mathcal{E}}_T + \mathcal{E}_{TV} + \frac{\text{std}(|R^*|)}{\sqrt{N}} \right)
  \]
Example 1: Semilinear Heat equation

- **Parabolic PDE with (globally) Lipschitz nonlinearity:**

  \[
  u_t = \Delta u + f(u), \quad (x, t) \in D \times [0, T],
  \]

  \[
  u(x, 0) = \bar{u}(x), \quad x \in D,
  \]

  \[
  u|_{\partial D} = 0.
  \]

- **Training set:** \( S = S_{\text{int}} \cup S_{\text{tb}} \cup S_{\text{sb}} \)
PINNs

- **Deep Neural networks**: \((x, t) \mapsto u_\theta(x, t), \theta \in \Theta.\)

- **Residuals**
  - **Interior Residual**: \(R_{\text{int}, \theta} = \partial_t u_\theta - \Delta u_\theta - f(u_\theta).\)
  - **Temporal boundary residual**: \(R_{\text{tb}, \theta} = u_\theta(\cdot, 0) - \bar{u}\)
  - **Spatial boundary residual**: \(R_{\text{sb}, \theta} = u_\theta|_{\partial D}\).

- **Loss function**:

\[
J = \sum_{n=1}^{N_{\text{tb}}} w_{n}^{tb}|R_{\text{tb}, \theta}(x_n)|^2 + \sum_{n=1}^{N_{\text{sb}}} w_{n}^{sb}|R_{\text{sb}, \theta}(x_n, t_n)|^2 + \sum_{n=1}^{N_{\text{int}}} w_{n}^{\text{int}}|R_{\text{int}, \theta}|^2.
\]
Estimate on Generalization error

- **Generalization error:** 
  \[(E_G)^2 := \int_0^T \int_D |u(x, t) - u^*(x, t)|^2 dx dt.\]

- **Training errors:**

  \[E_T^2 := \sum_{n=1}^{N_{tb}} w_n^{tb} |R_{tb, \theta^*}|^2 + \sum_{n=1}^{N_{sb}} w_n^{sb} |R_{sb, \theta^*}|^2 + \sum_{n=1}^{N_{int}} w_n^{int} |R_{int, \theta^*}|^2.\]

- **Estimate:**

  \[E_G \leq C_1 \left( E_T^{tb} + E_T^{int} + C_2 (E_T^{sb})^{1/2} + C_q^{1/2} \left( N_{tb}^{-\frac{\alpha_{tb}}{2}} + N_{int}^{-\frac{\alpha_{int}}{2}} + N_{sb}^{-\frac{\alpha_{sb}}{4}} \right) \right).\]
Results for 1-d Heat Eqn with Random points
Results for Multi-d Heat Eqn with Random points

- PINN with Depth 4, Width 20, Interior training points $2^{16}$, Boundary points $2^{15}$

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Training Error</th>
<th>Generalization error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$2.8 \times 10^{-5}$</td>
<td>0.0035%</td>
</tr>
<tr>
<td>5</td>
<td>0.0002</td>
<td>0.016%</td>
</tr>
<tr>
<td>10</td>
<td>0.0003</td>
<td>0.03%</td>
</tr>
<tr>
<td>20</td>
<td>0.006</td>
<td>0.79%</td>
</tr>
<tr>
<td>50</td>
<td>0.006</td>
<td>1.5%</td>
</tr>
<tr>
<td>100</td>
<td>0.004</td>
<td>2.6%</td>
</tr>
</tbody>
</table>

- No Curse of dimensionality !!

Siddhartha Mishra  Deep Learning and PDEs
Example 2: Viscous scalar conservation laws

▶ Nonlinear hyperbolic-parabolic PDE.

\[ u_t + \text{div} \ f(u) = \nu \nabla u \quad (x, t) \in D \times [0, T], \]
\[ u(x, 0) = \bar{u}(x), \quad x \in D, \quad u|_{\partial D} = 0. \]

▶ Analysis can be readily extended to
  ▶ Periodic and other boundary conditions.
  ▶ More general viscous terms.
  ▶ Systems of equations.

▶ Training set: \( S = S_{\text{int}} \cup S_{\text{tb}} \cup S_{\text{sb}} \)
Deep Neural networks: $(x, t) \mapsto u_\theta(x, t), \theta \in \Theta$.

Residuals
- Interior Residual: $R_{int, \theta} = \partial_t u_\theta + \text{div } f(u_\theta) - \nu \Delta u_\theta$.
- Temporal boundary residual: $R_{tb, \theta} = u_\theta(\cdot, 0) - \bar{u}$
- Spatial boundary residual: $R_{sb, \theta} = u_\theta|_{\partial D}$.

Loss function:
$$J = \sum_{n=1}^{N_{tb}} w_n^{tb} |R_{tb, \theta}(x_n)|^2 + \sum_{n=1}^{N_{sb}} w_n^{sb} |R_{sb, \theta}|^2 + \sum_{n=1}^{N_{int}} w_n^{int} |R_{int, \theta}|^2,$$

where $\varepsilon_{tb}^T$ and $\varepsilon_{sb}^T$ represent the temporal and spatial boundary errors, respectively.
Generalization error: \((E_G)^2 := \int_0^T \int_D |u(x, t) - u^*(x, t)|^2 dx dt\).

Estimate:

\[
E_G^2 \leq Ce^{CT} \left( (E_{tb})^2 + (E_{int})^2 + C_b(E_{sb}) \right) \\
+ Ce^{CT} C_q \left( N_{tb}^{-\alpha_{tb}} + N_{int}^{-\alpha_{int}} + N_{sb}^{-\frac{\alpha_{sb}}{2}} \right)
\]

Constants: \(C = C (\|\nabla u\|_{L^\infty})\)

From viscous profiles, we know that \(\|\nabla u\|_{L^\infty} \sim \frac{1}{\sqrt{\nu}}\)

Generalization error can blow up near shocks!!
Results for 1-D Burgers’ with Compression

- Sobol points, $N_{int} = 8192$, $N_{tb} = N_{sb} = 256$, Depth 8, Width 20.

$\nu = 10^{-2}$  \hspace{2cm} $\nu = 10^{-3}$  \hspace{2cm} $\nu = 0$

<table>
<thead>
<tr>
<th>Viscosity</th>
<th>Training Error</th>
<th>Generalization error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{0.01}{\pi}$</td>
<td>0.0005</td>
<td>1.0%</td>
</tr>
<tr>
<td>$\frac{0.005}{\pi}$</td>
<td>0.0008</td>
<td>1.2%</td>
</tr>
<tr>
<td>$\frac{0.001}{\pi}$</td>
<td>0.009</td>
<td>11.0%</td>
</tr>
<tr>
<td>0</td>
<td>0.08</td>
<td>23.1%</td>
</tr>
</tbody>
</table>
Results for 1-D Burgers’ with Rarefaction

- Sobol points, $N_{int} = 8192$, $N_{tb} = N_{sb} = 512$, Depth 4, Width 20.

\[ \nu = 10^{-2} \quad \nu = 10^{-3} \quad \nu = 0 \]

<table>
<thead>
<tr>
<th>Viscosity</th>
<th>Training Error</th>
<th>Generalization error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01 ( \pi )</td>
<td>0.0043</td>
<td>2.2%</td>
</tr>
<tr>
<td>0.005 ( \pi )</td>
<td>0.0034</td>
<td>1.8%</td>
</tr>
<tr>
<td>0.001 ( \pi )</td>
<td>0.0004</td>
<td>1.6%</td>
</tr>
<tr>
<td>0 ( \pi )</td>
<td>0.0003</td>
<td>1.2%</td>
</tr>
</tbody>
</table>
Example 3: Incompressible Euler equations

- Nonlinear PDE of form
  \[ u_t + (u \cdot \nabla)u + \nabla p = f, \quad (x, t) \in D \times [0, T], \]
  \[ \text{div } u = 0, \quad (x, t) \in D \times [0, T], \]
  \[ u(x, 0) = \bar{u}(x), \quad x \in D, \quad u \cdot n|_{\partial D} = 0. \]

- Analysis can be readily extended to
  - Periodic and other boundary conditions.
  - Navier-Stokes Equations.

- Training set: \( S = S_{\text{int}} \cup S_{tb} \cup S_{sb} \)
PINNs

- Deep Neural networks: \((x, t) \mapsto u_\theta(x, t), p_\theta(x, t), \theta \in \Theta\).

- Residuals
  - Velocity Residual: \(R_{u,\theta} = \partial_t u_\theta + (u_\theta \cdot \nabla)u_\theta + \nabla p_\theta - f\)
  - Divergence Residual: \(R_{\text{div},\theta} \equiv \text{div} u_\theta\)
  - Temporal and Spatial boundary residuals as before.

- Loss function:

\[
J(\theta) = \sum_{n=1}^{N_{tb}} w_n^{tb} |R_{tb,\theta}(x_n)|^2 + \sum_{n=1}^{N_{sb}} w_n^{sb} |R_{sb,\theta}|^2
\]

\[
\left( \mathcal{E}_{tb}^2 \right) + \sum_{n=1}^{N_{int}} w_n^{int} |R_{u,\theta}|^2
\]

\[
\left( \mathcal{E}_{u}^2 \right)
\]

\[
\sum_{n=1}^{N_{int}} w_n^{int} |R_{\text{div},\theta}|^2 \left( \mathcal{E}^d_T \right)^2
\]

\[
\sum_{n=1}^{N_{int}} w_n^{int} |R_{\text{div},\theta}|^2 \left( \mathcal{E}^d_T \right)^2
\]
Generalization error: \((\mathcal{E}_G)^2 := \int_0^T \int_D |u(x, t) - u^*(x, t)|^2 \, dx \, dt\).

Estimate:

\[
\mathcal{E}_G^2 \leq C e^{CT} \left( (\mathcal{E}_{tb})^2 + (\mathcal{E}_u)^2 + C_b (\mathcal{E}_{sb} + \mathcal{E}_d) \right) \\
+ C e^{CT} C_q \left( N_{tb}^{-\alpha_{tb}} + N_{int}^{-\frac{\alpha_{int}}{2}} + N_{sb}^{-\frac{\alpha_{sb}}{2}} \right)
\]

Constants: \(C = C (\|\nabla u\|_{L^\infty})\)

Finite \(C\) for 2-D Euler (or NS) equations with Smooth data.

Estimate can blow up in 3-D!!
Results for 2-D Taylor vortex
Results for 2-D Double Shear Layer

\[ \omega(x, y), \ t=0.0 \]

\[ \omega(x, y), \ t=2.0 \]

\[ \omega(x, y), \ t=4.0 \]
Example 4: Radiative Transfer Equations

- **Radiative Intensity** $u \in L^p((0, T) \times \mathbb{R}^d \times S^{d-1} \times \mathbb{R}_+)$:

$$\frac{1}{c} u_t + n \cdot \nabla u + (k(x, \nu) + \omega(x, \nu)) u - \frac{\sigma(x, \nu)}{4\pi} \int_{R_+} \int_{S} \Phi(n, n', \nu, \nu') u \, dn' \, d\nu' = f(x, t, n, \nu).$$

- With Scattering kernel $\Phi$
- In general, can be a 8-dimensional Integro-Differential PDE.
- Straightforward to use PINNs (SM, Molinaro, forthcoming)
Monochromatic 1-D with scattering
Polychromatic 3-D with scattering
PINNs for Inverse problems

- Started with Raissi, Perdikaris, Karniadakis, 2018.
- Extensively developed for various inverse problems by Karniadakis and collaborators, 2018-
- We focus on one particular class of inverse problems here.
Data Assimilation or Unique continuation problem

- $\mathbb{D}$ (Space-time) domain, $\mathbb{D}'$ Observation domain.
- Abstract PDE is
  \[ \mathbb{D}(u) = f \text{ in } \mathbb{D}, \quad \mathbb{D} : X^* \subset X \mapsto Y^* \subset Y. \]
- Incomplete information on Boundary conditions.
- Measurements, possibly noisy, on $\mathbb{D}'$ of Observable:
  \[ \mathbb{L}(u) = g \text{ in } \mathbb{D}', \quad \mathbb{L} : X^* \mapsto Z^* \subset Z. \]
- Goal: Reconstruct $u$ given $f, g$. 

Siddhartha Mishra
Deep Learning and PDEs
PINNs for Data Assimilation problem

- For Parameters $\theta \in \Theta$, $u_\theta : \mathbb{D} \mapsto \mathbb{R}^m$ is a DNN, with $u_\theta \in X^*$
- Define PDE Residual:
  \[ R := R_\theta(y) = \mathcal{D}(u_\theta(y)) - f(y), \quad y \in \mathbb{D} \quad R_\theta \in Y^*, \quad \forall \theta \in \Theta \]
- Define Data Residual:
  \[ R_d := R_{d,\theta}(y) = \mathcal{L}(u_\theta(z)) - g(z), \quad z \in \mathbb{D}' \quad R_{d,\theta} \in Z^*, \quad \forall \theta \in \Theta \]
- PINNs are minimizers of
  \[ \|R_\theta\|_Y + \|R_{d,\theta}\|_Z \sim \int_{\mathbb{D}} |R_\theta(y)|^{p_y} dy + \int_{\mathbb{D}'} |R_{d,\theta}(z)|^{p_z} dz \]
- Replace Integrals by Quadrature!
- PINN for approximating PDE is defined as $u^* = u_{\theta_*}$ such that
  \[ \theta_* = \arg \min_{\theta \in \Theta} \sum_{i=1}^{N_{int}} w_i R_\theta(y_i) + \sum_{i=1}^{N_d} w_d^i R_{d,\theta}(z_j) \]
Assumptions on the Continuous Problem

▶ Solutions need to satisfy **Conditional Stability** estimate:

\[ \|u - \bar{u}\|_{L^p(E)} \leq C_{pd} \left( \|D(u) - D(\bar{u})\|_Y^{T_p} + \|L(u) - L(\bar{u})\|_Z^{T_d} \right), \]

▶ with \( D' \subset E \subset D \) and \( C_{pd} = C_{pd} \left( \|u\|_\hat{X}, \|\bar{u}\|t_\hat{X} \right). \)

▶ A slightly weaker version,

\[ \|u - \bar{u}\|_{L^p(E)} \leq C_{pd} \omega \left( \|D(u) - D(\bar{u})\|_Y + \|L(u) - L(\bar{u})\|_Z \right), \]

▶ with modulus of continuity \( \omega \).

▶ Often derived from

▶ Three-balls inequalities

▶ Carleman estimates.
Observe that $R^* = D(u^*) - D(u)$, $R_d^* = L(u^*) - L(u)$.

Training errors

\[ E_{p,T} = \sum_{i=1}^{N_{int}} w_i R_{\theta^*}(y_i), \quad E_{d,T} = \sum_{i=1}^{N_d} w^d_j R_{d,\theta^*}(z_j) \]

Error estimate:

\[
\mathcal{E}_G = \| u - u^* \|_{L^p(E)} \\
\leq C_{pd} \left( \| D(u^*) - D(u) \|_{\ell^p}^{\tau_p} + \| L(u^*) - L(u) \|_{\ell^d}^{\tau_d} \right) (\text{stability}) \\
\leq C_{pd} \left( \| R^* \|_{\ell^p}^{\tau_p} + \| R_d^* \|_{\ell^d}^{\tau_d} \right) \\
\Rightarrow \mathcal{E}_G \leq C_{pd} \left( \mathcal{E}_{p,T}^{\tau_p} + \mathcal{E}_{d,T}^{\tau_d} + C_q^{p_T} N_{\text{int}}^{\alpha_T^{\tau_p}} + C_{q_d}^{p_T} N_d^{\alpha_d^{\tau_d}} \right)
\]
Example 1: Poisson Equation

- Model Elliptic PDE with Data assimilation problem:

\[-\Delta u = f, \quad \text{in } D \subset \mathbb{R}^d,\]
\[u|_{D'} = g \quad \text{in } D' \subset D.\]

- Conditional stability by Three balls inequality (Alessandrini et al, 2009).

- Generalization error estimate is:

\[
\|u - u^*\|_{H^1} \leq C \left| \log \left( \mathcal{E}_{p,T} + \mathcal{E}_{d,T} + C_{q}^{\frac{1}{2}} N_{\text{int}}^{\frac{-\alpha}{2}} + C_{qd}^{\frac{1}{2}} N_{d}^{\frac{-\alpha d}{2}} \right) \right|^{-\tau}.
\]

- With \( C = C \left( \|u\|_{H^1}^{1-\tau} + \|u^*\|_{H^1}^{1-\tau} \right) \) and \( 0 < \tau < 1. \)
Results in 2-D
Results in 2-D

- No Noise: Depth 4, Width 24

<table>
<thead>
<tr>
<th>N</th>
<th>Training Error</th>
<th>Generalization error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$20^2$</td>
<td>0.0008</td>
<td>1.1%</td>
</tr>
<tr>
<td>$40^2$</td>
<td>0.0006</td>
<td>1.0%</td>
</tr>
<tr>
<td>$80^2$</td>
<td>0.0005</td>
<td>0.9%</td>
</tr>
<tr>
<td>$160^2$</td>
<td>0.0004</td>
<td>0.8%</td>
</tr>
</tbody>
</table>

- With 1% noise:

<table>
<thead>
<tr>
<th>N</th>
<th>Training Error</th>
<th>Generalization error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$20^2$</td>
<td>0.012</td>
<td>2.3%</td>
</tr>
<tr>
<td>$40^2$</td>
<td>0.013</td>
<td>1.9%</td>
</tr>
<tr>
<td>$80^2$</td>
<td>0.013</td>
<td>1.7%</td>
</tr>
<tr>
<td>$160^2$</td>
<td>0.013</td>
<td>1.3%</td>
</tr>
</tbody>
</table>
Example 2: Heat Equation

- Model Parabolic PDE with Data assimilation problem:

\[
\begin{align*}
    u_t - \Delta u &= f, \quad \text{in } D_T = D \times (0, T), \\
    u|_{\partial D} &= 0, \\
    u|_{D'_T} &= g \quad \text{in } D'_T \subset D_T
\end{align*}
\]

- Conditional stability by Carleman estimate (Imanuvilov, 1995).

- Generalization error estimate is:

\[
\| u - u^* \|_{L^2(H^1)} \leq C \left( \mathcal{E}_{d,T} + \mathcal{E}_{int,T} + \mathcal{E}_{sb,T} \right) \\
+ CC_q \left( N_{int}^{-\frac{\alpha}{2}} + N_{sb}^{-\frac{\alpha_{sb}}{2}} + N_{d}^{-\frac{\alpha_d}{2}} \right).
\]
Results

Siddhartha Mishra
Deep Learning and PDEs
Results with Random points

• 1-D : Depth 4, Width 24

<table>
<thead>
<tr>
<th>N</th>
<th>Training Error</th>
<th>Generalization error</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 × 50</td>
<td>0.001</td>
<td>0.55%</td>
</tr>
<tr>
<td>16 × 100</td>
<td>0.0009</td>
<td>0.52%</td>
</tr>
<tr>
<td>16 × 200</td>
<td>0.0007</td>
<td>0.47%</td>
</tr>
</tbody>
</table>

• Multi-D with $N = 2^{14}$ training points

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Training Error</th>
<th>Generalization error</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.0003</td>
<td>0.044%</td>
</tr>
<tr>
<td>10</td>
<td>0.0003</td>
<td>0.07%</td>
</tr>
<tr>
<td>20</td>
<td>0.0003</td>
<td>0.62%</td>
</tr>
<tr>
<td>50</td>
<td>0.0003</td>
<td>1.68%</td>
</tr>
<tr>
<td>100</td>
<td>0.0003</td>
<td>2.0%</td>
</tr>
</tbody>
</table>
Example 3: Wave Equation

- Model **Hyperbolic** PDE with Data assimilation problem:

  \[ u_{tt} - \Delta u = f, \quad \text{in } D_T = D \times (0, T), \]
  \[ u|_{\partial D} = 0, \]
  \[ u|_{D'_T} = g \quad \text{in } D'_T \subset D_T \]

- Conditional stability by **Carleman estimate** (Bardos, Lebeau, Rauch, 1989) under a **Geometric control condition**.

- Generalization error estimate is:

\[
\| u - u^* \|_{C(L^2)} \leq \kappa \left( E_d, T + E_{int}, T + E_{sb}, T \right) \\
+ \kappa C_q \left( N_{int}^{-\alpha_2} + N_{sb}^{-\alpha_{sb}^2} + N_{d}^{-\alpha_d^2} \right).
\]
1-D results under GCC

\[ u(t, x) \]

\[ u^*(t, x) \]

\[ |u(t, x) - u^*(t, x)| \]
1-D results without GCC

\[ D_T \]

\[ D'_T \]

\[ u(t, x) \]

\[ u^*(t, x) \]

\[ |u(t, x) - u^*(t, x)| \]
## Comparison

- **With GCC:**

<table>
<thead>
<tr>
<th>N</th>
<th>Training Error</th>
<th>Generalization error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$60^2$</td>
<td>0.0012</td>
<td>0.29%</td>
</tr>
<tr>
<td>$90^2$</td>
<td>0.0011</td>
<td>0.28%</td>
</tr>
<tr>
<td>$120^2$</td>
<td>0.0008</td>
<td>0.2%</td>
</tr>
</tbody>
</table>

- **Without GCC:**

<table>
<thead>
<tr>
<th>N</th>
<th>Training Error</th>
<th>Generalization error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$60^2$</td>
<td>0.0011</td>
<td>2.2%</td>
</tr>
<tr>
<td>$90^2$</td>
<td>0.0009</td>
<td>1.5%</td>
</tr>
<tr>
<td>$120^2$</td>
<td>0.0008</td>
<td>1.4%</td>
</tr>
</tbody>
</table>
Model **Indefinite PDE** with Data assimilation problem:

\[
\Delta u + \nabla p = f, \quad \text{in } \mathbb{D} \subset \mathbb{R}^d, \\
\left. u \right|_{\mathbb{D}'} = g \quad \text{in } \mathbb{D}' \subset \mathbb{D}.
\]

Conditional stability by **Three balls inequality** (Lin, Uhlmann, Wang, 2010).

Generalization error estimate is:

\[
\|u - u^*\|_{L^2(B_R)} \sim O \left( \mathcal{E}^{1-\tau}_{p,T} (1 + \mathcal{E}^\tau_{d,T}) + N^{-(1-\tau)}\alpha (1 + N^{-\alpha d}\tau) \right)
\]

With \(0 < \tau < 1\).
2-D Results

<table>
<thead>
<tr>
<th>N</th>
<th>Training Error</th>
<th>$\mathcal{E}_G(u)$</th>
<th>$\mathcal{E}_G(p)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$20^2$</td>
<td>0.0007</td>
<td>2.3%</td>
<td>5.6%</td>
</tr>
<tr>
<td>$40^2$</td>
<td>0.0004</td>
<td>1.7%</td>
<td>4.0%</td>
</tr>
<tr>
<td>$80^2$</td>
<td>0.0004</td>
<td>1.5%</td>
<td>3.5%</td>
</tr>
</tbody>
</table>
2-D Results

\begin{align*}
&u_1(x_1, x_2) \\
&u_2(x_1, x_2) \\
&p(x_1, x_2) \\
&p'(x_1, x_2)
\end{align*}
Supervised learning for observables in Parametric PDEs.

Use of QMC training points

Very successful for Many-query problems (UQ, PDE constrained optimization).

PINNs for forward and inverse problems for PDEs.

Very successful for high-dimensional PDEs, Data assimilation.

Caveats:

- Error estimate of form: $\mathcal{E}_G \sim \mathcal{O}(\mathcal{E}_T + N^{-\alpha})$
- No A priori control on training error.
- Training error computed A posteriori.
- Training is solving very high-dim non-convex optimization
- Subject to stochasticity + saturation (local minima)

Think of error estimate as if it trains well, it generalizes well.