Recent Advances in Physics-Informed Machine Learning
AAAI 2024 Tutorial
Yiping Lu (NYU & Northwestern), Grant M. Rotskoff (Stanford)
2024/2
Table of Contents
1 Why Physics-Informed Machine Learning

- Why Physics-Informed Machine Learning
- Formulation for Physics-Informed Machine Learning
- Differential Equation Solving Examples
- Theory Behind Physics-Informed Neural Network Advanced PINN
- Operator Learning System Identification
- Summary
Scientific Paradigm
1 Why Physics-Informed Machine Learning

Physical Science · · · · · theoretical derivation combined with experimental verification to study natural phenomena
Numerical Science · · · · · numerical simulation to understand complex real systems

Paul M. Dirac (Ŵżŵż)
"The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws lead to equations much too complicated to be soluble."

Accurate "constitutive equation" — e.g. Newton's gravitation law! Kepler's Law
Scientific Paradigm

1 Why Physics-Informed Machine Learning

Physical Science • • • • •
theoretical derivation combined with experimental verification to study natural phenomena

Numerical Science • • • • •
umerical simulation to understand complex real systems

Paul M. Dirac (1929)

“The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble.”
Scientific Paradigm

1 Why Physics-Informed Machine Learning

Physical Science · · · · ·
- theoretical derivation combined with experimental verification to study natural phenomena
Numerical Science · · · · ·
- numerical simulation to understand complex real systems

Paul M. Dirac (1929)

“The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble.”

- Accurate “constitutive equation”
  - e.g. Newton’s gravitational law → Kepler’s Law
- Efficient algorithms required
Challenges for Computational Physics

1 Why Physics-Informed Machine Learning

Traditional numerical methods approximate general functions using polynomials or piecewise polynomials, however...
Traditional numerical methods approximate general functions using polynomials or piecewise polynomials, however...

**Curse of Dimensionality**

The cost to represent a function is exponential in the dimensionality.

Physical systems *require* high-dimensional representations, *e.g.* dimension of quantum many-body problem $\propto \#$ electrons
Inverse problems/optimal design involve solving

$$\min_x \mathcal{L}(F(x)),$$

where $F$ is the forward process, such as a physics simulation, and $\mathcal{L}$ is the objective aim to optimize. Even when a single iteration of this forward process is manageable, the overall task becomes computationally infeasible due to the iterative optimization process.
Combating the Curse of Dimensionality with ML

1 Why Physics-Informed Machine Learning

- Physical Science
  - theoretical derivation combined with experimental verification to study natural phenomena
- Numerical Science
  - numerical simulation to understand complex real systems

Combating these challenges using Machine (Deep) Learning!
Combating Curse of Dimensionality with ML

1 Why Physics-Informed Machine Learning

- Physical Science
  - theoretical derivation combined with experimental verification to study natural phenomena
- Numerical Science
  - numerical simulation to understand complex real systems
- Machine Learning
  - understand and build models that leverage empirical data to improve performance

Neural Networks provide tools to build flexible, universal, and efficient approximations for complex high-dimensional functions and functionals.

- **In practice**
  - Imagenet (32x32 dimension)
  - Alpha Go (19x19 dimension)
  - Large Language Models ($d_{model} \sim \mathcal{O}(10^3)$)
- **In theory**
  - Separation to Kernel (Linear) Methods
  - Depth Separation
What is Physics-Informed Machine Learning?

1 Why Physics-Informed Machine Learning

Physical Science  · · · · · theoretical derivation combined with experimental verification to study natural phenomena
Numerical Science  · · · · · numerical simulation to understand complex real systems
Machine Learning  · · · · · understand and build models that leverage empirical data to improve performance

Physics-Informed Machine Learning study potential benefits for machine learning models by incorporating the physical prior such as

- **Differential Equations**: ODEs, PDEs, S(P)DEs
- **Law of conservation, Symmetry** ...
What is Physics-Informed Machine Learning?

1. Why Physics-Informed Machine Learning

<table>
<thead>
<tr>
<th>Physical Science</th>
<th>theoretical derivation combined with experimental verification to study natural phenomena</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numerical Science</td>
<td>numerical simulation to understand complex real systems</td>
</tr>
<tr>
<td>Machine Learning</td>
<td>understand and build models that leverage empirical data to improve performance</td>
</tr>
</tbody>
</table>

Physics-informed Machine Learning study potential benefits for machine learning models by incorporating the physical prior such as

- **Differential Equations**: ODEs, PDEs, S(P)DEs
- **Law of conservation, Symmetry** ...

Applications include

- Quantum Many-body Problem
- Turbulence Models
- Modeling Rare Events

What are the Challenges in PIML?

1 Why Physics-Informed Machine Learning

**Representation: Higher Dimension**

Imagenet only 32 x 32 dimension, which can only simulate ~300 molecules. Thus we need to understand:

- function space that we can approximate in high-dimension.
- physical prior can help to represent functions more efficiently.
- approximation theory in infinite dimensional.
What are the Challenges in PIML?

1 Why Physics-Informed Machine Learning

**Representation: Higher Dimension**

Imagenet only 32 x 32 dimension, which can only simulate $\sim 300$ molecules. Thus we need to understand

- function space that we can approximate in high-dimension.
- physical prior can help to represent functions more efficiently.
- approximation theory in infinite dimensional.

**Generalization: Expensive Data Collection**

Labeling data for scientific research is expensive, thus we need to consider the generalization theory for physics-informed machine learning.

- **Small Data**
  - Lots of Physics
- **Big Data**
  - No Physics
What is this tutorial about?
1. Why Physics-Informed Machine Learning

How to represent a physical solution and why it generalizes for

- Solving Differential Equations and Optimal Control
- Better Sampling for scientific problems
What is this tutorial about?
1 Why Physics-Informed Machine Learning

How to represent a physical solution and why it generalizes for

- Solving Differential Equations and Optimal Control
- Better Sampling for scientific problems

with applications in

- Inverse Problem
- Quantum Many-body Problem
- Rare Event (Transition Path) Sampling
- Large Deviations
Table of Contents

2 Formulation for Physics-Informed Machine Learning

► Why Physics-Informed Machine Learning

► Formulation for Physics-Informed Machine Learning

► Differential Equation Solving
  Examples

► Theory Behind Physics-Informed Neural Network
  Advanced PINN

► Operator Learning
  System Identification

► Summary
Physics-Informed Machine Learning

physics-informed machine learning as a structured risk minimization problem

\[
\min_{f \in \mathcal{H}} \mathcal{L}(f, \mathcal{D}) + \Omega(f)
\]

(1)

- **Data** \( \mathcal{D} \): we could augment the dataset utilizing available physical prior like symmetry
- **Model** \( f \): we could embed physical prior into the model design
- **Regularization** \( \Omega \): regularization terms using given physical priors like differential equations
Formulation for Physics-Informed Machine Learning

2 Formulation for Physics-Informed Machine Learning

Physics-Informed Machine Learning

physics-informed machine learning as a structured risk minimization problem

$$\min_{f \in \mathcal{H}} \mathcal{L}(f, \mathcal{D}) + \underbrace{\Omega(f)}_{\text{physical prior}}$$  (1)

- **Data** $\mathcal{D}$: we could *augment the dataset utilizing available physical prior* like symmetry
- **Model** $f$: we could embed physical prior into the model design
- **Regularization** $\Omega$: regularization terms using given physical priors like differential equations

Tasks that we are interested in
- Solving physical equations (First principle modeling)
- Operator Learning
- System Identification/Scientific Discovery

---

Partial Differential Equations (PDEs)

2 Formulation for Physics-Informed Machine Learning

**Definition:** PDEs

A PDE is a relation of the following type, parameterized by $\lambda \in \mathbb{R}^m$:

$$F(x_1, \ldots, x_n, \ldots; u_{x_1}, \ldots, u_{x_n}; u_{x_1 x_1}, u_{x_1 x_2}, \ldots; \lambda) = 0$$

with suitable boundary conditions

$$B(u, \vec{x}) = 0, \quad \vec{x} \in \partial \Omega,$$

Solving a PDE $\rightarrow$ find a $u$ function satisfying the governing equation.

where

- $u = u(x_1, \ldots x_n)$ is a unknown function of $n$ variables, i.e., $u : \mathbb{R}^d \rightarrow \mathbb{R}$; ($\vec{u}$ can be a vector, i.e., $\vec{u} \in \mathbb{R}^d$, here, we assume it to be a scalar for simplicity)
- $u_{x_i} = \frac{\partial u}{\partial x_i}$, $u_{x_i x_j} = \frac{\partial^2 u}{\partial x_i \partial x_j}$, ....
- $f$ is the governing equation.
- $B$ is the boundary condition.
Governing equation: Linear vs. Non-linear

2 Formulation for Physics-Informed Machine Learning

Linear PDE

A PDE is linear if and only if $f$ is linear with respect to $u$ and all its derivatives.

$$ f(\vec{x}; u; u_{x_1}, \ldots, u_{x_n}; u_{x_1x_1}, u_{x_1x_2}; \lambda) = 0, \quad \vec{x} \in \Omega, $$ (3)
**Linear PDE**

A PDE is linear if and only if $f$ is linear with respect to $u$ and all its derivatives.

\[ f(\vec{x}; u; u_{x_1}, \ldots, u_{x_n}; u_{x_1x_1}, u_{x_1x_2}; \ldots; \lambda) = 0, \quad \vec{x} \in \Omega, \quad (3) \]

**Non-linear PDE**

- **Semilinear PDE** where $f$ is nonlinear only with respect to $u$ but is linear with respect to all its derivatives;
- **Quasi-linear PDE** where $f$ is linear with respect to the highest order derivatives of $u$;
- **Fully nonlinear PDE** where $f$ is nonlinear with respect to the highest order derivatives of $u$. 
Order of a PDE

The **highest order** of differentiation occurring in the equation is the order of the equation.

\[ f(\vec{x}; u, u_{x_1}, \ldots, u_{x_n}; u_{x_1x_1}, u_{x_1x_2}; \lambda) = 0, \quad \vec{x} \in \Omega, \]  

(4)
### Order of a PDE

The **highest order** of differentiation occurring in the equation is the order of the equation.

\[ f(\vec{x}; u; u_{x_1}, ..., u_{x_n}; u_{x_1x_1}, u_{x_1x_2}, ..., \lambda) = 0, \quad \vec{x} \in \Omega, \]  

(4)

### Second order PDEs

Most commonly used in engineering applications.

\[ a u_{xx} + 2bu_{xy} + cu_{yy} + du_x + eu_y + hu = f \]

where \( a, ..., f \) are smooth (e.g. \( C^2 \)) functions of \( x, y \).

- **Elliptic**: \( b^2 - ac < 0 \), e.g., Laplace equation \( u_{xx} + u_{yy} = 0 \)
- **Parabolic**: \( b^2 - ac = 0 \), e.g., Diffusion equation \( u_t - Du_{xx} = 0 \)
- **Hyperbolic**: \( b^2 - ac > 0 \), e.g., Wave equation \( u_{tt} - c^2 u_{xx} = 0 \)
Boundary conditions: Types of boundary value problems

2 Formulation for Physics-Informed Machine Learning

- Dirichlet: specifies the boundary value of $u$: $u|_{\partial \Omega} = f$
- Neumann: specifies the value of the normal derivative of the $u$: $u_x|_{\partial \Omega} = f$
- Robin: $c_0 u|_{\partial \Omega} + c_1 u_x|_{\partial \Omega} = f$
- Cauchy: Dirichlet and Neumann, i.e, $u|_{\partial \Omega} = f$ and $u_x|_{\partial \Omega} = f$
- Mixed: different location ($x$) have different boundary condition.

Figure: Boundary value problem

https://en.wikipedia.org/wiki/Boundary_value_problem

Note 1: https://en.wikipedia.org/wiki/Boundary_value_problem
Forward problem
2 Formulation for Physics-Informed Machine Learning

**Forward problem: Given a fixed $\lambda$, solve for $u(x)$**

Consider the following PDE parameterized by $\lambda \in \mathbb{R}^m$:

$$F(x_1, \ldots, x_n, \ldots; u_{x_1}, \ldots, u_{x_n}; u_{x_1x_1}, u_{x_1x_2}, \ldots; \lambda) = 0$$

Traditionally, solved with finite difference method (FDM), finite element method (FEM), etc.
Forward problem: Given a fixed $\lambda$, solve for $u(x)$

Consider the following PDE parameterized by $\lambda \in \mathbb{R}^m$:

$$F(x_1, \ldots, x_n, \ldots; u_{x_1}, \ldots, u_{x_n}; u_{x_1x_1}, u_{x_1x_2}, \ldots; \lambda) = 0$$

Traditionally, solved with finite difference method (FDM), finite element method (FEM), etc.

**Advantages:** Accurate, Reliable

**Challenges:** Expensive and time consuming, Hard to incorporate in downstream applications
Forward problem

Given a fixed $\lambda$, solve for $u(x)$

Consider the following PDE parameterized by $\lambda \in \mathbb{R}^m$:

$$F(x_1, \ldots, x_n, \ldots; u_{x_1}, \ldots, u_{x_n}, u_{x_1x_1}, u_{x_1x_2}, \ldots; \lambda) = 0$$

Traditionally, solved with finite difference method (FDM), finite element method (FEM), etc.

Advantages: Accurate, Reliable

Challenges: Expensive and time consuming, Hard to incorporate in downstream applications

Learning is suitable for:

- Surrogate modeling: find a cheap model to surrogate the PDE governed system, i.e., $u = f_\theta(x; \lambda)$.
- Incorporating physical knowledge (PDE) information for downstream tasks.
Inverse problems
2 Formulation for Physics-Informed Machine Learning

Inverse problem: Given a set of observed $u(x)$, find $\lambda$

Consider the following PDE parameterized by $\lambda \in \mathbb{R}^m$:

$$F(x_1, \ldots, x_n, \ldots; u_{x_1}, \ldots, u_{x_n}; u_{x_1 x_1}, u_{x_1 x_2}, \ldots; \lambda) = 0$$

- Identifying unknown parameters in PDEs/boundary/initial conditions
- Data driven (with partial physics knowledge) spatio-temporal modeling

Traditionally, formulated as a PDE constraint optimization problem, and solved with adjoint method.
Inverse problems
2 Formulation for Physics-Informed Machine Learning

Inverse problem: Given a set of observed \( u(x) \), find \( \lambda \)

Consider the following PDE parameterized by \( \lambda \in \mathbb{R}^m \):

\[
F(x_1, \ldots, x_n, \ldots; u_{x_1}, \ldots, u_{x_n}; u_{x_1x_1}, u_{x_1x_2}, \ldots; \lambda) = 0
\]

- Identifying unknown parameters in PDEs/boundary/initial conditions
- Data driven (with partail physics knowledge) spatio-temporal modeling

Traditionally, formulated as a PDE constraint optimization problem, and solved with adjoint method.

Learning is suitable for:

- Incorporating PDE information in inverse problems
- Surrogate modeling 1: find a cheap model to surrogate (inversion of) the PDE governed system, i.e., \( u = f_\theta(x; \lambda) \) (or \( \lambda = f_{\theta'}(u) \)).
Table of Contents

3 Differential Equation Solving

► Why Physics-Informed Machine Learning

► Formulation for Physics-Informed Machine Learning

► Differential Equation Solving
  Examples

► Theory Behind Physics-Informed Neural Network
  Advanced PINN

► Operator Learning
  System Identification

► Summary
Most of physics can be formulated as $A(u) = f$ where $A$ is a differential operator.
Physics-Informed Neural Network

3 Differential Equation Solving

Most of physics can be formulated as $A(u) = f$ where $A$ is a differential operator.

**PINN solving $A(u) = f$**

Suppose we observe $(x_i, f(x_i))_{i=1}^n$, can we solve $A(u) = f$?
Most of physics can be formulated as \( A(u) = f \) where \( A \) is a differential operator. 

**PINN solving \( A(u) = f \)**

Suppose we observe \((x_i, f(x_i))_{i=1}^n\), can we solve \( A(u) = f \)? PINN minimizes the equation residual on observed data points, *i.e.*

\[
   u^* = \arg\min_{u \in \mathcal{H}} \sum_{i=1}^{n} \| A(u)(x_i) - f(x_i) \|^2
\]
The core idea behind PINN is transforming solving $A(u) = f$ to a minimization problem $\min ||A(u) - f||$. New transformation can bring new methods!
The core idea behind PINN is transforming solving $A(u) = f$ to a minimization problem
\[ \min \| A(u) - f \|. \] New transformation can bring new methods!

**Deep Ritz Methods** Using Variational form, i.e. $Ax = b \iff \min x^\top Ax - 2bx$

Not all PDEs admit a variational form.

Yu B. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Communications in Mathematics and Statistics, 2018
Methods Beyond PINN

3 Differential Equation Solving

The core idea behind PINN is transforming solving $A(u) = f$ to a minimization problem $\min ||A(u) - f||$. New transformation can bring new methods!

**Deep Ritz Methods** Using Variational form, i.e. $Ax = b \iff \min x^T Ax - 2b x$

Not all PDEs admit a variational form.

Yu B. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Communications in Mathematics and Statistics, 2018

**Weak Adversarial Network** Solving equation $A(u) = f$ equalvalent to

$$\min_u \max \frac{\langle v, A(u) - f \rangle}{\|v\| \leq 1}$$

and change the constraint to a log penalization.

The core idea behind PINN is transforming solving $A(u) = f$ to a minimization problem $\min ||A(u) - f||$. New transformation can bring new methods!

**Deep Ritz Methods** Using Variational form, i.e. $Ax = b \iff \min x^T Ax - 2bx$

Not all PDEs admit a variational form.

Yu B. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. Communications in Mathematics and Statistics, 2018

**Weak Adversarial Network** Solving equation $A(u) = f$ equalvalent to

$$\min_u \max_{\|v\| \leq 1} \langle v, A(u) - f \rangle$$

and change the constraint to a log penalization.


**Adversarial training** $L^2$ loss is not strong enough (for regualrity of PDE structure), we should use $L^\infty$ loss for some PDEs.

A neural network is not the only ansatz (a high bias list...)

- **Gaussian Process**


A neural network is not the only ansatz (a high bias list...)

- **Gaussian Process**
  


- **Diffusion map**


A neural network is not the only ansatz (a high bias list...)

- **Gaussian Process**
  


- **Diffusion map**
  


- **Tensor Network**
  

  Richter L, Sallandt L, Nüsken N. Solving high-dimensional parabolic PDEs using the tensor train format International Conference on Machine Learning, 2021.

So far, we have only built loss functions. How should we sample data?

\[ I(u) = \int_{\Omega} \mathcal{L}(x, u) d\nu(x), \]

- Simplest case \( d\nu(x) = dx \)
- Challenging case \( d\nu(x) = e^{-U(x)} dx \)
Adaptive Importance Sampling

3 Differential Equation Solving

On-the-fly variance reduction via adaptive importance sampling

\[ \mathcal{L}(x,f) = \frac{1}{2} |\nabla f(x)|^2, \quad d\nu(x) = e^{-\beta V(x)} dx \quad (\beta > 0) \]

- Idea 1: Window sampling with instantaneous solution \( f \) with a bias
- Idea 2: Reweight the expectation with these importance sampled points

\[ W_l(x) \geq 0 \text{ with } l = 1, \ldots, L \text{ such that } \forall x \in \Omega : \sum_{l=1}^{L} W_l(x) = 1, \]

\[ \mathbb{E}_\nu \phi = \sum_{l=1}^{L} \int_{\mathbb{R}^d} \phi(x) W_l(x) d\nu(x) = \sum_{l=1}^{L} w_l \mathbb{E}_l \phi \]

GM Rotskoff, AR Mitchell, E Vanden-Eijnden Mathematical and Scientific Machine Learning, 757-780
For any $\phi$, define

$$E_l \phi = Z_l^{-1} \int_{\mathbb{R}^d} \phi(x) W_l(x) d\nu(x) \quad \text{where} \quad Z_l = \int_{\mathbb{R}^d} W_l(x) d\nu(x)$$

(5)

as well as the weights

$$w_l = E_\nu W_l.$$  

(6)

By choosing $\phi(x) = W_{l'}(x)$ in this expression, we deduce that the weights satisfy the eigenvalue problem (Thiede et al 2016)

$$w_{l'} = \sum_{l=1}^L w_l p_{l'l'}, \quad l' = 1, \ldots, L, \quad \text{subject to} \quad \sum_{l=1}^L w_l = 1,$$

(7)

where we defined

$$p_{l'l'} = \langle W_{l'} \rangle_1.$$  

(8)
Sample $Z_l^{-1} W_l(x) d\nu(x)$ with MCMC biased by $-\log W_l(x)$ so that

$$\mathbb{E}_l \phi \approx \frac{1}{n} \sum_{i=1}^{n} \phi(x_{i,l}), \quad x_{i,l} \sim Z_l^{-1} W_l(x) d\nu(x)$$

This allows us to estimate $\mathbb{E}_l \phi$ in (7) as well as $p_{ll'}$ in (8) — can solve eigenvalue problem! the weights $w_l$, and finally estimate $\mathbb{E}_{\nu} \phi$. 
Adaptive Importance Sampling

3 Differential Equation Solving
Important Examples of high-dimensional problems

• **Optimal Control**: Hamilton-Jacobi Equation
  – Dimension: State Space Dimension
Important Examples of high-dimensional problems

- **Optimal Control**: Hamilton-Jacobi Equation
  - Dimension: State Space Dimension

- **Metastability**: Backward Kolmogorov / Feynman-Kac
  - Dimension: State Space Dimension
Important Examples of high-dimensional problems

- **Optimal Control**: Hamilton-Jacobi Equation
  - Dimension: State Space Dimension

- **Metastability**: Backward Kolmogorov / Feynman-Kac
  - Dimension: State Space Dimension

- **Nonequilibrium Dynamics**: Compute large deviation function
  - Dimension: State Space Dimension
Examples of High-Dimensional PDEs

3 Differential Equation Solving

Important Examples of high-dimensional problems

- **Optimal Control**: Hamilton-Jacobi Equation
  - Dimension: State Space Dimension

- **Metastability**: Backward Kolmogorov / Feynman-Kac
  - Dimension: State Space Dimension

- **Nonequilibrium Dynamics**: Compute large deviation function
  - Dimension: State Space Dimension

- **Quantum Many-body problem**: Eigenvalue Problems
  - dimension $\propto$ number of particles
Dynamics driven by an SDE,

\[ dX_t = -\nabla V(X_t) dt + \sqrt{2\beta^{-1}} dW_t. \]

Various quantities satisfy Backward Kolmogorov Equation, including the committor probability:

\[ q(x) := \mathbb{P}_x(t_B < t_A) \]

where \( t_A = \inf \{ t : x(t) \in A \} \) and \( t_B \) is defined analogously. GM Rotskoff, AR Mitchell, E Vanden-Eijnden. Active importance sampling for variational objectives dominated by rare events: Consequences for optimization and generalization. Mathematical and Scientific Machine Learning, 757-780, 2022.
We want to solve the PDE,

\[
\begin{cases}
(Lq)(x) = 0 & \text{for } x \notin A \cup B \\
q(x) = 0 & \text{for } x \in A \\
q(x) = 1 & \text{for } x \in B.
\end{cases}
\]

where \(-L\) is the infinitesimal generator of the process defined by (??):

\[Lq = \nabla V \cdot \nabla q - \beta^{-1} \Delta q.\]
We want to solve the PDE,
\[
\begin{cases}
(Lq)(x) = 0 & \text{for } x \notin A \cup B \\
q(x) = 0 & \text{for } x \in A \\
q(x) = 1 & \text{for } x \in B.
\end{cases}
\]

where \(-L\) is the infinitesimal generator of the process defined by (??):
\[
Lq = \nabla V \cdot \nabla q - \beta^{-1} \Delta q.
\]

Variational formulation:
\[
C(q) = \int_{\mathbb{R}^d} |\nabla q(x)|^2 d\nu(x) \quad \text{with} \quad d\nu(x) = Z^{-1} e^{-\beta V(x)} \, dx
\]
Example: low-dimensional metastable system

3 Differential Equation Solving
Example: 66-dimensional molecular systems

3 Differential Equation Solving
Molecular dynamics becomes Markovian on a lag-time $\tau$.

\[
\mathcal{T}_{A \cup B}^{\tau} [\phi](x) = \mathbb{E}_x \phi(X_{\tau^*}) \quad \text{where} \quad \tau^* = \min(\tau, T)
\]

\[
q(x) = 0 \quad x \in A
\]

\[
q(x) = 1 \quad x \in B
\]

Then, the committor satisfies:

\[
(\mathcal{T}_{A \cup B}^{\tau} [q] - \text{Id})(x) = 0 \quad x \in (A \cup B)^c
\]

FIG. 7. Illustration of some key properties of the Holton Mass model relevant to the prediction problems considered here. Red and yellow ellipses approximately mark the projections of states A and B, respectively, on the collective variables. The background color shows the average time to hit state B, clipped to a maximum of 1300 days to show detail. Black contours show the negative logarithm of the stationary density marginalized on these collective variables. Three transition paths harvested from a long simulation are shown in white.

The two states of interest in this model are a strong polar vortex, with large positive $U(z,t)$ (meaning eastward wind, marked as state A in Figure 7), and a weak polar vortex, with a weak wind profile in which $U(z,t)$ sometimes dips negative (marked as state B in Figure 7). Specifically, we define A and B as spheres centered on the model's two stable fixed points $(a, U_a)$ and $(b, U_b)$ in the 75-dimensional state space. The two spheres have radii of 8 and 20 respectively, with distances measured in the non-dimensionalized state space specified in [34]. In physical units, these correspond to the ellipsoids $A = \ldots$ and $B = \ldots$.

The two states of interest in this model are a strong polar vortex, with large positive $U(z,t)$ (meaning eastward wind, marked as state A in Figure 7), and a weak polar vortex, with a weak wind profile in which $U(z,t)$ sometimes dips negative (marked as state B in Figure 7). Specifically, we define A and B as spheres centered on the model's two stable fixed points $(a, U_a)$ and $(b, U_b)$ in the 75-dimensional state space. The two spheres have radii of 8 and 20 respectively, with distances measured in the non-dimensionalized state space specified in [34]. In physical units, these correspond to the ellipsoids $A = \ldots$ and $B = \ldots$.

To generate an initial data set, we sampled 30,000 points uniformly in $U(30 \text{ km})$ and $|\Psi|(30 \text{ km})$ from a long (50,000 days) trajectory and ran two ten-day trajectories from each starting point. Simulation details are reported in [34]. We simulated with a time step of 0.005 days, and saved the state of the system every 0.1 days. To validate our neural network results, we use a long trajectory of 500,000 days to compute $h_q(s) = E[B(X(\tau)) | u^{\phi_\ast}(X(0))]^{2} [s, s+ s]$ for $s \in [0, 1]$. (29)

Aim to calculate the large deviation rate function for an observable, giving information about the asymptotic probability distribution.

Large deviations on path measures

\[ P(A_T \in [a, a + da]) \approx e^{-TI(a)} \]

where \( P \) is the path measure associated with

\[ dX_t = b(X_t)dt + \sigma(X_t)dW_t \]

and

\[ A_T = \int_0^T f(X_t)dt + \int_0^T g(X_t) \circ dX_t \]

Control problem hidden in a rare events problem

\[ dX^u_t = u_t(X^u_t) \, dt + \sigma \, dW_t, \quad (9) \]

and then the SCGF can be estimated simply by reweighting the average

\[ \psi(\lambda) = \lim_{T \to \infty} \frac{1}{T} \log E_{X^u} \left( e^{\lambda T A_T} \frac{dP[X^u]}{dP_u[X^u]} \right). \quad (10) \]

\[ \psi(\lambda) = \lim_{T \to \infty} \frac{1}{T} \log E[e^{\lambda T A_T}] = \sup_u \lim_{T \to \infty} \left\{ \lambda E_u[A_T] - \frac{1}{T} D_{KL}[dP_u||dP] \right\}. \quad (11) \]
Current in the asymmetric exclusion process; comparison with tensor network approach.
Active nonequilibrium matter, quantifying entropy production fluctuations.

\[ \psi(\lambda) \times 10^4 \]

\( \lambda = -10.0 \)

\( \lambda = 0.00 \)

\( \lambda = 10.0 \)
Quantum Monte Carlo aims to calculate the wave function (eigenfunction). Monte Carlo here means handle the multi-dimensional integrals that arise in the different formulations of the many-body problem.
Quantum Monte Carlo aims to calculate the wave function (eigenfunction). Monte Carlo here means handle the multi-dimensional integrals that arise in the different formulations of the many-body problem.

**Variational Monte Carlo**

We can recast the problem that finding the eigenvalue of $\mathcal{H}$ as

\[
\min_{\theta} \mathcal{L}(\theta) = \frac{\langle \Psi_\theta, \mathcal{H} \Psi_\theta \rangle}{\langle \Psi_\theta, \Psi_\theta \rangle} = \frac{\int_{x \in \mathcal{X}} \Psi_\theta^*(x) \cdot (\mathcal{H} \Psi_\theta)(x) dx}{\int_{x \in \mathcal{X}} \Psi_\theta^*(x) \cdot \Psi_\theta(x) dx}
\]
Quantum Monte Carlo aims to calculate the wave function (eigenfunction). Monte Carlo here means handle the multi-dimensional integrals that arise in the different formulations of the many-body problem.

**Variational Monte Carlo**

We can recast the problem that finding the eigenvalue of $\mathcal{H}$ as

$$
\min_{\theta} \mathcal{L}(\theta) = \frac{\langle \Psi_\theta, \mathcal{H} \Psi_\theta \rangle}{\langle \Psi_\theta, \Psi_\theta \rangle} = \frac{\int_{x \in \mathcal{X}} \Psi_\theta^*(x) \cdot (\mathcal{H} \Psi_\theta)(x) \, dx}{\int_{x \in \mathcal{X}} \Psi_\theta^*(x) \cdot \Psi_\theta(x) \, dx}
$$

$$
= \int_{x \in \mathcal{X}} \frac{\Psi_\theta^*(x) \cdot \Psi_\theta(x)}{\int_{x \in \mathcal{X}} \Psi_\theta^*(x) \cdot \Psi_\theta(x) \, dx} \cdot \frac{\mathcal{H} \Psi_\theta(x)}{\Psi_\theta(x)} \, dx = \mathbb{E}_{\pi_\theta(x)} E_\theta(x) \tag{12}
$$
Variational Monte Carlo

- **Fisher-Rao Gradient**

\[
\nabla_\theta \mathcal{L}_\theta = 2E_{\pi_\theta}(x) \left[ (E_\theta(x) - E_{\pi_\theta}(x) E_\theta(x)) \nabla_\theta \log |\Psi_\theta| \right]
\]

\[\text{deviation of local energy} \quad \text{score}\]

Variational Monte Carlo

3 Differential Equation Solving

Variational Monte Carlo

- **Fisher-Rao Gradient**

\[
\nabla_\theta \mathcal{L}_\theta = 2 \mathbb{E}_{\pi_\theta(x)} \left[ (E_\theta(x) - \mathbb{E}_{\pi_\theta(x)} E_\theta(x)) \nabla_\theta \log |\Psi_\theta| \right]
\]

\[\text{deviation of local energy}\]
\[\text{score}\]


- **Wasserstein Gradient**

\[
\nabla_\theta \mathcal{L}_\theta = \mathbb{E}_\theta \nabla_\theta \left[ -2 \nabla_x E_{\text{loc}}(x^{(i)}) , \nabla_x \log \pi(x^{(i)}, \theta) \right]
\]

\[\text{gradient of local energy}\]

Why VMC is hard?

3 Differential Equation Solving

- Design of anti-symmetric Neural Network $\Psi(x_{\sigma(1)}, \cdots, x_{\sigma(n)}) = \operatorname{sgn}(\sigma)\Psi(x_1, \cdots, x_n)$
  (Slater) Determinant is slow and exists an exponential approximation lower bound.


Why VMC is hard?

3 Differential Equation Solving

- Design of ani-symmetric Neural Network \( \Psi(x_{\sigma(1)}, \cdots, x_{\sigma(n)}) = \text{sgn}(\sigma)\Psi(x_1, \cdots, x_n) \)

(Slater) Determinant is slow and exists an exponential approximation lower bound.


- The calculation of \( \Delta \) is slow in high dimension

\[
\mathcal{H} := -\frac{1}{2} \sum_i \Delta_i + \sum_{i>j} \frac{1}{|r_i - r_j|} - \sum_{i \neq I} \frac{Z_I}{|r_i - R_I|} + \sum_{I \neq J} \frac{Z_I Z_J}{|R_I - R_J|}
\]  

(13)

Why VMC is hard?

3 Differential Equation Solving

- Design of ani-symmetric Neural Network $\Psi(x_{\sigma(1)}, \cdots, x_{\sigma(n)}) = \text{sgn}(\sigma)\Psi(x_{1}, \cdots, x_{n})$ (Slater) Determinant is slow and exists an exponential approximation lower bound.


- The calculation of $\Delta$ is slow in high dimension

  $$\mathcal{H} := -\frac{1}{2} \sum_i \Delta_i + \sum_{i>j} \frac{1}{|r_i - r_j|} - \sum_{iI} \frac{Z_I}{|r_i - R_I|} + \sum_{I>J} \frac{Z_I Z_J}{|R_I - R_J|}$$  


- Non-convex landscape and overfitting

PINN-like idea can be used beyond physics. Generally speaking, you can always fusion modeling with learning with PINN, for example

- **Auction Design**


PINN-like idea can be used beyond physics. Generally speaking, you can always fusion modeling with learning with PINN, for example

- **Auction Design**
  


- **Neural Rendering**
  

Table of Contents
4 Theory Behind Physics-Informed Neural Network

► Why Physics-Informed Machine Learning
► Formulation for Physics-Informed Machine Learning
► Differential Equation Solving Examples
► Theory Behind Physics-Informed Neural Network Advanced PINN
► Operator Learning System Identification
► Summary
What is the optimal sample complexity for learning with prior information $A(u) = f$?
What is the **optimal sample complexity** for learning with prior information $A(u) = f$? Is PINN Optimal?
What is the **optimal sample complexity** for learning with prior information $A(u) = f$? Is PINN Optimal? Are All Losses Created Equal?
What is the **optimal sample complexity** for learning with prior information $A(u) = f$?

Is PINN Optimal? Are All Losses Created Equal?

**Recast Solving PDE as a Statistical Problem**

Example: Solving $\Delta u = f$
Statistics of Physics-Informed Neural Network

4 Theory Behind Physics-Informed Neural Network

What is the **optimal sample complexity** for learning with prior information $A(u) = f$?

Is PINN Optimal? Are All Losses Created Equal?

**Recast Solving PDE as a Statistical Problem**

Example: Solving $\Delta u = f$

- **Hypothesis Space**: the solution $u$ in (Sobolev, Besov, Barron space...)

What is the optimal sample complexity for learning with prior information $A(u) = f$?

Is PINN Optimal? Are All Losses Created Equal?

**Recast Solving PDE as a Statistical Problem**

**Example: Solving** $\Delta u = f$

- **Hypothesis Space**: the solution $u$ in (Sobolev, Besov, Barron space...)
- **Observation Data**:

$$ (u(x_i), f(x_i) = \Delta u(x_i) + \text{noise})_{i=1}^n $$
What is the **optimal sample complexity** for learning with prior information $A(u) = f$?

Is PINN Optimal? Are All Losses Created Equal?

**Recast Solving PDE as a Statistical Problem**

**Example:** Solving $\Delta u = f$

- **Hypothesis Space:** the solution $u$ in (Sobolev, Besov, Barron space...)
- **Observation Data:**
  
  $$(u(x_i), f(x_i) = \Delta u(x_i) + \text{noise})_{i=1}^n$$

Now we recast a solving PDE problem as a non-parametric estimation problem, so that we can

- Using Fano, ... methods to know the **lower bound**
- Using empirical process, ... methods to build the **upper bound**

What is the **optimal sample complexity** for learning with prior information $A(u) = f$?

Is PINN Optimal? Are All Losses Created Equal?

### Information Theoretical Lower Bound

If $A$ is a $t$–th order linear differential operator, then any Estimator $H$ using $(X_i, f_i)_{i=1}^{n}$ can’t do better than

\[
\inf_H \sup_{u \in \mathcal{C}^\alpha(\Omega)} \mathbb{E} \|H(\{X_i, f_i\}_{i=1, \ldots, n}) - u^*\|_{W^s_2} \gtrsim n^{-\frac{2\alpha-2s}{2\alpha-2t+d}},
\]

- Solving a PDE equal to reconstructing a function with gradient information

**inf** means best estimator and **sup** means the hardest problem.
What is the **optimal sample complexity** for learning with prior information $A(u) = f$?

Is PINN Optimal? Are All Losses Created Equal?

### Information Theoretical Lower Bound

If $A$ is a $t$–th order linear differential operator, then any Estimator $H$ using $(X_i, f_i)_{i=1}^n$ can’t do better than

$$\inf_H \sup_{u \in C^\alpha(\Omega)} \mathbb{E}\|H(\{X_i, f_i\}_{i=1,\ldots,n}) - u^*\|_{W^s_2} \gtrsim n^{-\frac{2\alpha-2s}{2\alpha-2t+d}},$$

- Solving a PDE equal to reconstructing a function with gradient information
- $\inf$ means best estimator and $\sup$ means the hardest problem

**Take Home Message** PINN is Optimal!
What is the **optimal sample complexity** for learning with prior information $A(u) = f$? Is PINN Optimal? Are All Losses Created Equal?

### Information Theoretical Lower Bound

If $A$ is a $t-$th order linear differential operator, then any Estimator $H$ using $(X_i, f_i)_{i=1}^n$ can’t do better than

$$\inf_H \sup_{u \in C^\alpha(\Omega)} \mathbb{E}\|H(\{X_i, f_i\}_{i=1}^n) - u^*\|_{W^s_2} \gtrsim n^{-\frac{2\alpha - 2s}{2\alpha - 2t + d}}.$$  

- Solving a PDE equal to reconstructing a function with gradient information
- $\inf$ means best estimator and $\sup$ means the hardest problem

**Take Home Message** PINN is Optimal! Not every consistent loss function is optimal! We need case by case studying!

Why Deep Ritz Method is sub-optimal?
Why Deep Ritz Method is sub-optimal? Solving a simple PDE $\Delta u = f$ using Fourier Basis. Using Deep Ritz Methods, the objective function is

$$\min \int \frac{1}{2} \| \nabla f(x) \|^2 - u(x)f(x) \, dx$$
A Fourier Basis View
4 Theory Behind Physics-Informed Neural Network

Why Deep Ritz Method is sub-optimal? Solving a simple PDE $\Delta u = f$ using Fourier Basis. Using Deep Ritz Methods, the objective function is

$$\min \int \frac{1}{2} \| \nabla f(x) \|^2 - u(x)f(x) \, dx$$

- **Estimator 1**: First learn $f$, the god solves the equation computationally intractable

Estimator 1

First Estimate $f$ then solve $u, f_z = \frac{1}{n} \sum f(x_i) \phi_z(x_i)$, then $u = \sum \frac{1}{\|z\|^2} f_z \phi_z(x)$
Why Deep Ritz Method is sub-optimal? Solving a simple PDE $\Delta u = f$ using Fourier Basis. Using Deep Ritz Methods, the objective function is

$$\min \int \frac{1}{2} \| \nabla f(x) \|^2 - u(x)f(x) \, dx$$

- **Estimator 1**: First learn $f$, the god solves the equation computationally intractable

**Estimator 1**

First Estimate $f$ then solve $u$, $f_z = \frac{1}{n} \sum f(x_i) \phi_z(x_i)$, then $u = \sum \frac{1}{\|z\|^2} f_z \phi_z(x)$

- **Estimator 2**: Deep ritz methods

**Estimator 2**

Plug $u = \sum u_z \phi_z(x)$ into the Deep Ritz Objective function

$$\frac{1}{n} \sum_{i=1}^{n} \left( \sum_z u_z \nabla \phi_z(x_i) \right)^2 + \sum_z u_z \phi_z(x_i) f(x_i)$$
A Fourier Basis View
4 Theory Behind Physics-Informed Neural Network

- **Estimator 1**: The Fourier coefficient of the solution of Estimator 1 is

\[
\mathbf{u}_{1,z} = \text{diag} \left( \| z \| \frac{2}{2} \right)^{-1} \left\{ \| z \|_\infty \leq Z \mathbf{f}_z \right\}
\]

- **Estimator 2**: The Fourier coefficient of the solution of Estimator 2 is

\[
\mathbf{u}_{2,z} = \left( \frac{1}{n} \sum_{i=1}^{n} \nabla \phi_i(x_i) \nabla \phi_j(x_i) \right)^{-1} \left\{ \| i \|_\infty \leq Z, \| j \|_\infty \leq Z \mathbf{f}_z \right\}
\]
A Fourier Basis View
4 Theory Behind Physics-Informed Neural Network

- **Estimator 1:** The Fourier coefficient of the solution of Estimator 1 is

\[
\mathbf{u}_{1,z} = \text{diag} \left( \left\| \mathbf{Z} \right\|_2^2 \right)^{-1} \left\| \mathbf{Z} \right\|_\infty \leq Z f_z. \tag{14}
\]

- **Estimator 2:** The Fourier coefficient of the solution of Estimator 2 is

\[
\mathbf{u}_{2,z} = \left( \frac{1}{n} \sum_{i=1}^{n} \nabla \phi_i(x_i) \nabla \phi_j(x_i) \right)^{-1} f_z, \quad ||i||_\infty \leq Z, ||j||_\infty \leq Z. \tag{15}
\]

Suboptimality of Deep Ritz Methods
introduce a new variance \( \text{Var}(\| \nabla u(x) \|^2 - \Delta u(x) u(x)) \), but neglectable in high-dimension

Let's consider the simplest PDE $\Delta u = f$. If $f$ can be represented by a NN, can $u$ be represented by a NN?
Let’s consider the simplest PDE $\Delta u = f$. If $f$ can be represented by a NN, can $u$ be represented by a NN?

The answer is **YES**. This helps us to understand the implicit bias of NN to solve PDEs.

- Parametric Complexity Bounds for Approximating PDEs with Neural Networks Tanya Marwah, Zachary C. Lipton, Andrej Risteski Neural Information Processing Systems (NeurIPS), 2021
- Neural Network approximations of PDEs Beyond Linearity: A Representational Perspective Tanya Marwah, Zachary C. Lipton, Jianfeng Lu, Andrej Risteski International Conference on Machine Learning (ICML), 2023
Let’s consider the simplest PDE $\Delta u = f$. If $f$ can be represented by a NN, can $u$ be represented by a NN?

The answer is **YES**. This helps us to understand the implicit bias of NN to solve PDEs.

- Parametric Complexity Bounds for Approximating PDEs with Neural Networks. Tanya Marwah, Zachary C. Lipton, Andrej Risteski Neural Information Processing Systems (NeurIPS), 2021
- Neural Network approximations of PDEs Beyond Linearity: A Representational Perspective. Tanya Marwah, Zachary C. Lipton, Jianfeng Lu, Andrej Risteski International Conference on Machine Learning (ICML), 2023

**Reason:** Neural Network can perform (preconditioned) gradient flow.

- Similar to the recent line of that transformer can perform gradient descent for in-context learning.
- Precondition is essential for infinite-dimensional due to infinite condition number!
Will different loss function affects optimization speed?

1). **Physics-Informed** \( \int (\Delta u(x) - f(x))^2 dx \)

2). **Deep Ritz** \( \int \|\Delta u(x)\|^2 - 2u(x)f(x) dx \)
Will different loss function affects optimization speed?

1). **Physics-Informed** $\int (\Delta u(x) - f(x))^2 dx$

2). **Deep Ritz** $\int \|\Delta u(x)\|^2 - 2u(x)f(x) dx$

**Traditional Thoughts** 1) is much harder, for it involves condition number of $\Delta^2$ while 2) only involves $\Delta$
Optimization of Physics-Informed Neural Network

4 Theory Behind Physics-Informed Neural Network

Will different loss function affect optimization speed?

1). Physics-Informed \( \int (\Delta u(x) - f(x))^2 dx \)  

2). Deep Ritz \( \int \| \Delta u(x) \|^2 - 2u(x)f(x) dx \)

Traditional Thoughts 1) is much harder, for it involves condition number of \( \Delta^2 \) while 2) only involves \( \Delta \)

Machine Learning is a Kernelized gradient flow. Physics equations can precondition machine learning!

Optimization of Physics-Informed Neural Network

4 Theory Behind Physics-Informed Neural Network

Will different loss function affects optimization speed?

1). Physics-Informed \( \int (\Delta u(x) - f(x))^2 \, dx \)

2). Deep Ritz \( \int \| \Delta u(x) \|^2 - 2u(x)f(x) \, dx \)

Traditional Thoughts 1) is much harder, for it involves condition number of \( \Delta^2 \) while 2) only involves \( \Delta \)

Machine Learning is a Kernelized gradient flow. Physics equations can precondition machine learning!


Using Sobolev norm \( \int \| \nabla^k (\Delta u(x) - f(x)) \|^2 \, dx \) as loss function can further accelerates training accelerates optimization

- Sobolev training for physics-informed neural networks, with J. W. Jang, W. J. Han, and H. J. Hwang, 2023
Sobolev Training vs L2 training
4 Theory Behind Physics-Informed Neural Network

Sobolev Training vs L2 training for function fitting.
Sobolev Training vs L2 training for solving heat equation.
Computing and even back prop $\Delta u = u_{x_1x_1} + \cdots + u_{x_dx_d}$ is hard when $d$ is high!

$\underbrace{d \text{ times computation}}$
Computation of PINN in High Dimension

4 Theory Behind Physics-Informed Neural Network

Computing and even back prop $\Delta u = u_{x_1x_1} + \cdots + u_{x_dx_d}$ is hard when $d$ is high!

\[ \Delta u \text{ is hard when } d \text{ is high!} \]

$d$ times computation

**Idea 1:** Stein's Lemma: $u = \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma^2 I)} f(x + \delta)$, then $\nabla_x u = \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma^2 I)} [\frac{\delta}{\sigma^2} f(x + \delta)]$

- Relates to Feyman-Kac
- Finite Difference with random direction!

Computation of PINN in High Dimension

4 Theory Behind Physics-Informed Neural Network

Computing and even back prop $\Delta u = u_{x_1x_1} + \cdots + u_{x_dx_d}$ is hard when $d$ is high!

$d$ times computation

**Idea 1:** Stein's Lemma: $u = \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma^2 I)} f(x + \delta)$, then $\nabla_x u = \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma^2 I)} \left[ \frac{\delta}{\sigma^2} f(x + \delta) \right]$

- Relates to Feynman-Kac
- Finite Difference with random direction!


**Idea 2:** Sketching: random select dimension to descent

$$\Delta u(x) = \mathbb{E}_i \frac{d^2}{dx_i^2} u(x)$$

Consider solving equation \( \frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0 \) whose solution is \( u(x, t) = u(x - \beta t, 0) \) using PINN.

**Propagation Failure:** some collocation points start converging to trivial solutions before the correct solution from initial/boundary points is able to reach them.
Consider solving equation $\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0$ whose solution is $u(x, t) = u(x - \beta t, 0)$ using PINN.

**Propagation Failure:** some collocation points start converging to trivial solutions before the correct solution from initial/boundary points is able to reach them


5 Operator Learning

- Why Physics-Informed Machine Learning
- Formulation for Physics-Informed Machine Learning
- Differential Equation Solving
  - Examples
- Theory Behind Physics-Informed Neural Network
  - Advanced PINN
- Operator Learning
  - System Identification
- Summary
Motivation of Operator Learning: Meta-PINN?

5 Operator Learning

Parametric PDE

We consider PDEs parametrized by coefficient $a(x)$:

$$-\nabla \cdot (a(x) \nabla u(x)) = f(x), \quad x \in D.$$
Motivation of Operator Learning: Meta-PINN?

We consider PDEs parametrized by coefficient $a(x)$:

$$-
abla \cdot (a(x) \nabla u(x)) = f(x), \; x \in D.$$ 

What if we have a dataset of $a(x)$
How can we build a resolution invariant machine learning system?

Idea: Directly learn the mapping between functions.


How can we build a resolution invariant machine learning system?

**Idea:** Directly learn the mapping between functions.


**Idea:** Directly learn the mapping between functions.

**Vocabulary:**
Man, woman, boy, girl, prince, princess, queen, king, monarch

**Discretized vector**

<table>
<thead>
<tr>
<th>Word</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>man</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>woman</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>boy</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>girl</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>prince</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>princess</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>queen</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>king</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>monarch</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Each word gets a 1x9 vector representation.
Neural Operators: Learning in the Function space

5 Operator Learning

Idea: Learning in the function space
Operator Learning: Framework

5 Operator Learning

Operator learning aims to build a parametric approximation $\mathcal{G}_\theta (\theta \in \mathbb{R}^p)$ to approximate a (non-linear) operator $\mathcal{G} : \mathcal{A} \rightarrow \mathcal{U}$

- Banach space $\mathcal{A} : \{a : D \rightarrow \mathbb{R}^{d_A}\}$ and $\mathcal{U} : \{a : D \rightarrow \mathbb{R}^{d_A}\}$ are all function space
- Idea1:

  ![Diagram showing process of encoding and decoding](image)

  - Linear Encoding from a function to $\mathbb{R}^{d_1}$ code
  - Transform a $\mathbb{R}^{d_1}$ code to a $\mathbb{R}^{d_2}$ code
  - Linear Decoding from $\mathbb{R}^{d_2}$ code to a function
We need to generalize operations in neural networks to function space
We need to generalize operations in neural networks to function space.

**Linear Transform**

- **Linear Encoding**: \( u \rightarrow \{ \int_x u(x)f_i(x)dx \}^{n}_{i=1} \)
We need to generalize operations in neural networks to function space.

### Linear Transform

- **Linear Encoding**: $u \rightarrow \{\int_x u(x)f_i(x)dx\}_{i=1}^n$
- a vector-input vector-output neural network
We need to generalize operations in neural networks to function space

**Linear Transform**

- **Linear Encoding**: \( u \rightarrow \{\int_x u(x)f_i(x)dx\}_{i=1}^n \)
- a vector-input vector-output neural network
- **Linear Decoding**: \( \{\beta_k\}_{k=1}^p \rightarrow \sum_{k=1}^p \beta_k \quad \tau_k \)

Universal approximation theorem of Chen & Chen (1995) states that DeepONets can approximate continuous operators
Operator Learning: Framework

5 Operator Learning

Operator learning aims to build a parametric approximation $G_\theta(\theta \in \mathbb{R}^p)$ to approximate a (non-linear) operator $G : \mathcal{A} \rightarrow \mathcal{U}$

- Banach space $\mathcal{A} : \{a : D \rightarrow \mathbb{R}^{d_a}\}$ and $\mathcal{U} : \{a : D \rightarrow \mathbb{R}^{d_a}\}$ are all function space

- Idea 1:
  - Linear Encoding from a function to $\mathbb{R}^{d_1}$ code
  - Transform a $\mathbb{R}^{d_1}$ code to a $\mathbb{R}^{d_2}$ code
  - Linear Decoding from $\mathbb{R}^{d_2}$ code to a function

- Idea 2: Directly feature extraction in the function space!
We need to generalize operations in neural networks to function space.

\[
\nu_{l+1}(s) = \sigma \left( W_l \nu_l(s) + \int_{D} k_l(s, z) \nu_l(z) \, dz + b_l(s) \right)
\]

We need to generalize operations in neural networks to function space

\[ v_{l+1}(s) = \sigma \left( W_l v_l(s) + \int_D k_l(s, z)v_l(z)dz + b_l(s) \right) \]


- Fast implementation: Fourier Neural Operator

  FFT->multiplication->iFFT->nonlinear activation

We need to generalize operations in neural networks to function space

**Attention**

Original Attention: Fourier Transform

\[ \int_{\Omega} (\xi_q(x_i) \phi_k(\xi)) v_j(\xi) \, d\xi \]

**Computation scales** \( O(n^2k) \): \( n \) number of pixels, \( k \) number of "Basis"

We need to generalize operations in neural networks to function space.

**Attention**

**Galerkin Attention:**

\[ z_j(x_i) = \sum_{j=1}^{d} \left( \int_{\Omega} k_l(\xi)v_j(\xi) d\xi \right) q_l(x_i) \]

**Computation scales** \(O(nk^2)\): \(n\) number of pixels, \(k\) number of "Basis"

Convergence Rate

- de Hoop M V, Nelsen N H, et al. Convergence rates for learning linear operators from noisy data. SIAM/ASA Journal on Uncertainty Quantification
Convergence Rate

- de Hoop M V, Nelsen N H, et al. Convergence rates for learning linear operators from noisy data. SIAM/ASA Journal on Uncertainty Quantification

Improved Rates by Multi-level Methods

The **Koopman** operator is a linear but infinite-dimensional operator that describes the evolution of observables in a finite dimensional dynamical system.

Mathematically, the Koopman operator is defined as:

$$ U^t g(x) = g \circ F^t(x) $$

where $U^t$ is the adjoint of the generator $F^t$. This allows us to understand how the distribution of state space evolves through the dynamic.
A linear operator is an “infinite-dimensional” matrix,
A linear operator is an “infinite-dimensional” matrix, operator learning equals to reconstruct a matrix using matrix-vector multiplication.


Multilevel algorithms are essential to achieve minimax optimality, which differs from finite-dimensional matrix reconstruction!

A linear operator is an “infinite-dimensional” matrix, operator learning equals to reconstruct a matrix using matrix-vector multiplication.


Multilevel algorithms are essential to achieve minimax optimality, which differs from finite-dimensional matrix reconstruction!

A linear operator is an “infinite-dimensional” matrix, operator learning equals to reconstruct a matrix using matrix-vector multiplication.

A linear operator is an “infinite-dimensional” matrix, operator learning equals to reconstruct a matrix using matrix-vector multiplication.


Optimal regularization differs for each row!

Evert Row is a Linear Regression

Multilevel algorithms are essential to achieve minimax optimality, which differs from finite-dimensional matrix reconstruction!

Hardness of learning in infinite dimensions

5 Operator Learning

Neural operators can approximate any continuous operator. Chen & Chen 1995, Nikola Kovachki et al. 2021
Neural operators can approximate any continuous operator.  

Chen & Chen 1995, Nikola Kovachki et. al. 2021

**Curse of Dimensionality**

The cost to represent a function is exponential to the dimensionality.

**Smoothness only is not enough to break the Curse of dimensionality!**
Different from finite dimension, **Smoothness** only is not enough to break the Curse of Dimensionality! Additional structure is needed, such as

- **Holomorphic Mappings**
  

- **PDE Operators**
  
Different from finite dimension, **Smoothness** only is not enough to break the Curse of Dimensionality! Additional structure is needed, such as

- **Holomorphic Mappings**


- **PDE Operators**


**Open Question**

What is the **general structure** that makes operator possible in infinite dimension?
Different structure that is used to break the curse of dimensionality includes:

- Darcy Flow, Navier-Stokes via PCA-Net [1]
- Hamilton-Jacobi Equation [2]
- ...

Different structure that is used to break the curse of dimensionality includes

- Darcy Flow, Navier-Stokes via PCA-Net [1]
- Hamilton-Jacobi Equation [2]
- ...

Idea Neural Network can approximate known "algorithms"

- Approximate convergent schemes such as spectral methods
- Approximate the Method of Characteristics

Similar to approximation theory for PINN


Neural Operator adaptive to certain structure is essential!
Approximate partial derivative using finite difference can be represent as convolution.
Approximate partial derivative using finite difference can be represent as convolution.

How to approximate $\partial_{xx}f(x, y)$?

**Question**

How can we map convolution kernels with finite difference operators?
Approximate partial derivative using finite difference can be represented as convolution.

How to approximate \( \partial_{xx} f(x, y) \)?

\[
\partial_{xx} f(x, y) \approx \frac{f(x-\Delta x, y) + f(x+\Delta x, y) - 2f(x, y)}{\Delta x^2}
\]

Question

How can we map convolution kernels with finite difference operators?
Approximate partial derivative using finite difference can be represented as a convolution.

\[ f(x, y) \approx f(x - \Delta x, y) + f(x + \Delta x, y) - 2f(x, y) \]

**How to approximate** \( \partial_{xx}f(x, y) \)?

\[ \partial_{xx}f(x, y) \approx \frac{f(x-\Delta x, y) + f(x+\Delta x, y) - 2f(x, y)}{\Delta x^2} \]

Equivalent to a convolution kernel \([-1, 2, 1]\).

**Question**

How can we map convolution kernels with finite difference operators?
What’s the property of a partial derivative?
Differentiation can cause low order polynomial to zero!
What’s the property of a partial derivative?
Differentiation can because low order polynomial to zero!

**Orders of sum rules**

For a filter $q$, we say $q$ to have sum rules of order $\alpha = (\alpha_1, \alpha_2)$, where $\alpha \in \mathbb{Z}_+^2$, provided that

$$\sum_{k \in \mathbb{Z}^2} k^\beta q[k] = 0$$

for all $\beta \in \mathbb{Z}_+^2$ with $|\beta| < |\alpha|$ and for all $\beta \in \mathbb{Z}_+^2$ with $|\beta| = |\alpha|$ but $\beta \neq \alpha$. If (16) holds for all $\beta \in \mathbb{Z}_+^2$ with $|\beta| < K$ except for $\beta \neq \beta_0$ with certain $\beta_0 \in \mathbb{Z}_+^2$ and $|\beta_0| = J < K$, then we say $q$ to have total sum rules of order $K \setminus \{J + 1\}$.

**Linear constraints on convolutional weights!**
PDE-Net is a neural network but can also represent a PDE with form

\[ u_t = f(u, u_x, u_{xx}, \cdots) \]
Build a big dictionary

$$\Theta(U) = [1, U, U^2, \cdots, \sin(U), \cdots, U_x, U_x^2, \cdots, U_{xx}, U_{xx}^2, \cdots]$$

possible dictionary

and then preform sparse regression methods


Intuitively speaking, the balls in our data set (whiffle balls, perhaps, excluded) are similar enough objects that the equations governing their trajectories should include similar terms. **Group Sparsity!**

<table>
<thead>
<tr>
<th>Ball</th>
<th>First drop</th>
<th>Second drop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Golf Ball</td>
<td>$\ddot{x} = -9.34 + 0.05v$</td>
<td>$\ddot{x} = -9.44 - 0.03v$</td>
</tr>
<tr>
<td>Baseball</td>
<td>$\ddot{x} = -8.51 + 0.14v$</td>
<td>$\ddot{x} = -7.56 + 0.14v$</td>
</tr>
<tr>
<td>Tennis Ball</td>
<td>$\ddot{x} = -9.08 - 0.13v$</td>
<td>$\ddot{x} = -8.64 - 0.12v$</td>
</tr>
<tr>
<td>Volleyball</td>
<td>$\ddot{x} = -8.11 - 0.08v$</td>
<td>$\ddot{x} = -9.64 - 0.23v$</td>
</tr>
<tr>
<td>Blue Basketball</td>
<td>$\ddot{x} = -6.71 + 0.15v$</td>
<td>$\ddot{x} = -7.50 + 0.07v$</td>
</tr>
<tr>
<td>Green Basketball</td>
<td>$\ddot{x} = -7.36 + 0.10v$</td>
<td>$\ddot{x} = -8.05 + 0.02v$</td>
</tr>
<tr>
<td>Whiffle Ball 1</td>
<td>$\ddot{x} = -8.24 - 0.34v$</td>
<td>$\ddot{x} = -9.44 - 0.43v$</td>
</tr>
<tr>
<td>Whiffle Ball 2</td>
<td>$\ddot{x} = -9.81 - 0.56v$</td>
<td>$\ddot{x} = -9.79 - 0.48v$</td>
</tr>
<tr>
<td>Yellow Whiffle Ball</td>
<td>$\ddot{x} = -8.50 - 0.47v$</td>
<td>$\ddot{x} = -8.45 - 0.46v$</td>
</tr>
<tr>
<td>Orange Whiffle Ball</td>
<td>$\ddot{x} = -7.83 - 0.35v$</td>
<td>$\ddot{x} = -8.03 - 0.42v$</td>
</tr>
</tbody>
</table>

The Reynolds number for a ball with diameter $D$ and velocity $v$ will then be

$$\text{Re} = 0.6667Dv \times 10^5$$

<table>
<thead>
<tr>
<th>Ball</th>
<th>Radius (m)</th>
<th>Mass (kg)</th>
<th>Density (kg/m)</th>
<th>$v_{\text{max}}$ (m/s)</th>
<th>Max Re</th>
</tr>
</thead>
<tbody>
<tr>
<td>Golf Ball</td>
<td>0.021963</td>
<td>0.045359</td>
<td>1022.066427</td>
<td>26.63</td>
<td>1.75 × 10^5</td>
</tr>
<tr>
<td>Baseball</td>
<td>0.035412</td>
<td>0.141747</td>
<td>762.037525</td>
<td>26.61</td>
<td>2.83 × 10^5</td>
</tr>
<tr>
<td>Tennis Ball</td>
<td>0.033025</td>
<td>0.056699</td>
<td>375.813253</td>
<td>21.95</td>
<td>2.18 × 10^5</td>
</tr>
<tr>
<td>Volleyball</td>
<td>0.105*</td>
<td>NA</td>
<td>NA</td>
<td>22.09</td>
<td>6.96 × 10^5</td>
</tr>
<tr>
<td>Blue Basketball</td>
<td>0.119366</td>
<td>0.510291</td>
<td>71.628378</td>
<td>24.80</td>
<td>8.88 × 10^5</td>
</tr>
<tr>
<td>Green Basketball</td>
<td>0.116581</td>
<td>0.453592</td>
<td>68.342914</td>
<td>25.06</td>
<td>8.77 × 10^5</td>
</tr>
<tr>
<td>Whiffle Ball 1</td>
<td>0.036287</td>
<td>0.028349</td>
<td>141.641937</td>
<td>16.91</td>
<td>1.84 × 10^5</td>
</tr>
<tr>
<td>Whiffle Ball 2</td>
<td>0.036287</td>
<td>0.028349</td>
<td>141.641937</td>
<td>16.35</td>
<td>1.78 × 10^5</td>
</tr>
<tr>
<td>Yellow Whiffle Ball</td>
<td>0.046155</td>
<td>0.042524</td>
<td>103.250857</td>
<td>15.30</td>
<td>2.12 × 10^5</td>
</tr>
<tr>
<td>Orange Whiffle Ball</td>
<td>0.046155</td>
<td>0.042524</td>
<td>103.250857</td>
<td>15.77</td>
<td>2.18 × 10^5</td>
</tr>
</tbody>
</table>

Complex secondary physical mechanisms, like unsteady fluid drag forces, can obscure the underlying law of gravitation, leading to an erroneous model.
Different Levels of Interpretability

5 Operator Learning

Fully white box, limited capacity

Gray box neural network

Physics knowledge as network structure, differentiable physics that integrate FEM/FDM solvers

Fully black box, universal approximator

SIDy

PDE-Net

DeepONet, FNO
In this tutorial, we introduced empirical and theoretical challenges to cooperate physical information $Au = f$ to machine learning systems.
Recent Advances in Physics-Informed Machine Learning

Thank you for listening!

Any questions?