Solving High-Dimensional PDEs Using Deep Learning: Original Insights and Recent Progress

Jiequn Han
Center of Computational Mathematics
Flatiron Institute, Simons Foundation

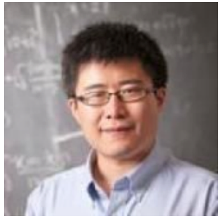
SIAM Conference on Computational Science and Engineering (CSE25)
March 3, 2025

Thanks to ...


Weinan E


Arnulf Jentzen


Roberto Car


Jianfeng Lu


Leslie Greengard

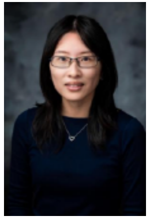

Joan Bruna

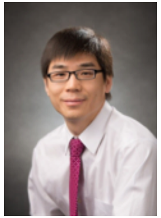

Jihao Long


Linfeng Zhang


Benjamin Moll


Qianxiao Li


Ruimeng Hu


Heng Xiao


Manas Rachh

and other brilliant collaborators - too many to list!

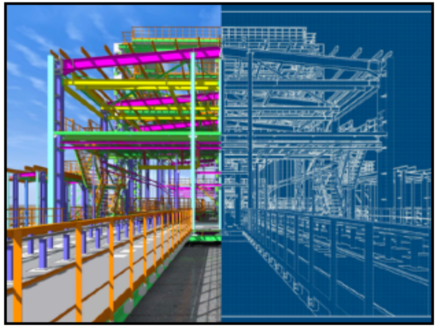
PRINCETON UNIVERSITY

SIMONS FOUNDATION

“Third Pillar” of Science

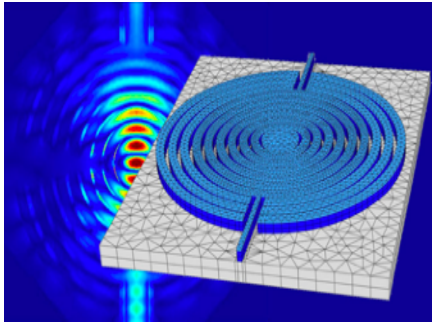
Together with theory and experimentation, computational science now constitutes the “third pillar” of scientific inquiry.

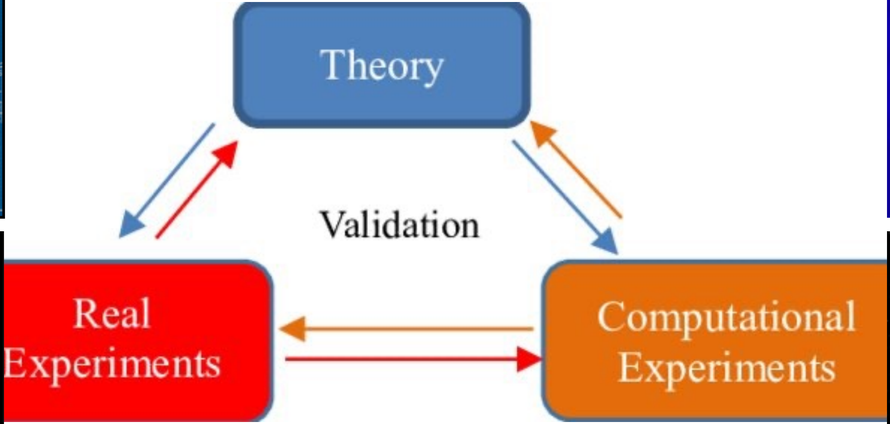
— President’s Information Technology Advisory Committee report (2005)

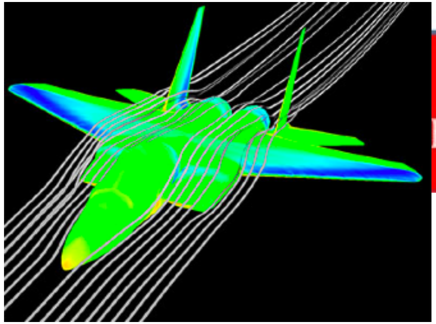


structural analysis

optics

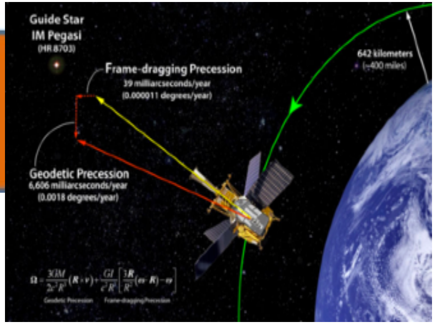






fluid dynamics

control

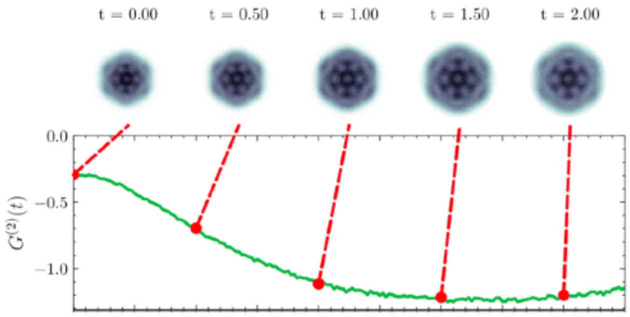


Examples of High-Dimensional PDE

Schrödinger equation

$$i\frac{\partial u}{\partial t}(t,x) = -\frac{1}{2}\Delta_x u(t,x) + V(t,x)u(t,x)$$

Dim of x = # of electrons $\times 3$

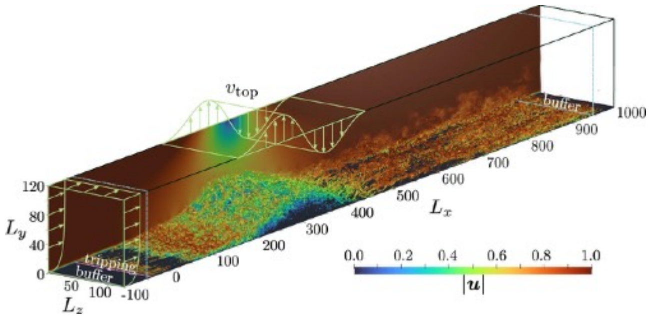


Nys et al., Nat. Commun. (2025)

The Hamilton-Jacobi-Bellman equation in optimal control

$$\frac{\partial u}{\partial t}(t,x) + \frac{1}{2}\Delta_x u(t,x) - \|\nabla u(t,x)\|^2 = 0$$

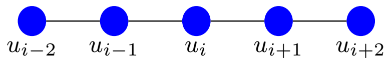
Dim of x = # of the state variable

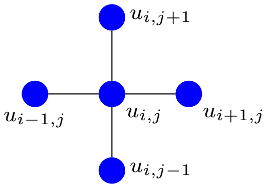


Font et al., Nat. Commun. (2025)

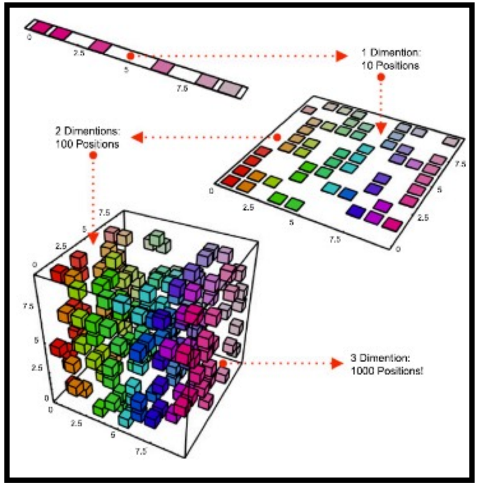
Curse of Dimensionality (1)

With $h = 1/N$, to have a solution with error $O(h^2)$:





- 1D problems $\rightarrow$ solving a system of size $O(N)$
- 2D problems $\rightarrow$ solving a system of size $O(N^2)$
- 3D problems $\rightarrow$ solving a system of size $O(N^3)$
- $\cdots$
- d-dim problems $\rightarrow$ solving a system of size $O(N^d)$

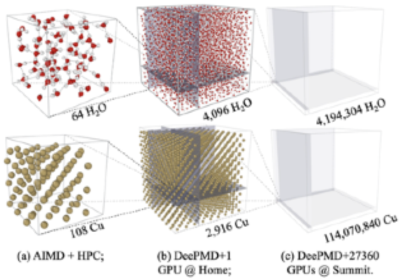


Curse of dimensionality:
\# of nodes per dimension = N $\rightarrow$ solution with error $O(h^2)$
As the dimension of variables d grows, the computational cost $O(N^d)$ grows exponentially with respect to d

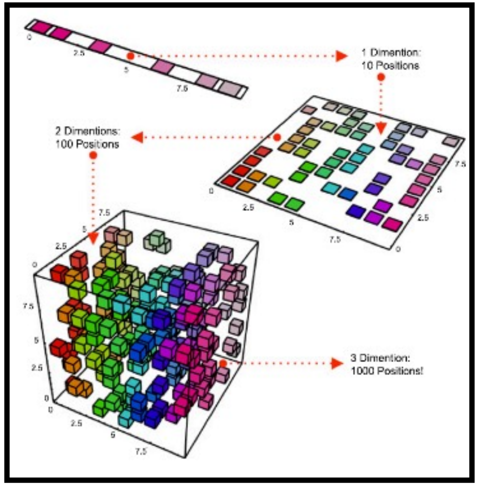
To reduce the error by a factor 4, the system size grows with a factor of 2^d

Curse of Dimensionality (2)

The curse of dimensionality is widespread in traditional numerical methods for PDEs. In many problems, we can only handle $d = 3 \sim 5$



The phenomenon also exists in other scientific computing problems whenever they involve many state variables



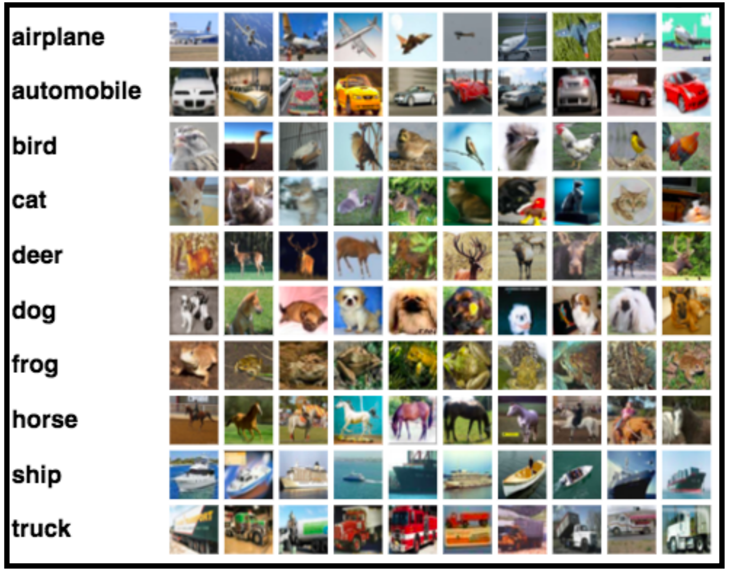
The core challenge: traditional methods need exponentially many basis functions to approximate high-dimensional functions.

Wonder of DL: Approximating High-Dim Functions

Deep learning: general function approximation

Given $S = \{(x_j, y_j = f^\star(x_j)),\ j = 1,\cdots,N\}$
learn (i.e., approximate) $f^\star$
which is a high-dimensional mapping

CIFAR 10 - input: each image is 32 * 32 * 3 = 3072 dimensional; output: 10 categories



Imagine we have something like “polynomials” but works in high dimensions!

Stochastic Formulation for High-Dimensional PDE

Formulation/
Loss Function

Network
Architecture

Optimizer

Original insight: stochastic formulations are well-suited for high-dimensional PDEs

Linear Parabolic PDEs (1)

Linear PDE: $$\frac{\partial u}{\partial t}(t,x) + \frac{1}{2}\Delta_x u(t,x) + \nabla u(t,x)\cdot\mu(t,x) + f(t,x) = 0, \quad u(T,x) = g(x)$$

$$(\Omega = [0,T]\times\mathbb{R}^d)$$

Examples: Black-Scholes, diffusion equation, … interested in the high-dim cases

Feynman-Kac formula:

$$\mathrm{d}X_t = \mu(t,X_t)\mathrm{d}t + \mathrm{d}W_t$$

$$\Longrightarrow \mathrm{d}u(t,X_t) = (u_t(t,X_t) + \nabla u(t,X_t)\cdot\mu(t,X_t) + \frac{1}{2}\Delta(t,X_t))\mathrm{d}t + [\nabla u(t,X_t)]^{\mathrm{T}}\mathrm{d}W_t$$ (Itô’s lemma)

$$\Longrightarrow \mathrm{d}u(t,X_t) = -f(t,X_t)\mathrm{d}t + [\nabla u(t,X_t)]^{\mathrm{T}}\mathrm{d}W_t$$

$$\Longrightarrow \mathbb{E}[u(T,X_T) - u(t,X_t)] = -\mathbb{E}\int_t^T f(s,X_s)\mathrm{d}s$$

$$\Longrightarrow u(t,x) = \mathbb{E}[\, g(X_T) + \int_t^T f(s,X_s)\mathrm{d}s \,|\, X_t = x \,]$$

$u(t,X_t)$ satisfies a backward stochastic differential equation (BSDE) with a given terminal condition

Linear Parabolic PDEs (2)

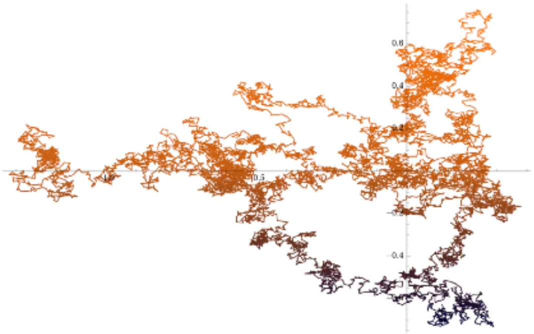
Linear PDE: $$\frac{\partial u}{\partial t}(t,x) + \frac{1}{2}\Delta_x u(t,x) + \nabla u(t,x)\cdot\mu(t,x) + f(t,x) = 0, \quad u(T,x) = g(x)$$

Feynman-Kac formula

$$\mathrm{d}X_t = \mu(t, X_t)\mathrm{d}t + \mathrm{d}W_t$$

Monte Carlo Simulation

$$\Longrightarrow \quad u(t,x) = \mathbb{E}\left[\, g(X_T) + \int_t^T f(s, X_s)\mathrm{d}s \,\middle|\, X_t = x \,\right]$$



Monte Carlo simulation:

$$u(t,x) \approx \frac{1}{N}\sum_{i=1}^{N}\left[g(X_T^{(i)}) + \int_t^T f(s, X_s^{(i)})\mathrm{d}s\right]$$

The convergence rate is $1/\sqrt{N}$ (N = # of paths), which is independent of dimensions!

Semilinear Parabolic PDEs

$$\frac{\partial u}{\partial t}(t,x) + \frac{1}{2}\Delta u(t,x) + \nabla u(t,x)\cdot\mu(t,x) \; + f(t,x,u(t,x),\nabla u(t,x)) = 0, \quad u(T,x) = g(x)$$

With the idea of Feynman-Kac formula

$$\mathrm{d}X_t = \mu(t,X_t)\mathrm{d}t + \mathrm{d}W_t$$

$$\implies \mathrm{d}u(t,X_t) = -f(t,X_t,u(t,X_t),\nabla u(t,X_t))\mathrm{d}t + [\nabla u(t,X_t)]^{\mathrm{T}}\mathrm{d}W_t$$

$$\implies u(t,x) = \mathbb{E}[g(X_T) + \int_t^T f(s,X_s,u(s,X_s),\nabla u(s,X_s))\mathrm{d}s \,|\, X_t = x]$$

But we do not know the value of $u(s,X_s), \nabla u(s,X_s)$ along the paths :(

E, Han, and Jentzen, CMS (2017); Han, Jentzen, and E, PNAS (2018)

Variational Formulation (1)

$$\frac{\partial u}{\partial t}(t,x) + \frac{1}{2}\Delta u(t,x) + \nabla u(t,x)\cdot\mu(t,x) + f(t,x,u(t,x),\nabla u(t,x)) = 0, \quad u(T,x) = g(x)$$

Rename $Y_t = u(t, X_t)$

Suppose $Y_0 = u(0, X_0),\ Z_t = \nabla u(t, X_t)$

$$\begin{aligned} \mathrm{d}X_t &= \mu(t,X_t)\mathrm{d}t + \mathrm{d}W_t \\ \mathrm{d}u(t,X_t) &= -f(t,X_t,u(t,X_t),\nabla u(t,X_t))\mathrm{d}t \\ &\quad + [\nabla u(t,X_t)]^{\mathrm{T}}\mathrm{d}W_t \\ u(T,x) &= g(x) \end{aligned} \longrightarrow \begin{aligned} \mathrm{d}X_t &= \mu(t,X_t)\mathrm{d}t + \mathrm{d}W_t \\ \mathrm{d}Y_t &= -f(t,X_t,Y_t,Z_t)\mathrm{d}t + (Z_t)^{\mathrm{T}}\mathrm{d}W_t \\ Y_T &= g(X_T) \end{aligned}$$

stochastic formulation

$$\inf_{Y_0,\{Z_t\}_{0\le t\le T}} \mathbb{E}|g(X_T) - Y_T|^2$$

$$\begin{aligned} s.t.\quad \mathrm{d}X_t &= \mu(t,X_t)\mathrm{d}t + \mathrm{d}W_t \\ \mathrm{d}Y_t &= -f(t,X_t,Y_t,Z_t)\mathrm{d}t + (Z_t)^{\mathrm{T}}\mathrm{d}W_t \end{aligned}$$

A control perspective to match terminal condition

E, Han, and Jentzen, CMS (2017); Han, Jentzen, and E, PNAS (2018)

Variational Formulation (2)

Rename $Y_t = u(t, X_t)$

Suppose $Y_0 = u(0, X_0), \ Z_t = \nabla u(t, X_t)$

$$dX_t = \mu(t, X_t)dt + dW_t$$

$$du(t, X_t) = -f(t, X_t, u(t, X_t), \nabla u(t, X_t))dt + [\nabla u(t, X_t)]^T dW_t$$

$$u(T, x) = g(x)$$

$$\longrightarrow \inf_{Y_0, \{Z_t\}_{0 \le t \le T}} \mathbb{E}|g(X_T) - Y_T|^2$$

$$s.t. \quad dX_t = \mu(t, X_t)dt + dW_t$$

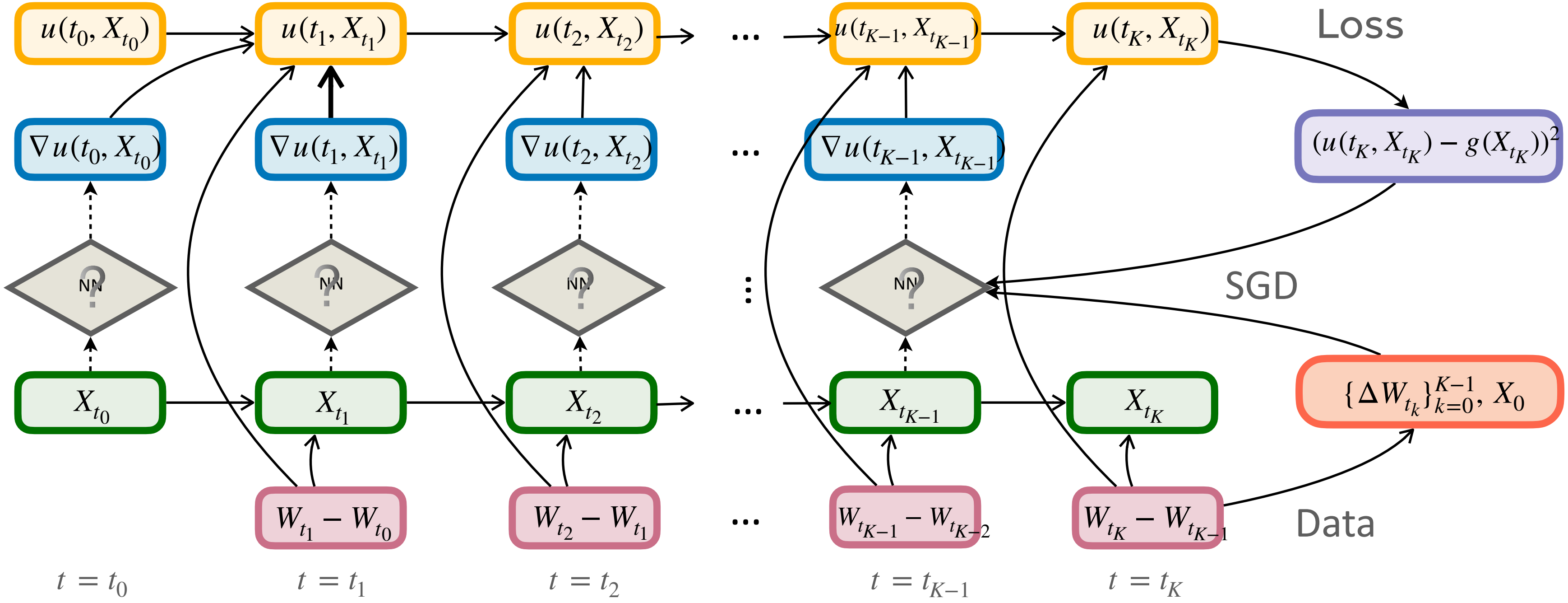
$$dY_t = -f(t, X_t, Y_t, Z_t)dt + (Z_t)^T dW_t$$

1. The PDE solution is the minimizer
2. The minimizer is unique (BSDE theory)
3. 1+2: The minimizer is the PDE solution
4. An approximate minimizer is an approximate PDE solution

E, Han, and Jentzen, CMS (2017); Han, Jentzen, and E, PNAS (2018)

Deep BSDE Method

$$X_{t_{k+1}} = X_{t_k} + \mu(t_k, X_{t_k})\Delta t_k + \Delta W_k \quad Y_{t_{k+1}} = Y_{t_k} - f(t_k, X_{t_k}, Y_{t_k}, \mathcal{N}_k(t_k, X_{t_k}))\Delta t_k + [\mathcal{N}_k(t_k, X_{t_k})]^{\mathrm{T}}\Delta W_k$$



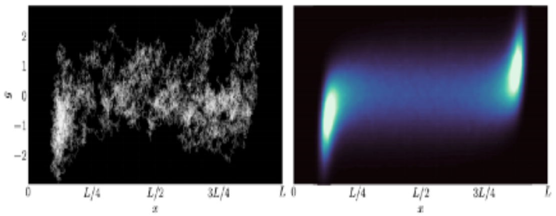
E, Han, and Jentzen, CMS (2017); Han, Jentzen, and E, PNAS (2018), code available at https://github.com/frankhan91/DeepBSDE

Applications

… We believe that this opens up a host of possibilities in economics, finance, operational research, and physics, by considering all participating agents, assets, resources, or particles together at the same time, instead of making ad hoc assumptions on their interrelationships.

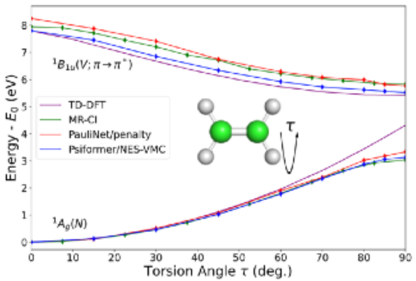
— Han, Jentzen, and E, PNAS (2018)

Fokker-Planck Equation



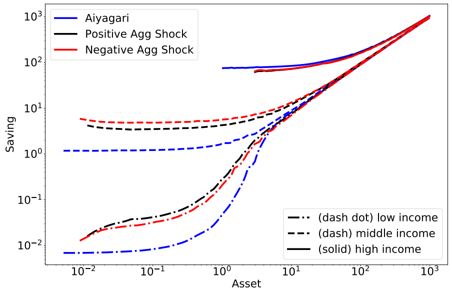
Boffi and Vanden-Eijnden (2024)

Schrödinger Equation



Han, Zhang, and E (2018),
Hermann, Schätzle, and Noé (2020)
Pfau, Spencer, and Matthews (2020)

HJB equation (Optimal Control/ Game Theory/Economics/Finance)



Han, Yang, and E (2021)

All these works leverage the stochastic formulation of the original problem

Numerical Analysis

4. An approximate minimizer is an approximate PDE solution

Theorem (A posteriori error estimate, Han-Long, 2020)

Denote by π the temporal discretization: $0 = t_0 < t_1 < \ldots < t_K = T$, and $h = \max_{0 \le k \le K-1} \Delta t_k$.

Under some assumptions, there exists a constant C independent of d and h such that for sufficiently small h

gradient of solution u

loss function

$$\sup_{t \in [0,T]} \mathbb{E}|Y_t - \hat{Y}_t^\pi|^2 + \int_0^T \mathbb{E}|Z_t - \hat{Z}_t^\pi|^2 \mathrm{d}t \le C\left(h + \mathbb{E}|g(X_T) - Y_T^\pi|^2\right),$$

solution u

time mesh size

where $\hat{Y}_t^\pi = Y_{t_k}^\pi, \ \hat{Z}_t^\pi = Z_{t_k}^\pi$ for $t \in [t_k, t_{k+1})$.

Different Formulations for Learning to Solve PDEs

Neural network approximations for PDEs with alternative formulations:

- Physics-informed neural networks (least-squares)
- Deep Ritz method (variational formulation)
- Neural quantum states/neural wavefunction (variational Monte Carlo)
- Weak Adversarial Networks (weak formulation)

Optimization difficulties arise due to non-regression loss functions

In contrast, optimization in computer vision and natural language processing can often scale to large problems with regression-type loss functions

Can we achieve similar optimization benefits in solving (high-dimensional) PDEs?

New Idea: Regression via Picard Iteration

Recall Feynman-Kac formula: $$u(t,x) = \mathbb{E}[\ g(X_T) + \int_t^T f_u(s, X_s)\mathrm{d}s \,|\, X_t = x\ ]$$

Picard iteration for the fixed-point: $$u_{k+1}(t,x) = \mathbb{E}[\ g(X_T) + \int_t^T f_{u_k}(s, X_s)\mathrm{d}s \,|\, X_t = x\ ]$$

Deep Picard iteration (DPI): For $k = 1,2,\ldots,$ do

1. Sample $\{t_i, x_i\}_{i=1}^M$ according the region of interest
2. For each i, simulate paths $X_t^{(i,j)}$ starting at $X_{t_i}^{(i,j)} = x_i$ with different Brownian motion $W_t^{(i,j)}$
3. Generate labels $y_i = \dfrac{1}{N}\displaystyle\sum_{j=1}^N \left[g(X_T^{(i,j)}) + \int_t^T f_{u_k}(s, X_s^{(i,j)})\mathrm{d}s\right]$
4. Optimize $\displaystyle\sum_{i=1}^M |y_i - u_\theta(t_i, x_i)|^2$ to get u_{k+1}

Han, Hu, Long, and Zhao (2024)

Gradient-Augmented Regression

Bismut-Elworthy-Li formula: $u(t,x) = \mathbb{E}\Big[\dfrac{g(X_T)(W_T - W_t)}{T-t} + \displaystyle\int_t^T \dfrac{f_u(s,X_s)(W_s - W_t)}{s-t}\mathrm{d}s \,|\, X_t = x\Big]$

Variance reduction with control variate:

$$u(t,x) = \mathbb{E}\Big[\frac{(g(X_T) - g(x))(W_T - W_t)}{T-t} + \int_t^T \frac{(f_u(s,X_s) - f_u(t,x))(W_s - W_t)}{s-t}\mathrm{d}s \,|\, X_t = x\Big]$$

Deep Picard iteration (DPI): For $k = 1,2,\ldots,$ do

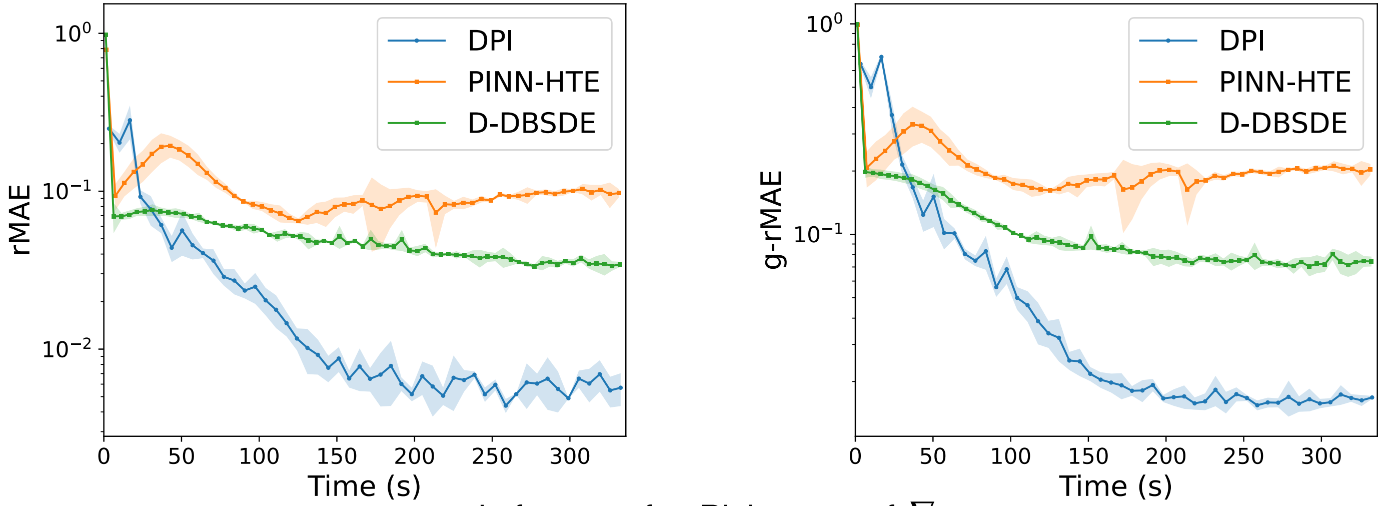
1. Sample $\{t_i, x_i\}_{i=1}^M$ according the region of interest
2. For each i, simulate paths $X_t^{(i,j)}$ starting at $X_{t_i}^{(i,j)} = x_i$ with different Brownian motion $W_t^{(i,j)}$
3. Generate labels $y_i = \dfrac{1}{N}\displaystyle\sum_{j=1}^{N}\Big[g(X_T^{(i,j)}) + \int_t^T f_{u_k}(s, X_s^{(i,j)})\mathrm{d}s\Big]$ and z_i similarly
4. Optimize $\displaystyle\sum_{i=1}^{M} |y_i - u_\theta(t_i,x_i)|^2 + \frac{\lambda}{d}|z_i - \nabla_x u_\theta(t_i,x_i)|^2$ to get u_{k+1}

Han, Hu, Long, and Zhao (2024)

Better Optimization

100-d Burgers-type equation:

$$\partial_t u(t,x) + \frac{\sigma^2}{2}\Delta u(t,x) + \left[\frac{\kappa\sigma^2}{\sqrt{d}}(u-\frac{1}{2}) - \frac{\sqrt{d}}{\kappa}\right]\sum_{i=1}^{d}\frac{\partial u}{\partial x_i}(t,x) = 0.$$

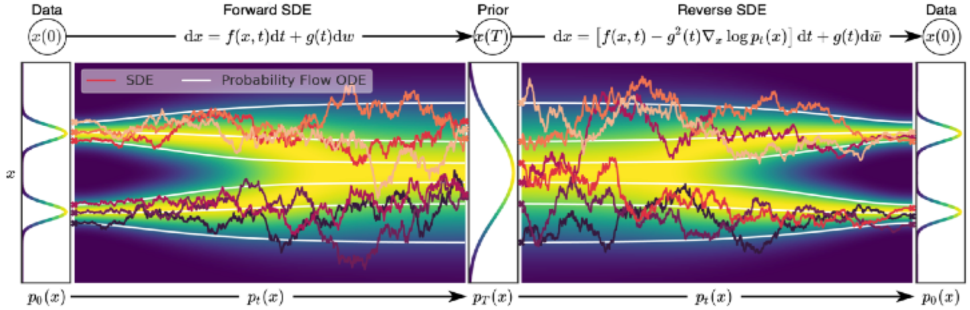


Left: error of u; Right: error of ∇u

Han, Hu, Long, and Zhao (2024)

High-Dimensional Sampling

Score-based diffusion for generative modeling (given a large dataset)

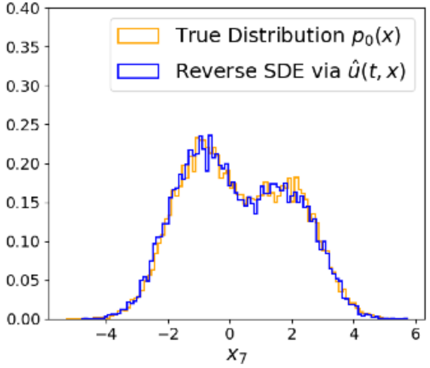


Song et al. ICLR (2021)
Han and Bruna, NeurIPS (2024)

HJB for sampling (no given data) using score-based diffusion:

$$\frac{\partial u}{\partial t}(t,x) - \Delta_x u(t,x) - \|\nabla u(t,x)\|^2 = 0$$

$$u(T,x) = -\log p_0(x) \quad \text{(target density)}$$

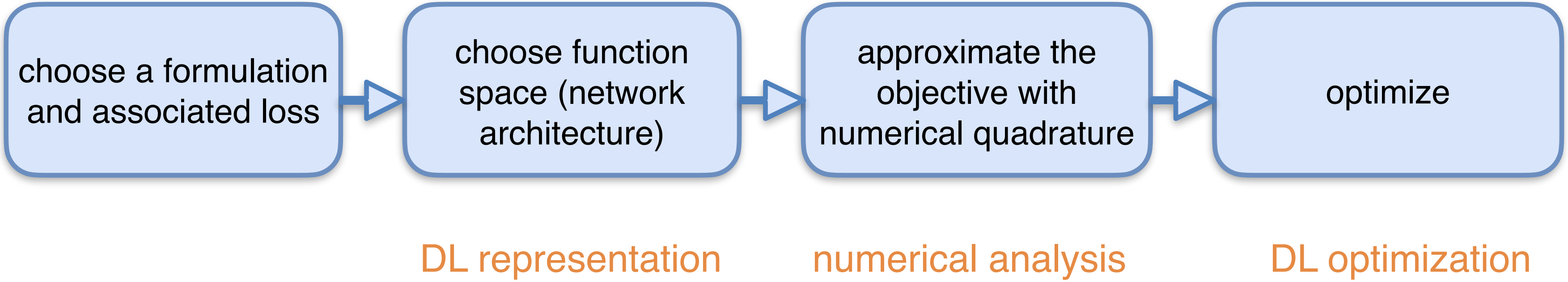


10-d Gaussian mixture

Han, Hu, Long, and Zhao (2024)

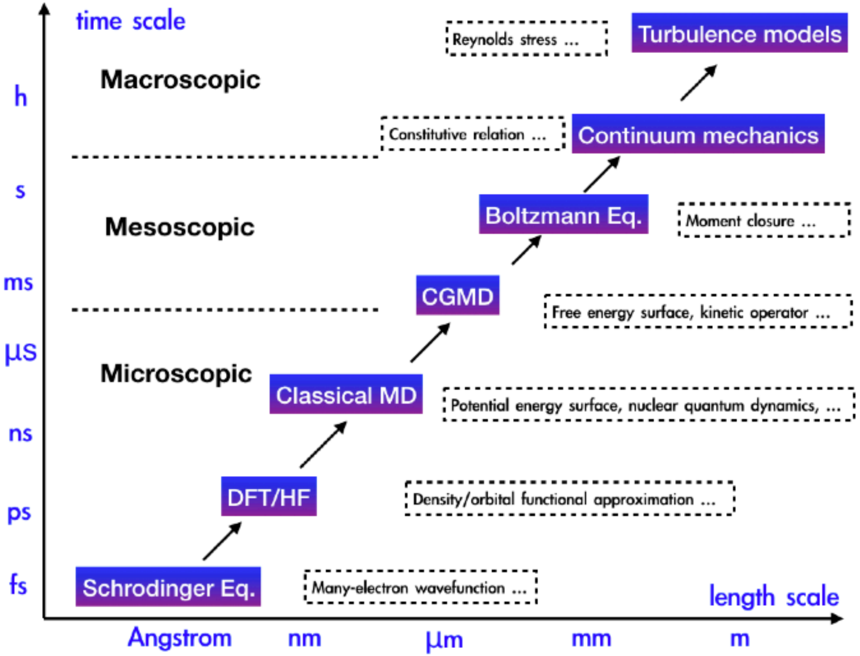
Summary

For a long time, solving high-dimensional PDEs suffers from the curse of dimensionality. Deep learning provides us efficient tools for overcoming this difficulty.



See the review paper *Algorithms for solving high dimensional PDEs: from nonlinear Monte Carlo to machine learning,* E, Han, and Jentzen (2022)

AI for Science and Multiscale Modeling



Closure/Reduced-Order Modeling across scales:

Molecular Dynamics/Coarse-Grained Molecular Dynamics/Kinetic Equations/ Turbulence Modeling/Climate Modeling …

See the review paper *Machine-learning-assisted modeling*, E, Han, and Zhang (2021)

Thanks for your attention!