北京大学
长沙计算与数字经济研究院
PKU-Changsha Institute for Computing and Digital Economy

DL for PDEs: deep adaptive sampling and surrogate modeling

Kejun Tang 唐科军

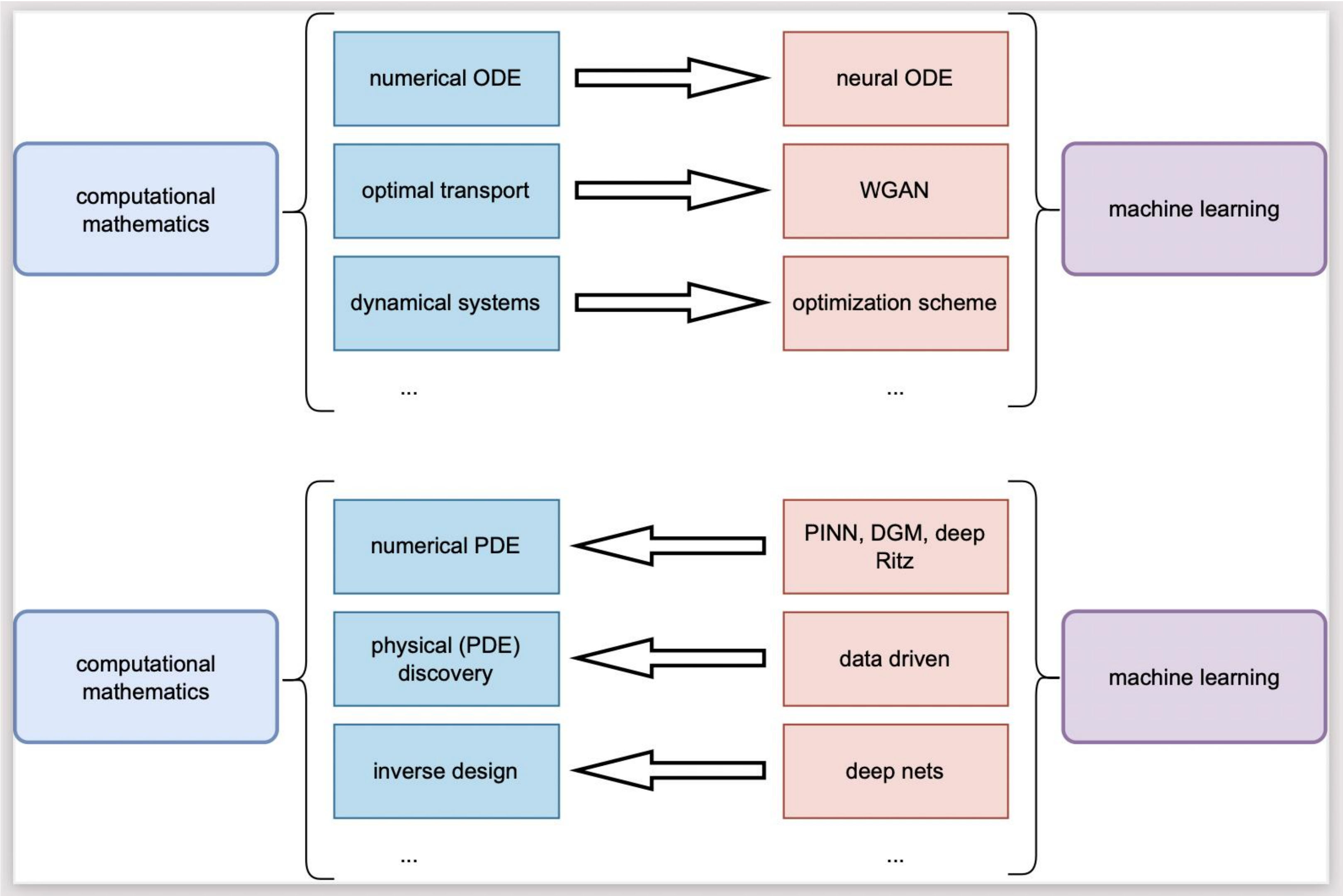
tangkejun@icode.pku.edu.cn

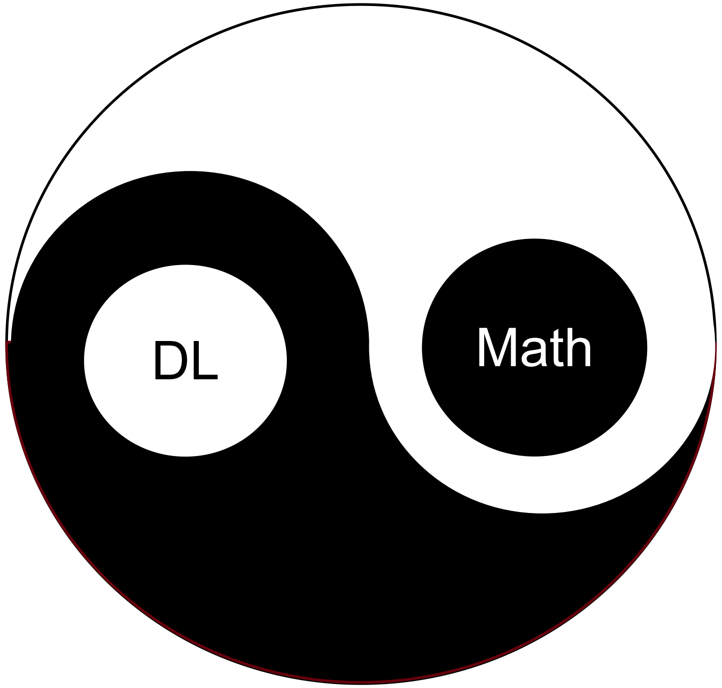
2023.11

Henan University

Content

Computational Mathematics & Machine (Deep) Learning





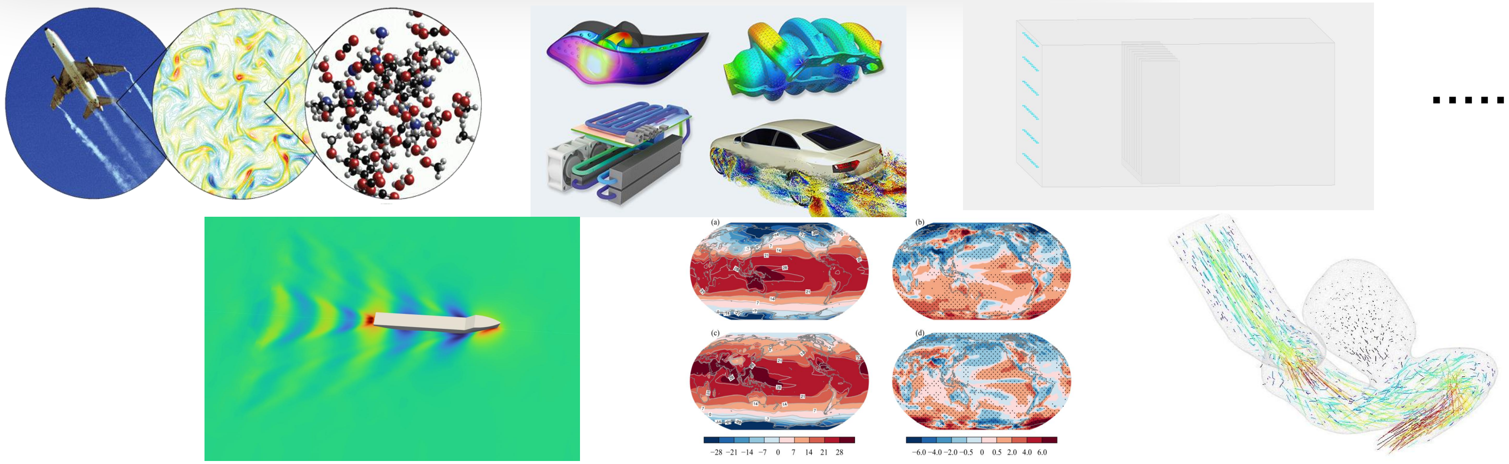
The relationship between math and DL

Deep learning for solving PDEs

Motivation: Many physical laws can be expressed in the form of **partial differential equations (PDEs).**

DL for PDEs: For challenging problems governed by PDEs, deep learning based AI solutions are becoming **an attractive alternative.**

Application fields: acoustics, molecular dynamics, electromagnetics, fluid mechanics, etc.



......

Maxwell

$$\begin{cases} \nabla \times H = J + \partial D/\partial t, & \nabla \times E = \partial B/\partial t, \\ \nabla \cdot B = 0, & \nabla \cdot D = 0. \end{cases}$$

Logistic

$$\frac{dI}{dt} = rI\left(1 - \frac{I}{K}\right), \quad r = \beta K$$

Schrodinger

$$i\hbar\frac{\partial v}{\partial t}\Psi(t,x) = \left(-\frac{1}{2}\Delta + V\right)\Psi(t,x)$$

Allen-Cahn

$$\frac{\partial u}{\partial t} = \Delta u + u - u^3$$

Navier-Stokes

$$\rho\left(\frac{\partial v}{\partial t} + v \cdot \nabla v\right) = \nabla P + \rho g + \mu \nabla^2 v$$

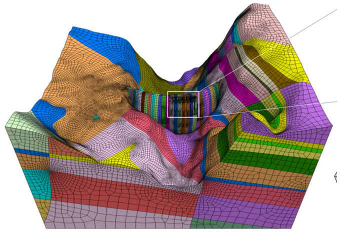
Boltzmann

$$\frac{\partial f}{\partial t} + \xi \cdot \frac{\partial f}{\partial r} + a \cdot \frac{\partial f}{\partial \xi} = \iint (f' f_1' - f f_1) d_D^2 |g| \cos\theta \, d\Omega \, d\xi_1$$

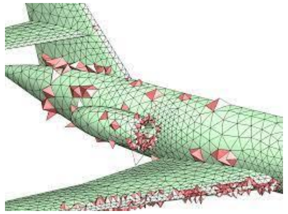
......

Potential advantages of DL for PDEs

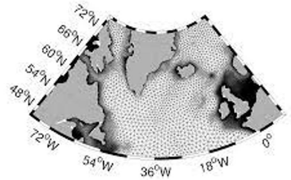
classical numerical methods



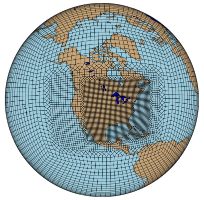
Dam



Aircraft



Ocean

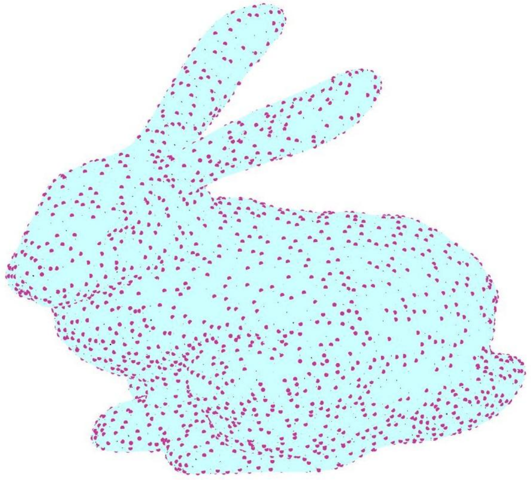


Atmospheric

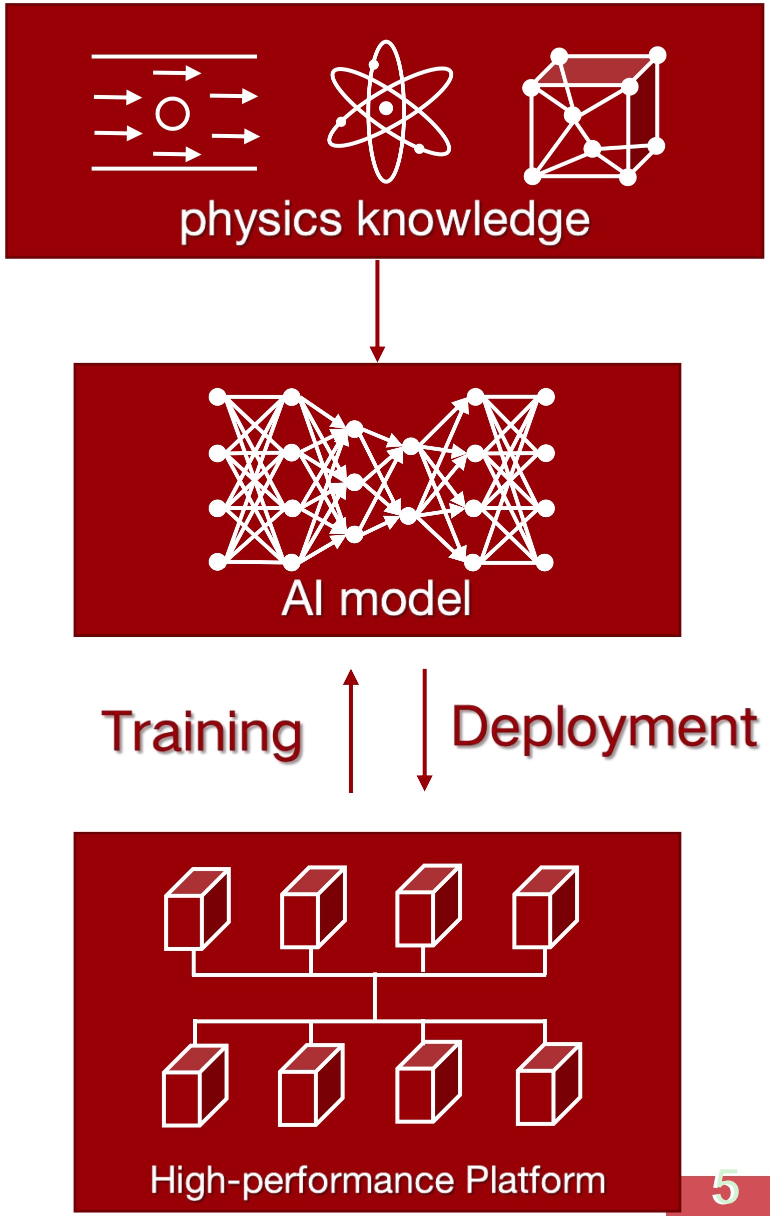
Mesh-based

VS

DL for PDEs

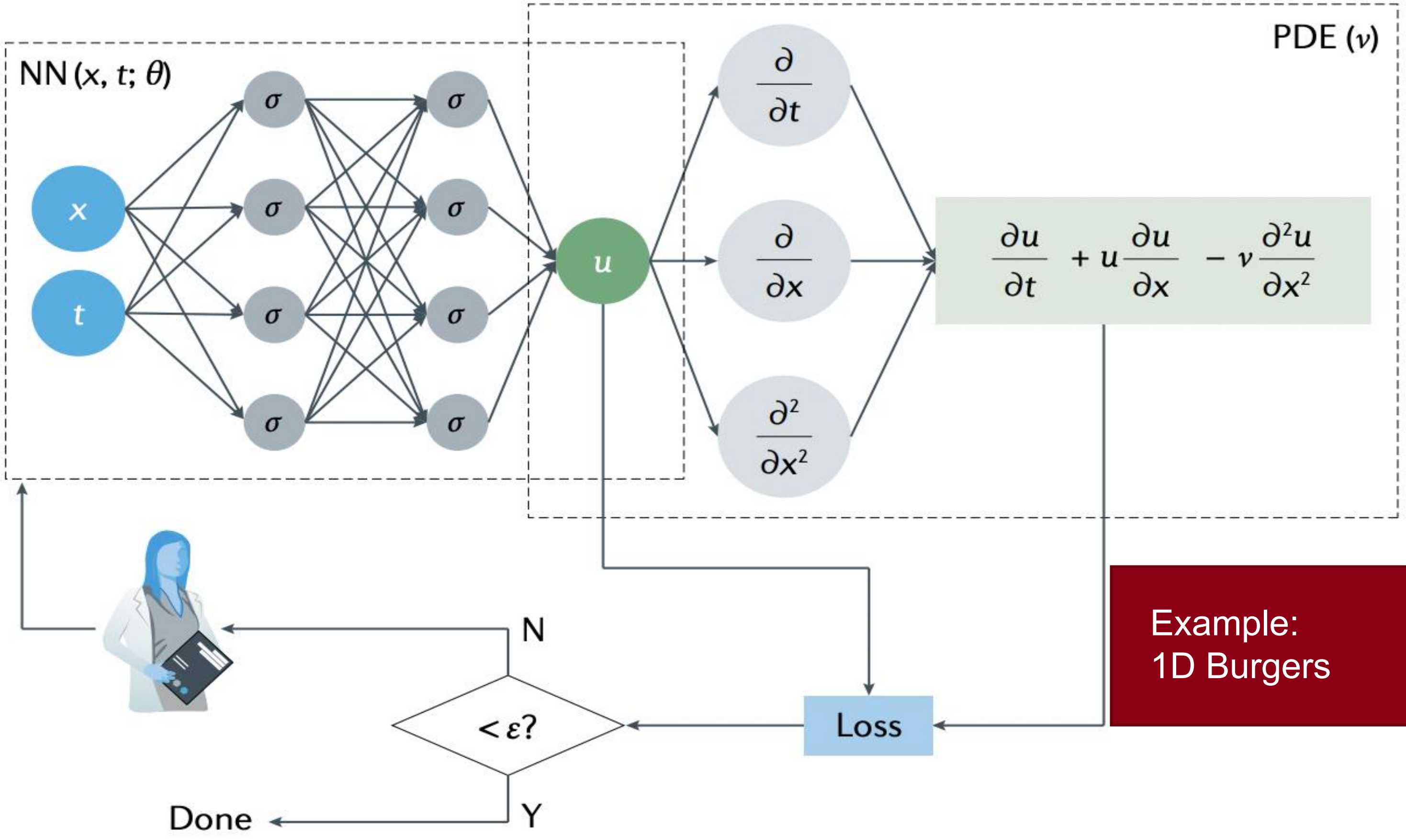


Meshfree



- ✓ High dimensional
- ✓ Naturally meshfree
- ✓ Intrinsically nonlinear
- ✓ surrogate modeling

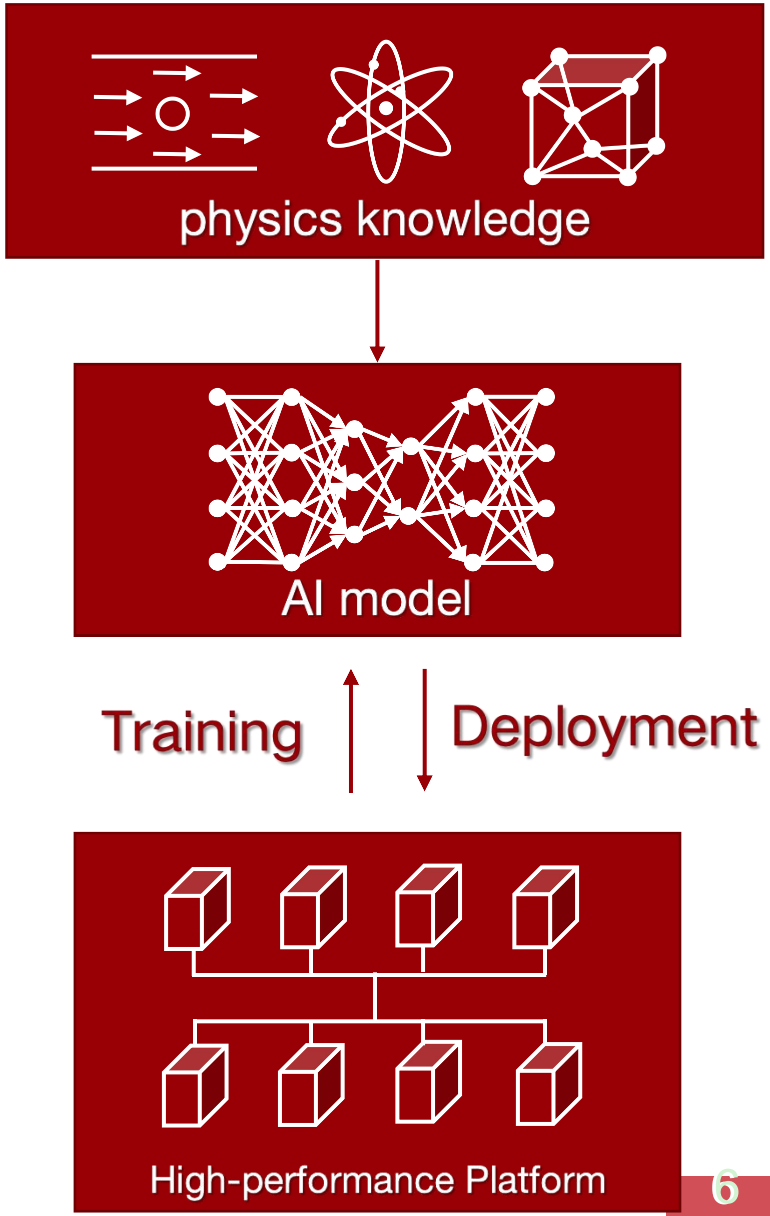
Physics-based DL model



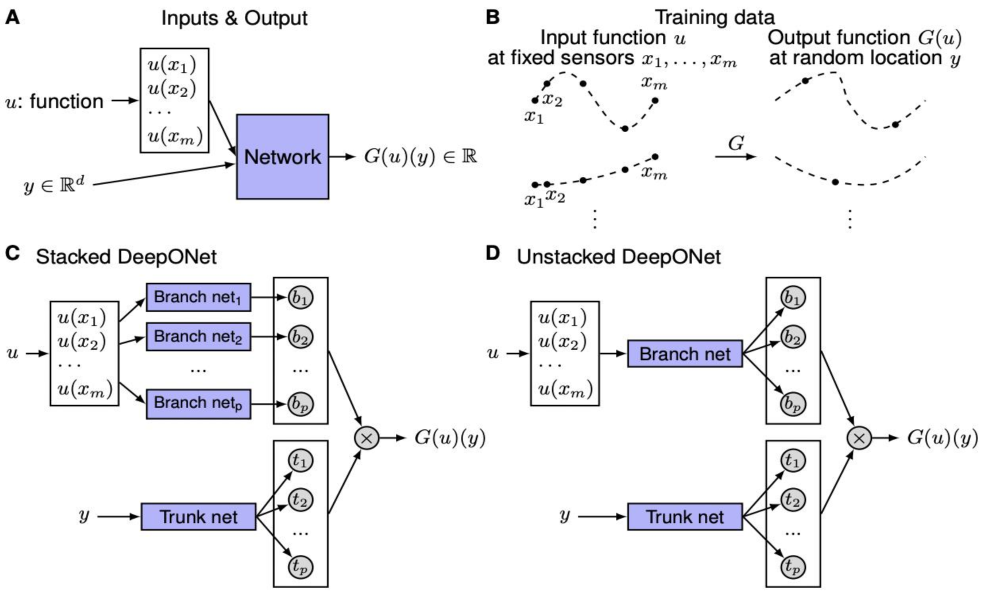
Example:
1D Burgers

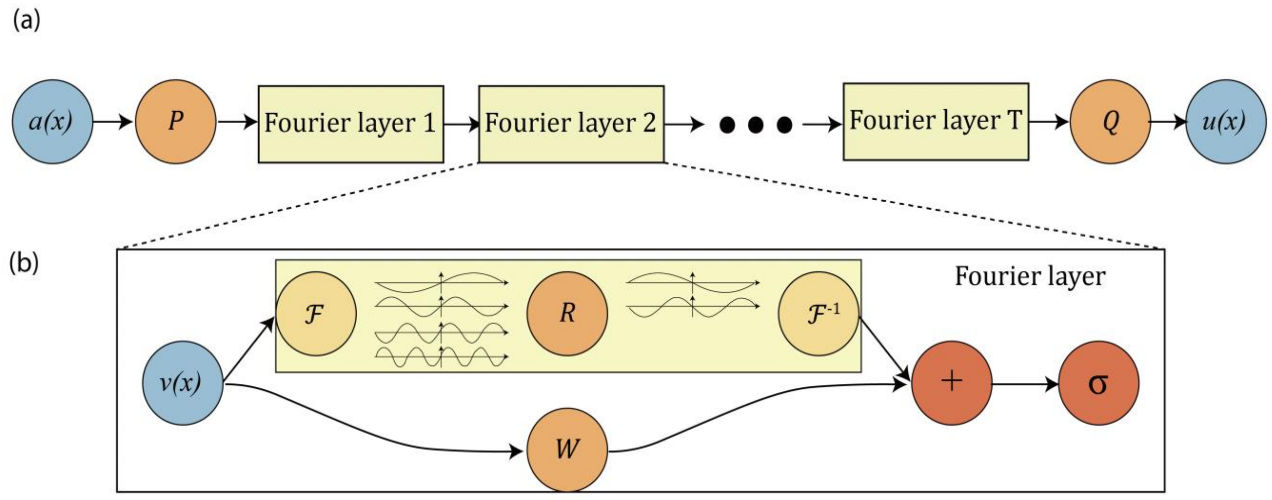
- ✓ High dimensional
- ✓ Naturally meshfree
- ✓ Intrinsically nonlinear

Meshfree



Surrogate modeling - Operator learning





operator: function to function

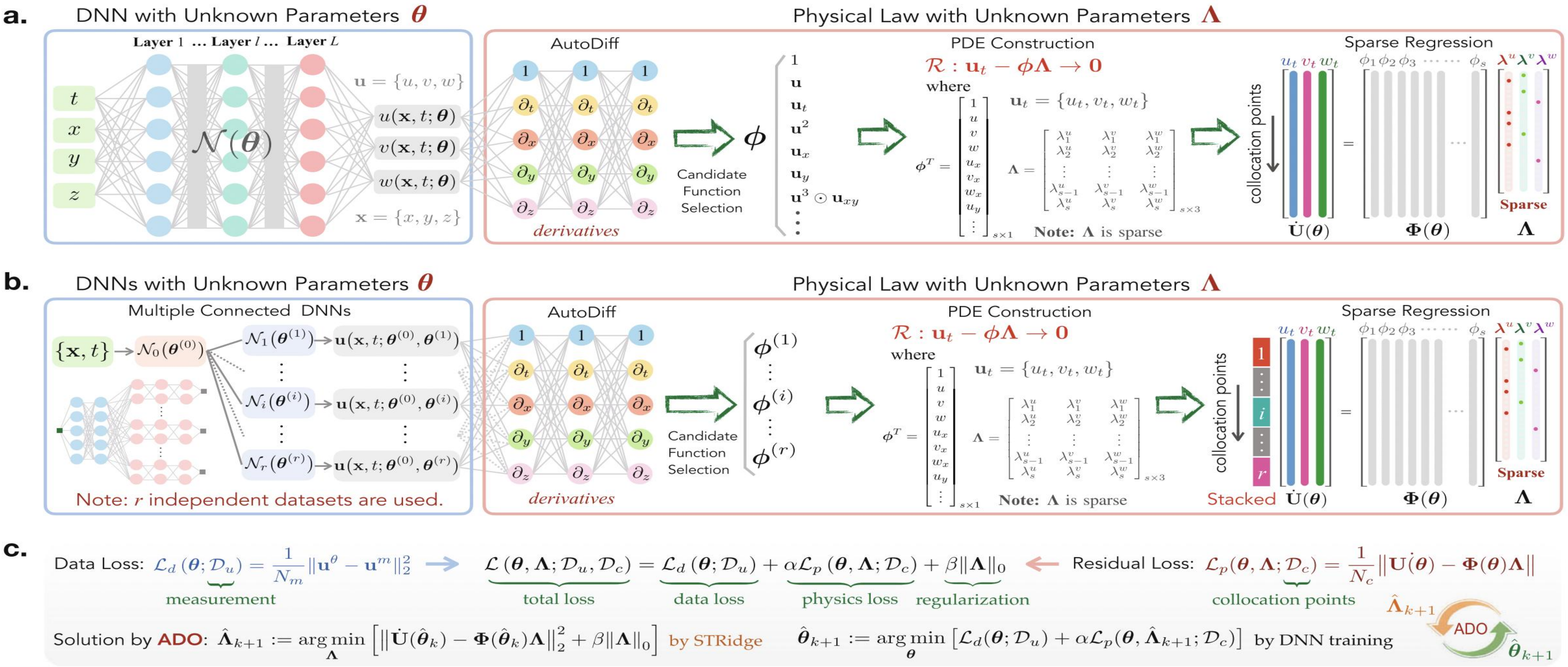
fast solver for parametric PDEs and Bayesian inverse problems

$$\mathcal{L}(u(x)) = s(x)$$

different s(x), different solutions

Lu, Lu, et al., Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nature machine intelligence 3.3 (2021): 218-229.
Li, Zongyi, et al., Fourier neural operator for parametric partial differential equations, arXiv preprint arXiv:2010.08895 (2020).

Physics discovery based on DL



Given some snapshots (say data produced by some PDEs), can we discover the physics model?

Chen, Zhao, Yang Liu, and Hao Sun, Physics-informed learning of governing equations from scarce data, Nature communications 12.1 (2021): 6136.

Content

This talk: focus on adaptive sampling, and surrogate modeling (for parametric optimal control)

The overall process

$$\mathcal{L}[u(x,t)] = s(x,t) \quad \forall (x,t) \in \Omega \times [0,T]$$

$$\mathcal{B}[u(x,t)] = g(x,t) \quad \forall (x,t) \in \partial\Omega \times [0,T]$$

$$\mathcal{I}[u(x,t=0)] = h(x) \quad \forall x \in \Omega$$

$\mathcal{L}$: partial differential operator, e.g., Laplacian

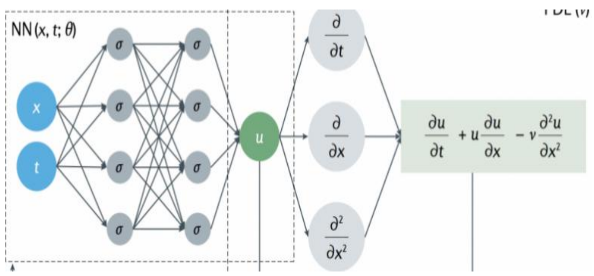
$\mathcal{B}$: boundary operator, e.g., Dirichlet boundary

$\mathcal{I}$: initial operator

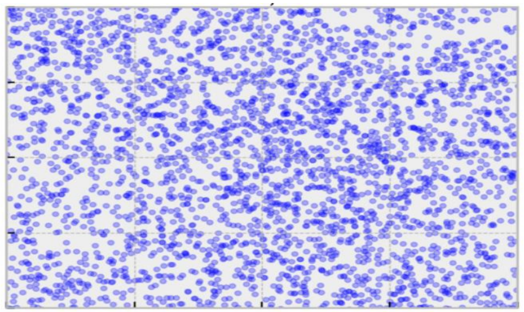
Ω : computational domain, e.g., [-1,1]

$\partial\Omega$: computational domain, e.g., {-1,1}

NN approximation $u(x,t;\theta) \rightarrow u(x,t)$



construct NN



draw training samples

Ω : $\{(x_\Omega^{(i)}, t^{(i)})\}_{i=1}^{N_r}$

$\partial\Omega$: $\{(x_{\partial\Omega}^{(i)}, t^{(i)})\}_{i=1}^{N_b}$

$$r(x,t;\theta) = \mathcal{L}[u(x,t;\theta)] - s(x,t)$$

$$b(x,t;\theta) = \mathcal{B}[u(x,t;\theta)] - g(x,t)$$

$$l(x,t=0;\theta) = \mathcal{I}[u(x,t=0;\theta)] - h(x)$$

compute loss and train

$$J_N(u(,;\theta)) = \frac{1}{N_r} r^2(x_\Omega^{(i)}, t^{(i)};\theta) + \gamma_1 \frac{1}{N_b} b^2(x_{\partial\Omega}^{(i)}, t^{(i)};\theta) + \gamma_2 \frac{1}{N_r} l^2(x_\Omega^{(i)};\theta).$$

The construction of NN

$$\mathcal{L}[u(x,t)] = s(x,t) \quad \forall (x,t) \in \Omega \times [0,T]$$

$$\mathcal{B}[u(x,t)] = g(x,t) \quad \forall (x,t) \in \partial\Omega \times [0,T]$$

$$\mathcal{I}[u(x,t=0)] = h(x) \quad \forall x \in \Omega$$

$\mathcal{L}$: partial differential operator, e.g., Laplacian

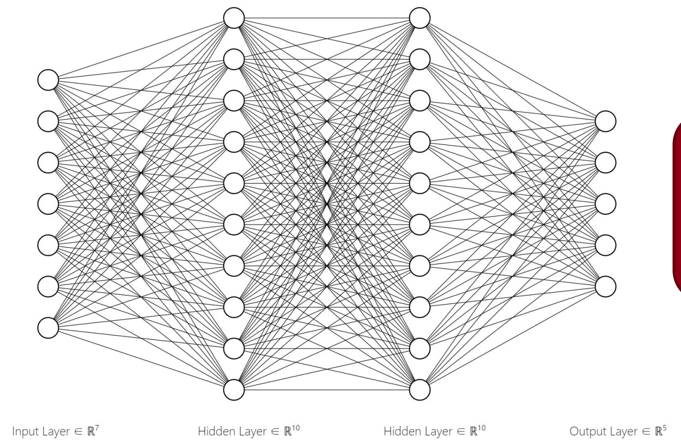
$\mathcal{B}$: boundary operator, e.g., Dirichlet boundary

$\mathcal{I}$: initial operator

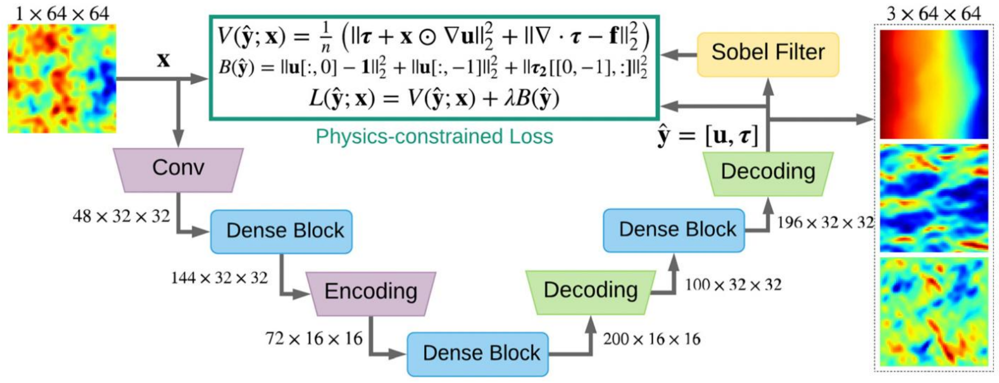
Ω : computational domain, e.g., [-1,1]

$\partial\Omega$: computational domain, e.g., {-1,1}

NN approximation $u(x,t;\theta) \to u(x,t)$



fully connected NN, mesh free

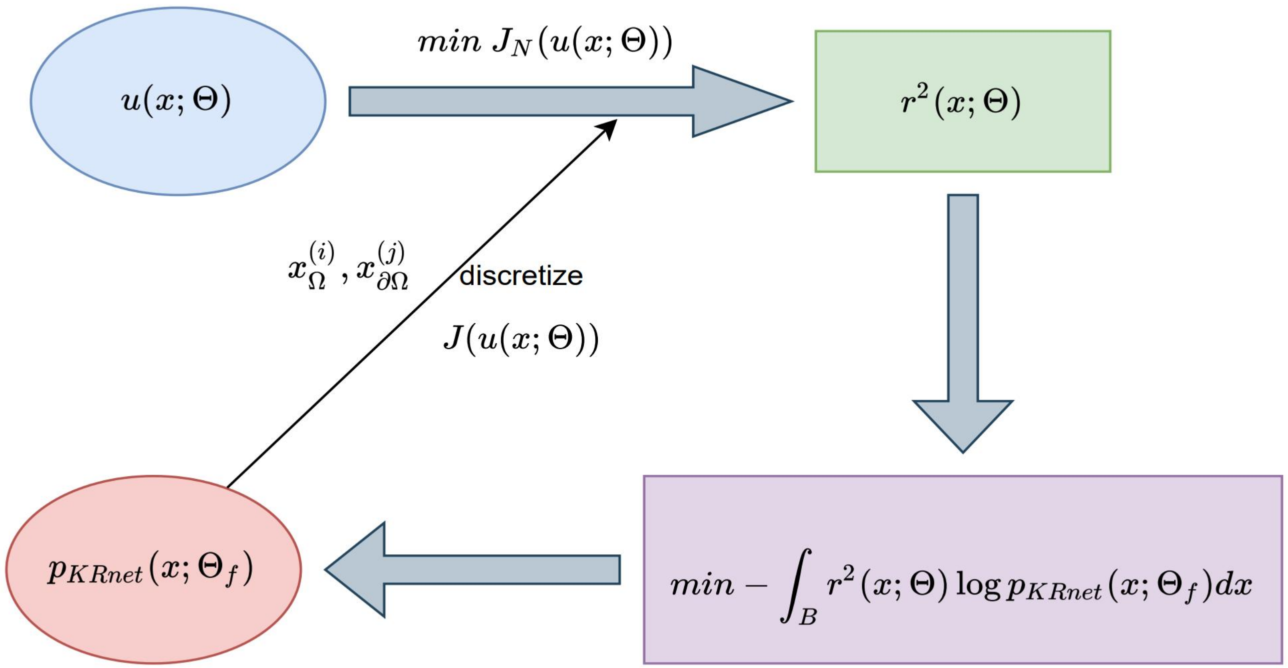


CNN based model if using structure mesh

The sampling step

Sampling Methods

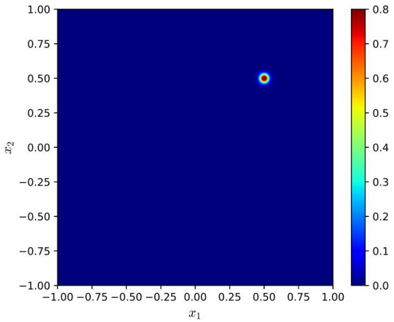
- Uniform sampling
- Random sampling
- Importance sampling
- Quasi random sampling
- Deep adaptive sampling (DAS)[1]



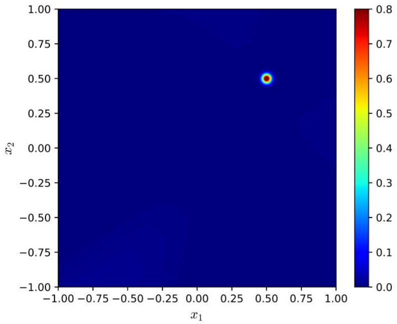
The framework of DAS

Case: Two-dimensional peak problem

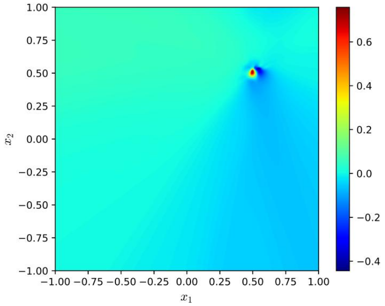
$$
\begin{aligned}
-\Delta u(x_1, x_2) &= s(x_1, x_2) \quad \text{in } \Omega, \\
u(x_1, x_2) &= g(x_1, x_2) \quad \text{on } \partial\Omega,
\end{aligned}
$$



Exact solution



DAS



Uniform sampling

K. Tang, X. Wan and C. Yang, DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations, Journal of Computational Physics, vol: 476, 2023.

The optimization step

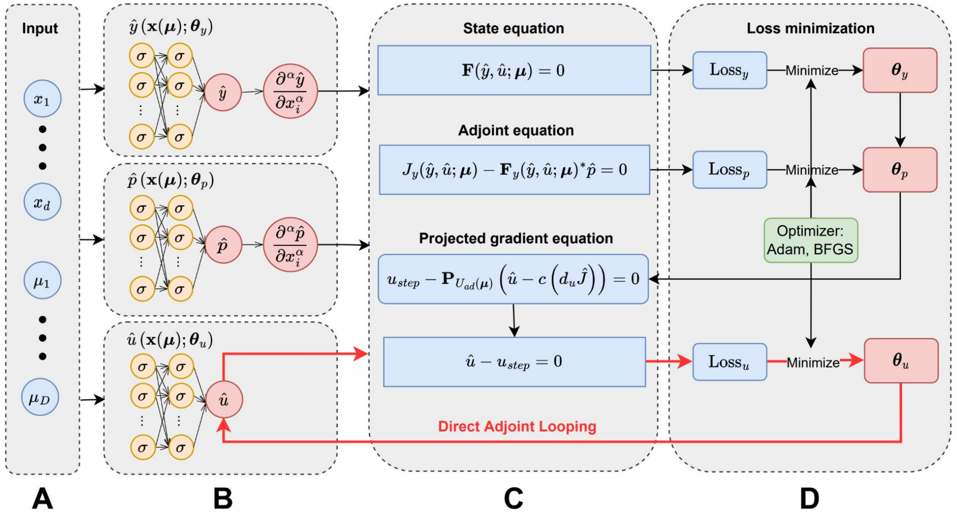
Optimization Methods

- Adam (popular)
- RMSProp
- Vanilla SGD

} first order method

- (L) BFGS — second order method

AONN for parametric optimal contorl problems

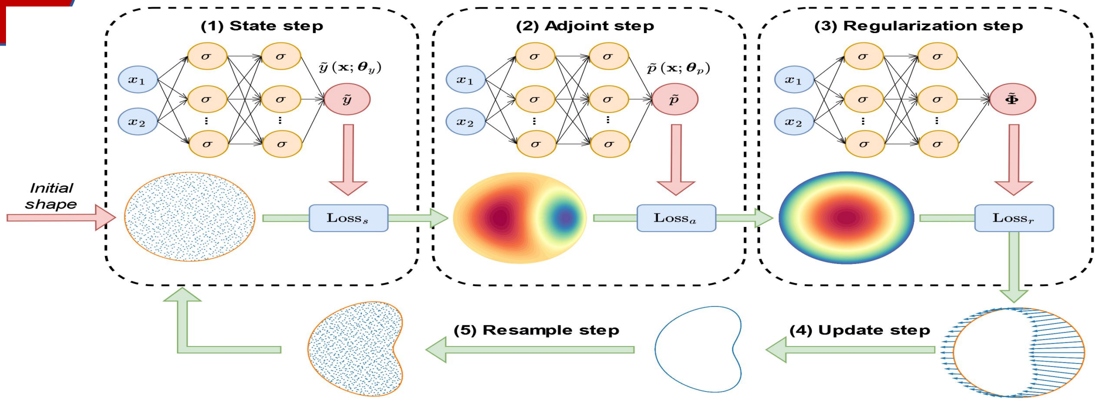


Key to PDE constrained optimal control problem, shape optimazation problem

first order methods:
- ✓ High dimensional
- ✓ Fast
- ✓ Rough

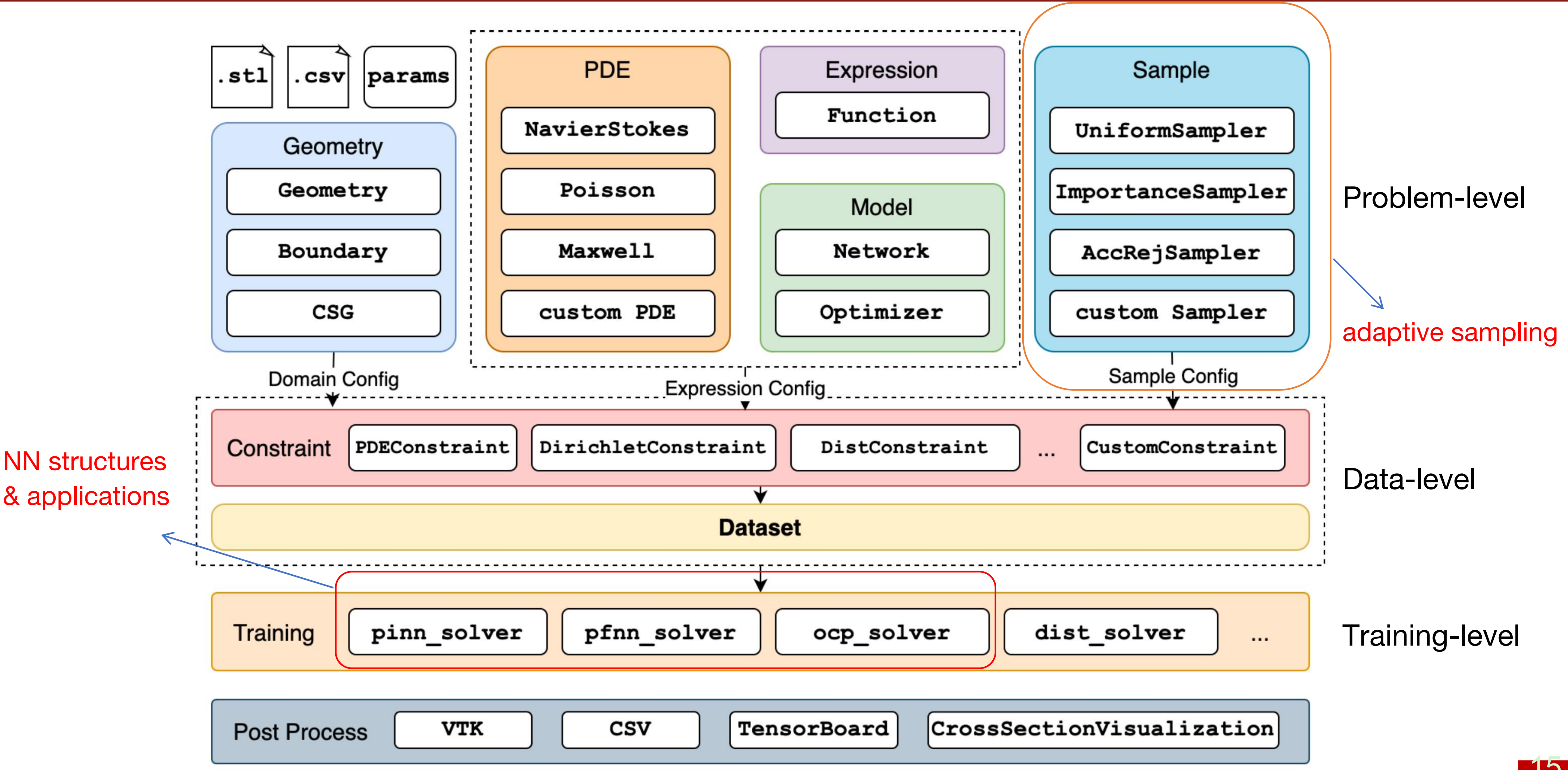
second order methods:
- ✓ Low dimensional
- ✓ Cost
- ✓ Accurate

AONN-2 for shape optimization



P. Yin, G Xiao, K. Tang, and C. Yang, AONN: An adjoint-oriented neural network method for all-at-once solutions of parametric optimal control problems, SIAM Journal on Scientific Computing, accepted, 2023.
X. Wang*, P. Yin*, B. Zhang, and C. Yang, AONN-2: An adjoint-oriented neural network method for PDE-constrained shape optimization, submitted, 2023

Software Architecture with Modular Design



Content

Constraints

- Original (classical) form of PDEs
- **Weak (variational) form of PDEs**
  - Ritz
  - Galerkin
- Length factor: **Penalty-free**

PINN ⟷ least square FEM

Deep Ritz ⟷ Ritz method

PFNN ⟷ Weak form with penalty free

Not all PDEs have a Ritz form.

$$\mathcal{L}(\mathrm{x}; u(\mathrm{x})) = s(\mathrm{x}) \qquad \forall \mathrm{x} \in \Omega,$$
$$\mathfrak{b}(\mathrm{x}; u(\mathrm{x})) = g(\mathrm{x}) \qquad \forall \mathrm{x} \in \partial\Omega.$$

Example $\mathcal{L} = -\Delta$

PINN

$$J(u(x;\Theta)) = \|r(x;\Theta)\|^2_{2,\Omega} + \|b(x;\Theta)\|^2_{2,\partial\Omega},$$

where $r(x;\Theta) = \mathcal{L}u(x;\Theta) - s(x)$, and $b(x;\Theta) = \mathfrak{b}u(x;\Theta) - g(x)$

Deep Ritz

$$\min_{u \in H} I(u)$$

where

$$I(u) = \int_\Omega \left(\frac{1}{2} |\nabla u(x)|^2 - f(x)u(x) \right) dx$$

PFNN

$$w_{\boldsymbol{\theta}}(\boldsymbol{x}) = g_{\boldsymbol{\theta}_1}(\boldsymbol{x}) + \ell(\boldsymbol{x}) f_{\boldsymbol{\theta}_2}(\boldsymbol{x}), \qquad \begin{cases} \ell(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Gamma_D, \\ \ell(\boldsymbol{x}) > 0, & \text{otherwise.} \end{cases}$$

Convert PDE to an optimization problem

use steady-state equations to illustrate the idea

$$\mathcal{L}\left(x; u\left(x\right)\right) = s(x) \qquad \forall\left(x\right) \in \Omega,$$
$$\mathfrak{b}\left(x; u\left(x\right)\right) = g(x) \qquad \forall\left(x\right) \in \partial\Omega.$$

$\mathcal{L}$: partial differential operator, $\mathfrak{b}$: boundary operator.

How deep methods do: a deep net $u(\mathbf{x};\Theta) \to u(\mathbf{x})$

$$J(u(\mathbf{x};\Theta)) = \|r(\mathbf{x};\Theta)\|_{2,\Omega}^2 + \gamma\|b(\mathbf{x};\Theta)\|_{2,\partial\Omega}^2,$$

where $r(\mathbf{x};\Theta) = \mathcal{L}u(\mathbf{x};\Theta) - s(\mathbf{x})$, $b(\mathbf{x};\Theta) = \mathfrak{b}u(\mathbf{x};\Theta) - g(\mathbf{x})$, and

$$\|r(\mathbf{x};\Theta)\|_{2,\Omega}^2 = \int_\Omega r^2(\mathbf{x};\Theta)d\mathbf{x}$$

An optimization problem: $\min\limits_{\Theta} J(u(\mathbf{x};\Theta))$

The penalty term brings the difficulty

Consider the following boundary-value problem:

$$\begin{cases} -\nabla \cdot \Big(\rho(|\nabla u|)\nabla u\Big) + h(u) = 0, & \text{in } \Omega \subset \mathbb{R}^d, \\ u = \varphi, & \text{on } \Gamma_D, \\ \Big(\rho(|\nabla u|)\nabla u\Big) \cdot \boldsymbol{n} = \psi, & \text{on } \Gamma_N, \end{cases}$$

where $\boldsymbol{n}$ is the outward unit normal, $\Gamma_D \cup \Gamma_N = \partial\Omega$ and $\Gamma_D \cap \Gamma_N = \varnothing$.

$$w_{\boldsymbol{\theta}}(\boldsymbol{x}) = g_{\boldsymbol{\theta}_1}(\boldsymbol{x}) + \ell(\boldsymbol{x}) f_{\boldsymbol{\theta}_2}(\boldsymbol{x}),$$

The structure of penalty free methods

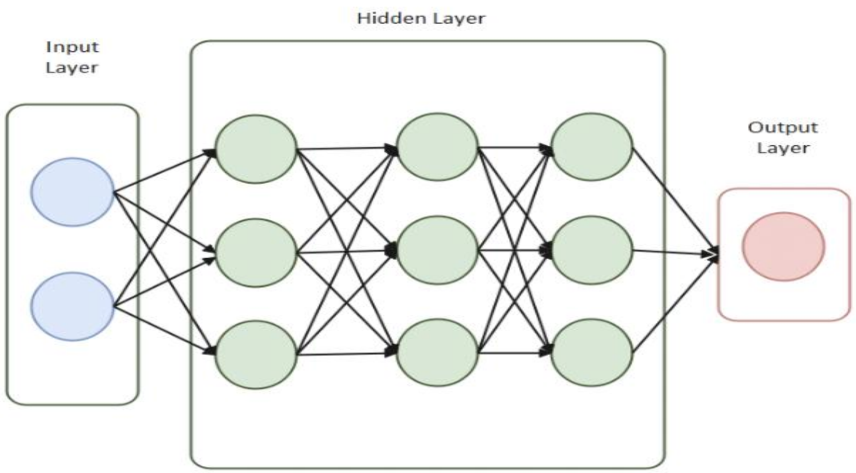
g_{θ_1} — for essential boundary conditions can be pretrained

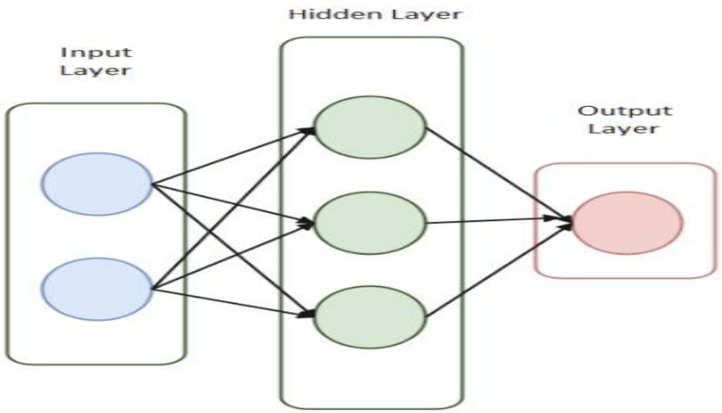
independent

f_{θ_2} — for the other parts

H. Sheng, C. Yang, PFNN: A penalty-free neural network method for solving a class of second order boundary-value problems on complex geometries, J. Comput. Phys. 428 (2021) 110085.

Penalty free methods





f_{θ_2} Two neural networks instead of one g_{θ_1}

$$w_{\boldsymbol{\theta}}(\boldsymbol{x}) = g_{\boldsymbol{\theta}_1}(\boldsymbol{x}) + \ell(\boldsymbol{x}) f_{\boldsymbol{\theta}_2}(\boldsymbol{x}),$$

$$\begin{cases} \ell(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \Gamma_D, \\ \ell(\boldsymbol{x}) > 0, & \text{otherwise.} \end{cases}$$

$\ell(\boldsymbol{x})$ can also be the distance function

$$\begin{cases} l_k(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \gamma_k, \\ l_k(\boldsymbol{x}) = 1, & \boldsymbol{x} \in \gamma_{k_o}, \\ 0 < l_k(\boldsymbol{x}) < 1, & \text{otherwise} \end{cases} \qquad l_k(\boldsymbol{x}) = \sum_{i=1}^{m_k} a_i \phi(\boldsymbol{x}; \widehat{\boldsymbol{x}}^{k,i}) + \boldsymbol{b} \cdot \boldsymbol{x} + c,$$

$$\phi(\boldsymbol{x}; \widehat{\boldsymbol{x}}) = \left(e^2 + \|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|^2\right)^{-1/2}$$

$\ell(\boldsymbol{x})$

$$I[w] := \int_{\Omega} (P(w) + H(w))\, d\boldsymbol{x} - \int_{\Gamma_N} \psi w d\boldsymbol{x},$$

where

$$P(w) := \int_0^{|\nabla w|} \rho(s) s ds \quad \text{and} \quad H(w) := \int_0^{w} h(s) ds.$$

$$u^* = \arg\min_{w \in \mathcal{H}} \Psi[w],$$

where

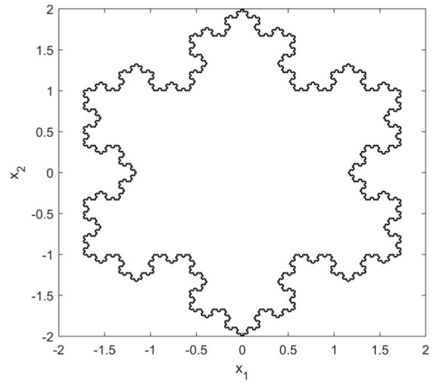
$$\Psi[w] := \frac{|\Omega|}{\#S(\Omega)} \sum_{\boldsymbol{x}^i \in S(\Omega)} \left(P(w(\boldsymbol{x}^i)) + H(w(\boldsymbol{x}^i))\right) - \frac{|\Gamma_N|}{\#S(\Gamma_N)} \sum_{\boldsymbol{x}^i \in S(\Gamma_N)} \psi(\boldsymbol{x}^i) w(\boldsymbol{x}^i)$$

Plugging $w_{\boldsymbol{\theta}}(\boldsymbol{x}) = g_{\boldsymbol{\theta}_1}(\boldsymbol{x}) + \ell(\boldsymbol{x}) f_{\boldsymbol{\theta}_2}(\boldsymbol{x}),$ into the above loss function

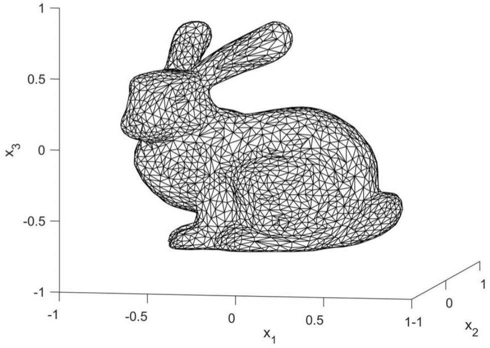
Penalty free methods: results

Method	Deep Ritz		Deep Nitsche		PFNN
Unknowns	811		811		742
$L=5$	$\beta=100$	0.454%±0.072%	$\beta=100$	0.535%±0.052%	**0.288%±0.030%**
	$\beta=300$	1.763%±0.675%	$\beta=300$	1.164%±0.228%	
	$\beta=500$	5.245%±1.943%	$\beta=500$	3.092%±1.256%	
$L=6$	$\beta=100$	0.747%±0.101%	$\beta=100$	0.483%±0.095%	**0.309%±0.064%**
	$\beta=300$	3.368%±0.690%	$\beta=300$	0.784%±0.167%	
	$\beta=500$	4.027%±1.346%	$\beta=500$	2.387%±0.480%	
$L=7$	$\beta=100$	0.788%±0.041%	$\beta=100$	0.667%±0.149%	**0.313%±0.071%**
	$\beta=300$	2.716%±0.489%	$\beta=300$	1.527%±0.435%	
	$\beta=500$	4.652%±1.624%	$\beta=500$	1.875%±0.653%	

Method		Deep Ritz		Deep Nitsche		PFNN
Unknowns		821		821		762
$p=1.2$	$\lambda=0.6$	$\beta=100$	0.612%±0.213%	$\beta=100$	0.659%±0.129%	**0.513%±0.116%**
		$\beta=300$	0.540%±0.153%	$\beta=300$	0.593%±0.136%	
		$\beta=500$	0.560%±0.218%	$\beta=500$	0.563%±0.122%	
	$\lambda=1.2$	$\beta=100$	0.555%±0.120%	$\beta=100$	0.643%±0.135%	**0.489%±0.121%**
		$\beta=300$	0.513%±0.085%	$\beta=300$	0.608%±0.109%	
		$\beta=500$	0.532%±0.159%	$\beta=500$	0.584%±0.098%	
$p=4.0$	$\lambda=0.6$	$\beta=100$	27.646%±0.310%	$\beta=100$	28.548%±2.849%	**0.699%±0.467%**
		$\beta=300$	16.327%±0.294%	$\beta=300$	21.236%±1.326%	
		$\beta=500$	12.034%±0.538%	$\beta=500$	17.972%±2.020%	
	$\lambda=1.2$	$\beta=100$	25.133%±0.823%	$\beta=100$	30.375%±2.387%	**0.722%±0.393%**
		$\beta=300$	16.330%±0.484%	$\beta=300$	21.938%±2.369%	
		$\beta=500$	11.573%±0.458%	$\beta=500$	18.998%±1.872%	



(a) Koch Snowflake ($L=5$)



(b) Stanford Bunny

$$-\nabla\cdot\left(\frac{\nabla u}{\sqrt{1+|\nabla u|^2}}\right)=0,$$

Minimal surface equation on a Koch snowflake

$$-\nabla\cdot(|\nabla u|^{p-2}\nabla u)-\lambda\exp(u)+c=0,$$

p-Liouville-Bratu equation on the Stanford Bunny

Convert PDE to an optimization problem

use steady-state equations to illustrate the idea

$$\begin{aligned} \mathcal{L}\left(x; u\left(x\right)\right) &= s(x) \qquad \forall \left(x\right) \in \Omega, \\ \mathfrak{b}\left(x; u\left(x\right)\right) &= g(x) \qquad \forall \left(x\right) \in \partial\Omega. \end{aligned}$$

$\mathcal{L}$: partial differential operator, $\mathfrak{b}$: boundary operator.

How deep methods do: a deep net $u(\mathbf{x};\Theta) \to u(\mathbf{x})$

$$J(u(\mathbf{x};\Theta)) = \|r(\mathbf{x};\Theta)\|_{2,\Omega}^2 + \gamma \|b(\mathbf{x};\Theta)\|_{2,\partial\Omega}^2,$$

where $r(\mathbf{x};\Theta) = \mathcal{L}u(\mathbf{x};\Theta) - s(\mathbf{x})$, $b(\mathbf{x};\Theta) = \mathfrak{b}u(\mathbf{x};\Theta) - g(\mathbf{x})$, and

$$\|r(\mathbf{x};\Theta)\|_{2,\Omega}^2 = \int_\Omega r^2(\mathbf{x};\Theta)d\mathbf{x}$$

An optimization problem: $\min J(u(\mathbf{x};\Theta))$

Key point: $\min_\Theta J(u(\mathbf{x};\Theta)) \to \min_\Theta J_N(u(\mathbf{x};\Theta))$ discretize the loss by uniform sampling in general (or other quasi-random methods based on uniform samples)

Look at errors

$$u(\mathbf{x};\Theta^*) = \arg\min_{\Theta} J(u(\mathbf{x};\Theta)),$$

$$u(\mathbf{x};\Theta_N^*) = \arg\min_{\Theta} J_N(u(\mathbf{x};\Theta)).$$

$$\mathbb{E}\left(\|u(\mathbf{x};\Theta_N^*) - u(\mathbf{x})\|_\Omega\right) \le \underbrace{\mathbb{E}\left(\|u(\mathbf{x},\Theta_N^*) - u(\mathbf{x};\Theta^*)\|_\Omega\right)}_{\text{statistical error}} + \underbrace{\|u(\mathbf{x};\Theta^*) - u(\mathbf{x})\|_\Omega}_{\text{approximation error}}$$

Our work: focus on how to reduce the statistical error

the capability of neural networks → approximation error

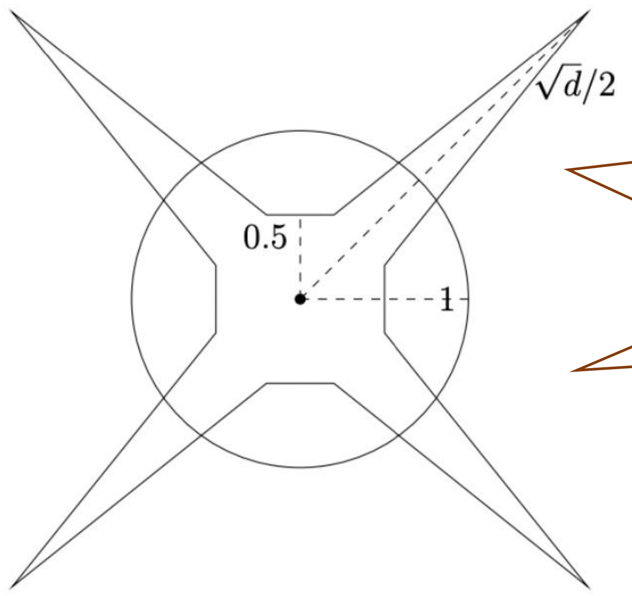
the strategy of loss discretization → statistical error

Key point: how to sample?

北京大学 长沙计算与数字经济研究院
PEKING UNIVERSITY Changsha Institute for Computing and Digital Economy

Geometric properties of high-dimensional spaces

uniformly distributed points in high-dimensional spaces



Question: if the support of the solution is concentrate on the origin, what will happen?

Most of the volume of a high-dimensional cube is located around its corner [Vershynin, High-Dimensional Probability, 2020]. Cube: $[-1, 1]^d$

$$\mathbb{P}(\|\mathbf{x}\|_2^2 \leq 1) \leq \exp(-\frac{d}{10}).$$

Some observations

Question: is uniform sampling optimal for deep methods?

FEM → Adaptive FEM

mesh → refine mesh in a certain type of adaptivity

Observation:

1. uniform mesh is not optimal for FEM
2. choosing uniform samples is not a good choice for high-dimensional problems

Deep methods

lack of adaptivity $\to$ develop adaptive schemes

Localized residual

Assume

$$\zeta = \int_\Omega 1_I(\mathbf{x})d\mathbf{x} \approx \int_\Omega r^2(\mathbf{x})d\mathbf{x} \ll 1.$$

A rare event!

Consider a Monte Carlo estimator of ζ in terms of uniform samples

$$\hat{P}_{\mathrm{MC}} = \frac{1}{N}\sum_{i=1}^{N} 1_I(\mathbf{x}^{(i)}).$$

The relative error of $\hat{P}_{\mathrm{MC}}$ is

$$\frac{\mathrm{Var}^{1/2}(\hat{P}_{\mathrm{MC}})}{\zeta} = N^{-1/2}((1-\zeta)/\zeta)^{1/2} \approx (\zeta N)^{-1/2}.$$

sample size $O(1/\zeta)$ $\rightarrow$ relative error $O(1)$.

choosing uniform samples is not efficient for low reglularity problems

- How does FEM do?

Error estimator

general framework: using an error estimator to refine mesh

- How does deep method do?

???

we need a general framework...

Estimate the residual

$$\int_\Omega r^2(\mathbf{x};\Theta)d\mathbf{x} \approx \frac{1}{N_r}\sum_{i=1}^{N_r} r^2(\mathbf{x}_\Omega^{(i)};\Theta),$$

key point

- reduce the variance of r^2

$$J_r(u(\mathbf{x};\Theta)) = \int_\Omega r^2(\mathbf{x};\Theta)d\mathbf{x} = \int_\Omega \frac{r^2(\mathbf{x};\Theta)}{p(\mathbf{x})}p(\mathbf{x})d\mathbf{x} \approx \frac{1}{N_r}\sum_{i=1}^{N_r}\frac{r^2(\mathbf{x}_\Omega^{(i)};\Theta)}{p(\mathbf{x}_\Omega^{(i)})},$$

where $\{\mathbf{x}_\Omega^{(i)}\}_{i=1}^{N_r}$ from $p(\mathbf{x})$ instead of a uniform distribution.

Importance sampling

$$p^* = \frac{r^2(\mathbf{x};\Theta)}{\mu},\quad \mu = \int_\Omega r^2(\mathbf{x};\Theta)d\mathbf{x}$$

Deep adaptive sampling method (DAS)

Sample from $p(\mathbf{x})$ for a fixed Θ: a deep generative model

$$p_{KRnet}(\mathbf{x}; \Theta_f) \approx \mu^{-1} r^2(\mathbf{x}; \Theta)$$

where $p_{KRnet}(\mathbf{x}; \Theta_f)$ is a PDF induced by KRnet [Tang, Wan and Liao, 2020]; [Tang, Wan and Liao, 2021]

"Error estimator": $\hat{r}_X(\mathbf{x}) \propto r^2(\mathbf{x}; \Theta)$

$$D_{KL}(\hat{r}_X(\mathbf{x}) \| p_{KRnet}(\mathbf{x}; \Theta_f)) = \int_B \hat{r}_X \log \hat{r}_X d\mathbf{x} - \int_B \hat{r}_X \log p_{KRnet} d\mathbf{x}.$$

$$\min_{\Theta_f} H(\hat{r}_X, p_{KRnet}) = -\int_B \hat{r}_X \log p_{KRnet} d\mathbf{x}.$$

Challenge

- design a valid PDF model for efficient sampling

A deep generative model for sampling

KRnet: construct a PDF model via Knothe-Rosenblatt rearrangement, [Tang, Wan and Liao, 2021]

$$\mathbf{z} = f_{KRnet}(\mathbf{x}) = L_N \circ f_{[K-1]}^{\text{outer}} \circ \cdots \circ f_{[1]}^{\text{outer}}(\mathbf{x}),$$

$$p_{KRnet}(\mathbf{x}) = p_{\mathbf{Z}}(f_{KRnet}(\mathbf{x})) \left| \det \nabla_{\mathbf{x}} f_{KRnet} \right|,$$

where $f_{[i]}^{\text{outer}}$ is defined as

$$f_{[k]}^{\text{outer}} = L_S \circ f_{[k,L]}^{\text{inner}} \circ \cdots \circ f_{[k,1]}^{\text{inner}} \circ L_R.$$

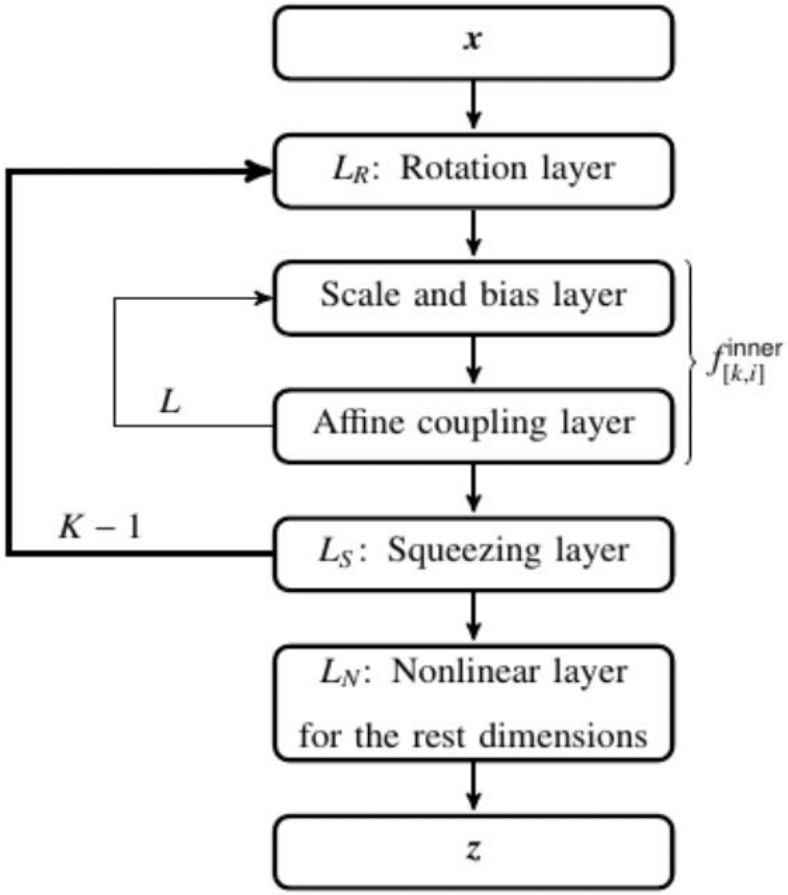
Advantages

- GAN and VAE can not provide an explicit PDF though they can generate samples efficiently
- KRnet provides an explicit PDF
- KRnet can generate samples efficiently

A deep generative model for sampling

structure of KRnet

- squeezing layer
- rotation layer
- affine coupling layer
- nonlinear layer



The framework of DAS (see [Tang, Wan and Yang, 2022] for more details)
1

// solve PDE
Sample m samples $\mathbf{x}_{\Omega,k}^{(i)}$ and Sample m samples $\mathbf{x}_{\partial\Omega,k}^{(j)}$.
Update $u(\mathbf{x};\Theta)$ by descending the stochastic gradient of $J_N(u(\mathbf{x};\Theta))$.
// Train KRnet
Sample m samples from $\mathbf{x}_{\Omega,k}^{(i)}$.
Update $p_{KRnet}(\mathbf{x};\Theta_f)$ by descending the stochastic gradient of $H(\hat{r}_X, \hat{p}_{KRnet})$.
// Refine training set (replace all points: DAS-R; the number of points increases gradually: DAS-G)
Generate $\mathbf{x}_{\Omega,k+1}^{(i)} \subset \Omega$ through $p_{KRnet}(\mathbf{x};\Theta_f^{*,(k+1)})$.
Repeat until stopping criterion satisfies

Theorem (Tang, Wan and Yang, 2022)

Let $u(\mathbf{x};\Theta_N^{*,(k)}) \in F$ be a solution of DAS at the k-stage where the collocation points are independently drawn from $\hat{p}_{KRnet}(\mathbf{x};\Theta_f^{*,(k-1)})$. Given $0<\varepsilon<1$, the following error estimate holds under certain conditions

$$\left\| u(\mathbf{x};\Theta_N^{*,(k)}) - u(\mathbf{x}) \right\|_{2,\Omega} \leq \sqrt{2}C_1^{-1}\left(R_k + \varepsilon + \left\| b(\mathbf{x};\Theta_N^{*,(k)}) \right\|_{2,\partial\Omega}^2 \right)^{\frac{1}{2}}.$$

with probability at least $1-\exp(-2N_r\varepsilon^2/(\tau_2-\tau_1)^2)$.

Corollary (Tang, Wan and Yang, 2022)

If the boundary loss $J_b(u)$ is zero, then the following inequality holds

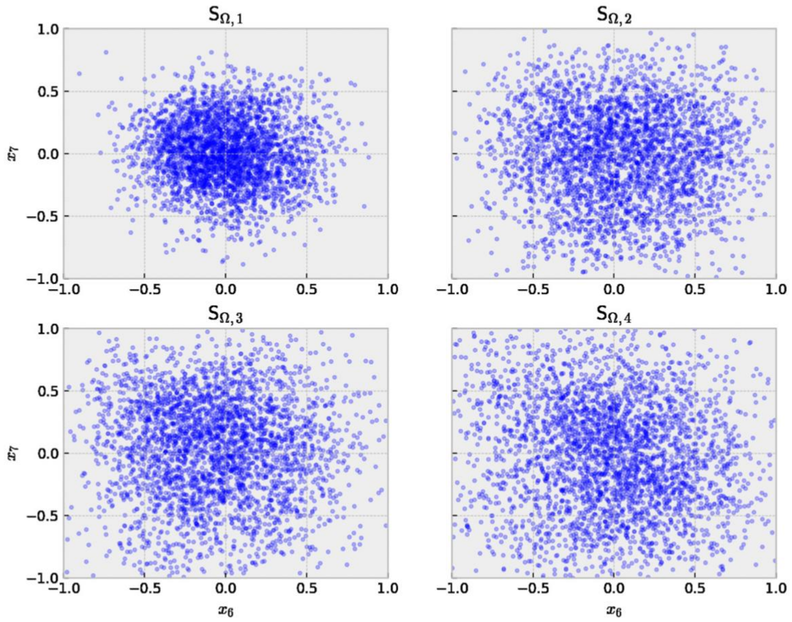
$$\mathbb{E}(R_{k+1}) \leq \mathbb{E}(R_k)$$

Some results of DAS

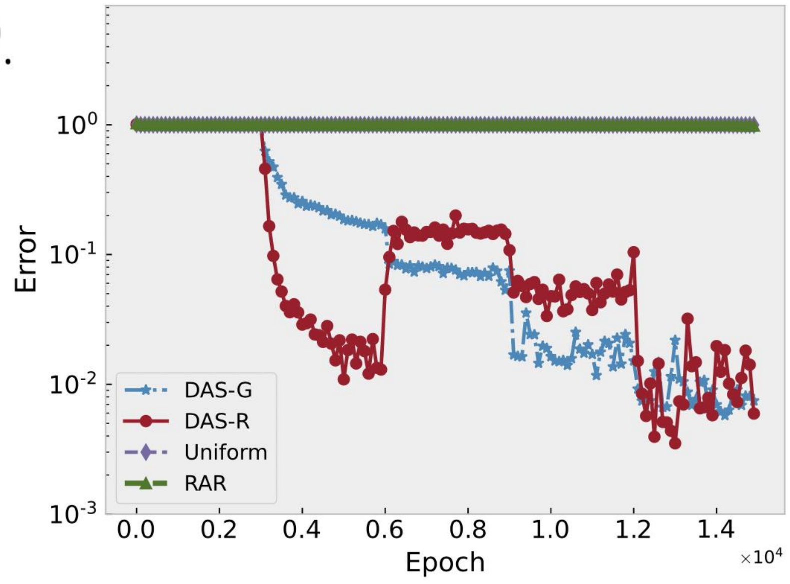
$$-\Delta u(\boldsymbol{x}) + u(\boldsymbol{x}) - u^3(\boldsymbol{x}) = s(\boldsymbol{x}), \quad \boldsymbol{x} \text{ in } \Omega = [-1,1]^{10}.$$

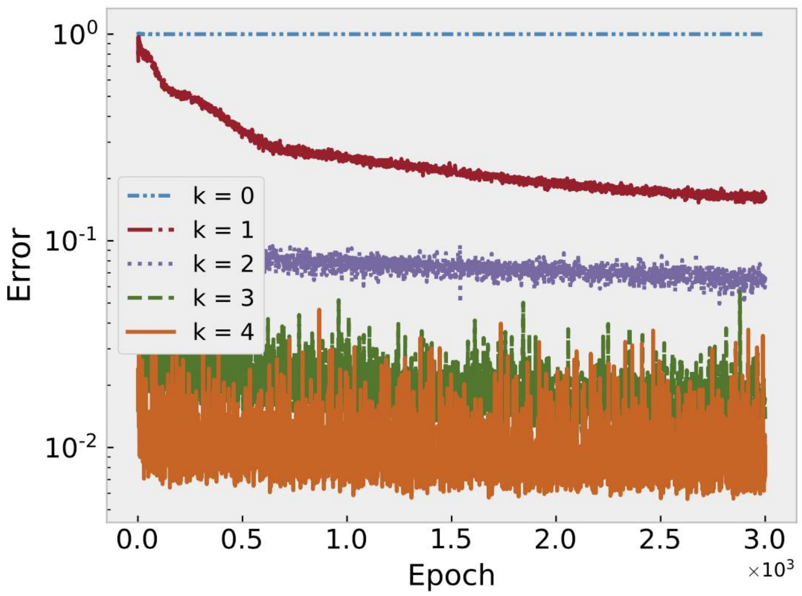
with an exact solution

$$u(\boldsymbol{x}) = e^{-10\|\boldsymbol{x}\|_2^2},$$



The evolution of samples





Training time and error for different $|S_\Omega|$ and sampling strategies, ten-dimensional nonlinear test problem.

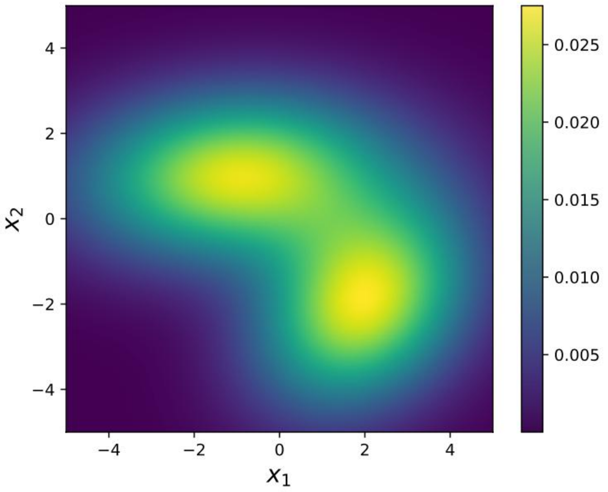
sampling strategy / $\vert S_\Omega \vert$	DAS-G		DAS-R		Uniform		RAR	
	time	error	time	error	time	error	time	error
5×10^4	1.82 h	0.042	3.44 h	0.062	1.84 h	1.008	1.42 h	0.999
10^5	3.65 h	0.020	6.92 h	0.054	3.86 h	1.001	2.97 h	1.002
1.5×10^5	5.81 h	0.010	10.41 h	0.037	5.73 h	1.002	4.63 h	0.993
2×10^5	7.82 h	0.009	13.87 h	0.013	7.80 h	0.996	5.75 h	0.983

The comparison of different sampling strategies

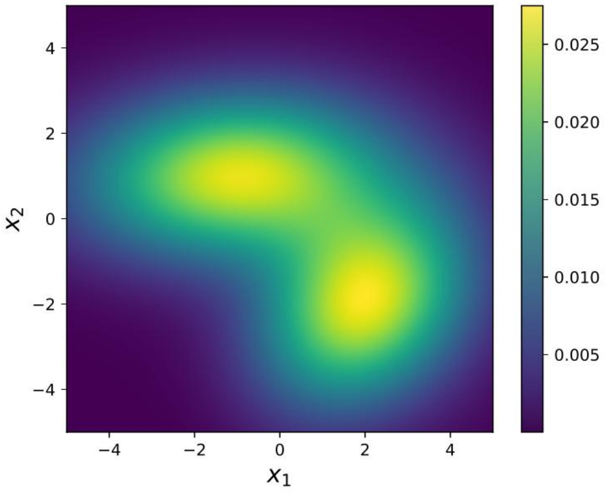
A special case: Fokker-Planck equations

setting

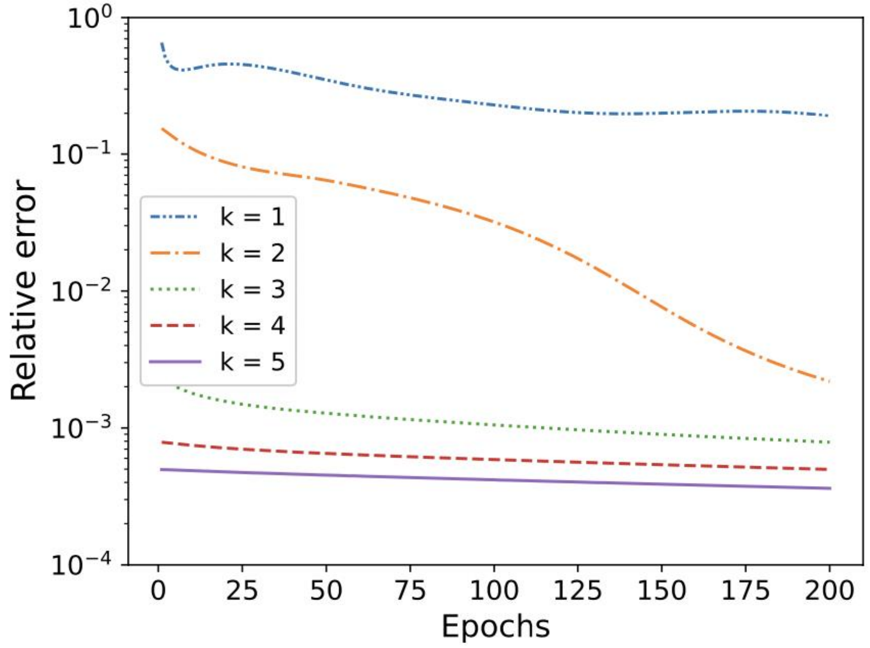
- $\frac{\partial p(x,t)}{\partial t} = \nabla \cdot [p(x,t)\nabla \log(\beta_1 p_1(x) + \beta_2 p_2(x))] + \nabla^2 p(x,t)$
- stationary solution
  $p_{st}(x) = \beta_1 p_1(x) + \beta_2 p_2(x), x \in \mathbb{R}^2, p_i(x)$: Gaussian distribution



(e) Exact solution $p(x)$



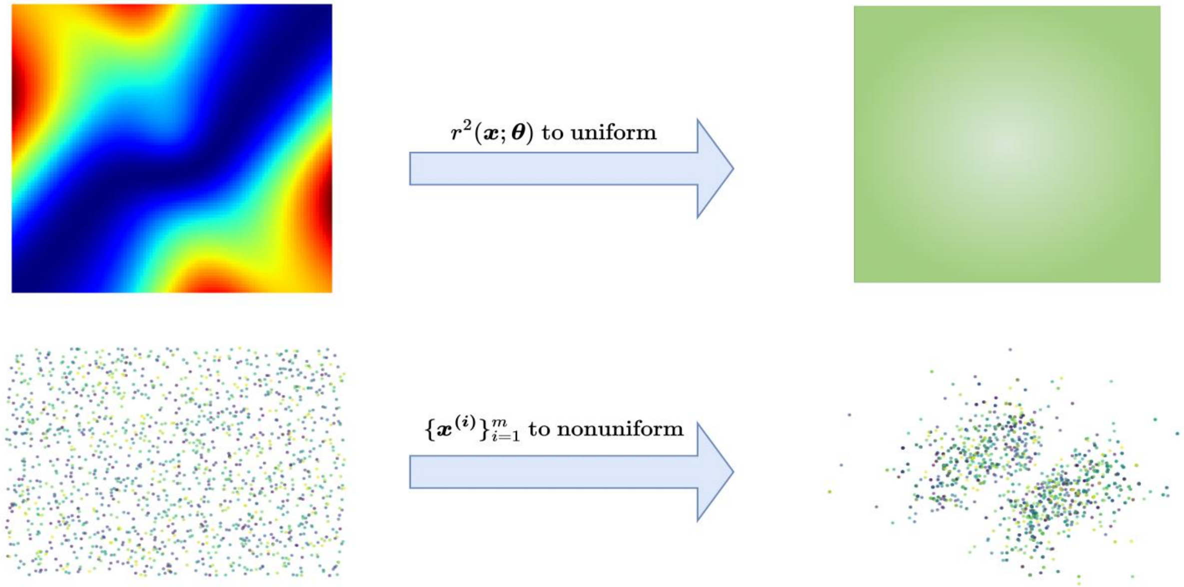
(f) ADDA approximation



A min-max formulation

Two things

- minimize the residual:
- endeavor to maintain a smooth profile of the residual



A min-max formulation

- minimize the residual: $\min_\theta r(\mathbf{x};\theta)$
- maintain a smooth profile of the residual

$$\min_{\theta} \max_{p_\alpha \in V} \mathcal{J}(u_\theta, p_\alpha) = \int_\Omega r^2(\mathbf{x};\theta) p_\alpha(\mathbf{x}) d\mathbf{x},$$

For simplicity, we remove the boundary residual term.

$$\min_{\theta} \max_{p \in V} \mathcal{J}(u_\theta, p) = \int_\Omega r^2(\mathbf{x};\theta) p(\mathbf{x}) d\mathbf{x}.$$

where

$$p_\alpha(\mathbf{x}) = p_{\mathbf{Z}}(f_\alpha(\mathbf{x}))|\nabla_{\mathbf{x}} f_\alpha|.$$

is a flow model.

How can this min-max formulation achieve our goal?

- Optimal transport theory
- Some constraints for V

Wasserstein distance

$$d_{W^M}(\mu, \nu) = \inf_{\pi \in \Pi(\Omega \times \Omega)} \int_{\Omega \times \Omega} d_M(\mathbf{x}, \mathbf{y})\, d\pi(\mathbf{x}, \mathbf{y}),$$

Typically,

$$V := \{p(\boldsymbol{x}) \,|\, \|p\|_{\text{Lip}} \leq 1,\ 0 \leq p(\boldsymbol{x}) \leq M\},$$

where M is a positive number, or

$$\hat{V} = \{p(\boldsymbol{x}) \,|\, \|p\|_{\text{Lip}} \leq 1,\ p(\boldsymbol{x}) \geq 0,\ \int_{\Omega} p(\boldsymbol{x}) d\boldsymbol{x} = 1\}.$$

The min-max formulation

$$\inf_{u} \sup_{p \in \hat{V}} \mathcal{J}(u, p) = \int_{\Omega} r^2(u(\boldsymbol{x}))p(\boldsymbol{x})\, d\boldsymbol{x},$$

The constraint for p is important.
Otherwise, the maximization step will yield a delta measure

$$\delta(\boldsymbol{x} - \boldsymbol{x}_0) = \arg\max_{p>0, \int_{\Omega} p d\boldsymbol{x}=1} \int_{\Omega} r^2(\boldsymbol{x}; \theta)p(\boldsymbol{x})d\boldsymbol{x},$$

where $\boldsymbol{x}_0 = \arg\max_{\boldsymbol{x} \in \Omega} r^2(\boldsymbol{x}; \theta)$.

How this maximization step push the residual-induced distribution to a uniform one?

$$
\begin{aligned}
&\sup_{p\in V}\int_{\Omega} r^2(\boldsymbol{x};\theta)p(\boldsymbol{x})d\boldsymbol{x}\\
=&\sup_{p\in V}\int_{\Omega} r^2(\boldsymbol{x};\theta)p(\boldsymbol{x})d\boldsymbol{x}-\int_{\Omega} r^2(\boldsymbol{x};\theta)d\boldsymbol{x}\int_{\Omega} p(\boldsymbol{x})d\boldsymbol{x}+\int_{\Omega} r^2(\boldsymbol{x};\theta)d\boldsymbol{x}\int_{\Omega} p(\boldsymbol{x})d\boldsymbol{x}\\
\leq&\int_{\Omega} r^2(\boldsymbol{x};\theta)d\boldsymbol{x}\left(\sup_{p\in V}\left[\int_{\Omega} p(\boldsymbol{x})d\mu_r-\int_{\Omega} p(\boldsymbol{x})d\mu_u\right]+\sup_{p\in V}\int_{\Omega} p(\boldsymbol{x})d\boldsymbol{x}\right)\\
\leq&(d_{WM}(\mu_r,\mu_u)+M)\int_{\Omega} r^2(\boldsymbol{x};\theta)\,d\boldsymbol{x},
\end{aligned}
$$

μ_u is a uniform distribution.

Theorem

Under certain conditions, $\lim_{n\to\infty} \mathcal{J}(u_n, p_n) = 0$*, for some sequence of functions* $\{p_n\}_{n=1}^{\infty}$ *satisfying the constraints defined in the min-max formulation. Meanwhile, this optimization sequence has the following two properties:*

1. *The residual sequence* $\{r(u_n)\}_{n=1}^{\infty}$ *of* $\{u_n\}_{n=1}^{\infty}$ *converges to 0 in* $L^2(d\mu)$.
2. *The renormalized squared residual distributions*

$$d\nu_n \triangleq \frac{r^2(u_n)}{\int_\Omega r^2(u_n(\mathbf{x}))\, d\mathbf{x}}\, d\mu(\mathbf{x})$$

converge to the uniform distribution μ *in the Wasserstein distance* d_{W^M}.

K. Tang,* J. Zhai*, X. Wan and C. Yang, Adversarial Adaptive Sampling: Unify PINN and Optimal Transport for the Approximation of PDEs preprint, 2023.

How can we implement the min-max optimization problem?

- the minimization step is straightforward
- the maximization step is not trival because of the constraints

A formulation for practical implementation

$$\min_{\theta} \max_{\substack{p_\alpha > 0, \\ \int_\Omega p_\alpha(\mathbf{x}) d\mathbf{x} = 1}} \mathcal{J}(u_\theta, p_\alpha) = \int_\Omega r^2(\mathbf{x}; \theta) p_\alpha(\mathbf{x}) d\mathbf{x} - \beta \int_\Omega |\nabla_{\mathbf{x}} p_\alpha(\mathbf{x})|^2 d\mathbf{x},$$

This formulation makes that p is well-posed

$$\begin{cases} 2\beta \nabla^2 p^* + r^2(\mathbf{x}; \theta) - \frac{1}{|\Omega|} \int_\Omega r^2(\mathbf{x}; \theta) d\mathbf{x} = 0, & \mathbf{x} \in \Omega, \\ \frac{\partial p^*}{\partial \mathbf{n}} = 0, & \mathbf{x} \in \partial\Omega. \end{cases}$$

- minimize the residual

$$\int_\Omega r^2\left[u_\theta(\boldsymbol{x})\right] p_\alpha(\boldsymbol{x}) d\boldsymbol{x} \approx \frac{1}{m}\sum_{i=1}^{m} r^2\left[u_\theta(\boldsymbol{x}_\alpha^{(i)})\right]$$

- maximization step

$$\mathcal{J}(u_\theta, p_\alpha) \approx \frac{1}{m}\sum_{i=1}^{m} \frac{r^2\left[u_\theta(\boldsymbol{x}_{\alpha'}^{(i)})\right] p_\alpha(\boldsymbol{x}_{\alpha'}^{(i)})}{p_{\alpha'}(\boldsymbol{x}_{\alpha'}^{(i)})} - \beta \cdot \frac{1}{m}\sum_{i=1}^{m} \frac{|\nabla_{\boldsymbol{x}} p_\alpha(\boldsymbol{x}_{\alpha'}^{(i)})|^2}{p_{\alpha'}(\boldsymbol{x}_{\alpha'}^{(i)})}$$

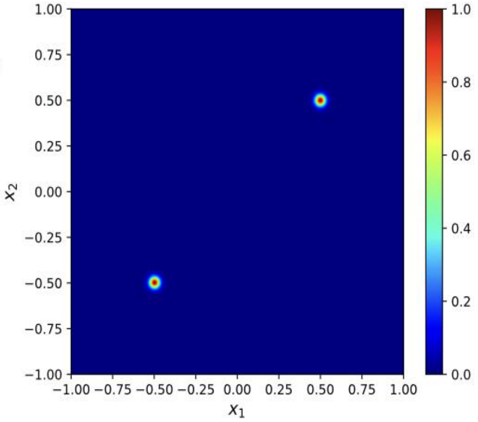
Training style is similar to WGAN

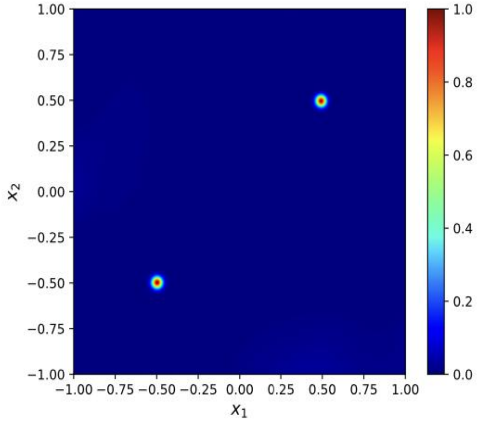
- simultaneously optimize the approximate solution and the random samples

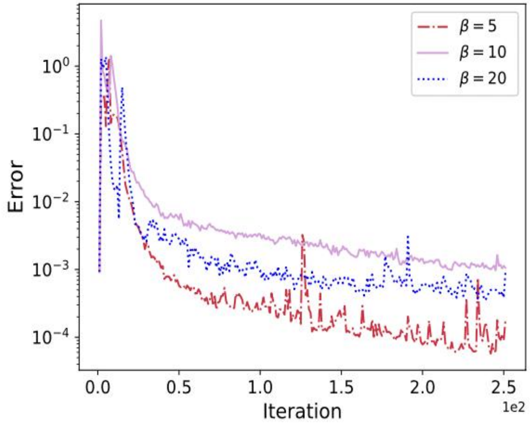
Results of AAS

$$-\nabla \cdot [u(\boldsymbol{x})\nabla v(\boldsymbol{x})] + \nabla^2 u(\boldsymbol{x}) = s(\boldsymbol{x}) \quad \text{in } \Omega,$$
$$u(\boldsymbol{x}) = g(\boldsymbol{x}) \quad \text{on } \partial\Omega,$$

Two peak problem

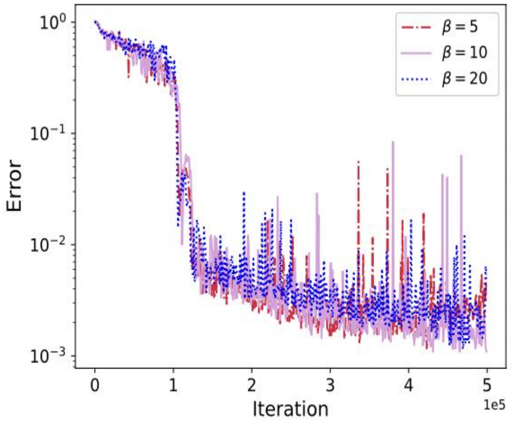


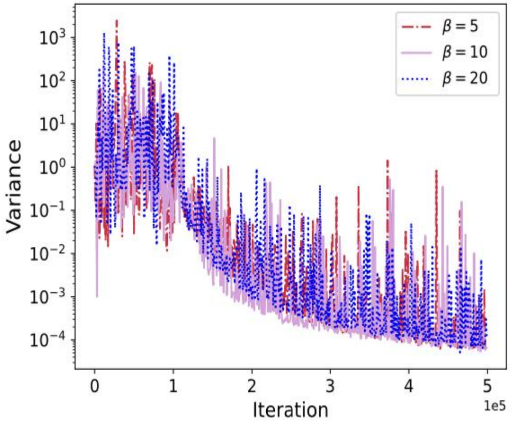


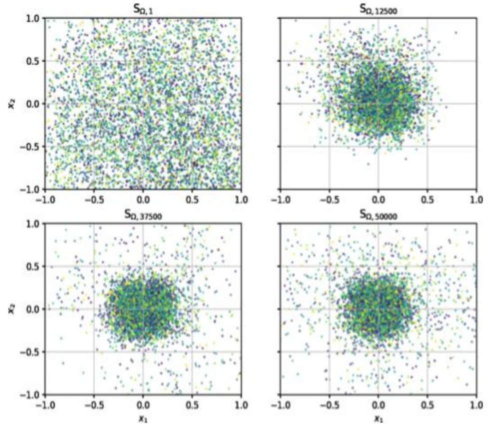


$$-\Delta u(\mathbf{x}) + u(\mathbf{x}) - u^3(\mathbf{x}) = s(\mathbf{x}), \quad \mathbf{x} \text{ in } \Omega = [-1,1]^{10}$$
$$u(\mathbf{x}) = g(\mathbf{x}), \quad \mathbf{x} \text{ on } \partial\Omega.$$

High-dimensional nonlinear problem







Problem setup

OCP($\boldsymbol{\mu}$) Parametric optimal control problem: for any $\boldsymbol{\mu}$, find the solution to

$$\min_{(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}))\in Y\times U} J(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu});\boldsymbol{\mu}),$$
$$\text{s.t.}\quad \mathbf{F}(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu});\boldsymbol{\mu})=0 \ \text{ in } \Omega(\boldsymbol{\mu}), \ \text{ and } \ u(\mathbf{x},\boldsymbol{\mu})\in U_{ad}(\boldsymbol{\mu}),$$

- $\boldsymbol{\mu}\in\mathcal{P}\subset\mathbb{R}^D$: a vector that collects a finite number of parameters
- $\Omega(\boldsymbol{\mu})\subset\mathbb{R}^d$: a spatial domain depending on $\boldsymbol{\mu}$
- $\mathbf{x}\in\Omega(\boldsymbol{\mu})$: a spatial variable
- $J: Y\times U\times\mathcal{P}\mapsto\mathbb{R}$: a parameter-dependent objective functional. Y and U are two proper function spaces defined on $\Omega(\boldsymbol{\mu})$
- $\mathbf{F}$: the governing equation, parameter-dependent PDEs
- $U_{ad}(\boldsymbol{\mu})$: a parameter-dependent bounded closed convex subset of U

OCP($\boldsymbol{\mu}$) Parametric optimal control problem: for any $\boldsymbol{\mu}$, find the solution to

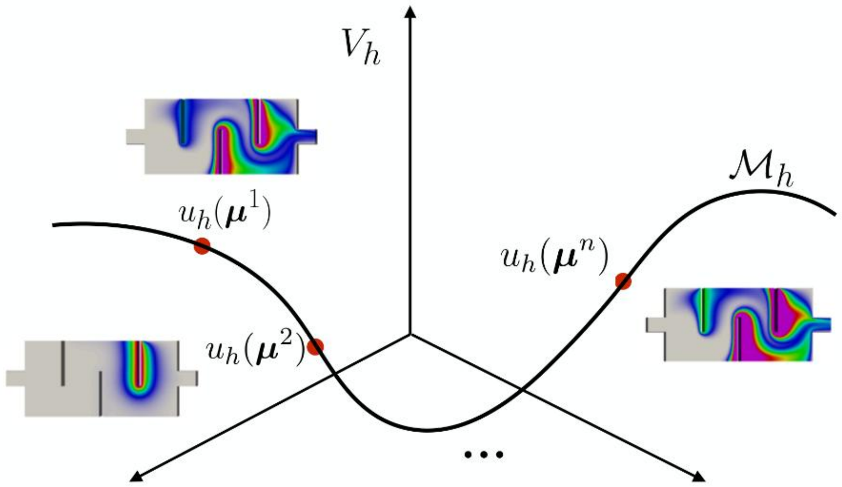
$$\min_{(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}))\in Y\times U} J(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu});\boldsymbol{\mu}),$$
$$\text{s.t.}\quad \mathbf{F}(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu});\boldsymbol{\mu})=0\ \text{ in }\Omega(\boldsymbol{\mu}),\ \text{ and }\ u(\mathbf{x},\boldsymbol{\mu})\in U_{ad}(\boldsymbol{\mu}),$$

- The presence of parameters introduces extra prominent complexity
- Obtaining all-at-once solutions is challenge
- Additional constraints (e.g. box constraints) make NN-based methods hard to train

AONN (adjoint-oriented neural network)

- The AONN methods can efficiently deal with a series of challenging PDE-constrained optimization problems.

Parameter-dependent challenges



μ-dependent solutions form a manifold

- Parameter space discretization.
- Inter-parameter dependence.
- Essentially high-dimensional.

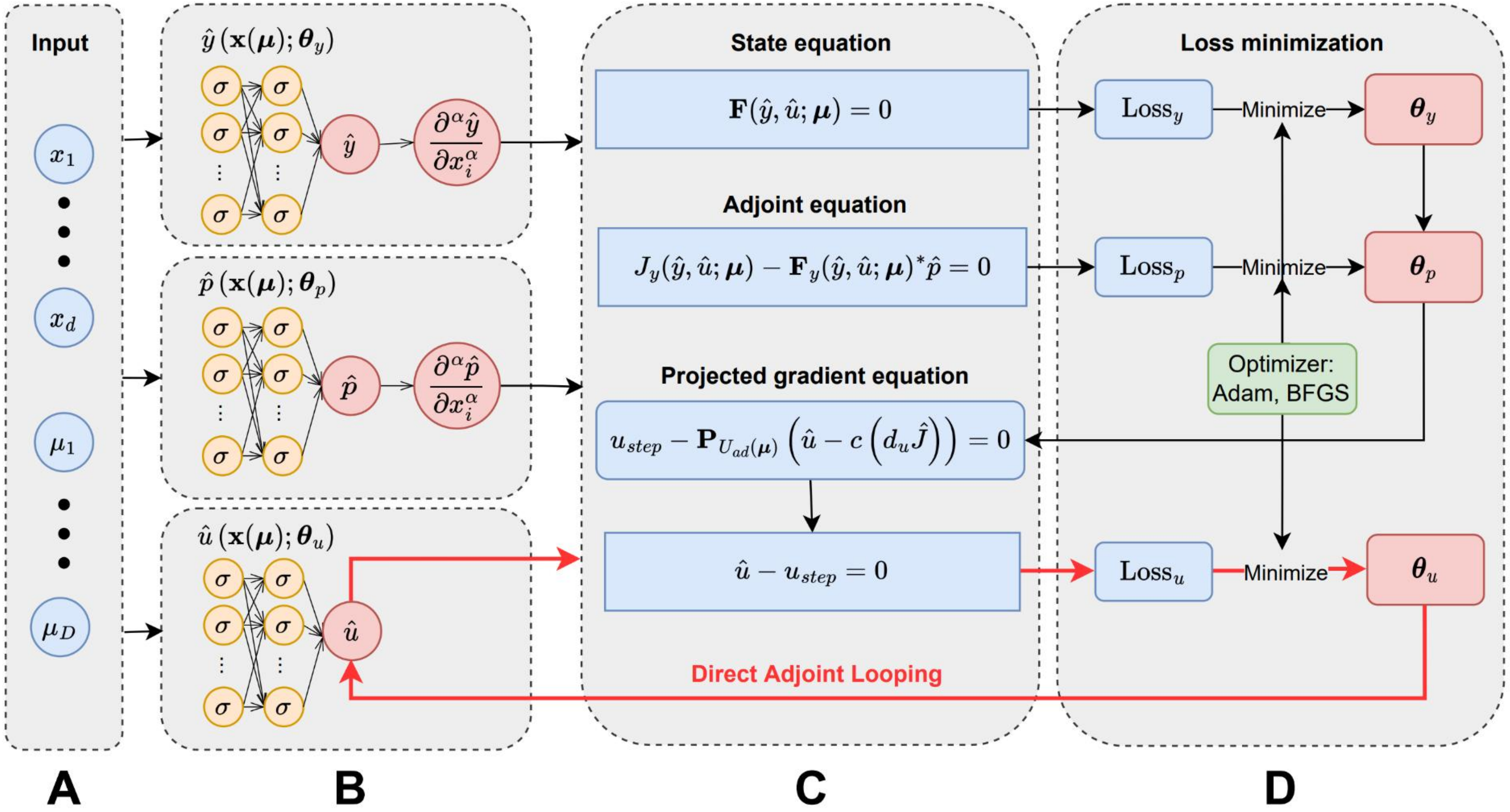
AONN for solving parametric optimal control problems

- Parametric optimal control problem:

$$\begin{cases} \min\limits_{y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu})} \mathcal{J}(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu});\boldsymbol{\mu}), \\ \text{s.t.} \quad \mathbf{F}(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu});\boldsymbol{\mu}) = 0 \quad \text{in } \Omega(\boldsymbol{\mu}), \\ u(\mathbf{x},\boldsymbol{\mu}) \in U_{ad}(\boldsymbol{\mu}). \end{cases}$$

- The parameter **μ** could involve:

$\mathcal{J}(\cdot;\boldsymbol{\mu})$: model parameter
$\mathbf{F}(\cdot;\boldsymbol{\mu})$: physical parameter
$U_{ad}(\boldsymbol{\mu})$: control parameter
$\Omega(\boldsymbol{\mu})$: geometrical parameter



(A) Input parameters. (B) Neural networks. (C) Loss functions. (D) Optimizer.

P. Yin, G. Xiao, K. Tang, C. Yang, AONN: An adjoint-oriented neural network method for all-at-once solutions of parametric optimal control problems, preprint.

Main idea

The KKT system

$$\begin{cases} J_y(y^*(\boldsymbol{\mu}), u^*(\boldsymbol{\mu}); \boldsymbol{\mu}) - \mathbf{F}_y^*(y^*(\boldsymbol{\mu}), u^*(\boldsymbol{\mu}); \boldsymbol{\mu}) p^*(\boldsymbol{\mu}) = 0, \\ \mathbf{F}(y^*(\boldsymbol{\mu}), u^*(\boldsymbol{\mu}); \boldsymbol{\mu}) = 0, \\ (\mathrm{d}_u J(y^*(\boldsymbol{\mu}), u^*(\boldsymbol{\mu}); \boldsymbol{\mu}), v(\boldsymbol{\mu}) - u^*(\boldsymbol{\mu})) \geq 0, \ \forall v(\boldsymbol{\mu}) \in U_{ad}(\boldsymbol{\mu}). \end{cases}$$

Solving this KKT system to get the optimal solution

- three neural networks to approximate $y^*(\boldsymbol{\mu})$, $u^*(\boldsymbol{\mu})$ and $p^*(\boldsymbol{\mu})$ separately
- deal with the parameters

goal: obtain the optimal solution for any parameters

$$\mathcal{L}_s(\boldsymbol{\theta}_y, \boldsymbol{\theta}_u) = \left(\frac{1}{N} \sum_{i=1}^{N} \left| r_s \left(\hat{y}(\mathbf{x}(\boldsymbol{\mu})_i; \theta_y), \hat{u}(\mathbf{x}(\boldsymbol{\mu})_i; \theta_u); \boldsymbol{\mu}_i \right) \right|^2 \right)^{\frac{1}{2}}, \tag{1a}$$

residual of the state equation

$$\mathcal{L}_a(\boldsymbol{\theta}_y, \boldsymbol{\theta}_u, \boldsymbol{\theta}_p) = \left(\frac{1}{N} \sum_{i=1}^{N} \left| r_a \left(\hat{y}(\mathbf{x}(\boldsymbol{\mu})_i; \theta_y), \hat{u}(\mathbf{x}(\boldsymbol{\mu})_i; \theta_u), \hat{p}(\mathbf{x}(\boldsymbol{\mu})_i; \theta_p); \boldsymbol{\mu}_i \right) \right|^2 \right)^{\frac{1}{2}} \tag{1b}$$

residual of the adjoint equation

$$\mathcal{L}_u(\boldsymbol{\theta}_u, u_{\text{step}}) = \left(\frac{1}{N} \sum_{i=1}^{N} \left| \hat{u}(\mathbf{x}(\boldsymbol{\mu})_i; \theta_u) - u_{\text{step}}(\mathbf{x}(\boldsymbol{\mu})_i) \right|^2 \right)^{\frac{1}{2}}. \tag{1c}$$

$$r_s(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) \triangleq \mathbf{F}(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}), \tag{2a}$$

$$r_a(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}), p(\boldsymbol{\mu}); \boldsymbol{\mu}) \triangleq J_y(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) - \mathbf{F}_y^*(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu}) p(\boldsymbol{\mu}), \tag{2b}$$

Some key ingredients

- the state equation and the adjoint equation: solving two parametric PDEs in $\Omega_{\mathcal{P}} = \{\mathbf{x}(\boldsymbol{\mu}) : \mathbf{x} \in \Omega(\boldsymbol{\mu})\}$
- projection gradient descent for inequality constraints in the KKT system

$$\mathbf{P}_{U_{ad}(\boldsymbol{\mu})}(u(\boldsymbol{\mu})) = \arg \min_{v(\boldsymbol{\mu}) \in U_{ad}(\boldsymbol{\mu})} \|u(\boldsymbol{\mu}) - v(\boldsymbol{\mu})\|_2,$$

$$u_{\text{step}}(\boldsymbol{\mu}) = \mathbf{P}_{U_{ad}(\boldsymbol{\mu})}\left(u(\boldsymbol{\mu}) - c\,\mathrm{d}_u J(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu})\right).$$

Because the optimal control function $u^*(\boldsymbol{\mu})$ satisfies

$$u^*(\boldsymbol{\mu}) - \mathbf{P}_{U_{ad}(\boldsymbol{\mu})}\left(u^*(\boldsymbol{\mu}) - c\,\mathrm{d}_u J(y^*(\boldsymbol{\mu}), u^*(\boldsymbol{\mu}); \boldsymbol{\mu})\right) = 0, \quad \forall c \geq 0.$$

The residual for the control function

$$r_v(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}), p(\boldsymbol{\mu})) \triangleq u(\boldsymbol{\mu}) - \mathbf{P}_{U_{ad}(\boldsymbol{\mu})}\left(u(\boldsymbol{\mu}) - c\,\mathrm{d}_u J(y(\boldsymbol{\mu}), u(\boldsymbol{\mu}); \boldsymbol{\mu})\right).$$

AONN algorithm

- training $\hat{y}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_y)$ for the state function

$$\boldsymbol{\theta}_y^k = \arg\min_{\boldsymbol{\theta}_y} \mathcal{L}_s\left(\boldsymbol{\theta}_y, \boldsymbol{\theta}_u^{k-1}\right).$$

- updating $\hat{p}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_p)$ for the adjoint function

$$\boldsymbol{\theta}_p^k = \arg\min_{\boldsymbol{\theta}_p} \mathcal{L}_a\left(\boldsymbol{\theta}_y^k, \boldsymbol{\theta}_u^{k-1}, \boldsymbol{\theta}_p\right).$$

- refining $\hat{u}(\mathbf{x}(\boldsymbol{\mu}); \boldsymbol{\theta}_u)$ for the control function

$$\boldsymbol{\theta}_u^k = \arg\min_{\boldsymbol{\theta}_u} \mathcal{L}_u\left(\boldsymbol{\theta}_u, u_{\text{step}}^{k-1}\right).$$

Optimal control with geometrical parametrization

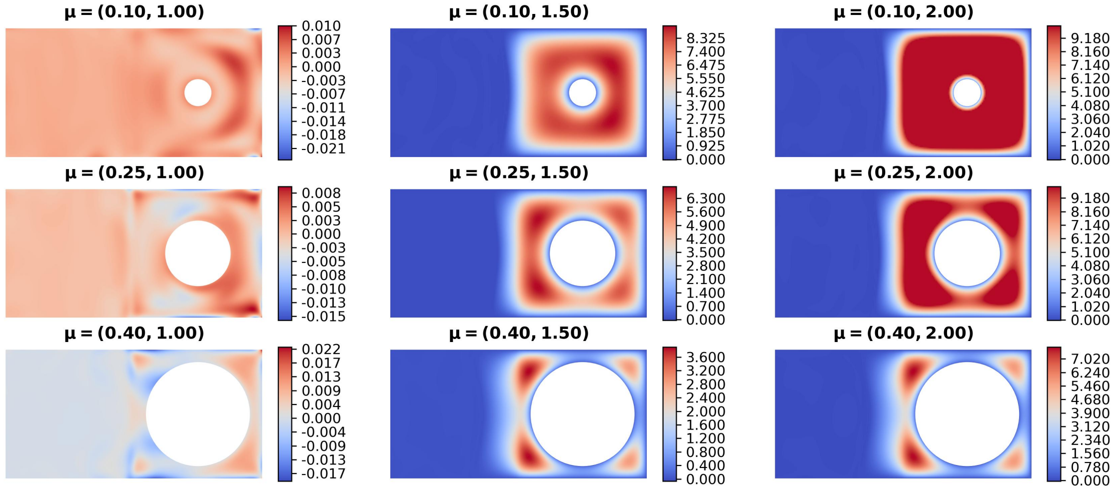
$$\begin{cases} \min\limits_{y(\boldsymbol{\mu}),u(\boldsymbol{\mu})} J(y(\boldsymbol{\mu}),u(\boldsymbol{\mu})) = \frac{1}{2}\|y(\boldsymbol{\mu})-y_d(\boldsymbol{\mu})\|^2_{L_2(\Omega(\boldsymbol{\mu}))} + \frac{\alpha}{2}\|u(\boldsymbol{\mu})\|^2_{L_2(\Omega(\boldsymbol{\mu}))}, \\ \text{subject to } \begin{cases} -\Delta y(\boldsymbol{\mu}) = u(\boldsymbol{\mu}) & \text{in } \Omega(\boldsymbol{\mu}), \\ y(\boldsymbol{\mu}) = 1 & \text{on } \partial\Omega(\boldsymbol{\mu}), \end{cases} \\ \text{and } \quad u_a \le u(\boldsymbol{\mu}) \le u_b \quad \text{a.e. in } \Omega(\boldsymbol{\mu}), \end{cases}$$

where $\boldsymbol{\mu} = (\mu_1, \mu_2)$ is the paramter.
$\Omega(\boldsymbol{\mu}) = ([0,2]\times[0,1])\backslash B((1.5,0.5),\mu_1)$ and the desired state is given by

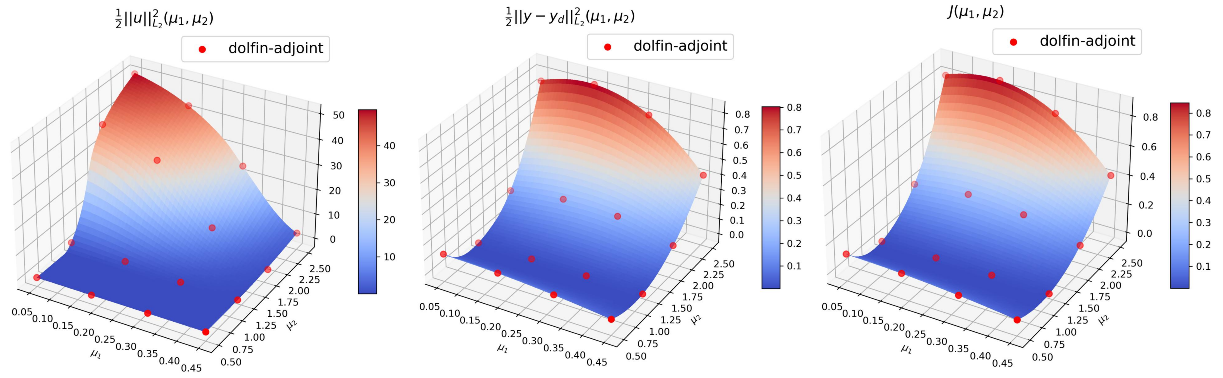
$$y_d(\boldsymbol{\mu}) = \begin{cases} 1 & \text{in } \Omega_1 = [0,1]\times[0,1], \\ \mu_2 & \text{in } \Omega_2(\boldsymbol{\mu}) = ([1,2]\times[0,1])\backslash B((1.5,0.5),\mu_1), \end{cases}$$

where $B((1.5,0.5),\mu_1)$ is a ball of radius μ_1 with center $(1.5,0.5)$, $\alpha = 0.001$ and $\boldsymbol{\mu} \in \mathcal{P} = [0.05,0.45]\times[0.5,2.5]$.

Solutions for different μ



Comparison with FEM



Parameter setting	Dolfin-adjoint Time (Intel i7-10510U)	AONN Time (Geforce RTX 2080)	AONN Error
$(\mu_1,\mu_2)\in\mathcal{P}$	-	21613s (training time)	-
$(\mu_1,\mu_2)\in\mathcal{P}_{4\times4}$	2244s	0.258s (evaluating time)	0.0483±0.0405
$(\mu_1,\mu_2)\in\mathcal{P}_{8\times8}$	9946s	0.347s (evaluating time)	0.0320±0.0295
$(\mu_1,\mu_2)\in\mathcal{P}_{16\times16}$	37380s	0.680s (evaluating time)	0.0338±0.0351

- ✓ All-at-once solutions
- ✓ Fast evaluation
- ✓ High accuracy

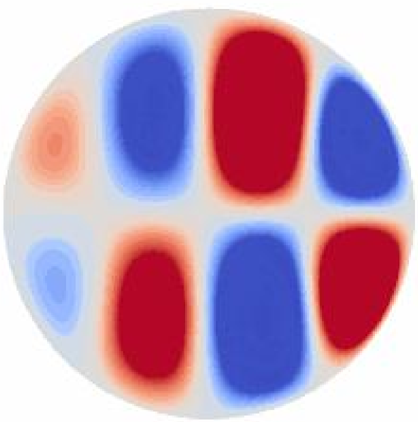
Optimal control with sparsity parametrization

- We seek the optimal sparse control $u(\beta)$ with parameter β controls the sparsity of u.
- The objective functional:

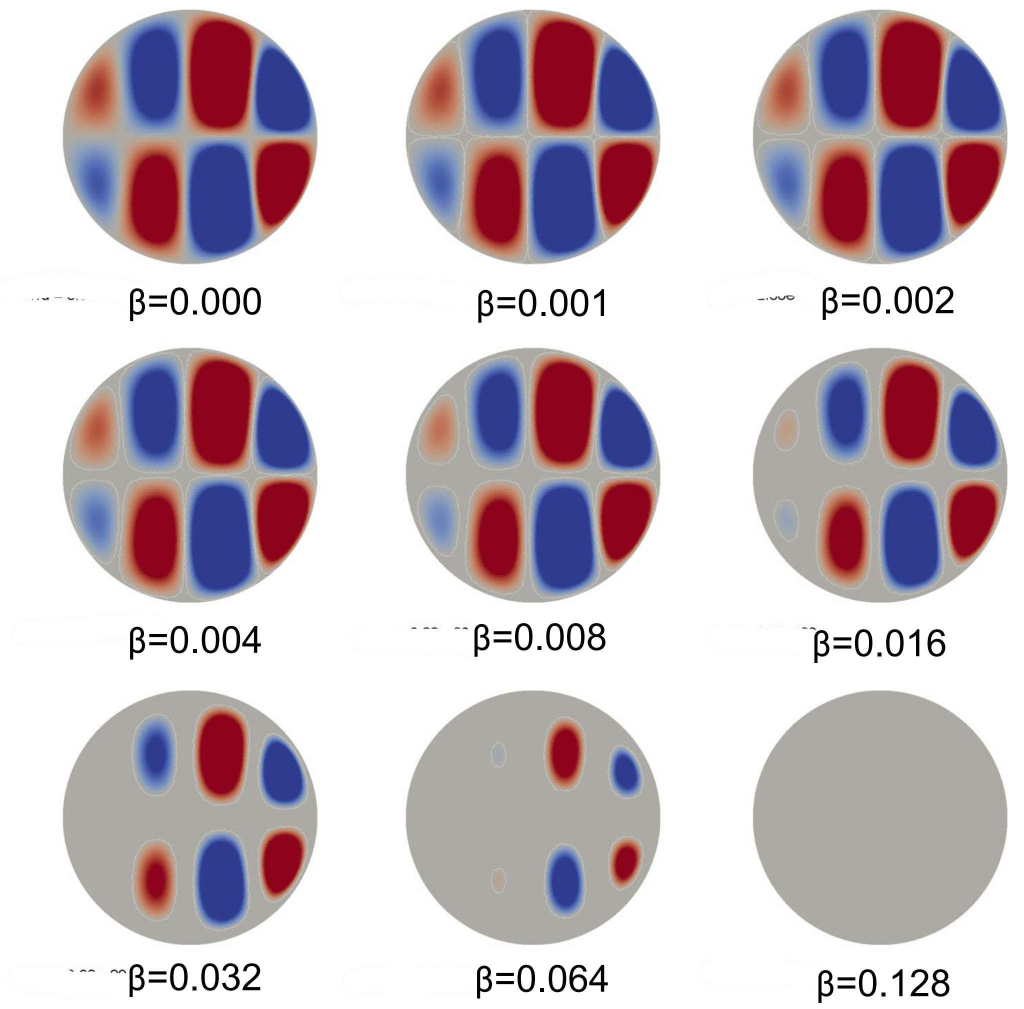
$$J(y,u) = \frac{1}{2}\|y - y_d\|_{L_2}^2 + \frac{\alpha}{2}\|u\|_{L_2}^2 + \beta\|u\|_{L_1}$$

- $\beta \in [0,\ 0.128]$

Solutions with continuously changing sparsity



FEM solutions for fixed sparsity



北京大学
长沙计算与数字经济研究院
PKU-Changsha Institute for Computing and Digital Economy

DL for computational mathematics, and computational mathematics for DL !

Thank you for your attention!

Kejun Tang
Email: tangkejun@icode.pku.edu.cn