ICLR
International Conference On Learning Representations

北京航空航天大學
BEIHANG UNIVERSITY

Score-based free-form architectures for high-dimensional Fokker-Planck equations

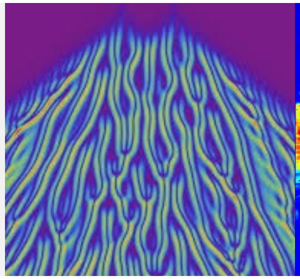
Feng Liu[1,3,4,5] **Faguo Wu**[1,4,5,6,*] **Xiao Zhang**[2,4,5,6,7,*]

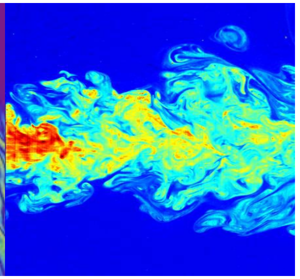
[1] School of Artificial Intelligence, Beihang University
[2] School of Mathematical Sciences, Beihang University
[3] National Superior College for Engineers, Beihang University
[4] Key Laboratory of Mathematics, Informatics and Behavioral Semantics, Ministry of Education
[5] Zhongguancun Laboratory
[6] Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing
[7] Hangzhou International Innovation Institute, Beihang University

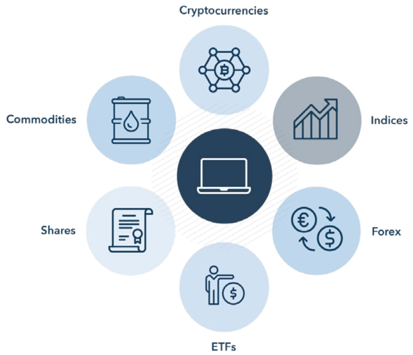
Background

Stochastic Dynamical Systems

The Fokker–Planck equation governs the time-varying probability density of dynamical systems driven by stochastic processes, with applications in statistical physics, finance, traffic flow prediction, epidemic spread simulation and climate modeling.

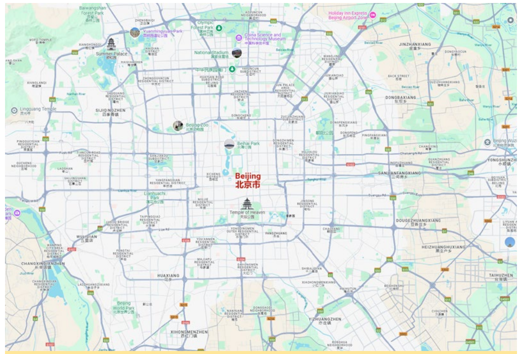






Stock Market

Governs stochastic processes in stock price movements, depicting risk modeling and option pricing.



Traffic Flow Prediction

Describes how cars or pedestrians move under uncertainty, helping optimize urban mobility.

Epidemic Spread

Simulates how diseases spread in populations with random interactions, aiding pandemic modeling.

Climate Modeling

Helps study stochastic climate variability, often represented in weather and ocean current simulations.

Preliminaries

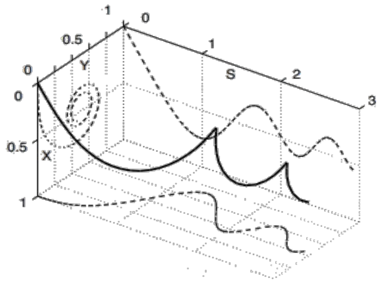
Fokker-Planck Equations and Challenges

- Stochastic differential equation: $dx = \mu(x,t)dt + \sigma(x,t)dW_t$
- Steady-state FP equation:

$$\frac{\partial p(\boldsymbol{x})}{\partial t} = \mathcal{L}p := -\sum_{i=1}^{d} \frac{\partial (p\mu_i)}{\partial x_i} + \sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial^2 (D_{i,j}p)}{\partial x_i \partial x_j} = -\nabla \cdot (p\boldsymbol{\mu}) + \nabla \cdot [\nabla \cdot (\boldsymbol{D}p)]$$

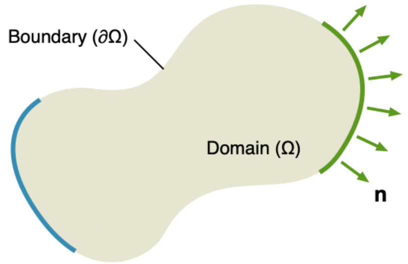
i.e.

$\mathcal{L}p(x) = 0, \qquad x \in R^d$



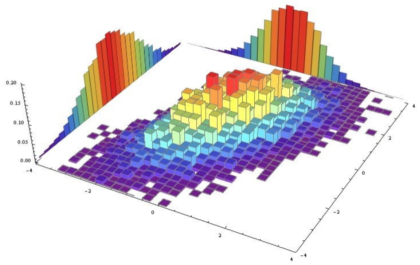
High-dimensional variables

$p(x) \to 0, \qquad (\|x\| \to \infty)$



Unbounded spatial domains

$\int_{R^d} p(x)dx = 1$



Normalization condition

Related Work

Physics-informed Neural Networks

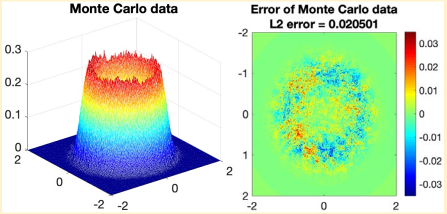
- Plain PDE loss: $\mathcal{J}_{\text{plain}}(p_\theta) = \mathbb{E}_{x\sim U(\Omega)}\left[\left|-\nabla\cdot\big(p_\theta(x)\mu(x)\big) + \nabla\cdot\Big(\nabla\cdot\big(D(x)p_\theta(x)\big)\Big)\right|^2\right]$
- Both p and Zp satisfy the SFP equation: $\mathcal{L}p = 0$
- Directly minimizing $\mathcal{J}_{\text{plain}}$ makes $p_\theta \approx Zp$ converge to a trivial solution.

Deep learning methods

Data Driven

Labeled data: $\{(x_j, p_j)\}_{j=1}^N$

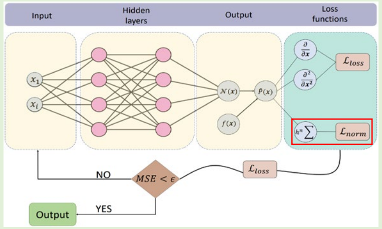
$$\mathcal{J}_{\text{label}}(p_\theta) = \frac{1}{N}\sum_{j=1}^{N}\big(p_\theta(x_j) - p_j\big)^2$$



Normalization Condition

Soft Constraints.

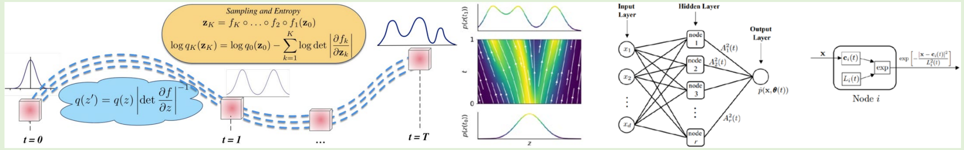
$$\mathcal{J}_{\text{norm}}(p_\theta) = \left(\frac{\nu(\Omega)}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} p_\theta(x) - 1\right)^2$$



Hard Constraints.

- Normalizing Flows: $p_X(x) = p_z(z)|\det \nabla_x f(x)|$
- Shape-morphing Gaussians:

$$p(x;\theta(t)) = \sum_{i=1}^{r} A_i^2(t)\exp\left[-\frac{|x - c_i(t)|^2}{L_i^2(t)}\right]$$



Related Work

Limitations

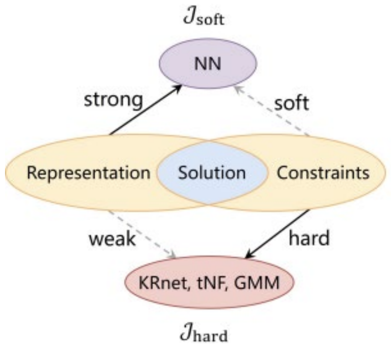
- **Data-driven:** few labels, low accuracy, computational burden.
- **Normalization:** trade-offs between representation capabilities and normalization constraints.

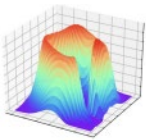
Model and Constraints

- **Soft constraints** often **violate the physical laws**.
- **Hard constraints** may **sacrifice the representation capacity**.

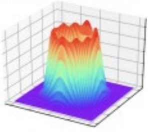
Optimize dynamics

- $\mathcal{J}_{\text{plain}}$ tends to push the solution to zero.
- $\mathcal{J}_{\text{norm}}$ prevents the zero solution on domain.
- These **conflicting forces** result in a **tortuous optimization process** that often requires delicate manual balancing.





(a) Real NVP



(b) GMM

Table 1: Deep learning methods for SFP equations.

Methods	Loss function	Model	Arbitrary NN?	Whether strictly satisfy NC?
Data-Driven	$\mathcal{J}_{\text{plain}} + \lambda \mathcal{J}_{\text{label}}$	$p_\theta(\boldsymbol{x}) = p_{\text{NN}}(\boldsymbol{x};\theta)$	✓	✗
Normalization Condition	$\mathcal{J}_{\text{plain}} + \lambda \mathcal{J}_{\text{norm}}$	$p_\theta(\boldsymbol{x}) = p_{\text{NN}}(\boldsymbol{x};\theta)$	✓	✗
	$\mathcal{J}_{\text{plain}}(p_\theta)$	$p_\theta(\boldsymbol{x}) = p_{\text{KRnet}}(\boldsymbol{x};\theta),\ p_{\text{GMM}}(\boldsymbol{x};\theta),\ p_{\text{TNN}}(\boldsymbol{x};\theta)\cdots$	✗	✓
	$\mathcal{J}_{\text{score}}(\widetilde{p}_\theta)$	$\widetilde{p}_\theta(\boldsymbol{x}) = \widetilde{p}_{\text{NN}}(\boldsymbol{x};\theta),\ p_\theta(\boldsymbol{x}) = \frac{\widetilde{p}_\theta(\boldsymbol{x})}{\int \widetilde{p}_\theta(\boldsymbol{y})d\boldsymbol{y}}$	✓	✓

Our Method

Fokker-Planck Neural Network

- We propose a Fokker-Planck neural network (FPNN) framework to efficiently solve high-dimensional SFP equations, decoupling **the score learning** and **the density normalization** into two stages.
- **Stein Score** $\nabla_x \log p$: $p_\theta(\boldsymbol{x}) = \frac{\widetilde{p}_\theta(\boldsymbol{x})}{Z_\theta}$, $Z_\theta = \int \widetilde{p}_\theta(\boldsymbol{x}) d\boldsymbol{x}$ $\log p_\theta(\boldsymbol{x}) = \log \widetilde{p}_\theta(\boldsymbol{x}) - \log Z_\theta$ $\nabla \log p_\theta(\boldsymbol{x}) = \nabla \log \widetilde{p}_\theta(\boldsymbol{x})$
- **Key idea:** Since p_θ and $\tilde{p}_\theta$ share the same score, we can derive a score PDE loss of SFP equations to bypass the normalization condition.
- **Derivation of Score PDE Loss:**

$$\begin{aligned}\mathcal{L}p(x) &= -\nabla \cdot (p\mu) + \nabla \cdot \nabla \cdot (Dp) \\ &= -\nabla \cdot (p\mu - (\nabla \cdot D)p - D\nabla p) \\ &= -\nabla \cdot \big(p(\mu - \nabla \cdot D - D\nabla \log p)\big) \\ &= -\nabla \cdot (p\tilde{\mu})\end{aligned}$$

$$\tilde{\mu} := \mu - \nabla \cdot D - D\nabla \log p$$

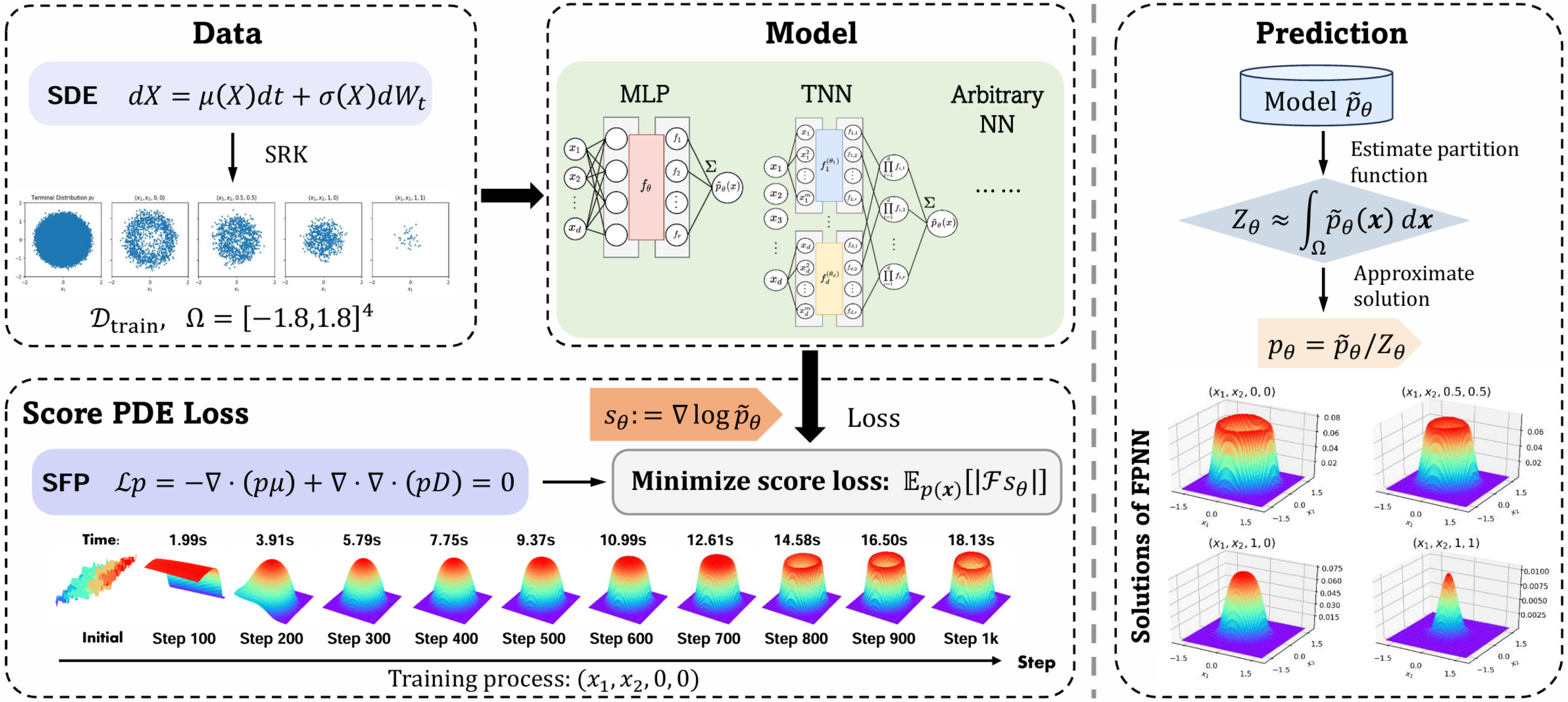
$$\begin{aligned}\mathcal{J}_{\text{plain}}(p_\theta) &= \mathbb{E}_{q(x)}[|\mathcal{L}p_\theta(x)|] = \int_\Omega \frac{1}{\nu(\Omega)} |\mathcal{L}p_\theta(x)| dx \\ &= \frac{1}{\nu(\Omega)} \int_\Omega |\nabla p_\theta \cdot \tilde{\mu} + p_\theta(\nabla \cdot \tilde{\mu})| dx \\ &= \frac{1}{\nu(\Omega)} \mathbb{E}_{p_\theta(x)}[|\nabla \log p_\theta \cdot \tilde{\mu} + (\nabla \cdot \tilde{\mu})|] \\ &\approx \frac{1}{\nu(\Omega)} \mathbb{E}_{p(x)}[|\mathcal{F}s_\theta(x)|] = \frac{1}{\nu(\Omega)} \mathcal{J}_{\text{score}}(s_\theta)\end{aligned}$$

Our Method

Framework

- **Score PDE Loss:** $\mathbb{E}_{p(\boldsymbol{x})}[|\mathcal{F}s_\theta(\boldsymbol{x})|] := \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[|s_\theta(\boldsymbol{x})\cdot\tilde{\boldsymbol{\mu}}(\boldsymbol{x}) + \nabla\cdot\tilde{\boldsymbol{\mu}}(\boldsymbol{x})|]$

$$\tilde{\boldsymbol{\mu}}(\boldsymbol{x}) := \boldsymbol{\mu}(\boldsymbol{x}) - \nabla\cdot\boldsymbol{D}(\boldsymbol{x}) - \boldsymbol{D}(\boldsymbol{x})s_\theta(\boldsymbol{x})$$



Tensor Neural Networks (TNN)

$$\tilde{p}_\theta(\boldsymbol{x}) = \sum_{j=1}^{r}\prod_{i=1}^{d} f_{i,j}(x_i;\theta_i)$$

$$Z_\theta \approx \sum_{j=1}^{r}\prod_{i=1}^{d}\left(\sum_{n_i=1}^{N_i} w_i^{(n_i)} f_{i,j}\left(x_i^{(n_i)};\theta_i\right)\right)$$

Multi-Layer Perceptron (MLP)

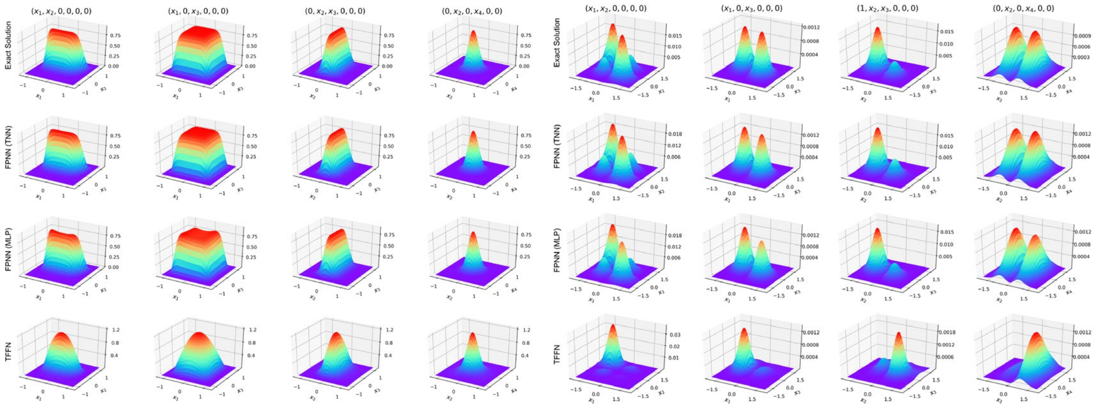
$$\tilde{p}_\theta(\boldsymbol{x}) = \sum_{j=1}^{r} f_j(\boldsymbol{x};\theta)$$

$$Z_\theta \approx \int_\Omega \tilde{p}_\theta(\boldsymbol{x})d\boldsymbol{x} \approx \frac{|\Omega|}{|\mathcal{D}_{\text{norm}}|}\sum_{\boldsymbol{x}_i\in\mathcal{D}_{\text{norm}}}\tilde{p}_\theta(\boldsymbol{x}_i)$$

Experiments

Accuracy

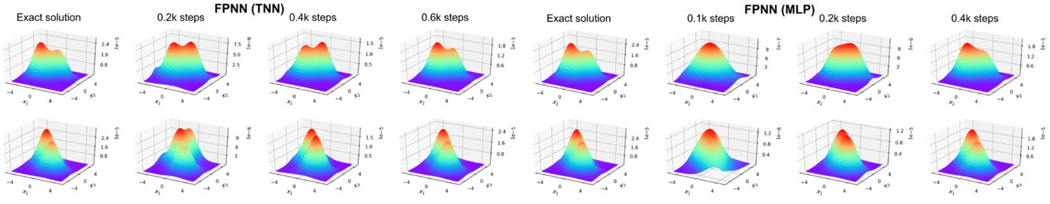
- **TFFN** can not learn the solution correctly with **50,688** parameters and 20k steps in 6D Unimodal problem.
- **FPNN** gets average relative errors of **11.36%**, **13.87%** and **12.72%** for 4D Ring, 6D Unimodal and 6D Multi-modal problems respectively, requiring only **256**, **980**, and **980** parameters.



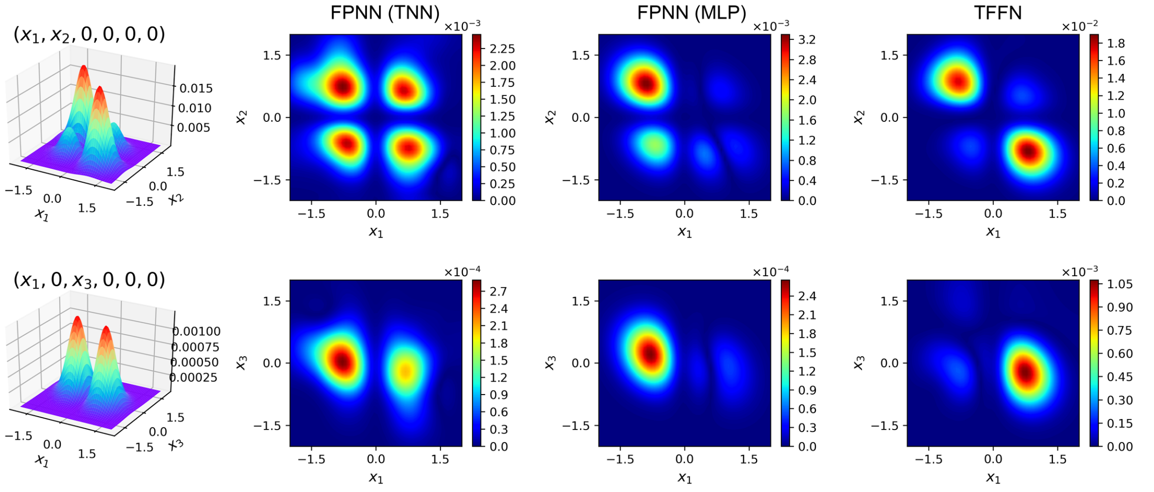
6D Unimodal **6D Multi-modal**

Table 2: Experimental results of TFFN and FPNN on 4-6 dimensional SFP equations.

SFP equations	Domain Ω	TFFN MAE	TFFN MAPE	FPNN (Ours) MAE	FPNN (Ours) MAPE
4D Ring	$[-1.8, 1.8]^4$	3.61×10^{-3}	49.25%	5.56×10^{-4}	3.84%
6D Unimodal	$[-1.2, 1.2]^6$	4.00×10^{-2}	293%	1.48×10^{-3}	4.33%
6D Multi-modal	$[-2, 2]^6$	1.84×10^{-3}	92.90%	1.98×10^{-4}	12.18%



10D Gaussian mixture: predicted solutions

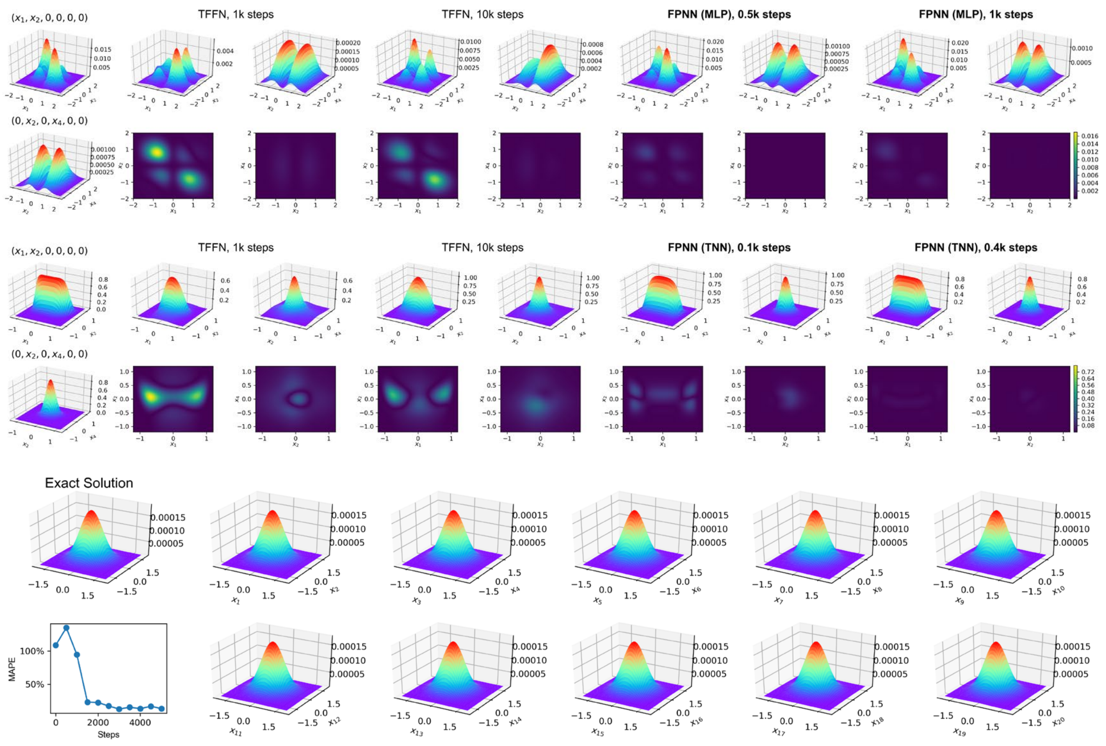


6D Multi-modal: absolute errors

Experiments

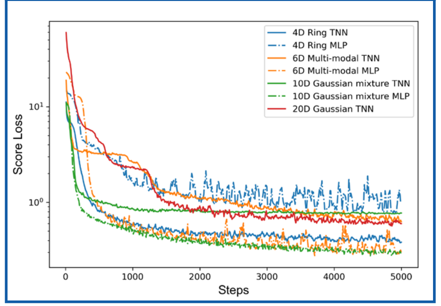
Efficiency

- **FPNN** successfully learn the multi-modal at **1k** steps, and capture the 6-dimensional peak in **400** steps. Score PDE loss makes the training dynamics more coherent and efficient, so we solve SFP equations much faster than **TFFN**.

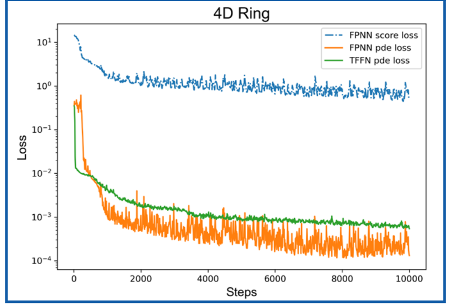


20D Gaussian: predicted solutions

$\mathcal{J}_{\text{score}}$ for SFP problems



$\mathcal{J}_{\text{score}}$ vs. $\mathcal{J}_{\text{plain}}$



Model	Layers	Parameters	MAE	MAPE
	[m, hidden layers, r]			
	[1, 64, 128]	33,792	7.27×10^{-3}	99.82%
	[3, 20, 20, 20, 20]	5,360	4.25×10^{-3}	65.41%
TNN	[3, 64, 64, 64]	34,304	6.91×10^{-4}	7.58%
	[5, 64, 128]	34,816	8.20×10^{-4}	5.66%
	[8, 64, 128]	35,584	5.56×10^{-4}	3.84%
	[d, hidden layers]			
	[4, 8, 8, 8, 8]	256	1.26×10^{-3}	11.36%
MLP	[4, 20, 20, 20, 20]	1,360	9.74×10^{-4}	8.74%
	[4, 64, 64, 64]	8,640	8.81×10^{-4}	6.48%
	[4, 128, 128]	17,152	6.21×10^{-4}	5.39%

4D Ring: FPNN with different networks

Conclusion

- We successfully solve **4-20** dimensional steady-state FP equations for complex physical systems and comprehensive experimental results show great gains in **efficiency**, **accuracy**, **memory** and **computational resource usage**.

- **Time-varying** probability densities have broader applications in physics, finance, mean-field games, and diffusion models, but still face challenges for the normalization condition at any time.

- In the future, we aim to develop an efficient deep-learning solver for general Fokker-Planck equations and look forward to its applications in broader fields.

Thank you！

Paper: https://openreview.net/forum?id=5qg6JPSgCj
Code: https://github.com/niuffs/FPNN