Machine learning to solve functional renormalization group

Takeru Yokota

RIKEN, iTHEMS

Ref: TY, Physics-informed neural network for solving functional renormalization group on lattice, arXiv:2312.16038

Functional Renormalization Group at Niigata 2024, Jan 8 2024, online+Kaishi Professional University Yoneyama Campus

Contents

- Introduction
  - Functional differential equation
  - Taylor-expansion-based method and its limitation
- Physics-informed neural networks for solving FRG
  - Physics-informed neural network
  - Application to Wetterich equation
- Demonstration: 0-dimensional $\mathrm{O}(N)$ model

Contents

- Introduction
  - Functional differential equation
  - Taylor-expansion-based method and its limitation
- Physics-informed neural networks for solving FRG
  - Physics-informed neural network
  - Application to Wetterich equation
- Demonstration: 0-dimensional $\mathrm{O}(N)$ model

Functional differential equation (FDE)

- **Partial differential equation (PDE) involving functional & functional diff.**
- **A comprehensive tool to describe the statistical properties of a system**

Various examples in physics

- **Functional renormalization group (FRG)**

  Wetterich eq.: exact formulation of RG of the effective action

$$\partial_k \Gamma_k[\varphi] = \frac{1}{2}\mathrm{Tr}\left[\partial_k R_k \left(\frac{\delta^2 \Gamma_k}{\delta\varphi\delta\varphi}[\varphi] + R_k\right)^{-1}\right]$$

  k: RG scale
  R_k: Regulator

  Wetterich, PLB (1993)

- **Hopf eq.** *Hopf, J. Rat. Mech. Anal. (1952)*

  An exact formulation of the time evolution of the statistical properties of turbulence

- Density functional theory, Schwinger-Dyson, Fokker-Planck, …

FDE is exact, but…

there is no universal, accurate, and efficient solution method!

Taylor-expansion-based method and its limitations

To facilitate numerical analysis,
approximation of FDE is needed.

$$\partial_k \Gamma_k[\varphi] = \frac{1}{2}\mathrm{Tr}\left[\partial_k R_k \left(\frac{\delta^2 \Gamma_k}{\delta\varphi\delta\varphi}[\varphi] + R_k\right)^{-1}\right]$$

Common approach: Taylor series expansion & truncation

E.g.1) Derivative expansion $\Gamma_k[\varphi] \approx \int dx \left(U_k(\varphi(x)) + \frac{1}{2}Z_k(\varphi(x))\left(\partial_\mu \varphi(x)\right)^2 + \cdots\right)$

E.g.2) Vertex expansion **Functional Taylor expansion around** $\varphi(x) = \varphi_{\mathrm{target}}(x)$

❌ **The solution is applicable to limited configurations of** $\varphi(x)$

- Derivative expansion: $\varphi(x) \approx \mathrm{const}.$
- Vertex expansion: $\varphi(x) \approx \varphi_{\mathrm{target}}(x)$

→

- Prior knowledge of the ground state is required.
- Treatment of complex field configurations, such as inhomogeneous states, is tough.

❌ **Usually, improving the order of the truncation is not easy.**

- Many coordinate (momentum) integrals

Other solution methods?

Another attempt: FDE as high-dim. PDE

FDE: Infinite-dimensional partial differential equation (PDE)

To realize numerical analysis, the input dimensions should be truncated.

Method 1) Introduction of finite spatial lattice

$$\boldsymbol{\varphi} = \{\varphi_{n,\alpha}\}_{n,\alpha},\ n = (n_1, \ldots, n_d),\ 0 \le n_i < L,\ \alpha = 1, \ldots, N_{\mathrm{IDOF}} \text{ (internal DOF)}$$

Total DOF of φ: $N_{\mathrm{DOF}} = L^d N_{\mathrm{IDOF}}$

Method 2) Basis function expansion

See, e.g., *Venturi, PR (2018), Venturi, Dektor, Res. Math. Sci (2021)*

$$\varphi(x,\alpha) \approx \sum_{i=1}^{N_{\mathrm{DOF}}} \varphi_i b_i(x,\alpha) = \boldsymbol{\varphi} \cdot \boldsymbol{b}(x,\alpha)\ (b_i(x,\alpha)\text{: orthonormal basis function})$$

$$\partial_k \Gamma_k[\varphi] = \frac{1}{2}\mathrm{Tr}\left[\partial_k R_k \left(\frac{\delta^2 \Gamma_k}{\delta\varphi\delta\varphi}[\varphi] + R_k\right)^{-1}\right] \quad \Rightarrow \quad \text{$(N_{\mathrm{DOF}}+1)$-dim. PDE}$$

$$\partial_k \Gamma_k(\boldsymbol{\varphi}) = \frac{1}{2}\mathrm{tr}\left[\partial_k R_k \left(\frac{\partial^2 \Gamma_k}{\partial\varphi\partial\varphi}(\boldsymbol{\varphi}) + R_k\right)^{-1}\right]$$

The original continuum theory is obtained for increasing N_{DOF}.

Failure of computational grids

How can we solve $(N_{\mathrm{DOF}}+1)$-dim. PDE with large N_{DOF}

Taking computational grids for φ is infeasible.

# of grid points $\sim$ exp(N_{DOF}) if grids are assigned for each $\varphi_{n,\alpha}$

Grid-based methods (finite-element method, Runge-Kutta, …) can not be used.

c.f.) Some people attempt to use tensor decomposition to mitigate computational complexity of representing the solution for linear FDE, such as Hopf eq.

But only the results with $N_{\mathrm{DOF}} \lesssim 6$ has been reported…

Venturi, PR (2018), Venturi, Dektor, Res. Math. Sci (2021)

Grid-free method for high-dim. PDE is required

Purpose of this talk

My idea:

Machine learning allows us to calculation with large N_{DOF}!

- Grid-free method for solving PDE
- The solution is obtained simultaneously for a **domain** of the configuration space of $\boldsymbol{\varphi}$ rather than one configuration.

Advantageous for applications to inhomogeneous systems

Solitons in scalar models, electrons in materials, …

In this talk, I will share the idea and show a numerical demonstration.

Contents

- Introduction
  - Functional differential equation
  - Taylor-expansion-based method and its limitation
- Physics-informed neural networks for solving FRG
  - Physics-informed neural network
  - Application to Wetterich equation
- Demonstration: 0-dimensional $\mathrm{O}(N)$ model

Machine learning for partial differential equations (PDE)

Recently, there have been many applications of machine learning to PDEs

- **Physics-informed neural network (PINN)**

  Psichogios, Ungar, AIChE (1992)
  Lagaris, Likas, Fotiadis, IEEE Transactions on Neural Networks (1998)
  Raissi, Perdikaris, Karniadakis (2017)
  Raissi, Perdikaris, Karniadakis, Journal of Computational Physics (2019),…

Applicable to various types of PDEs

- **Rayleigh-Ritz variational method**

  E, Yu, Comm. Math. Stat. (2018)
  Khoo, Lu, Ying, Res. Math. Sci. (2018),…

- **Backward stochastic differential equation**

  E, Han, Jentzen, Comm. Math. Stat. (2017)
  Han, Jetzen, E (2018)
  Rassi (2018)
  Beck, E, Jentzen, J. Nonlinear Science (2019),…

⋮

Physics-informed neural network (PINN)

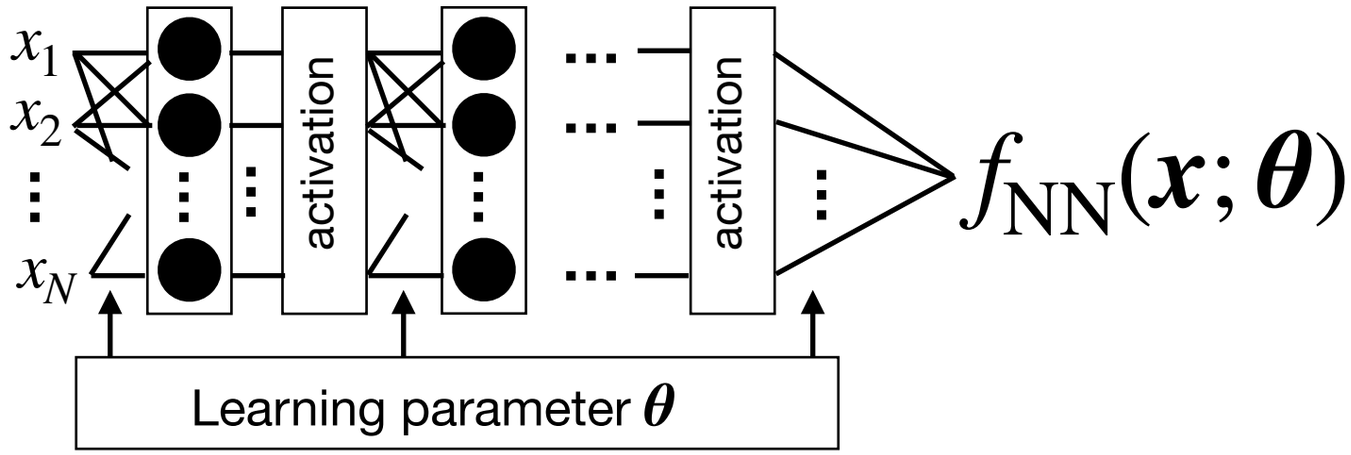
I.E. Lagaris, A. Likas, D. I. Fotiadis, IEEE Transactions on Neural Networks (1998)
M. Raissi, P. Perdikaris, G. E. Karniadakis (2017)
M. Raissi, P. Perdikaris, G.E. Karniadakis, Journal of Computational Physics (2019)

The solution is represented by an NN and optimized to satisfy PDE and BC with backpropagation.

PDE: $I_{\mathrm{DE}}(f(\boldsymbol{x}), \partial_\mu f(\boldsymbol{x}), \cdots) = 0 \qquad \boldsymbol{x} \in \Omega$ (Domain)

Boundary condition: $I_{\mathrm{BC}}(f(\boldsymbol{x}), \partial_\mu f(\boldsymbol{x}), \cdots) = 0 \qquad \boldsymbol{x} \in \partial\Omega$

$f(\boldsymbol{x}) \approx f_{\mathrm{NN}}(\boldsymbol{x}; \boldsymbol{\theta})$



Loss function (taking the minimum when PDE & BC are satisfied)

例: $L(\boldsymbol{\theta}) = \underset{\boldsymbol{x}\in\Omega}{\mathrm{average}} \| I_{\mathrm{DE}}(f_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\theta}), \partial_\mu f_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\theta}), \cdots) \|^2 + \underset{\boldsymbol{x}\in\partial\Omega}{\mathrm{average}} \| I_{\mathrm{BC}}(f_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\theta}), \partial_\mu f_{\mathrm{NN}}(\boldsymbol{x};\boldsymbol{\theta}), \cdots) \|^2$

PINN's capability of handling high-dim. PDEs

Advantage of PINN: Grid free → Applicable to high-dim. PDEs

Applications to high-dim. PDEs

- *L. Guo, H. Wu, X. Yu, T. Zhou, Computer Methods in Applied Mechanics and Engineering (2022)*
- *D. He, S. Li, W. Shi, X. Gao, J. Zhang, J. Bian, L. Wang, T.-Y. Liu, In International Conference on Artificial Intelligence and Statistics (2023)*
- *J. Cen, X. Chen, M. Xu, Q. Zou, arXiv:2305.06863.*
- *K. Tang, X. Wan, C. Yang, Journal of Computational Physics, 476 (2023)*
- *Z. Hu, K.Shukla, G. E. Karniadakis, K. Kawaguchi, arXiv:2307.12306*
- *Z. Hu, Z. Yang, Y. Wang, G. E. Karniadakis, K. Kawaguchi, arXiv:2311.15283*
- *Z. Hu, Z. Shi, G. E. Karniadakis, K. Kawaguchi, arXiv:2312.14499*

10^5-dim. PDE

Hamilton-Jacobi-Bellman eq. $$\partial_t u(\boldsymbol{x},t) + \Delta u(\boldsymbol{x},t) - \frac{1}{d}\sum_{i=1}^{d} |\partial_{\boldsymbol{x}_i} u|^c = -2, \quad \boldsymbol{x} \in \mathbb{R}^d, t \in [0,T]$$

Black-Scholes-Barenblatt eq. $$u_t = -\frac{1}{2}\sigma^2 \sum_{i=1}^{d} \boldsymbol{x}_i^2 u_{\boldsymbol{x}_i \boldsymbol{x}_i} + r(u - \sum_{i=1}^{d} \boldsymbol{x}_i u_{\boldsymbol{x}_i}),$$

PDE	10^2	10^3	10^4	10^5
HJB-Lin	34min	68min	81min	310min
BSB	31min	57min	118min	41min

Table 1: This table presents the convergence time required by our SDGD for different PDEs. In the HJB-Lin equation, as the dimensionality increases from 100 to 100,000, the dimensionality grows by a factor of 1000, while the time only increases by a factor of ten. This indicates that our method can withstand the curse of dimensionality. In the second BSB equation, surprisingly, the high-dimensional case converges faster than the low-dimensional case, demonstrating the so-called blessing of dimensionality and that our method can harness the blessing of dimensionality.

Moreover, the solution is obtained simultaneously for a **domain** of the inputs rather than one input.

PINN for Wetterich equation

PINN allows us to solve the Wetterich equation with large N_{DOF}!

Procedures proposed in *TY, arXiv:2312.16038*

1) NN for $\Gamma_k(\boldsymbol{\varphi})$

Most nontrivial part of $\Gamma_k(\boldsymbol{\phi})$ is the interaction-induced RG part.

$$\Gamma_k^{\boldsymbol{\theta}}(\boldsymbol{\varphi}) = S(\boldsymbol{\varphi}) + \Delta S_{\mathrm{free}}(k) + \gamma_{\boldsymbol{\theta}}(k, \boldsymbol{\varphi})$$

Constant shift from free part

Interaction-induced RG part

- We replace this with an NN
- Init. cond. $\gamma(k_{\mathrm{UV}}, \boldsymbol{\varphi}) = 0$

A conceivable choice:

$$\gamma_{\boldsymbol{\theta}}(k, \boldsymbol{\varphi}) \approx \mathrm{NN}_{\boldsymbol{\theta}}(k, \boldsymbol{\varphi}) - \mathrm{NN}_{\boldsymbol{\theta}}(k_{\mathrm{UV}}, \boldsymbol{\varphi})$$

L_{BC} is not necessary.

2) Training of Wetterich equation

$$L_{\boldsymbol{\theta}} = \mathop{\mathbb{E}}_{\substack{\boldsymbol{\varphi} \sim \mathscr{P}_{\boldsymbol{\varphi}} \\ k \sim \mathscr{P}_k}} \left[\left(\partial_k \Gamma_k^{\boldsymbol{\theta}}(\boldsymbol{\varphi}) - \frac{1}{2} \mathrm{tr} \left[\partial_k R_k \left(\frac{\partial^2 \Gamma_k^{\boldsymbol{\theta}}(\boldsymbol{\varphi})}{\partial \boldsymbol{\varphi} \partial \boldsymbol{\varphi}} + R_k \right)^{-1} \right] \right)^2 \right]$$

- The expectation is evaluated on a finite number of collocation points $(k, \boldsymbol{\varphi})$.
- $\mathscr{P}_{\boldsymbol{\varphi},k}$ is some probability distribution.

Contents

- Introduction
  - Functional differential equation
  - Taylor-expansion-based method and its limitation
- Physics-informed neural networks for solving FRG
  - Physics-informed neural network
  - Application to Wetterich equation
- Demonstration: 0-dimensional $\mathrm{O}(N)$ model

0-dim. O(N) model

$$S(\boldsymbol{\varphi}) = \frac{1}{2}m^2\boldsymbol{\varphi}^2 + \frac{g}{4!}(\boldsymbol{\varphi}^2)^2$$

- $N_{\mathrm{DOF}} = N$

The Wetterich equation is an $(N+1)$-dim. PDE.

We can investigate the scalability with N_{DOF} by increasing N.

* We do not reduce Wetterich eq. to 2-dim. PDE with k and $\rho = \varphi^2/2$

- Exact results and results by perturbative, large-N expansions are available.

  E.g., *Keitel, Bartosch, JPA (2012)*

  - The perturbative region is given by $\tilde{g} = Ng/m^4 \ll 1$
  - We calculate $\gamma(k, \boldsymbol{\varphi})$ and self-energy $\sigma = \partial^2_{\boldsymbol{\varphi}}\gamma(k, \boldsymbol{\varphi})$.

- Regulator: $R_k^{\alpha\alpha'} = k_{\mathrm{UV}}^2 e^{-2l},\ l = \ln(k_{\mathrm{UV}}/k)$
- Mass squared: $m^2/k_{\mathrm{UV}}^2 = 0.01$ ($\ll 1$ to validate the UV saddle-point cond.)

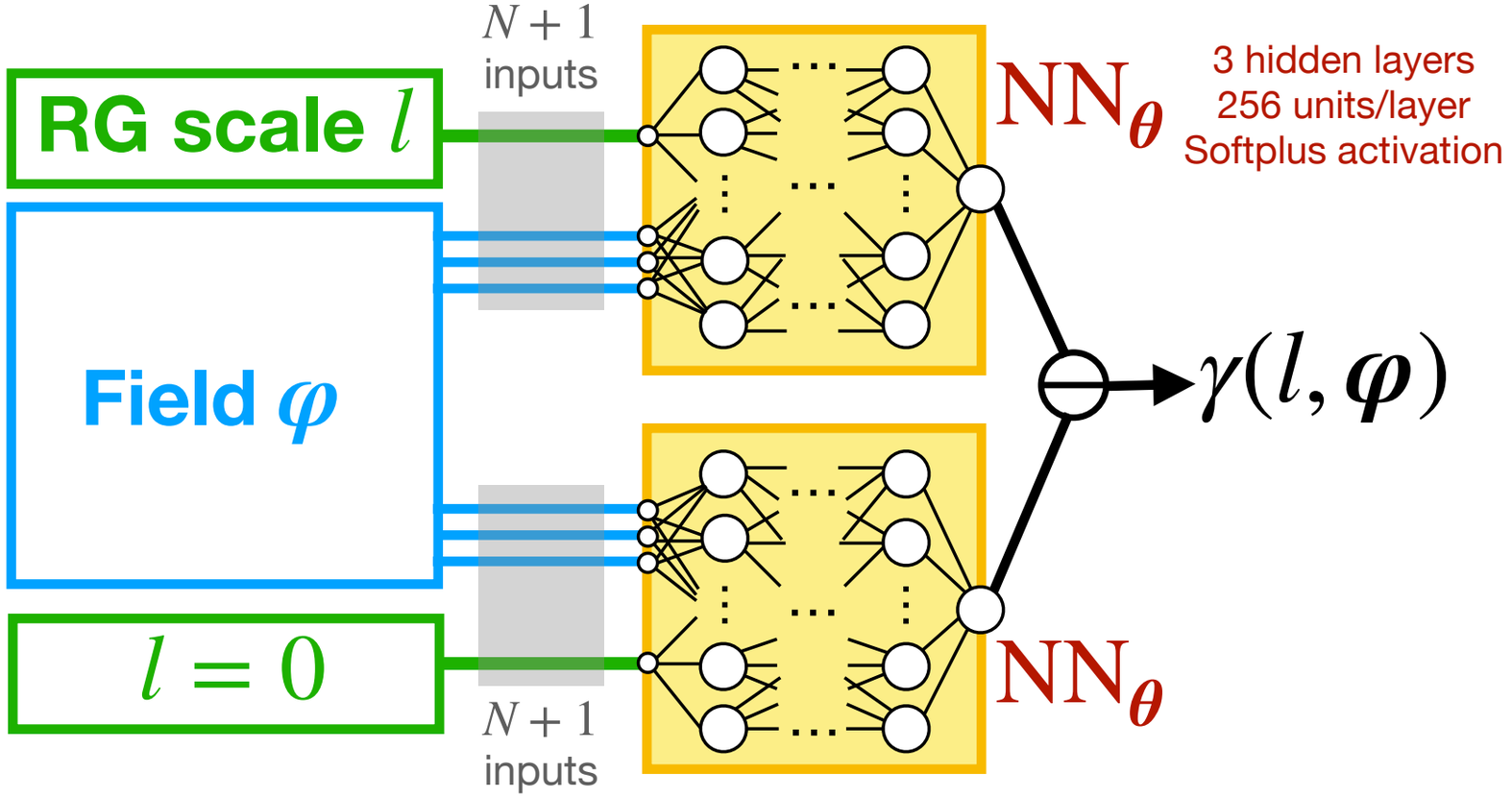
Neural network for effective action

$$\gamma_{\boldsymbol{\theta}}(l,\boldsymbol{\varphi}) \approx \mathrm{NN}_{\boldsymbol{\theta}}(l,\boldsymbol{\varphi}) - \mathrm{NN}_{\boldsymbol{\theta}}(0,\boldsymbol{\varphi})$$

$$l = \ln(k_{\mathrm{UV}}/k)$$

$$\gamma_{\boldsymbol{\theta}}(0,\boldsymbol{\varphi}) = 0$$

- 3 hidden layers
- 256 units/layer
- Differentiable softplus activation



Pretraining

$$L_{\boldsymbol{\theta}} = \mathop{\mathbb{E}}_{\substack{\boldsymbol{\varphi} \sim \mathscr{P}_{\boldsymbol{\varphi}} \\ l \sim \mathscr{P}_l}} \left[\left(\partial_l \Gamma_l^{\boldsymbol{\theta}}(\boldsymbol{\varphi}) - \frac{1}{2} \mathrm{tr} \left[\partial_l R_l \left(\frac{\partial^2 \Gamma_l^{\boldsymbol{\theta}}(\boldsymbol{\varphi})}{\partial \boldsymbol{\varphi} \partial \boldsymbol{\varphi}} + R_l \right)^{-1} \right] \right)^2 \right]$$

The matrix must be regular during the training.

In our experience, this is frequently broken with randomly chosen $\boldsymbol{\theta}$.

Pretraining with some approximate analytic results remedies this problem

We use 1st-order perturbative result:

$$L_{\boldsymbol{\theta}}^{\mathrm{pre}} = \mathop{\mathbb{E}}_{\substack{\boldsymbol{\varphi} \sim \mathscr{P}_{\boldsymbol{\varphi}} \\ l \sim \mathscr{P}_l}} \left[\left(\gamma(l, \boldsymbol{\varphi}; \boldsymbol{\theta}) - \gamma^{\mathrm{1pt}}(l, \boldsymbol{\varphi}) \right)^2 \right]$$

Other details of numerical procedure

$$L_{\boldsymbol{\theta}} = \mathop{\mathbb{E}}_{\substack{\boldsymbol{\varphi}\sim\mathscr{P}_{\boldsymbol{\varphi}} \\ l\sim\mathscr{P}_l}} \left[\left(\partial_l \Gamma_l^{\boldsymbol{\theta}}(\boldsymbol{\varphi}) - \frac{1}{2}\mathrm{tr}\left[\partial_l R_l \left(\frac{\partial^2 \Gamma_l^{\boldsymbol{\theta}}(\boldsymbol{\varphi})}{\partial\boldsymbol{\varphi}\partial\boldsymbol{\varphi}} + R_l \right)^{-1} \right] \right)^2 \right]$$

- $\mathscr{P}_l$: uniform distribution in $[0, l_{\mathrm{end}}]$ with $l_{\mathrm{end}} = 5$
- $\mathscr{P}_{\boldsymbol{\varphi}}$: $\|\boldsymbol{\varphi}\|$ is sampled following $N(0, N/m^2)$ (w/o sign)
  $\hat{\boldsymbol{n}} = \boldsymbol{\varphi}/\|\boldsymbol{\varphi}\|$ is uniformly sampled

  * Other choices such as $N(\mathbf{0}, m^{-2}\mathbf{1})$ fail to sample the neighborhoods of $\boldsymbol{\varphi} = \mathbf{0}$ due to curse of dimensionality

- 500 collocation points are used to evaluate the expectation.
- Adam optimizer
- Pytorch
- The matrix inverse is evaluated by direct method
  - This may not be efficient but is easy to implement (torch.linalg.inv)
  - More efficient way: Hutchinson trace estimator (future work) *Hutchinson, Simul. Comp. (1990)*

Computational time & convergence

We conduct computations for all the combinations of

$N = 1,10,100$ and $\tilde{g} = 0.1,1,10$

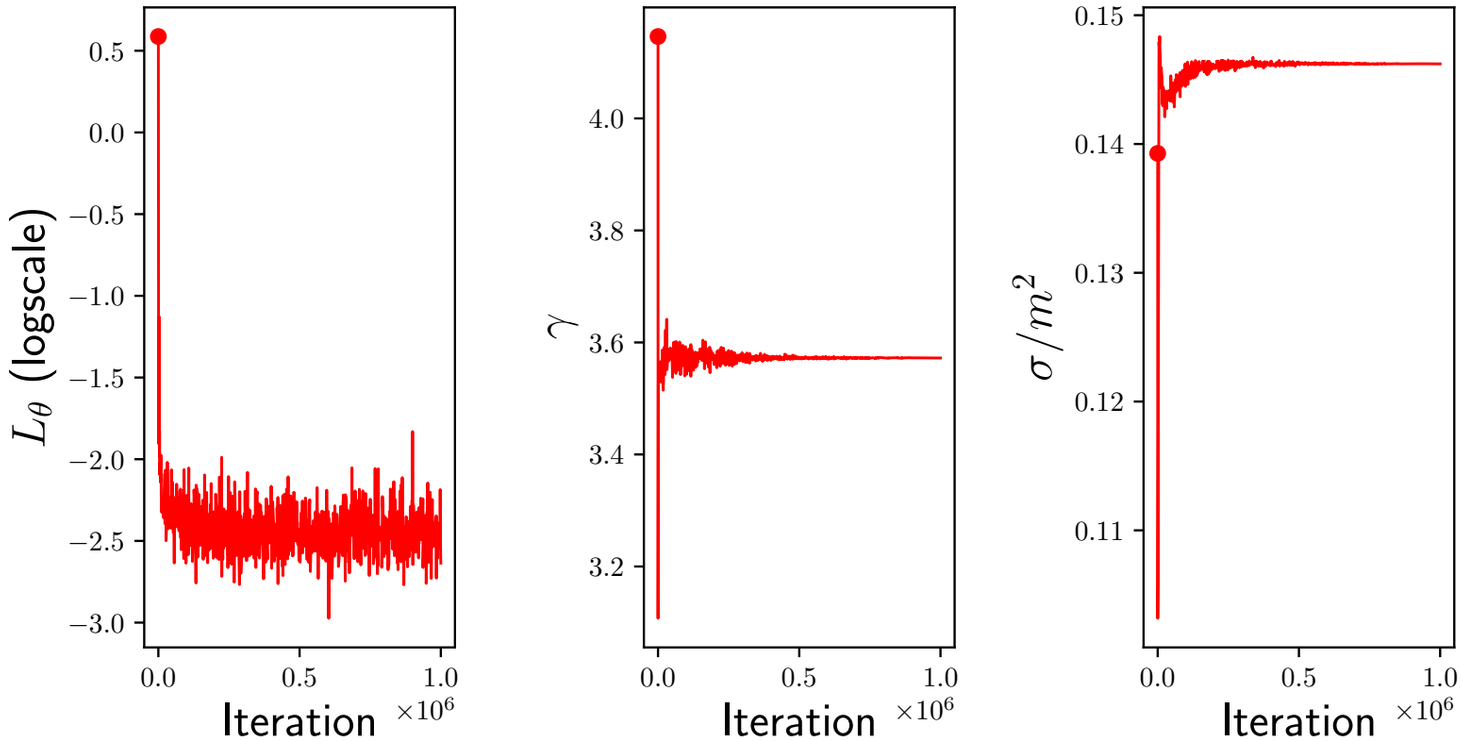
- Learning rate (Wetterich): 10^{-4} with exponential decay factor 0.99999
- Learning rate (pretraining): 10^{-3}

Comp. time on NVIDIA A100 GPU

N	1	10	100
Pretraining	4m	4m	6m
Wetterich	6h	7h	11h

Learning curve & histories of physical quantities

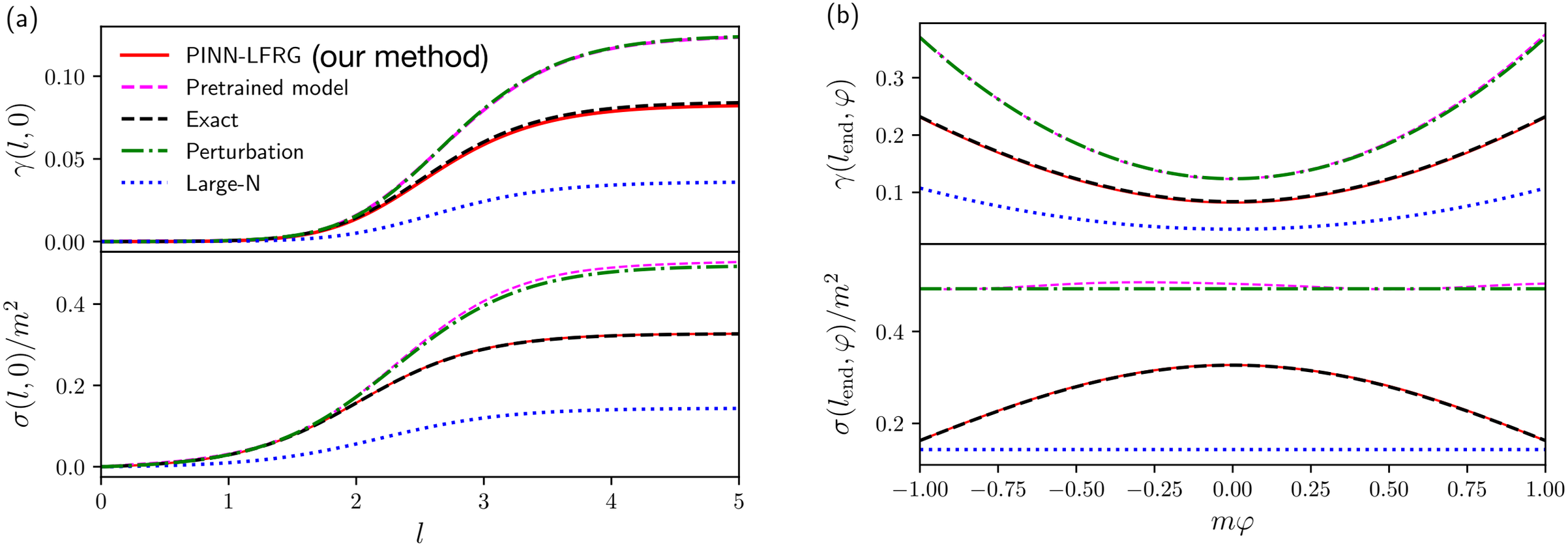
($N = 100$ & $\tilde{g} = 1$ case)



$N = 1$ and $\tilde{g} = 1$ case

RG scale l dependence at $\varphi = 0$

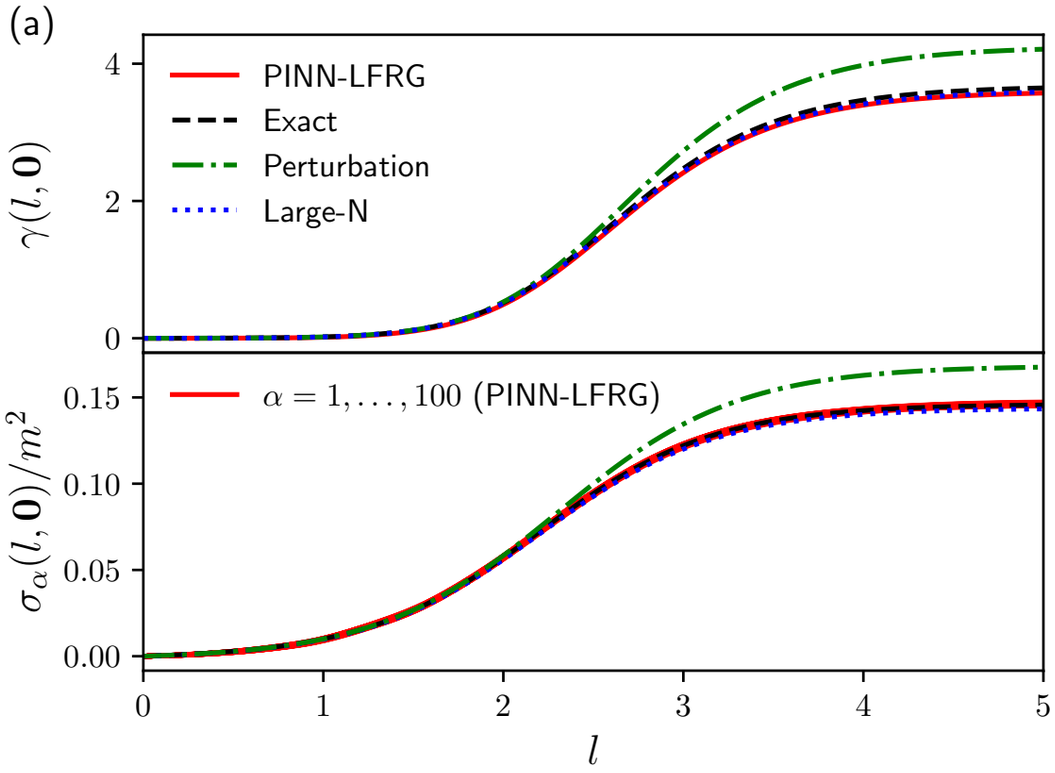
φ dependence at $l = l_{\mathrm{end}}$



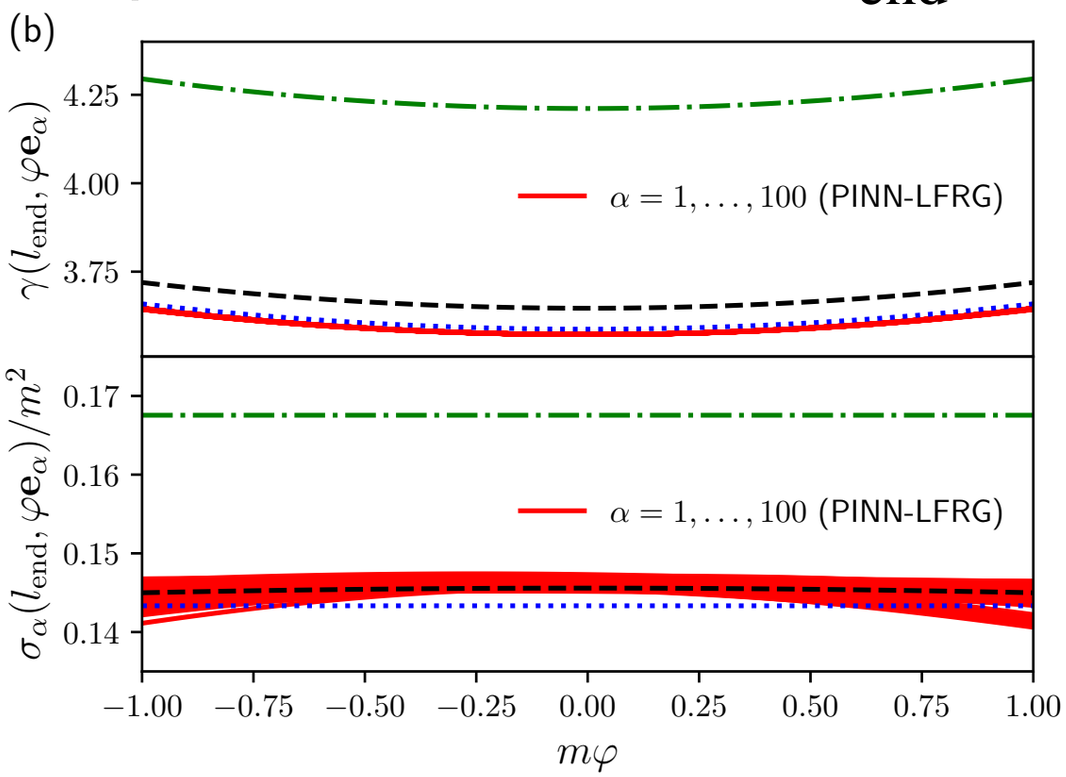
- $\gamma(l, \varphi)$ is simultaneously obtained for a domain of φ in our method (PINN-LFRG).
- PINN-LFRG shows accurate results compared to 1st-order perturbation & leading-order large-N expansion

$N = 100$ and $\tilde{g} = 1$ case

RG scale l dependence at $\varphi = 0$



φ dependence at $l = l_{\text{end}}$



Reduced self-energy $\sigma_\alpha(l, \boldsymbol{\varphi}) = \dfrac{\partial^2}{\partial \varphi_\alpha^2} \gamma(l, \boldsymbol{\varphi})$

- Except for $\gamma(l, \mathbf{0})$, PINN-LFRG results are given by $N = 100$ lines corresponding to the N-direction in $\boldsymbol{\varphi}$ space.
- PINN-LFRG shows comparable results with large-N expansion, which should be accurate for $N = 100$.
- $\mathrm{O}(N)$ symmetry is reproduced in PINN-LFRG.

Results at $\varphi = 0$ for different N and $\tilde{g}$

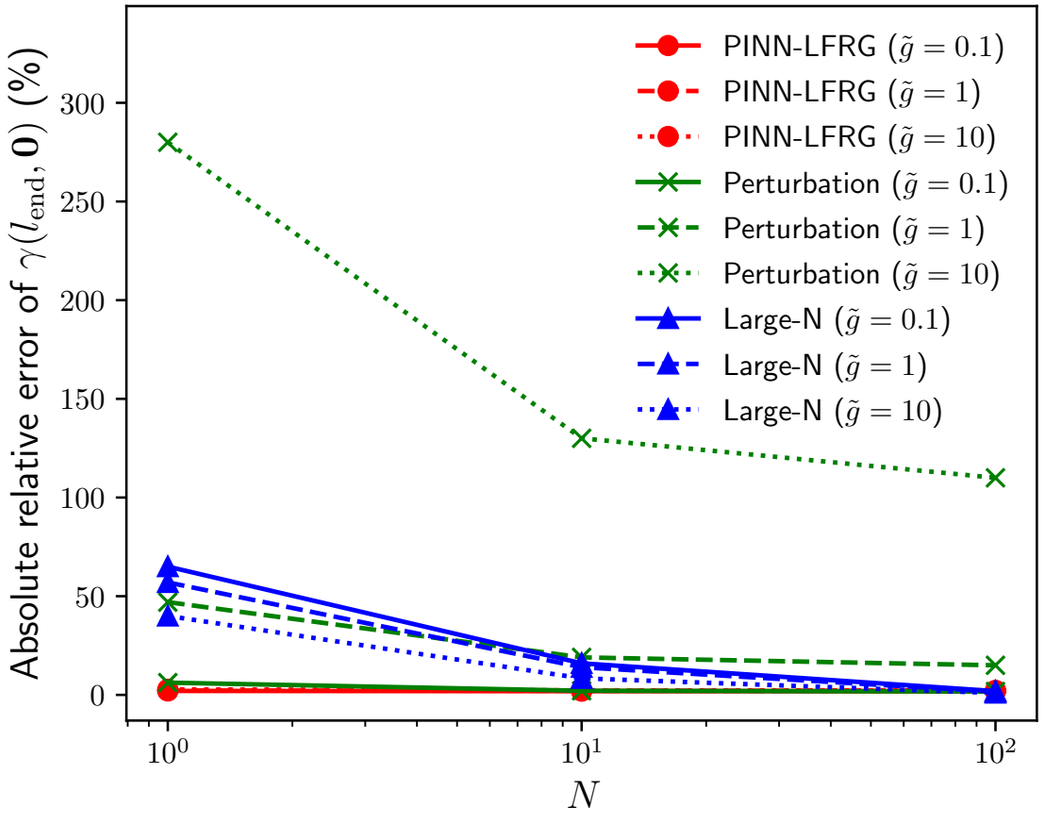
Relative error compared to exact results (minus indicates underestimation)

N		1			10			100		
$\tilde{g}$		0.1	1	10	0.1	1	10	0.1	1	10
Perturb. (%)	γ	6.2	47	275	2.1	19	129	1.7	15	110
	σ	7.6	51	228	2.3	19	109	1.7	15	92
Large-N (%)	γ	-65	-57	-40	-16	-14	-8.4	-1.9	-1.6	-0.95
	σ	-65	-56	-42	-16	-13	-8.2	-1.9	-1.5	-0.89
PINN-LFRG (%)	γ	-2.0	-2.2	-2.8	-1.9	-2.1	-2.3	-1.9	-2.0	-2.3
	σ	-0.17	0.12	0.76	0.16	0.46	0.42	-0.011	0.44	0.50
	$\Delta\sigma$	0	0	0	0.27	0.18	0.24	0.38	0.29	0.26

$$\gamma = \gamma(l_{\text{end}}, \mathbf{0})$$

$$\sigma = \frac{1}{N}\sum_{\alpha=1}^{N} \sigma_\alpha(l_{\text{end}}, \mathbf{0})$$

$$\Delta\sigma = \sqrt{\frac{1}{N}\sum_{\alpha=1}^{N} \left(\sigma_\alpha(l_{\text{end}}, \mathbf{0}) - \sigma\right)^2}$$



- For all $\tilde{g}$ and N, the errors of PINN-LFRG are within 3% for γ and 1% for σ.

- Even when $1/N$ and $\tilde{g}$ are not small, PINN-LFRG shows accurate results.

NNs are promising approximations independent of the existence of a small parameter.

Summary

Physics-informed neural network (PINN) provides a novel framework for solving the Wetterich equation with large DOF

- Addressing high-dim. PDE
- The solution is obtained simultaneously for a **domain** of the configuration space of φ rather than one configuration.

Advantageous for applications to inhomogeneous systems

The demonstration in the 0D O(N) model indicates the feasibility of calculations involving a substantial number of degrees of freedom, around 10^2 or more.

Ref: TY, Physics-informed neural network for solving functional renormalization group on lattice, arXiv:2312.16038

Outlook

- **Extension to finite-dimensional boson systems is straightforward.**

  Application: Inhomogeneous states in scalar models, such as solitons …

- **Fermion systems?**

  There is currently no efficient method for constructing NNs for Grassmann variables…

  However, one could apply our approach to fermionic systems by introducing **bosonic auxiliary fields**, for example.

An exciting application…**Density functional theory (DFT)**

Common approach for electrons, nuclei, **..,**

Effective action for density ρ (two-particle point irreducible FRG)

$$\partial_k \Gamma_\lambda[\rho] = \frac{1}{2}\int d\tau \int d\boldsymbol{x} \int d\boldsymbol{x}' U(\boldsymbol{x}-\boldsymbol{x}') \left[\rho(\tau,\boldsymbol{x})\rho(\tau,\boldsymbol{x}') + \left(\frac{\delta^2 \Gamma_\lambda[\rho]}{\delta\rho\delta\rho} \right)^{-1} (\tau,\boldsymbol{x},\tau,\boldsymbol{x}') - \rho(\tau,\boldsymbol{x})\delta(\boldsymbol{x}-\boldsymbol{x}') \right]$$

Polonyi, Sailer, PRB (2002), Schwenk, Polonyi (2004), Kemler, Braun, JPG (2013), Rentrop, Jakobs, Meden, JPA (2015), Kemler, Pospiech, Braun, JPG (2017), Liang, Niu, Hatsuda, PLB (2018), TY, Yoshida, Kunihiro PRC (2019); PTEP (2019), TY, Naito, PRB (2019), TY, Kasuya, Yoshida, Kunihiro, PTEP (2020), TY, Naito, PRR (2021), TY, Haruyama, Sugino, PRE (2021), TY, Naito, PRB (2022)