Newton Informed Neural Operator for Computing Multiple Solutions of Nonlinear Partials Differential Equations[1]

Yahong Yang

The Pennsylvania State University

NeurIPS, 2024

[1] Hao, W., Liu, X., Yang, Y. (2024). Newton Informed Neural Operator for Computing Multiple Solutions of Nonlinear Partials Differential Equations. *NeurIPS 2024*.

Newton Method for Functions

To approximate the root of $f(x) = 0$, the Newton method uses the following iteration:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \Longleftrightarrow f'(x_n)\, x_{n+1} = f'(x_n)\, x_n - f(x_n).$$

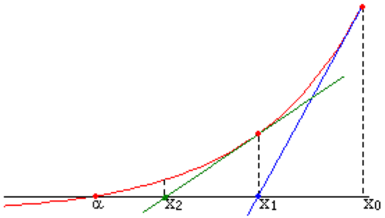


Figure: Newton method

Newton Method for Operators

For the Newton method applied to an operator, if we aim to find the solution of $\mathcal{F}(u) = 0$, the iteration can be written as:

$$\mathcal{F}'(u_n)\, u_{n+1} = \mathcal{F}'(u_n)\, u_n - \mathcal{F}(u_n) \Longleftrightarrow \mathcal{F}'(u_n)\, \delta u = -\mathcal{F}(u_n),$$

where $\delta u = u_{n+1} - u_n$.
In this context, $\mathcal{F}'(u)v$ is the (Fréchet) derivative of the operator, which is a linear operator with respect to v, defined as follows: To find $\mathcal{F}'(u)$ in $\mathcal{X}$, for any $v \in \mathcal{X}$,

$$\lim_{|v| \to 0} \frac{|\mathcal{F}(u+v) - \mathcal{F}(u) - \mathcal{F}'(u)v|}{|v|} = 0,$$

where $|\cdot|$ denotes the norm in $\mathcal{X}$.

Newton Method for Sovling PDEs

For the Newton method applied to a nonlinear PDE:

$$\begin{cases} \mathcal{L}u(\boldsymbol{x}) = f(u), & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\Omega, \end{cases}$$

where $\mathcal{F}(u) = \mathcal{L}u(\boldsymbol{x}) - f(u)$, each iteration becomes the solution of the following linear equation with respect to δu:

$$\begin{cases} (\mathcal{L} - f'(u))\delta u(\boldsymbol{x}) = -\mathcal{L}u + f(u), & \boldsymbol{x} \in \Omega, \\ \delta u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\Omega. \end{cases}$$

Newton Method for Sovling PDEs

For linear equation with respect to δu:

$$\begin{cases} (\mathcal{L} - f'(u))\delta u(\boldsymbol{x}) = -\mathcal{L}u + f(u), & \boldsymbol{x} \in \Omega, \\ \delta u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\Omega. \end{cases}$$

Assumption

(i): *For any $u \in \mathcal{X} \subset H^2(\Omega)$, we have that*

$$f'(u) \cap \{\textit{eigenvalues of } \mathcal{L}\} = \emptyset.$$

(ii): *There exists a constant F such that $\|f(x)\|_{W^{2,\infty}(\mathbb{R})} \leq F$.*
(iii): *All coefficients in $\mathcal{L}$ are C^1 and $\partial\Omega \in C^2$.*

When the above assumption holds, the map between u and δu is well-posed and continuous.

Operator Learning

Apply operator learning to approximate the map between u and δu denoted as $\mathcal{G}$.

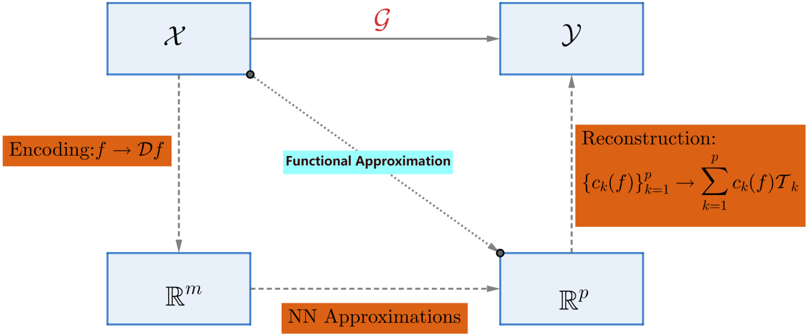


Figure: Operator learning

Operator Learning in Sobolev Training[2]

Theorem

For any $p, m \in \mathbb{N}^+$ and sufficiently large q, $\lambda \in [1,2]$, and $\mathcal{G} : \mathcal{X} \subset H^s(\mathbb{T}^d) \to H^s(\mathbb{T}^d)$ with $s > n + \frac{d}{2}$, where $\max\{\|f\|_{W^{n,\infty}(\Omega)}, \|f\|_{H^s(\Omega)}\} \le M$ for any $f \in \mathcal{X}$ and $M > 0$, satisfying proper assumption, there exist σ_2-NNs $\mathcal{T}(\boldsymbol{y}; \boldsymbol{\theta}_{2,k})$ with depth $3 + \log_2 d - 1 + \log_2 n - 1$, width $4n - 4 + 6d$, and a map $\mathcal{D}$ from $\mathcal{X} \to [-M, M]^m$, as well as σ_1-NNs $\mathcal{B}(\mathcal{D}f; \boldsymbol{\theta}_k)$ with $C_1 9^m q$ parameters such that:

$$\sup_{f \in \mathcal{X}} \left\| \mathcal{G}(f) - \sum_{k=1}^{p} \mathcal{B}(\mathcal{D}f; \boldsymbol{\theta}_k) \mathcal{T}(\boldsymbol{y}; \boldsymbol{\theta}_{2,k}) \right\|_{H^2(\Omega)} \le C \left(p^{-\frac{n-2}{d}} + p^{\frac{2}{d}} m^{-\frac{s-s'}{d}} + m^{\frac{s'}{d}} p^{\frac{2}{d}} q^{-\frac{\lambda}{m}} \right),$$

where $s' \in \left(n + \frac{d}{2}, s\right)$, and C, C_1 are independent of m, p and q. Furthermore, for $\lambda = 1$, $\mathcal{B}(\boldsymbol{z}; \boldsymbol{\theta}_k)$ is a shallow neural network that can achieve this approximation rate. As λ approaches 2, the ratio between the width and depth of $\mathcal{B}$ decreases, implying that the network structure becomes deeper.

[2] Yang, Y. (2024). DeepONet for Solving PDEs: Generalization Analysis in Sobolev Training, arXiv preprint arXiv:2410.04344.

Training Neural Operator Networks

Mean Square Loss The Mean Square Error loss function is defined as:

$$\mathcal{E}_S(\boldsymbol{\theta}) := \frac{1}{M_u \cdot M_x} \sum_{j=1}^{M_u} \sum_{k=1}^{M_x} \left| \mathcal{G}(u_j)(\boldsymbol{x}_k) - \mathcal{O}(u_j; \boldsymbol{\theta})(\boldsymbol{x}_k) \right|^2$$

where $u_1, u_2, \ldots, u_{M_u} \sim \mu$ are independently and identically distributed (i.i.d) samples in $\mathcal{X}$, and $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{M_x}$ are uniformly i.i.d samples in Ω.

Theorem

If the proper assumption holds, then the generalization error is bounded by

$$\sup_{\boldsymbol{\theta} \in [-B,B]^{d_{\boldsymbol{\theta}}}} |\mathbf{E}(\mathcal{E}_S(\boldsymbol{\theta}) - \mathcal{E}_{Sc}(\boldsymbol{\theta}))| \leqslant C \left[\frac{1}{\sqrt{M_u}} \left(1 + C d_{\boldsymbol{\theta}} \log(CB\sqrt{M_u})^{2\kappa+1/2} \right) + \frac{d_{\boldsymbol{\theta}} \sqrt{\log M_x}}{\sqrt{M_x}} \right],$$

where $\mathcal{E}_{Sc}(\boldsymbol{\theta}) = \lim_{M_u, M_x \to \infty} \mathcal{E}_S(\boldsymbol{\theta})$, C, κ are constants independent of B, $d_{\boldsymbol{\theta}}$, M_x, and M_u. Here, B represents the bound of parameters and $d_{\boldsymbol{\theta}}$ is the number of parameters.

Training Neural Operator Networks

The error $\sup_{\boldsymbol{\theta}\in[-B,B]^{d_{\boldsymbol{\theta}}}} |\mathbf{E}(\mathcal{E}_S(\boldsymbol{\theta}) - \mathcal{E}_{Sc}(\boldsymbol{\theta}))|$ suggests that the network can perform well based on the loss function $\mathcal{E}_S(\boldsymbol{\theta})$. The reasoning is as follows:
Let

$$\boldsymbol{\theta}_S = \arg \min_{\boldsymbol{\theta}\in[-B,B]^{d_{\boldsymbol{\theta}}}} \mathcal{E}_S(\boldsymbol{\theta}) \text{ and } \boldsymbol{\theta}_{S_c} = \arg \min_{\boldsymbol{\theta}\in[-B,B]^{d_{\boldsymbol{\theta}}}} \mathcal{E}_{Sc}(\boldsymbol{\theta}).$$

We aim for $\mathbf{E}\mathcal{E}_{Sc}(\boldsymbol{\theta}_S)$ to be small, which can be written as:

$$\mathbf{E}\mathcal{E}_{Sc}(\boldsymbol{\theta}_S) \le \mathcal{E}_{Sc}(\boldsymbol{\theta}_{S_c}) + \mathbf{E}(\mathcal{E}_S(\boldsymbol{\theta}_S) - \mathcal{E}_{Sc}(\boldsymbol{\theta}_S)) \le \mathcal{E}_{Sc}(\boldsymbol{\theta}_{S_c}) + \sup_{\boldsymbol{\theta}\in[-B,B]^{d_{\boldsymbol{\theta}}}} |\mathbf{E}(\mathcal{E}_S(\boldsymbol{\theta}) - \mathcal{E}_{Sc}(\boldsymbol{\theta}))|,$$

where $\mathcal{E}_{Sc}(\boldsymbol{\theta}_{S_c})$ is small, as demonstrated by Theorem 1 when B is sufficiently large.

Training Neural Operator Networks

Newton Loss Relying solely on the MSE loss function can require a significant amount of data to achieve the task. However, obtaining enough data can be challenging, especially when the equation is complex and the dimension of the input space is large. The Newton loss function is defined as:

$$\mathcal{E}_N(\boldsymbol{\theta}) := \frac{1}{N_u \cdot N_x} \sum_{j=1}^{N_u} \sum_{k=1}^{N_x} \left| (\mathcal{L} - f'(u_j(\boldsymbol{x}_k)))\mathcal{O}(u_j; \boldsymbol{\theta})(\boldsymbol{x}_k) - \mathcal{L}u_j(\boldsymbol{x}_k) - f(u_j(\boldsymbol{x}_k)) \right|^2$$

where $u_1, u_2, \ldots, u_{N_u} \sim \nu$ are independently and identically distributed (i.i.d) samples in $\mathcal{X}$, and $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{N_x}$ are uniformly i.i.d samples in Ω.

Corollary

If the proper assumption holds, then the generalization error is bounded by

$$\sup_{\boldsymbol{\theta} \in [-B,B]^{d_{\boldsymbol{\theta}}}} |\mathbf{E}(\mathcal{E}_N(\boldsymbol{\theta}) - \mathcal{E}_{Nc}(\boldsymbol{\theta}))| \leqslant C \left[\frac{1}{\sqrt{N_u}} \left(1 + C d_{\boldsymbol{\theta}} \log(CB\sqrt{N_u})^{2\kappa+1/2} \right) + \frac{d_{\boldsymbol{\theta}} \sqrt{\log N_x}}{\sqrt{N_x}} \right],$$

where $\mathcal{E}_{Nc}(\boldsymbol{\theta}) = \lim_{N_u, N_x \to \infty} \mathcal{E}_S(\boldsymbol{\theta})$ and C, κ are constants independent of B, $d_{\boldsymbol{\theta}}$, N_x, and N_u. Here, B represents the bound of parameters and $d_{\boldsymbol{\theta}}$ is the number of parameters.

Newton informed neural operator

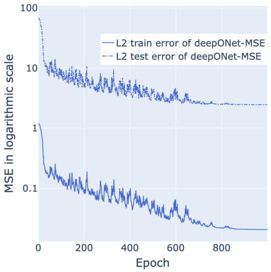
In this paper, we integrate Newton information into the loss function, defining it as follows:

$$\mathcal{E}(\boldsymbol{\theta}) := \lambda \mathcal{E}_S(\boldsymbol{\theta}) + \mathcal{E}_N(\boldsymbol{\theta}),$$

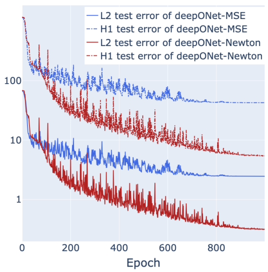
where $\mathcal{E}_N(\boldsymbol{\theta})$ represents the cost associated with the unsupervised learning data. If we lack sufficient data for $\mathcal{E}_S(\boldsymbol{\theta})$, we can adjust the parameters by selecting a small λ and increasing N_x and N_u. This strategy enables effective learning even when data for $\mathcal{E}_S(\boldsymbol{\theta})$ is limited. We refer to this neural operator, which incorporates Newton information, as the **Newton Informed Neural Operator**.

Better Generalization

We consider 2D convex problem $\mathcal{L}(u) - f(u) = 0$, where $\mathcal{L}(u) := -\Delta u$, $f(u) : -u^2 + \sin 5\pi(x+y)$ and $u = 0$ on $\partial\Omega$.



(a) The training and testing errors just using MSE.



(b) Comparison of models trained using MSE and both loss functions, the latter employing the defined loss function with $\lambda = 0.01$.

Figure: Various conditions.

Fast speed

We consider a 2D Non-convex problem,

$$\begin{cases} -\Delta u(x,y) - u^2(x,y) = -s\sin(\pi x)\sin(\pi y) \quad \text{in} \quad \Omega, \\ u(x,y) = 0, \quad \text{in} \quad \partial\Omega \end{cases}$$

where $\Omega = (0,1) \times (0,1)$.

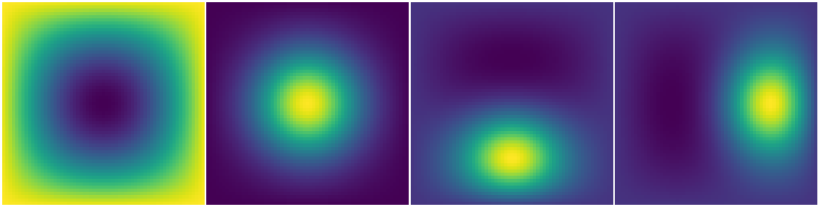


Figure: Solutions

Fast speed

We consider a 2D Non-convex problem,

$$\begin{cases} -\Delta u(x,y) - u^2(x,y) = -s\sin(\pi x)\sin(\pi y) \quad \text{in} \quad \Omega, \\ u(x,y) = 0, \quad \text{in} \quad \partial\Omega \end{cases}$$

where $\Omega = (0,1) \times (0,1)$.

Parameter	**Newton's Method**	**NINO**
Number of Streams	10	-
Data Type	float32	float32
Execution Time for 500 linear Newton systems (s)	31.52	1.1E-4
Execution Time for 5000 linear Newton systems (s)	321.15	1.4E-4

Table: Benchmarking the efficiency of Newton Informed Neural Operator

Find new solutions

The Gray-Scott model [3] describes the reaction and diffusion of two chemical species, A and S, governed by the following equations:

$$\frac{\partial A}{\partial t} = D_A \Delta A - SA^2 + (\mu + \rho)A,$$
$$\frac{\partial S}{\partial t} = D_S \Delta S + SA^2 - \rho(1 - S),$$

where D_A and D_S are the diffusion coefficients, and μ and ρ are rate constants.

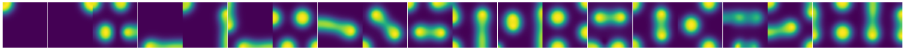
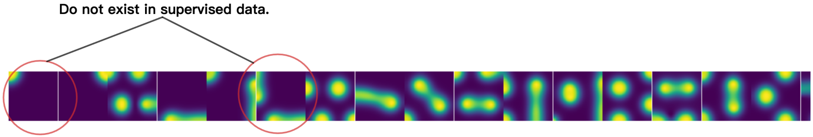
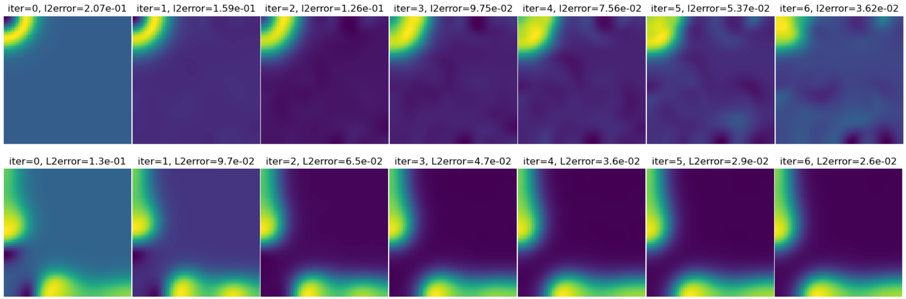


Figure: Examples of steady states of the Gray Scott model



Find new solutions



(a) Examples demonstrating how the neural operator maps the initial state to the steady state in a iterative manner

In Fig. (16), we use a ring-like pattern as the initial state to test our learned neural operator. This particular pattern does not appear in the supervised training dataset and lacks corresponding ground truth data. Despite this, our neural operator, trained using Newton's loss, is able to approximate the mapping of the initial solution to its correct steady state effectively.

Conclusion

In conclusion, we summarize the key contributions of this work as follows:

1. **Development of the Newton informed neural operator**: We introduced a novel approach by combining neural operator learning with classical Newton methods to address the challenge of solving nonlinear PDEs with multiple solutions in a well-posed manner.
2. **Theoretical and practical advantages**: Our theoretical analysis demonstrated that the proposed method can efficiently learn the Newton operator, reduce the dependency on supervised data, and discover solutions not present in the training data due to the incorporation of the Newton loss in the loss function.
3. **Experimental validation**: The experiments confirmed our theoretical results, showing that the Newton informed neural operator requires fewer supervised data, identifies previously unseen solutions, and achieves these results with reduced computational time compared to classical Newton methods.

Thank you for your listening!