J Syst Sci Complex (2024) 37(2): 413–440

NSNO: Neumann Series Neural Operator for Solving Helmholtz Equations in Inhomogeneous Medium*

CHEN Fukai · LIU Ziyang · LIN Guochang · CHEN Junqing · SHI Zuoqiang

DOI: 10.1007/s11424-024-3294-x
Received: 31 July 2023 / Revised: 14 October 2023

Abstract In this paper, the authors propose Neumann series neural operator (NSNO) to learn the solution operator of Helmholtz equation from inhomogeneity coefficients and source terms to solutions. Helmholtz equation is a crucial partial differential equation (PDE) with applications in various scientific and engineering fields. However, efficient solver of Helmholtz equation is still a big challenge especially in the case of high wavenumber. Recently, deep learning has shown great potential in solving PDEs especially in learning solution operators. Inspired by Neumann series in Helmholtz equation, the authors design a novel network architecture in which U-Net is embedded inside to capture the multiscale feature. Extensive experiments show that the proposed NSNO significantly outperforms the state-of-the-art FNO with at least 60% lower relative L^2-error, especially in the large wavenumber case, and has 50% lower computational cost and less data requirement. Moreover, NSNO can be used as the surrogate model in inverse scattering problems. Numerical tests show that NSNO is able to give comparable results with traditional finite difference forward solver while the computational cost is reduced tremendously.

Keywords Helmholtz equation, inverse problem, neumann series, neural network, solution operator.

CHEN Fukai
Department of Mathematical Sciences, Tsinghua University, Beijing 100084, *China*.
Email: cfk19@mails.tsinghua.edu.cn.
LIU Ziyang · LIN Guochang
Yau Mathematical Sciences Center, Tsinghua University, Beijing 100084, *China.*
Email: zy-liu20@mails.tsinghua.edu.cn; lingc19@mails.tsinghua.edu.cn.
CHEN Junqing
Department of Mathematical Sciences, Tsinghua University, Beijing 100084, *China.*
Email: jqchen@tsinghua.edu.cn.
SHI Zuoqiang (Corresponding author)
Yau Mathematical Sciences Center, Tsinghua University, Beijing 100084, *China; Yanqi Lake Beijing Institute of Mathematical Sciences and Applications, Beijing* 101408, *China.* Email: zqshi@tsinghua.edu.cn.
*This research was supported by the National Science Foundation of China under Grant No. 92370125 and the National Key R&D Program of China under Grant Nos. 2019YFA0709600 and 2019YFA0709602.
◇ *This paper was recommended for publication by Guest Editor YAN Zhenya.*

1 Introduction

Helmholtz equation is a fundamental partial differential equation (PDE) describing wave propagation in many areas of physics and engineering such as acoustics, electromagnetics[1], and medical imaging[2]. One of the most common scenarios is the propagation of waves in medium with spatially varying properties, which frequently occurs in the scattering of acoustic and electromagnetic waves[3], such that the governing equation is given by

$$\Delta u + k^2(1 + q(x))u = f(x), \tag{1}$$

where u is the scalar field representing the wave, $f(x)$ is the source term, $q(x)$ is the coefficient representing the inhomogeneity, which is usually compactly supported.

Traditional numerical methods for solving Helmholtz equation in inhomogeneous medium numerically such as finite difference method (FDM)[4] and finite element method (FEM)[5] require a fine grid or mesh to capture the high-frequency components of the solution when the wavenumber is large, which leads to a large scale indefinite linear system. Modern methods such as Krylov subspace methods[6] for solving large indefinte linear systems are known to have a slow convergence rate for Helmholtz problem, especially in inhomogeneous medium case[7]. Moreover, in inverse scattering problems, the Helmholtz equation (1) has to be solved repeatedly with different coefficients q and source terms f, which will be computationally intolerable if these equations are solved separately. Therefore, solving the inhomogeneous Helmholtz equation still remains a challenging task and it is of significant importance to seek for the solution operator mapping the coefficient q and source term f simultaneously to the solution to the Helmholtz equation (1).

Benefiting from the attractive capability of neural networks in approximating functions[8], deep learning has made remarkable progress in areas such as image recognition and natural language processing[9, 10]. Therefore, deep learning is recognized as a promising solution to solve partial differential equations and the solution operator from the parameter space to the solution space[11, 12]. By using neural networks with numerous learnable parameters as an ansatz to represent the solution or the solution operator, and training neural networks based on properly designed loss functions, the neural network is able to approximate the required solution or the solution operator.

In the literature, both solving a single PDE and PDE solution operators have attracted wide attention. In [13], the physics-informed neural network (PINN) taking the residual of the PDEs as the loss function to train the neural network is presented. Based on PINN, a neural network based solver for Helmholtz equations in homogeneous background has been proposed in [14], while plane wave activation functions are utilized in [15] to further improve accuracy. For PDE solution operators, two general neural operator framework named DeepONet and Fourier neural operator (FNO) have been developed in [16–18]. In DeepONet, the features in spatial coordinates and differential equation parameters are extracted separately by the trunk net and the branch net, and are then combined by dot product, while in FNO, the mapping from parameter space to solution space is formulated as an iterative integral, where

the integral kernel is parameterized in Fourier space. Furthermore, physics-informed loss is incorporate with DeepONet and FNO in [19] and [20], respectively, to explore the possibility of unsupervised solution operator learning. A solution operator to the heterogeneous Helmholtz equation mapping the sound speed distribution to the acoustic wavefield is approximated by neural networks based on an iterative scheme in [21].

However, two important issues are not fully considered in existing neural network based methods. For one thing, most existing solution operators either map the coefficient in the differential operator or the source term only to the solution. Since the coefficient q and the source term f in the Helmholtz equation (1) belong to different function spaces, it is a non-trival task to learn the solution operator mapping q and f simultaneously to u. For the other, fitting the high oscillations in the solution to the Helmholtz equation with large wavenumber always leads to unstable or even divergent training process[22]. Therefore, new network architecture should be designed to capture the multi-scale features in the solution to Helmholtz equations.

To address the two issues above, in the paper, we propose Neumann series neural operator (NSNO). Specifically, the solution to Helmholtz equation is rewritten in the form of a Neumann series. Each term in the Neumann series is the solution to a Helmholtz equation in homogeneous medium subject to different source terms. For the Helmholtz equation corresponding to the first term, the source term is exactly f, while q only appears in the source terms of the Helmholtz equations corresponding to the remaining terms, such that q and f are fully decoupled. Moreover, based on the Neumann series, we only need to solve the operator from the source term to the solution of Helmholtz equation in homogeneous medium. Instead of using the Fourier neural operator (FNO) directly to approximate the operator, we propose a novel network architecture combining FNO and U-Net[23] named UNO to capture the multi-scale property of the solution. By extracting and fusing information from different spatial resolutions, the proposed UNO is able to approximate the solution accurately even in cases of large wavenumbers. Our main contributions can be summarized as follows:

- We propose NSNO, a novel framework for neural operators mapping inhomogeneity coefficients and source terms simultaneously to the solution to Helmholtz equation in inhomogeneous medium based on Neumann series. To capture the multi-scale properties of Helmholtz equation, a novel network architecture UNO combining FNO and U-Net is also proposed as the building blocks of NSNO.
- Extensive numerical experiments show that NSNO significantly outperforms the state-of-the-art FNO with at least 60% lower L^2-error. The results also show that compared with FNO, the proposed UNO architecture is more efficient (50% less in computational cost), more accurate on multi-scale problems and has less training data requirement.
- We use NSNO as the surrogate model for the forward operator in inverse scattering problem where the scatterer is sampled from the MNIST dataset. NSNO is able to give comparable results with the traditional finite difference forward solver with over 20 times faster in speed.

The rest of this paper is organized as follows. The operator learning problem is setup in Section 2. In Section 3, we show the Neumann series reformulation and analyze the convergence of the Neumann series. The detailed network architecture and training process is presented in Section 4. Numerical experiments are shown in Section 5, and the application of NSNO in inverse scattering problem is given in Section 6. At last, conclusion remarks are made in Section 7.

2 Problem Setup

Consider the 2-dimensional Helmholtz equation in inhomogeneous medium subject to the Sommerfeld radiation condition at infinity[1],

$$\Delta u + k^2(1+q(x))u = f(x), \quad x \in \mathbb{R}^2, \tag{2}$$

$$\lim_{r\to\infty} \sqrt{r}\left(\frac{\partial u}{\partial r} - \mathrm{i}ku\right) = 0, \quad r = |x|, \tag{3}$$

where k is the wavenumber, $\mathrm{i} = \sqrt{-1}$ is the imaginary unit, $q(x)$ is the compactly supported coefficient representing the inhomogeneity, and $f(x)$ is the source term. In practice, the problem is usually reduced to a bounded domain Ω containing the support of q by introducing an artificial surface. For simplicity, we take Ω as a rectangle and employ the first-order absorbing boundary condition[24]:

$$\frac{\partial u}{\partial n} - \mathrm{i}ku = 0, \quad \text{on } \partial\Omega. \tag{4}$$

Note that in applications such as the inverse scattering problem, we normally needs multiple incident fields to reconstruct the scatterer q better, which can be computational expensive if the forward equations are solved case-by-case. Therefore, in this paper, we aim to learn the following operator:

$$\begin{aligned} &\mathcal{S} : L^\infty(\Omega) \times L^2(\Omega) \to L^2(\Omega), \\ &(q, f) \mapsto u, \end{aligned} \tag{5}$$

where u is the solution to

$$\Delta u + k^2(1+q(x))u = f(x), \quad \text{in } \Omega, \tag{6}$$

$$\frac{\partial u}{\partial n} - \mathrm{i}ku = 0, \quad \text{on } \partial\Omega. \tag{7}$$

3 Neumann Series

Note that for the solution operator $\mathcal{S}$, the input q and f belonging to different function spaces are coupled together, making it difficult to solve the solution operator mapping q and f simultaneously to u. Therefore, we consider utilizing the Neumann series to decouple q and f, such that the solution operator $\mathcal{S}$ is transformed into another operator G mapping the source term only to the solution. In this section, we first present some preliminary results on the variational formulation and stability estimate of Helmholtz equation, based on that the Neumann series is defined and its convergence property is analyzed.

3.1 Preliminaries

Suppose Ω is a rectangle in $\mathbb{R}^2$. Consider the Helmholtz equation

$$\begin{aligned} &\Delta u + k^2 u = g(x), \quad \text{in } \Omega, \\ &\frac{\partial u}{\partial n} - \mathrm{i}ku = 0, \quad \text{on } \partial\Omega. \end{aligned} \tag{8}$$

Its variational formulation is defined as

$$\begin{aligned} &\text{Find} \quad u \in H^1(\Omega) \quad \text{such that} \\ &a(u, v) = (g, v), \quad \forall v \in H^1(\Omega), \end{aligned} \tag{9}$$

where

$$a(u, v) = (\nabla u, \nabla v) - k^2(u, v) - \mathrm{i}k\langle u, v\rangle, \tag{10}$$

$(\cdot, \cdot)$ and $\langle \cdot, \cdot \rangle$ denote the L^2-inner product on Ω and $\partial\Omega$, respectively.

The existence of the solution to the variational problem (9) and the corresponding stability estimate are presented in the following theorem.

Theorem 3.1 *The variation problem* (9) *has a unique solution in* $H^1(\Omega)$*. Moreover, if the wavenumber* $k > 1$*, there exists a constant* C*, which only depends on the domain* Ω*, such that for any* $f \in L^2(\Omega)$*, the solution to the problem* (9) *satisfies*

$$k\|u\|_{L^2(\Omega)} \leq C\|g\|_{L^2(\Omega)}. \tag{11}$$

The proof of Theorem 3.1 can be found in [25]. Based on Theorem 3.1, we can define an operator G by

$$\begin{aligned} &G : L^2(\Omega) \to H^1(\Omega) \subset L^2(\Omega), \\ &g \mapsto u, \end{aligned} \tag{12}$$

such that u is the solution to the variational problem (9), i.e., the weak solution to the original Helmholtz equation. It can be seen that G is linear and bounded in $L^2(\Omega)$ with $\|G\|_{L^2(\Omega)} \leq C/k$.

3.2 Neumann Series

Note that the Helmholtz equation (6) can be rewritten as $\Delta u + k^2 u = f - k^2 q u$, which inspires the following iterative scheme

$$\Delta u_{n+1} + k^2 u_{n+1} = f - k^2 q u_n. \tag{13}$$

By the definition and linearity of G, the iterative scheme can be rewritten as

$$u_{n+1} = G(f - k^2 q u_n) = G(f) + G(-k^2 q u_n) = u_0 - (k^2 G q) u_n, \tag{14}$$

where $u_0 = G(f)$. By using the iterative scheme (13) recursively for N steps, we obtain the Neumann series

$$u_N = u_0 - (k^2 G q) u_0 + (k^2 G q)(k^2 G q) u_0 + \cdots + (-k^2 G q)^N u_0. \tag{15}$$

The convergence of this Neumann series is presented in the following theorem.

Theorem 3.2 (Convergence of Neumann series) *When $\|q\|_{L^\infty(\Omega)}$ is sufficiently small, the Neumann series* (15) *converges in $L^2(\Omega)$ as $N \to \infty$.*

Proof It suffices to prove $\| - k^2 Gq\|_{L^2(\Omega)} < 1$. Note that

$$\| - k^2 Gq\|_{L^2(\Omega)} = \sup_{g \in L^2(\Omega)} \frac{\| - k^2 G(qg)\|_{L^2(\Omega)}}{\|g\|_{L^2(\Omega)}} \leq \sup_{g \in L^2(\Omega)} \frac{Ck\|qg\|_{L^2(\Omega)}}{\|g\|_{L^2(\Omega)}} \leq Ck\|q\|_{L^\infty(\Omega)}. \quad (16)$$

It can be seen that the Neumann series converges as long as $\|q\|_{L^\infty(\Omega)} < \frac{1}{Ck}$. ∎

Therefore, based on the Neumann series (15), solving the solution operator $\mathcal{S}$ from (q, f) to u can be transformed into solving G, which is the solution operator from the source term only to the solution of Helmholtz equation.

4 Network Architecture and Training Process

In this section, we first give an overall illustration of the overall network architecture based on the Neumann series (15). Following that, we further elaborate on the approximation of operator G by neural networks. Finally, we discuss the training process, where physical-informed loss is utilized to further improve the performance of the learned operator.

4.1 Network Architecture

Figure 1 shows the overall network architecture of the proposed Neumann series neural operator, referred to as NSNO, where the Neumann series is truncated to $N+1$ items[†]. The operator G is approximated by $N+1$ separate neural networks $G_{\boldsymbol{\theta}} = \{G_{\theta_0}, G_{\theta_1}, \cdots, G_{\theta_N}\}$ with learnable parameters $\boldsymbol{\theta} = \{\theta_0, \theta_1, \cdots, \theta_N\}$, respectively. Note that although the neural networks $\{G_{\theta_0}, G_{\theta_1}, \cdots, G_{\theta_N}\}$ are all approximation of the same operator G, we still use different parameters since compared with the neural network reusing the same parameter multiple times, using different parameters gives the neural network stronger representation ability. The first neural network takes f as the input and outputs the approximation of the first item $v_0 \approx u_0$ in Neumann series (15). For the rest N neural networks, G_{θ_n} takes the multiplication of $-k^2 q$ and the output of the last neural network v_{n-1} as the input, and outputs the approximation of the $(n+1)$-th item $v_n \approx (-k^2 Gq)^n u_0$ in Neumann series (15).

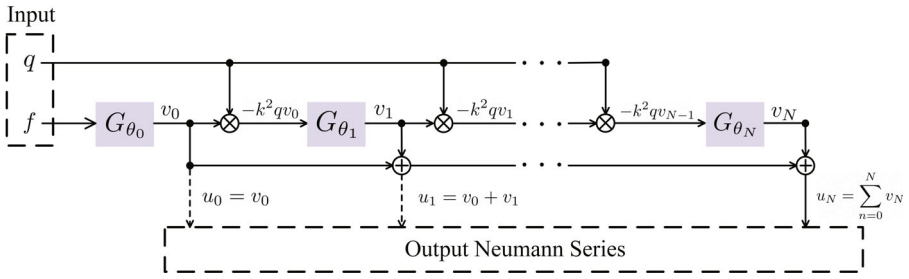


Figure 1 Overall network architecture of NSNO

[†]In the experiments, we have found that $N = 2$, i.e., only three items to construct the Neumann series, is sufficient to give an accurate result, as will be discussed in Subsection 5.5.

In practice, the domain $\Omega \in \mathbb{R}^2$ is discretized by a regularly spaced Cartesian grid of dimension $H \times W$ such that functions defined on Ω can be represented by tensors. The input and output of $G_{\boldsymbol{\theta}}$ has the same spatial dimensions as the grid and has two channels in most cases representing the real and imaginary parts. For G_{θ_0} the input has only one channel if f is real. The detailed structure of $G_{\boldsymbol{\theta}}$ will be specified in Subsection 4.2.

In summary, NSNO takes the tuple (q, f) as input, where f will only be fed into G_{θ_0}, while $-k^2 q$ will be multiplied with the outputs of $G_{\theta_0}, G_{\theta_1}, \cdots, G_{\theta_{N-1}}$. In this way, q and f belonging to different function spaces can be fully decoupled. The output of NSNO is the sum of the outputs of the $N+1$ neural networks $G_{\boldsymbol{\theta}}$. Besides, the intermediate results $u_0, u_1, \cdots, u_{N-1}$ can also be output if required, adding more flexibility to NSNO.

4.2 Network Architecture of $G_{\boldsymbol{\theta}}$

In this subsection, we introduce two types of network architecture for $G_{\boldsymbol{\theta}}$, i.e., the Fourier neural operator (FNO), which is a direct application of [17], and the U-shaped neural operator, which is a combination of the UNet architecture[23] and the FNO.

4.2.1 Fourier Neural Operator

One natural idea is to choose $G_{\boldsymbol{\theta}}$ as the Fourier neural operator (FNO) directly. As is shown in Figure 2, the architecture of FNO starts with a lifting layer, followed by a series of Fourier layers, and is ended with a projection layer.

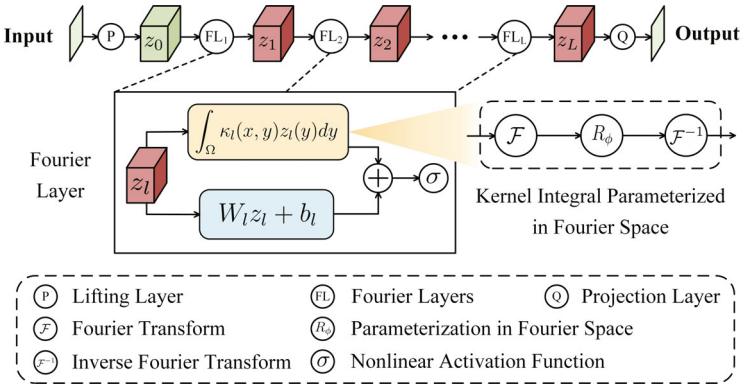


Figure 2 Network architecture of FNO

The lifting layer lifts the input to a higher dimension by a fully connected neural network P such that z_0 takes values in $\mathbb{R}^C$. A series of iterative Fourier layers is then applied to z_0, resulting in a sequence of functions $z_0 \mapsto z_1 \mapsto \cdots \mapsto z_L$ taking values in $\mathbb{R}^C$. Specifically, the iterative scheme is given by

$$z_{l+1}(x) = \sigma\left(\int_\Omega \kappa_l(x,y)z_l(y)dy + W_l z_l(x) + b_l\right), \quad l = 0, 1, \cdots, L-1, \tag{17}$$

where κ_l is an integral kernel, W_l is a linear transform, b_l is the bias, which are all learnable, and σ is a nonlinear activation function. By letting $\kappa_l(x,y) = \kappa_l(x-y)$ and using the convolution

therorem, the kernel integral in (17) can be rewritten as

$$\int_{\Omega} \kappa_l(x,y) z_l(y) dy = \mathcal{F}^{-1}\left(\mathcal{F}(\kappa_l) \cdot \mathcal{F}(z_l)\right)(x) := \mathcal{F}^{-1}\left(R_\phi \cdot \mathcal{F}(z_l)\right)(x), \tag{18}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier transform and its inverse, respectively, $R_\phi := \mathcal{F}(\kappa_l)$ is the parameterization of κ_l in the Fourier space with parameters ϕ. In the discrete case, the Fourier series is truncated such that higher modes are filtered and only $k_{\max}$ modes are reserved. Therefore, R_ϕ can be directly parameterized as a complex-valued $k_{\max} \times C \times C$ tensor. Finally, z_L is projected back to the output space with required dimension with another fully connected neural network Q.

4.2.2 U-Shaped Neural Operator

The architecture of FNO enables its superior accuracy in various applications, such as Burger's equation, Darcy flow, and so on. However, new challenges arise in solving Helmholtz equations with multi-scale features, especially in the high wavenumber regime due to the couple between the high frequency wave solutions and the numerical grid[26]. Building on global Fourier transform filtering higher frequency modes, FNO tends to learn an over-smooth solution. In multi-scale problems, FNO fails in capturing the intrinsic multis-cale features in the solution, leading to poor performance.

To address this issue, we utilize the U-Net structure widely used in image processing and computer vision[27, 28]. By using a hierarchical encoder-decoder structure and skip connections, U-Net is able to effectively capture and merge information at different scales, thus having capability to solve multi-scale problems. Therefore, we design a novel network architecture for $G_{\boldsymbol{\theta}}$ combining both U-Net and FNO structure termed UNO, where U-Net exploits the multi-scale structure of the solutions, while FNO achieves the transformation from the source term space to the solution space. Specifically, as is shown in Figure 3, UNO consists of three modules, i.e., the multiple-input multiple-output (MIMO) encoder, Fourier layers as skip connections and the multiple-input single-output (MISO) decoder, which will be detailed below.

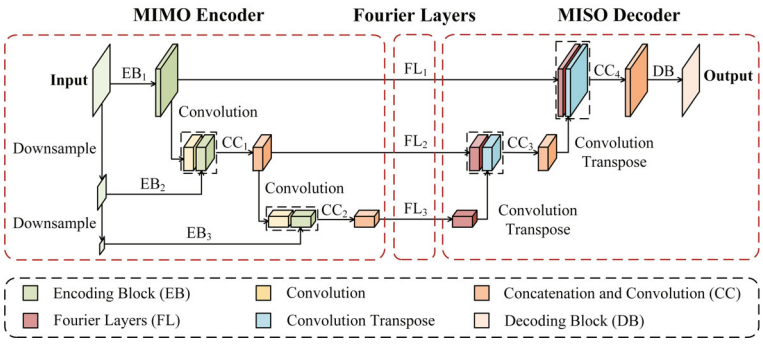


Figure 3 Network architecture of UNO

Multiple-Input Multiple-Output (MIMO) Encoder In the MIMO encoder module, we firstly downsample the input twice to spatial dimensions $\frac{H}{2} \times \frac{W}{2}$ and $\frac{H}{4} \times \frac{W}{4}$, respectively.

The original input and the two downsampled inputs are fed into three encoding blocks to extract features from different scales. As is shown in Figure 4, in the encoding blocks, the input is firstly fed into a double convolution module consists of two 3×3 convolution layers interleaved by a non-linear activation function, for which we choose GELU[29] in this paper. The output of the double convolution module is concatenated with the positional encoding, for which we simply take the Cartesian coordinates of the grid, and then passed to a shallow convolution module (SCM), in which we use two stacks of 3×3 and 1×1 convolutional layers[30].

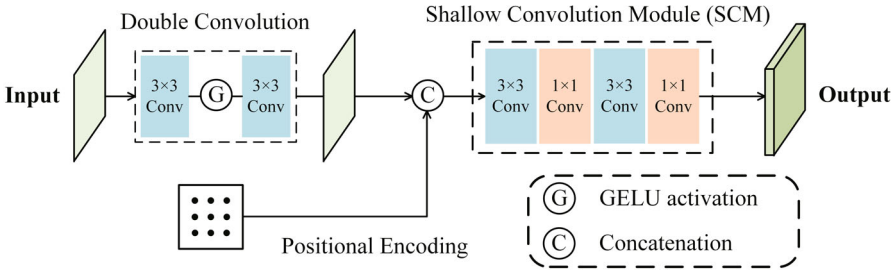


Figure 4 Network architecture of the encoding block in UNO

Moreover, the output of the EB block is combined with the feature extracted from the downsampled input, realizing the fusion of features in different scales. Specifically, the output of the EB block at the $(k-1)$-th level $\mathrm{EB}_{k-1}^{\mathrm{out}}$ is fed to a convolution layer with a stride of 2 and a doubled number of output channel, resulting in $(\mathrm{EB}_{k-1}^{\mathrm{out}})^{\downarrow}$, which is of the same size as $\mathrm{EB}_{k}^{\mathrm{out}}$ at the k-th level. The two tensors $(\mathrm{EB}_{k-1}^{\mathrm{out}})^{\downarrow}$ and $\mathrm{EB}_{k}^{\mathrm{out}}$ are thus concatenated and passed to a 1×1 convolution layer, realizing the integration of the feature extracted from both scales.

Fourier Layers Three separate Fourier layers defined in (17) serve as the skip connections in UNO, transforming the outputs of the MIMO encoder at different scales from the source term space to the solution space. Note that the channel dimension C differs in the three scales. The scale with finer mesh corresponds to fewer channels, which further improves efficiency by reducing the memory usage and computation cost in larger scales.

Multiple-Input Single-Output (MISO) Decoder The MISO decoder module takes the three outputs from the Fourier layers at different scales as input. The output of the Fourier layers at the k-th level $\mathrm{FL}_{k}^{\mathrm{out}}$ is fed to a convolution transpose layer with a stride of 2 and a halved number of output channel, resulting in $(\mathrm{FL}_{k-1}^{\mathrm{out}})^{\uparrow}$, which is of the same size as $\mathrm{FL}_{k-1}^{\mathrm{out}}$ at the $(k-1)$-th level. The two tensors $(\mathrm{FL}_{k}^{\mathrm{out}})^{\uparrow}$ and $\mathrm{FL}_{k-1}^{\mathrm{out}}$ are then concatenated and passed to a 1×1 convolution layer. In this way, the features extracted at different scales are merged together and finally passed to a decoding block, for which we simply choose a 3×3 convolution layer in this paper.

4.3 Training Process

In the training process, suppose that the training set contains N_{train} tuples of (q, f, u) sampled from the same distribution. The data loss function evaluates the discrepancy between the exact solution and the solution obtained by NSNO, which is given by

$$\mathcal{L}_{\text{data}} = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \|u_i - \widehat{u}_i\|_{L^2(\Omega)}, \tag{19}$$

where $u_i = \mathcal{S}(q_i, f_i)$ is the exact solution to (6) corresponding to (q_i, f_i), and $\widehat{u}_i = \text{NSNO}(q_i, f_i)$ is the numerical solution obtained by the proposed NSNO. Furthermore, to avoid overfitting and improve the generalization ability of the model, we also introduce the physics-informed loss to minimize the violation of the Helmholtz equation (6):

$$\mathcal{L}_{\text{pde}} = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \|\Delta\widehat{u}_i + k^2(1 + q_i)\widehat{u}_i - f_i\|_{L^2(\Omega)}, \tag{20}$$

where $\Delta\widehat{u}_i$ is computed by a five-point finite difference scheme. The total loss function for training the propsed NSNO is defined as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda\mathcal{L}_{\text{pde}}, \tag{21}$$

where λ is the weight balancing the two loss functions.

5 Experiments

In this section, a series of experiments are conducted under various data distributions and parameters settings to evaluate the performance of the proposed NSNO. Firstly we present the experiment setup and dataset generation. Following that, we show the benchmark results on the Helmholtz equation solution operator learning problem to demonstrate the effectiveness of the proposed NSNO. Besides, we discuss the necessity of introducing the physics-informed loss and the convergence properties of the Neumann series.

5.1 Experiment Setup

The basic settings of our experiments are listed below.

- **Domain discretization** In the experiments, the spatial domain Ω is set as $\Omega = [0, 1]^2$ and is discretized uniformly to a 256×256 grid.
- **Model hyperparameters** Unless otherwise specified, for the Fourier layers in FNO, we set $k_{\max} = 12$ and channel dimension $C = 32$ as is in the original FNO paper, and the number of iterations is set as $L = 4$. For the three Fourier layers at different scales in UNO, we also set $k_{\max} = 12$ while the channel dimension is set as $C = 8, 16, 32$ from the fine level to the coarse level, respectively. Besides, the number of iterations is set as $L = 3$ such that UNO has similar number of parameters as FNO. We have also found that using only three Neumann iterations steps is sufficient to give a satisfactory result.

- **Training hyperparamters** Unless otherwise specified, the training set has 1000 instances while the test set has 100 testing instances. We use the Adam optimizer[31] to train the neural network for 500 epochs with an initial learning rate of 0.001 that is halved every 100 epochs. The batchsize is set as 20. The weight λ in the loss function (21) is set as 0.05. All the experiments are conducted on a single Nvidia V100 GPU with 32 GB memory.

- **Performance evaluation** We use the average relative L^2-error on test sets defined as $\frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}\frac{\|u_i-\widehat{u}_i\|_{L^2(\Omega)}}{\|u_i\|_{L^2(\Omega)}}$ to evaluate the performance of the neural operators, where $u_i = \mathcal{S}(q_i, f_i)$ is the exact solution and $\widehat{u}_i = \text{NSNO}(q_i, f_i)$ is the numerical solution.

5.2 Benchmark Models and Dataset Generation

5.2.1 Benchmark Models

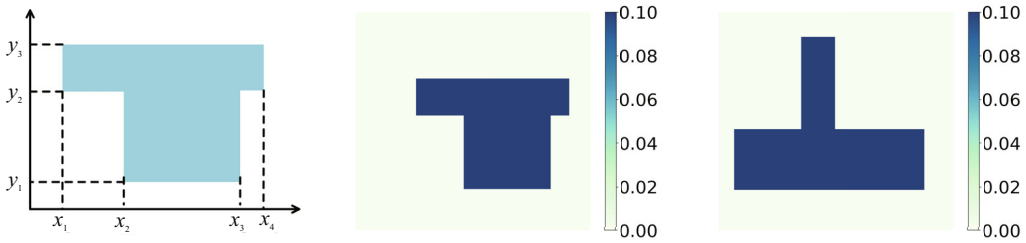
Note that instead of using the Neumann series to decouple q and f, another way is to directly use FNO or UNO to solve the solution operator mapping the tuple (q, f) to u. Therefore, by choosing whether to use the Neumann series and the proposed UNO architecture, the following four models are considered.

- **FNO:** Directly use FNO to learn the mapping from (q, f) to u.
- **UNO:** Directly use UNO to learn the mapping from (q, f) to u.
- **NS-FNO:** NSNO with $G_{\boldsymbol{\theta}}$ chosen as FNO.
- **NS-UNO:** NSNO with $G_{\boldsymbol{\theta}}$ chosen as UNO.

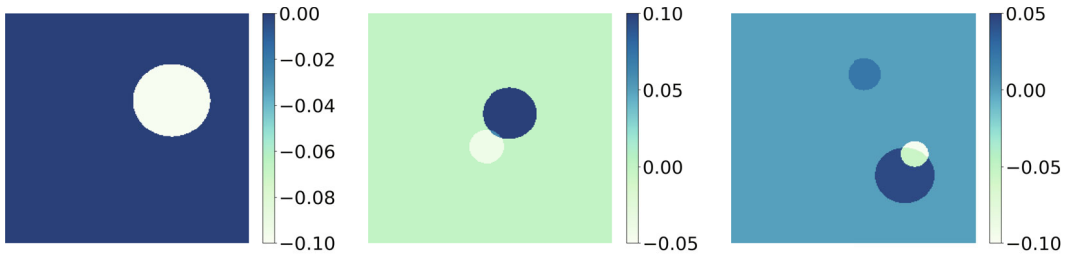
5.2.2 Dataset Generation

We generate various datasets for the coefficient q and source field f to give a thorough evaluation on the performance of the proposed models. For each dataset, we generate q and f separately from one of the following distributions.

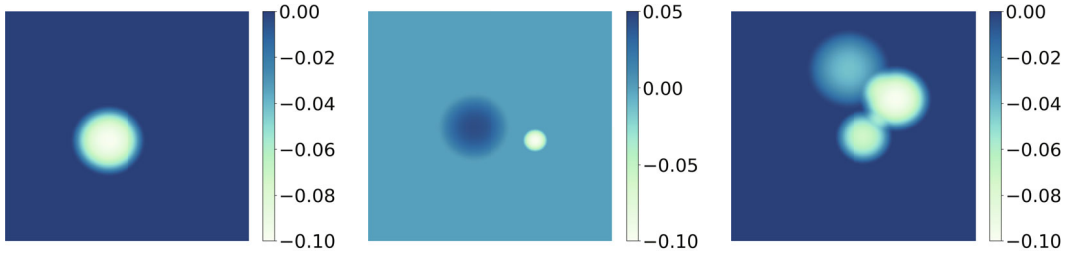
The three distributions for q are listed below, along with examples sampled from each distribution shown in Figure 5.



(a) Illustration of the T-shaped q (b) T-shaped, example 1 (c) T-shaped, example 2



(d) Random Circle, example 1 (e) Random Circle, example 2 (f) Random Circle, example 3



(g) Smoothed Circle, example 1 (h) Smoothed Circle, example 2 (i) Smoothed Circle, example 3

Figure 5 Examples of q. (a)–(c): T-shaped distribution. (d)–(f): Random circle distribution with the number of circles from 1 to 3. (g)–(i): Smoothed random circle distribution with the number of circles from 1 to 3

- **T-shaped:** As is illustrated in Figure 5(a), for T-shaped distribution, q is compactly supported with randomly generated T-shaped support. Specifically, four points are generated uniformly in [0.05, 0.95] and sorted, denoted as $x_1 < x_2 < x_3 < x_4$ as the coordinates of boundary points in the x-direction, then three points are generated uniformly in [0.05, 0.95] and sorted, denoted as $y_1 < y_2 < y_3$ as the coordinates of boundary points in the y-direction. The support of q is then taken as $[x_2, x_3] \times [y_1, y_2] \cup [x_1, x_4] \times [y_2, y_3]$. Finally, we rotate q by an angle chosen randomly in $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$. The function value in the T-shaped support is fixed as 0.1.

- **Random circle**[32]: For the random circle distribution, q is piecewise constant with the support taken as the union of randomly 1–3 circles. Specifically,

$$q = \sum_{i=1}^{N_c} \mu_i \chi_{D_i}, \quad D_i = \{(x, y) : (x - x_i)^2 + (y - y_i)^2 \le r_i^2\}, \quad N_c \in \{1, 2, 3\}, \tag{22}$$

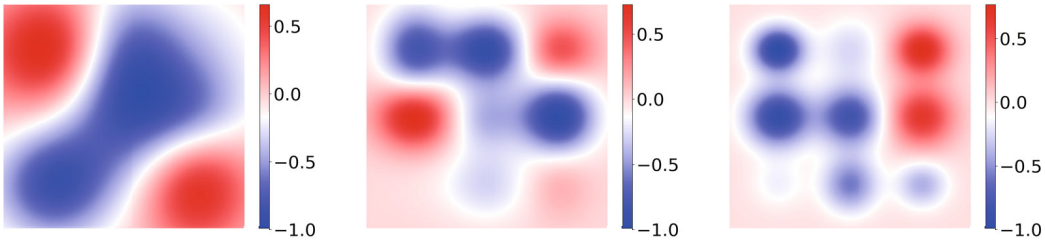
where χ_{D_i} is the indicator function on D_i, $x_i, y_i \sim U[0.2, 0.8]$, $r_i \sim U[0.05, 0.2]$, $\mu_i \sim U[-1, 1]$. Finally, q is normalized such that $\|q\|_{L^\infty(\Omega)} = 0.1$. Note that the circles are allowed to overlap with each other.

- **Smoothed circle:** The smoothed circle distribution is the smoothed version of the random circle distribution such that $q \in C_0^\infty(\Omega)$. Specifically,

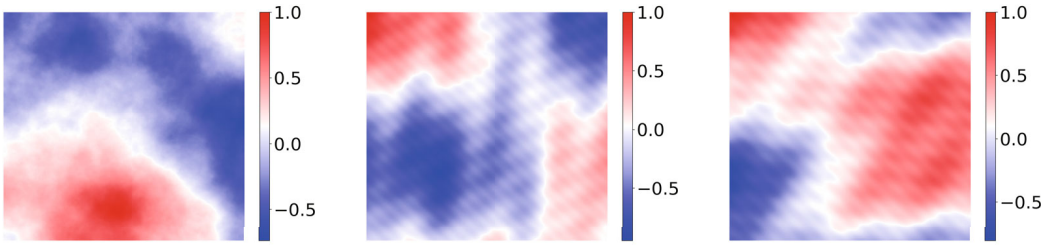
$$q = \sum_{i=1}^{N_c} \mu_i \chi_{D_i} \exp\left[-\frac{1}{1 - \frac{(x-x_i)^2 + (y-y_i)^2}{r_i^2}} \right], \tag{23}$$

where $D_i, N_c, \chi_{D_i}, x_i, y_i, r_i$, and μ_i are defined the same as the random circle distribution. Finally, q is also normalized such that $\|q\|_{L^\infty(\Omega)} = 0.1$.

The three distributions for f are listed below, along with examples sampled from each distribution shown in Figure 6. We normalize each f such that $\|f\|_{L^\infty(\Omega)} = 1$.



(a) Gaussian(10) (b) Gaussian(30) (c) Gaussian(50)



(d) GRF (e) Waves (f) Waves

Figure 6 Examples of f. (a)–(c): Gaussian distribution with different rates of decay. (d): One example sampled from the Gaussian random field (GRF) distribution. (e)–(f): Two examples sampled from the wave distribution

- **Gaussian(R)**[33]: f is taken as the sum of nine Gaussians. The centers of the Gaussians are fixed as $c_{i,j} = \left(\frac{3i-1}{10}, \frac{3j-1}{10}\right), i, j = 1, 2, 3$. The decay rates of the Gaussians are uniformly sampled from $[R, 2R]$, where R is an adjustable hyperparameter. Three examples with different decay rates are shown in Figures 6(a)–(c). It can be seen that the larger the decat rates are, the more compactly supported f is.

- **GRF**[17]: f is generated according to the Gaussian random field (GRF) $\mathcal{N}(0, (-\Delta+9I)^{-2})$ with zero Neumann boundary conditions on the Laplacian.

- **Wave:** The wave distribution is the weighted sum of six planar waves at different frequencies given as:

$$f(x, y) = \sum_{i=1}^{6} \frac{1}{\mu_i} \cos[\pi\mu_i(x\cos\theta_i + y\sin\theta_i)], \tag{24}$$

where $\mu_i \sim U[2^{i-1}, 1.5 \times 2^{i-1}]$, $\theta_i \sim U[0, 2\pi]$. Examples sampled from this distribution have explicit multi-scale property resulted from the superposition of waves at different frequencies.

After the coefficients q and source term f are generated, the corresponding exact solutions are then computed by finite difference method where the large linear system is solved by MUMPS[35].

5.3 Experiment Results

In this subsection, we first present the benchmark results for $k = 20$. Following that, further experiments are carried out to further show the advantages of the NS-UNO in higher wavenumber scenario, data efficiency and computational cost.

5.3.1 Benchmark Results

By combining the distributions for coefficient q and source term f, we present the benchmark results on the following six datasets as is shown in Table 1. It can be seen that compared with FNO and UNO, both NS-FNO and NS-UNO have lower relative L^2-error on the test sets, indicating the effectiveness of the proposed Neumann series based framework. Besides, among the four models, NS-UNO achieves the highest accuracy over all datasets, especially on the datasets where f has strong multi-scale characteristics, such as the GRF distribution. Compared with the state-of-the-art FNO, NS-UNO achieves at least 60% reduction of relative error. Therefore, the proposed NS-UNO network architecture cam better capture the multi-scale features of the solution the Helmholtz equations.

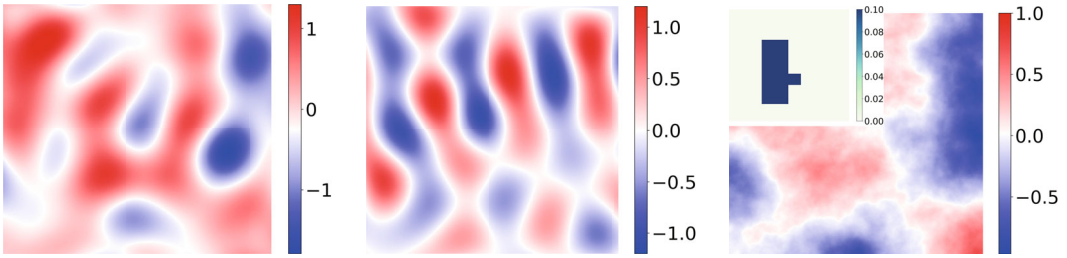
Table 1 Benchmark relative L^2-error ($\times 10^{-2}$) for $k = 20$

q	f	FNO	UNO	NS-FNO	NS-UNO
T-shaped	Gaussian(50)	3.22	2.80	1.34	**1.26**
	Gaussian(30)	4.78	3.10	1.72	**1.30**
	Gaussian(10)	10.23	3.36	4.82	**1.36**
T-shaped	GRF	16.80	15.94	4.87	**2.04**
Random circle	GRF	14.43	11.27	5.48	**1.54**
Smoothed circle	Waves	7.68	6.15	3.08	**1.43**

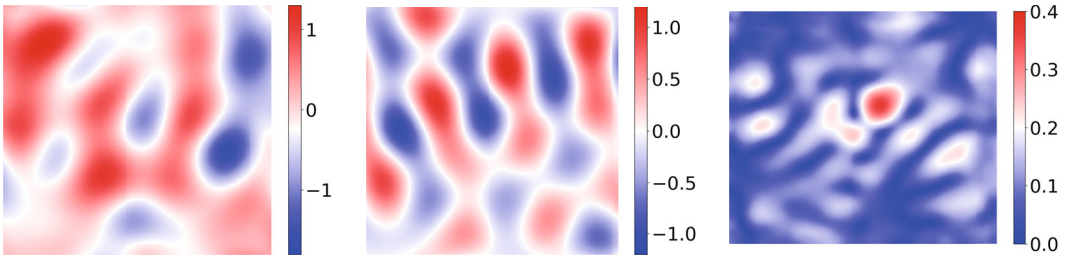
Moereover, we observe that as the decay rate of the Gaussians in the Gaussian dataset for f decreasing, the L^2-error of FNO-based methods significantly increases, while that of UNO-based methods only slightly increases. The reason is that for f consists of Gaussians with smaller decay rates, the Gaussians decaying at different rates will overlap with each other, as can be seen from Figures 6(a)–(c), adding more multi-scale features to f. Therefore, UNO-based methods which are more capable of handling multi-scale inputs outperform FNO-based methods when the decay rate decreases.

In Figure 7 and Figure 8, we showcase the exact solution and the numerical solution obtained by the four benchmark models for the dataset with T-shaped q and f generated by GRF as well as the dataset with smooth circles q and f generated by the wave distribution, respectively. The first row presents the real and imaginary parts of the exact solution and the corresponding parameters (q, f), respectively. The real and imaginary parts and the absolute error of FNO,

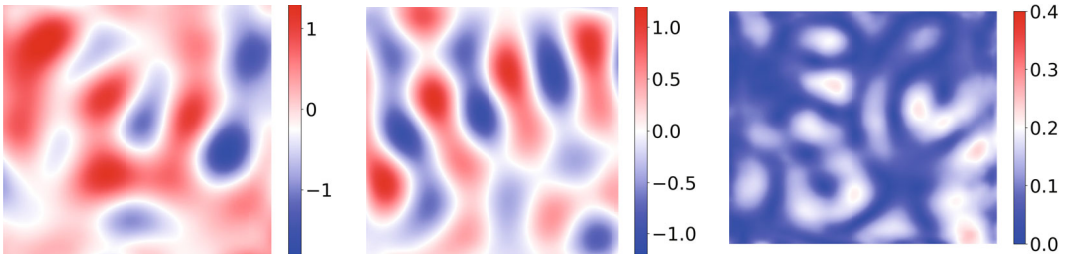
UNO, NS-FNO and NS-UNO are listed in the second to the fifth row in Figure 7 and Figure 8, respectively. It can be seen that in both datasets NS-UNO has the best accuracy. Compared with the other three models, the error is uniformly small, i.e., the numerical solution only largely deviates from the exact solution at few points, further indicating the effectiveness of the proposed NS-UNO.



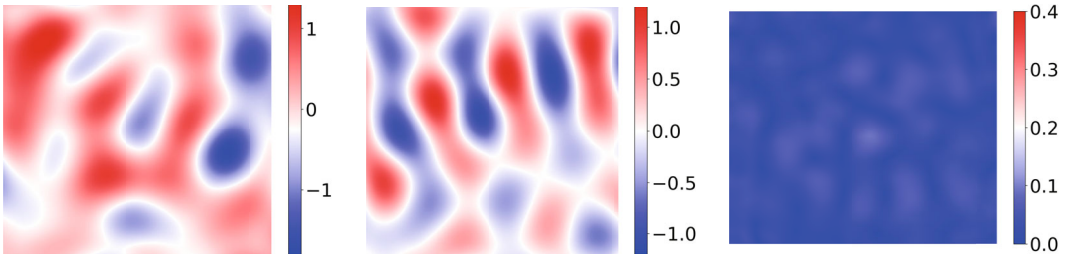
(a) Ground truth, real part (b) Ground truth, imaginary part (c) q (top left) and f



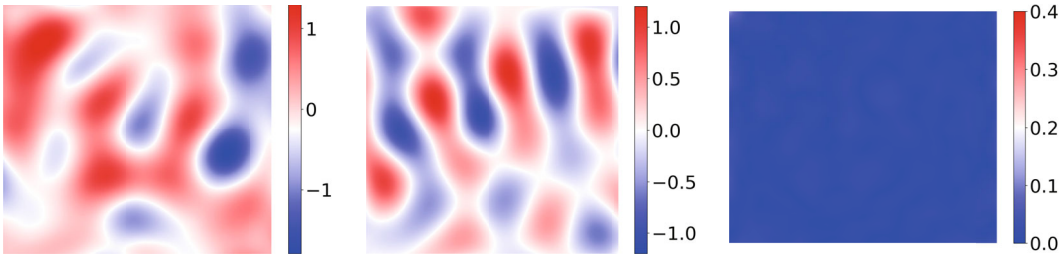
(d) FNO, real part (e) FNO, imaginary part (f) FNO, error, 16.26%



(g) UNO, real part (h) UNO, imaginary part (i) UNO, error, 15.45%

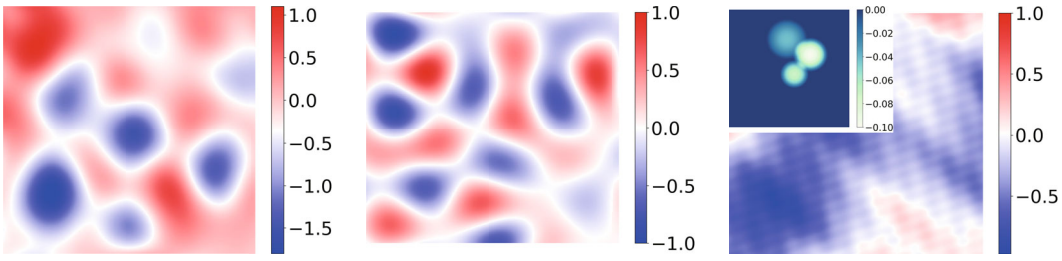


(j) NS-FNO, real part (k) NS-FNO, imaginary part (l) NS-FNO, error, 5.11%

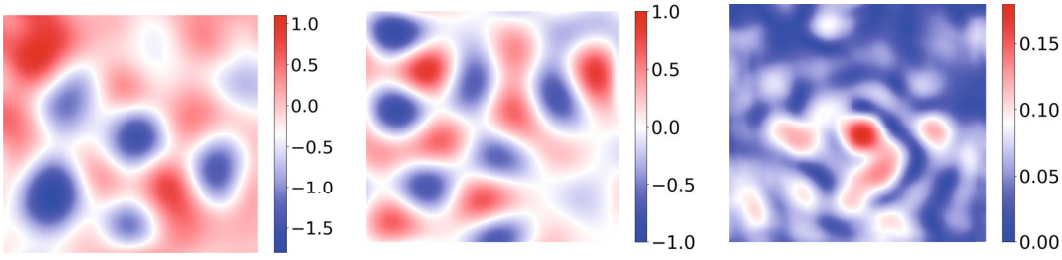


(m) NS-UNO, real part (n) NS-UNO, imaginary part (o) NS-UNO, error, 1.94%

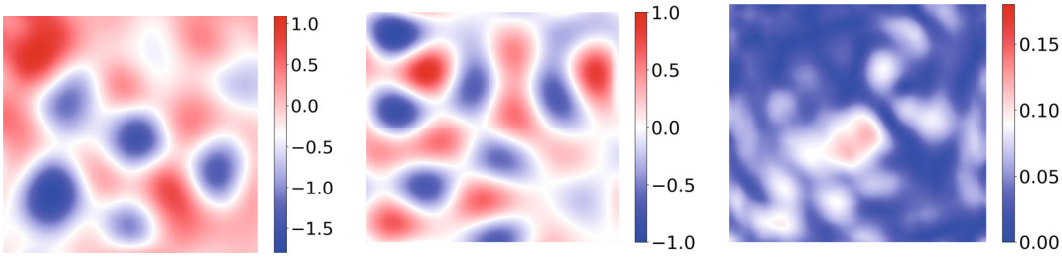
Figure 7 Examples of exact solution, numerical solutions and absolute error for dataset with T-shaped q and GRF f when $k = 20$



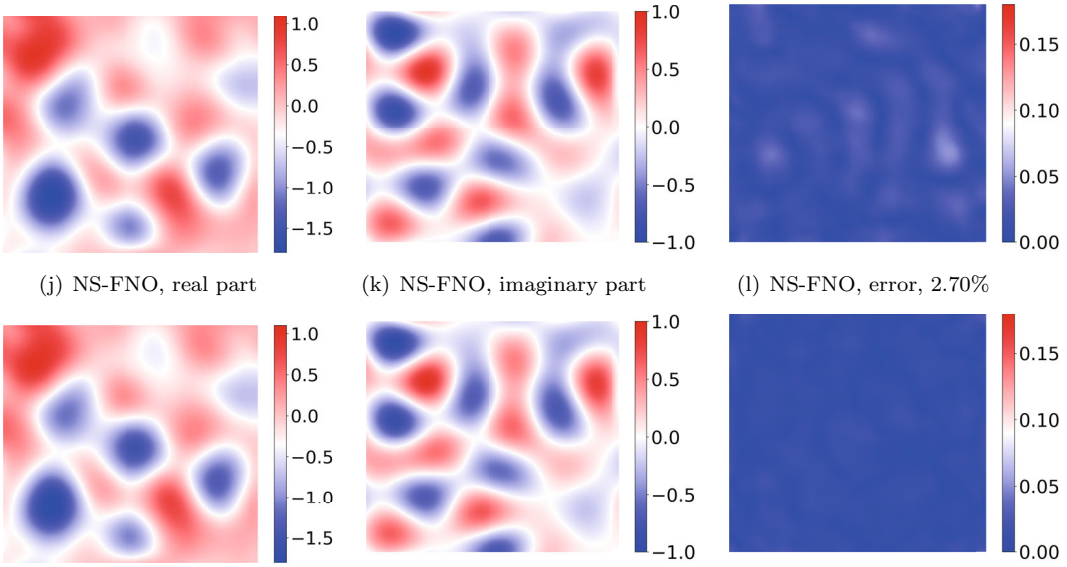
(a) Ground truth, real part (b) Ground truth, imaginary part (c) q (top left) and f



(d) FNO, real part (e) FNO, imaginary part (f) FNO, error, 9.57%



(g) UNO, real part (h) UNO, imaginary part (i) UNO, error, 7.70%



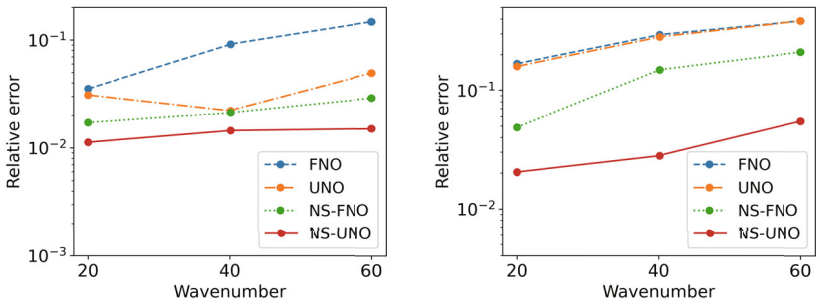
(j) NS-FNO, real part (k) NS-FNO, imaginary part (l) NS-FNO, error, 2.70%

(m) NS-UNO, real part (n) NS-UNO, imaginary part (o) NS-UNO, error, 1.32%

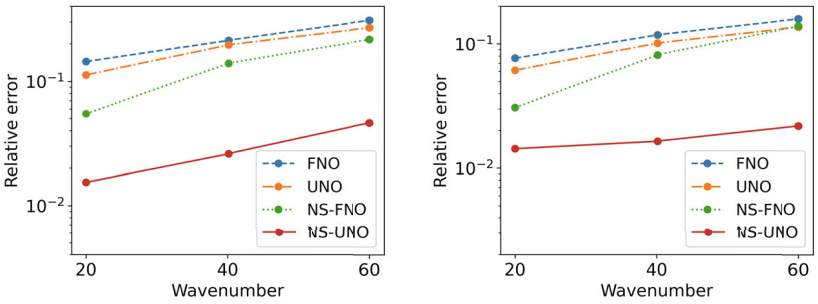
Figure 8 Examples of exact solution, numerical solutions and absolute error for dataset with smooth circle q and waves f when $k = 20$

5.3.2 Higher Wavenumber Scenario

To further test the performance of the models on multi-scale problems, we further present the results for $k = 40, 60$ in Figure 9. It can be seen that among the four models, NS-UNO has the lowest relative error, followed by NS-FNO, indicating the effectiveness of the proposed NSNO. Besides, for the datasets with f sampled from Gaussian random field or superposition of planar waves, which have significant multi-scale properties, NS-UNO shows superior advantage over the other models with nearly one order of magnitude lower relative error, which further reveals the potential of UNO architecture in dealing with multi-scale problems.



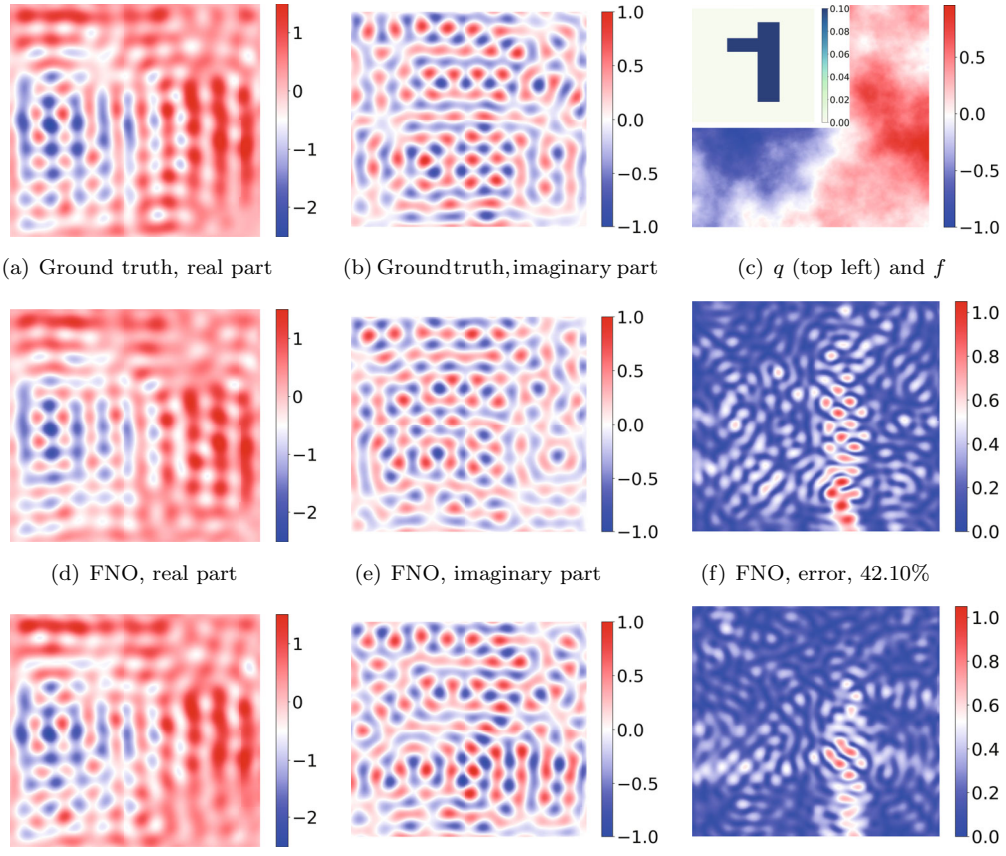
(a) q T-shaped, f Gaussian (30) (b) q T-shaped, f GRF



(c) q random circles, f GRF (d) q smoothed circles, f wave

Figure 9 Relative L^2-error of FNO, UNO, NS-FNO and NS-UNO on four datasets specified in the subcaptions for wavenumber $k = 20, 40, 60$

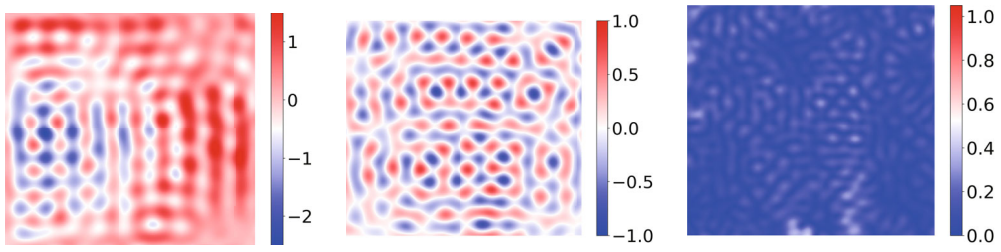
An example taken from the dataset with q being smooth circles and f being waves is shown in Figure 10 for $k = 60$. It can be seen that compared with the $k = 20$ cases shown in Figure 7 and Figure 8, the solutions are more complicated and oscillatory. However, the proposed NS-UNO is still able to learn the solutions with uniformly small error, showing superior power in capturing the multi-scale features of the solution the Helmholtz equations.



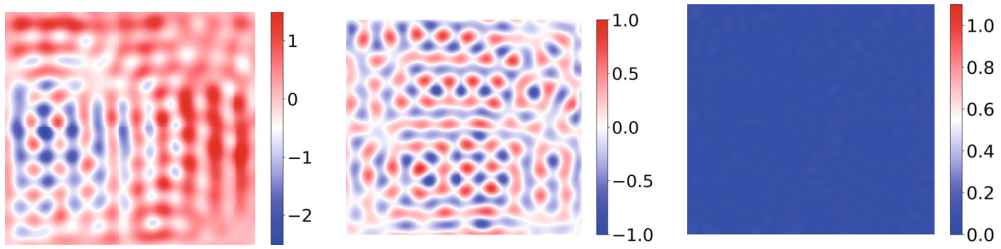
(a) Ground truth, real part (b) Groundtruth, imaginary part (c) q (top left) and f

(d) FNO, real part (e) FNO, imaginary part (f) FNO, error, 42.10%

(g) UNO, real part (h) UNO, imaginary part (i) UNO, error, 36.42%



(j) NS-FNO, real part (k) NS-FNO, imaginary part (l) NS-FNO, error, 17.82%

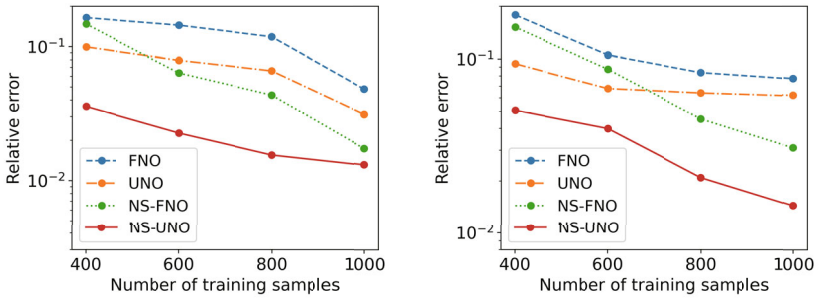


(m) NS-UNO, real part (n) NS-UNO, imaginary part (o) NS-UNO, error, 5.40%

Figure 10 Examples of exact solution, numerical solutions and absolute error for dataset with T-shaped q and GRF f when $k = 60$

5.3.3 Less Training Data

Figure 11 shows the relative error on test sets when we reduce the size of the training set to 800, 600, and 400. It can be seen that NS-UNO still significantly outperforms the other three models when less data is available, showing that NS-UNO has good data efficiency. We also observe that when there is insufficient data, UNO has lower relative error than NS-FNO, while NS-FNO outperforms UNO with the increase of training samples. This reveals that the difficulty in handling multi-scale problems can be alleviated by increasing the number of training data.



(a) q T-shaped, f Gaussian (30) (b) q smoothed circles, f wave

Figure 11 Relative L^2-error of FNO, UNO, NS-FNO and NS-UNO on two datasets specified in the subcaptions with the number of training samples $N = 400, 600, 800, 1000$

5.3.4 Training Computational Cost

A comparison of the training computational costs for NS-FNO and NS-UNO is given in Table 2. It can be seen that with similar number of parameters, NS-UNO has 50% lower computational cost than NS-FNO. This is because the number of channels in the finest level in UNO is a quarter of the number of channels in FNO, which largely reduces the size of tensors, leading to less memory usage and less training time. Therefore, NS-UNO is able to give more accurate results with less computational cost.

Table 2 Training computational costs for NS-FNO and NS-UNO

Model	Iters/sec	Memory (GB)	# of param (M)
NS-FNO	26	12.64	3.56
NS-UNO	12	5.94	3.55

5.4 Necessity of Physics-Informed Loss

In this section, we conducted tests on the weights for the physics-informed loss using the dataset where q is random circle and $f \sim \mathcal{N}(0, (-\Delta + 9I)^{-2})$. As is shown in Table 3, the optimal λ is 0.05, which coincides the choice of λ given in Subsection 5.1. When $\lambda = 0$, i.e., there is no physics-informed loss, both relative error on training and test sets are significantly higher than those with physics-informed loss, showing the necessity of physics-informed loss.

Table 3 Relative L^2-error ($\times 10^{-2}$) of NS-UNO versus weight for the physics-informed loss

λ	0	0.01	0.05	0.1	0.15	0.2
Training error	3.09	1.10	1.07	1.24	1.89	1.96
Test error	8.85	1.73	1.54	1.68	2.21	2.29
Generalization error	11.64	2.57	2.42	2.65	3.01	3.07

Moreover, we observe that the gap between training error and test error is considerably narrowed with the presence of physics-informed loss, indicating that the introduction of physics-informed loss can improve the generalization ability. To further investigate on the generalization ability, we test the generalization error of the learned models on a new dataset with $f \sim \mathcal{N}\big(0, (-\Delta + 9I)^{-\frac{3}{2}}\big)$. It can be seen that the although the generalization error is larger than the test error, the increment from test error to generalization error is larger in the case without physics-informed loss.

5.5 Influence of the Number of Items in Neumann Series

The relative L^2-error versus the number of items to construct the Neumann series is plotted in Figure 12. It can be seen that three items is sufficient to give an accurate result, and the relative error of the model with four items is almost the same as the model with three items. This is because as the depth of the network increases, the gradient backpropagation has to go through a longer path, leading to gradient vanishing problems[34]. The result also indicates that one Neumann series block in the neural network actually does not correspond to a single step in the exact Neumann series.

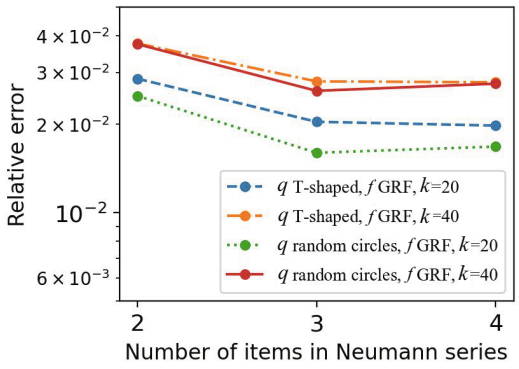


Figure 12 Relative L^2-error versus the number of items in Neumann series

5.6 Beyond Convergence of Neumann Series

As discussed in (16) in the proof of the convergence of Neumann series, the convergence rate is associated with the norm of the operator $-k^2Gq$, which is proportional to k and $\|q\|_{L^\infty(\Omega)}$. Overly large k and $\|q\|_{L^\infty(\Omega)}$ will lead to slow convergence and even divergence of the Neumann series. Surprisingly, we found that the proposed NS-UNO is able to break through this theoretical limitation and still gives reasonable results even the exact Neumann series diverges, as shown in Table 4.

Table 4 Relative L^2-error ($\times 10^{-2}$) of NS-UNO and Exact Neumann series (NS)

wavenumber	$\|q\|_{L^\infty(\Omega)}$	NS (3 terms)	NS (10 terms)	FNO	NS-UNO
$k = 20$	0.35	24.39	6.85	10.56	**6.42**
	0.4	44.11	57.12	17.90	**9.83**
$k = 40$	0.2	12.42	3.10	17.27	**2.92**
	0.25	25.16	35.52	20.19	**4.28**

Specifically, we use the dataset where q is T-shaped and f is sampled from Gaussian(30). We take $k = 20$ and $k = 40$, and compute the exact Neumann series by MUMPS. It can be seen that for $k = 20, \|q\|_{L^\infty(\Omega)} = 0.35$ and $k = 40, \|q\|_{L^\infty(\Omega)} = 0.2$, the Neumann series exhibits a slow convergence, while for $k = 20, \|q\|_{L^\infty(\Omega)} = 0.4$ and $k = 40, \|q\|_{L^\infty(\Omega)} = 0.25$, the Neumann series diverges, which coincides with the theoretical discussion in Subsection 3.2. However, the proposed NS-UNO outperforms the exact Neumann series with 10 terms using only three iteration steps, and can still maintain a relatively low L^2-error even the Neumann series diverges. Besides, NS-UNO still outperforms FNO for large k and $\|q\|_{L^\infty(\Omega)}$, which is not affected by this theoretical limitation of Neumann series.

6 Application in Inverse Scattering Problem

In this section, we solve an inverse scattering problem using the learned NSNO as the forward solver. We first demonstrate the setups for the inverse problem, including the governing

equation and data measurement. Traditional and neural network-based methods for solving the inverse problem are then introduced. Following that, we show the reconstructed results using traditional finite difference method and the proposed NS-UNO as the forward solver, respectively.

6.1 Problem Setup

We set the incident field as the plane wave $u^{in} = \mathrm{e}^{\mathrm{i}k\boldsymbol{x}\cdot\boldsymbol{d}}$, where $\boldsymbol{d}$ denotes the incoming direction. In our experiments, we discrete Ω to a 128×128 grid and generate plane waves from $M = 32$ different directions uniformly. Based on (6), the scattered field satisfies

$$\Delta u^s + k^2(1 + q(x))u^s = -k^2 q(x)u^{in}. \tag{25}$$

The mapping from the scatter q to the scattered field u^s can then be given by the solution operator defined in (5) as $q \mapsto \mathcal{S}(q, -k^2 q u^{in})$.

As for data, sensors are placed on each grid on the boundary of $\Omega = [0, 1]^2$ to collect the wave field data d_m for each incident wave u_m^{in}, $m = 1, 2, \cdots, M$. Therefore, the forward operator maps to scatter q to the data collected on the boundary $\mathcal{F}_m(q) = T \circ \mathcal{S}(q, -k^2 q u_m^{in})$, where T denotes the trace operator restricting the wave field on the boundary. The inverse problem is to reconstruct the scatterer q from the measured data. We generate the wave field measurement with a fine grid to avoid the inverse crime.

6.2 Solving the Inverse Problem

We solve the inverse problem by the traditional optimization approach, which means recovering an approximation of q by solving the following optimization problem:

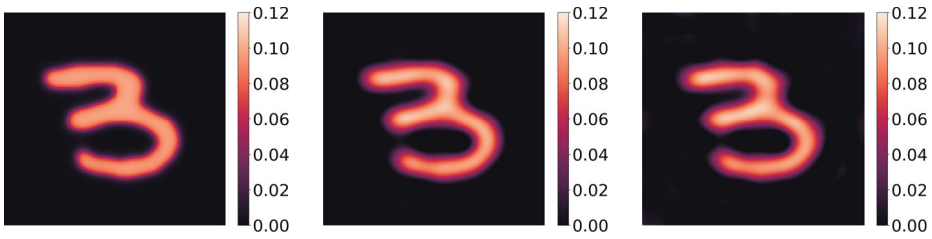
$$\operatorname*{argmin}_q J(q) = \sum_{m=1}^{M} J_j(q) = \sum_{m=1}^{M} \frac{1}{2}\|\mathcal{F}_m(q) - d_m\|_2^2. \tag{26}$$

We employ the L-BFGS algorithm[36] to address the minimization problem, initializing the iterative process with a value of 0. The gradient of the loss with respect to the model is computed using the adjoint state method[37], which will be detailed in the appendix. Our objective is to evaluate the effectiveness of utilizing a pre-trained neural network embedded as a forward solver within the optimization framework, as opposed to the conventional numerical solvers, for which we choose the finite difference method (FDM). For the relevant finite difference matrix, we select MUMPS as the direct solver.

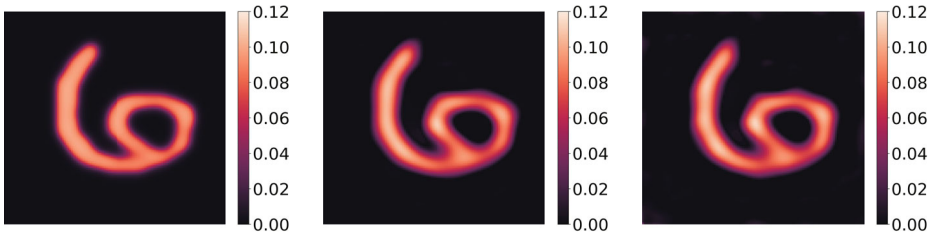
The neural network training process is outlined below. The dataset consists of scatterer samples q, which are extracted from the large MNIST dataset[38]. Specifically, these scatterer samples are initially resized to a resolution of 112×112 from the original MNIST dataset[39] and further padded to 128×128. To construct the training/test sets, we select 100/10 samples for each digit in the ranges 0–9. During the training process, only four plane waves with directions $0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$ are employed. The neural network is trained for a total of 1000 epochs with the learning rate initialized as 0.001 and halved every 200 epochs. The weight λ in the loss function (21) is set as 0.1. After the training is completed, the pre-trained neural network is evaluated on the test set and achieves an average relative L^2-error of 1.34%.

6.3 Reconstructing Results

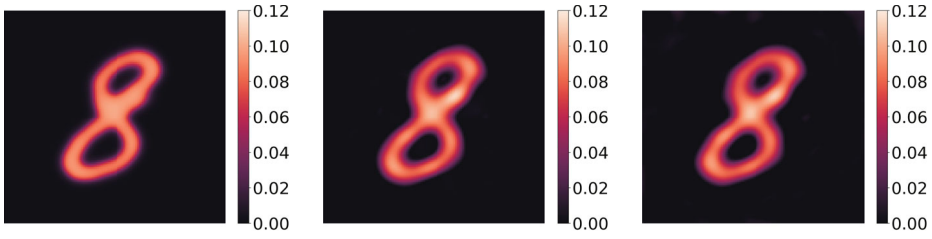
The reconstruction results obtained with the Finite difference method (FDM) and NS-UNO as the forward solver are depicted in Figure 13. The visual comparison demonstrates that NS-UNO produces results on par with the finite difference method, with only a slight 2% increase in relative L^2-error attributed to the inherent error of neural networks in solving forward problems. However, the key advantage of NS-UNO lies in its remarkable speed improvement. The neural network's capability to simultaneously solve all the forward problems essential for gradient computation results in NS-UNO being more than 20 times faster than the FDM with MUMPS. As a consequence, the adoption of NS-UNO as a surrogate model for the forward problem represents a significant enhancement in efficiency without causing substantial damage to accuracy.



(a) Ground truth (b) FDM, 9.67%, 25.1 s (c) NS-UNO, 11.48%, 1.21 s



(d) Ground truth (e) FDM, 10.41%, 38.4 s (f) NS-UNO, 12.43%, 1.31 s

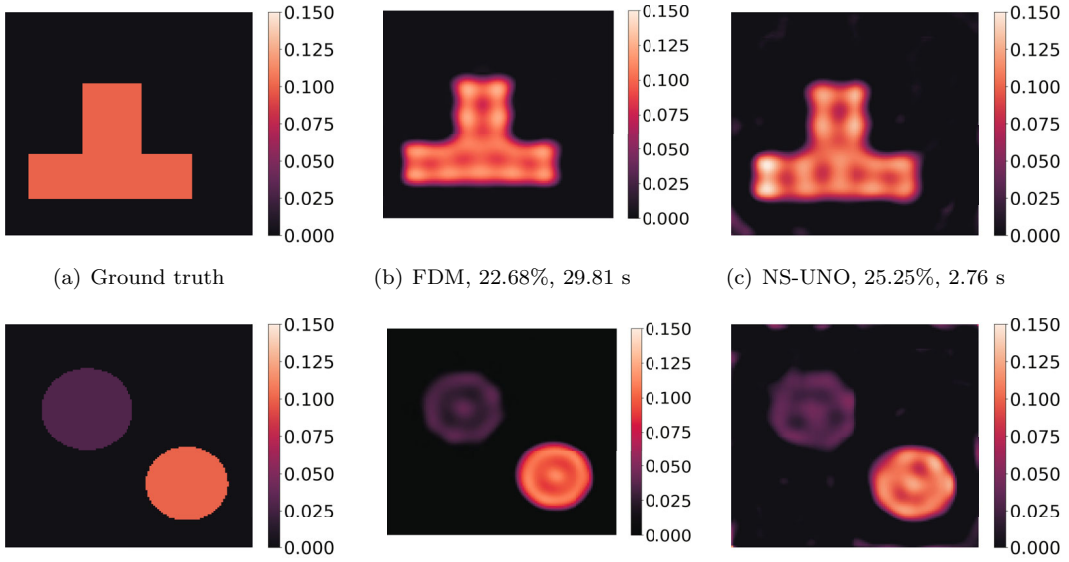


(g) Ground truth (h) FDM, 11.13%, 51.83 s (i) NS-UNO, 13.38%, 1.11 s

Figure 13 Relative L^2-error and reconstruction time of scatterer q using FDM and NS-UNO as the forward solver

To further show the generalization ability of the proposed NS-UNO, we directly use the neural network trained with MNIST dataset to recover q from T-shaped and random circle datasets, as is shown in Figure 14. It can be seen that the shape of the scatterer is accurately

reconstructed by NS-UNO, and the relative error is also comparable to the finite difference method.



(a) Ground truth (b) FDM, 22.68%, 29.81 s (c) NS-UNO, 25.25%, 2.76 s

(d) Ground truth (e) FDM, 20.61%, 41.56 s (f) NS-UNO, 26.40%, 1.28 s

Figure 14 Relative L^2-error and reconstruction time of scatterer q from T-shaped and random circle datasets using MUMPS and NS-UNO trained with MNIST dataset as the forward solver

7 Conclusions

In this paper, we propose a novel Neumann series neural operator (NSNO) to learn the solution operator of Helmholtz equation from inhomogeneity coefficients and source term to solutions. Specifically, by utilizing the Neumann series representation of the solution to the Helmholtz equation, the solution operator from both inhomogeneity coefficients and source terms to solutions can be decomposed into the solution operator from only source terms to solutions. A novel network architecture integrating U-Net structure has been designed to capture the multi-scale features in the solution to the Helmholtz equation.

Extensive experiments have been conducted to test the performance of NSNO. The results show that NSNO has superior accuracy compared to the state-of-the-art FNO especially in the large wavenumber case, and has lower computational cost and less data requirement. The application of NSNO in inverse scattering problem have also been discussed. We have shown that the proposed NSNO is able to serve as the surrogate model and give competitive results to traditional finite difference forward solver while the computational cost is reduced significantly.

The major limitation of our work is that the restriction on the maximum value of the inhomogeneity coefficients to guarantee the convergence of Neumann series hinders the application of NSNO in high-contrast medium. In addition, the proposed Neumann series is based on the specific form of Helmholtz equation. It still remains a challenge to extend NSNO to other

partial differential equations. For future work, we will try to address these issues by exploring more efficient network architecture to enlarge the application scope of NSNO, and developing alternative iterative schemes to deal with the solution operator to other partial differential equations.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] Colton D and Kress R, *Inverse Acoustic and Electromagnetic Scattering Theory*, Springer-Verlag, New York, 2013.

[2] Arridge S R, Optical tomography in medical imaging, *Inverse Problems*, 2007, **15**: 41–93.

[3] Colton D, Coyle J, and Monk P, Recent developments in inverse acoustic scattering theory, *SIAM Review*, 2000, **42**(3): 369–414.

[4] Singer I, Turkel E, High-order finite difference methods for the Helmholtz equation, *Comput. Methods Appl. Mech. Engrg.*, 1998, **163**(1–4): 343–358.

[5] Babuška I, Ihlenburg F, Paik E T, et al., A generalized finite element method for solving the Helmholtz equation in two dimensions with minimal pollution, *Comput. Methods Appl. Mech. Engrg.*, 1995, **128**(3–4): 325–359.

[6] Saad Y and Schultz M H, GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems, *SIAM. Sci. Stat. Comput.*, 1986, **7**(3): 856–869.

[7] Ernst O G and Gander M J, Why it is difficult to solve Helmholtz problems with classical iterative methods, *Numerical Analysis of Multiscale Problems*, Springer-Verlag, New York, 2011, 325–363.

[8] Hornik K, Stinchcombe M, and White H, Multilayer feedforward networks are universal approximators, *Neural Networks*, 1989, **2**(5): 359–366.

[9] Goodfellow I, Bengio Y, and Courville A, *Deep Learning*, MIT Press, Cambridge, 2016.

[10] Young T, Hazarika D, Poria S, et al., Recent trends in deep learning based natural language processing, *IEEE Comput. Intell. Mag.*, 2018, **13**(3): 55–75.

[11] Beck C, Hutzenthaler M, Jentzen A, et al., An overview on deep learning-based approximation methods for partial differential equations, 2020, arXiv: 2012.12348.

[12] Kovachki N, Li Z, Liu B, et al., Neural operator: Learning maps between function spaces, 2021, arXiv: 2108.08481.

[13] Raissi M, Perdikaris P, and Karniadakis G E, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.*, 2019, **378**: 686–707.

[14] Alkhalifah T, Song C, Waheed U B, et al., Wavefield solutions from machine learned functions constrained by the Helmholtz equation, *Artificial Intelligence in Geosciences*, 2021, **2**: 11–19.

[15] Cui T, Wang Z, and Xiang X, An efficient neural network method with plane wave activation functions for solving Helmholtz equation, *Computers and Mathematics with Applications*, 2022, **111**: 34–49.

[16] Lu L, Jin P, Pang G, et al., Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, *Nature Machine Intelligence*, 2021, **3**(3): 218–229.

[17] Li Z, Kovachki N, Azizzadenesheli K, et al., Fourier neural operator for parametric partial differential equations, 2020, arXiv: 2010.08895.

[18] Lu L, Meng X, Cai S, et al., A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data, *Computer Methods in Applied Mechanics and Engineering*, 2022, **393**: 114778.

[19] Wang S, Wang H, and Perdikaris P, Learning the solution operator of parametric partial differential equations with physics-informed DeepONets, *Science Advances*, 2021, **7**(40): eabi8605.

[20] Li Z, Zheng H, Kovachki N, et al., Physics-informed neural operator for learning partial differential equations, 2021, arXiv: 2111.03794.

[21] Stanziola A, Arridge S R, Cox B T, et al., A Helmholtz equation solver using unsupervised learning: Application to transcranial ultrasound, *J. Comput. Phys.*, 2021, **441**: 110430.

[22] Lin G, Hu P, Chen F, et al., BINet: Learning to solve partial differential equations with boundary integral networks, *CSIAM Trans. Appl. Math.*, 2023, **4**(2): 275–305.

[23] Ronneberger O, Fischer P, and Brox T, U-net: Convolutional networks for biomedical image segmentation, *Medical Image Computing and Computer-Assisted Intervention — MICCAI* 2015, Munich, 2015.

[24] Jin J, *The Finite Element Methods in Electromagnetics*, Wiley, New York, 2022.

[25] Hetmaniuk U, Stability estimates for a class of Helmholtz problems, *Commun. Math. Sci.*, 2007, **5**(3): 665–678.

[26] Liu X, Xu B, and Zhang L, Ht-net: Hierarchical transformer based operator learning model for multiscale pdes, 2022, arXiv: 2210.10890.

[27] Zhou Z, Rahman Siddiquee M M, Tajbakhsh N, et al., Unet++: A nested u-net architecture for medical image segmentation, *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support:* 4*th International Workshop*, Granada, 2018.

[28] Li X, Chen H, Qi X, et al, H-DenseUNet: Hybrid densely connected UNet for liver and tumor segmentation from CT volumes, *IEEE Trans. Med. Imag.*, 2018, **37**(12): 2663–2674.

[29] Hendrycks D and Gimpel K, Gaussian error linear units (gelus), 2016, arXiv: 1606.08415.

[30] Cho S J, Ji S W, Hong J P, et al., Rethinking coarse-to-fine approach in single image deblurring, *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Montreal, 2021.

[31] Kingma D P and Ba J, Adam: A method for stochastic optimization, 2014, arXiv: 1412.6980.

[32] Wei Z and Chen X, Deep-learning schemes for full-wave nonlinear inverse scattering problems, *IEEE Trans. Geosci. Remote Sens.*, 2018, **57**(4): 1849–1860.

[33] Zhang Z, Leung W T, and Schaeffer H, BelNet: Basis enhanced learning, a mesh-free neural operator, 2022, arXiv: 2212.07336.

[34] Hochreiter S, The vanishing gradient problem during learning recurrent neural nets and problem solutions, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 1998, **6**(2): 107–116.

[35] Amestoy P R, Duff I S, L'Excellent J Y, et al., A fully asynchronous multifrontal solver using distributed dynamic scheduling, *SIAM J. Matrix. Anal. Appl.*, 2001, **23**(1): 15–41.

[36] Liu D C and Nocedal J, On the limited memory BFGS method for large scale optimization, *Mathematical Programming*, 1989, **45**(1–3): 503–528.

[37] Plessix R E, A review of the adjoint-state method for computing the gradient of a functional

with geophysical applications, *Geophys. J. Int.*, 2006, **167**(2): 495–503.

[38] Jansson Y and Lindeberg T, Exploring the ability of CNNs to generalise to previously unseen scales over wide scale ranges, 2020 25*th International Conference on Pattern Recognition* (*ICPR*), Milan, 2021.

[39] Deng, L, The mnist database of handwritten digit images for machine learning research, *IEEE Signal Processing Mag.*, 2012, **29**(6): 141–142.

Appendix

Adjoint State Method

Without loss of generality, we consider the model with one incident wave denoted as u_0. The optimization problem is then given as

$$\min_q \quad J(q) = \frac{1}{2}\|Tu(q) - d\|_2^2 \tag{27}$$

$$\text{s.t.} \quad \Delta u + k^2(1+q)u = -k^2 q u_0, \quad \text{in } \Omega, \tag{28}$$

$$\frac{\partial u}{\partial n} = \mathrm{i}ku, \quad \text{on } \partial\Omega, \tag{29}$$

where T denotes the trace operator restricting the wave field on the boundary. In order to derive $\frac{\partial J}{\partial q}$, we apply the Lagrange multiplier method. Define the Lagrangian as

$$\mathcal{L}(u, \lambda, q) = J(q) - (\lambda, \Delta u + k^2(1+q)u + k^2 q u_0), \tag{30}$$

where $(f, g) = \mathrm{Re}\int_\Omega f\overline{g}$ denotes the real part of the inner product in $L^2(\Omega)$. After two integrations by part we obtain

$$\begin{aligned}
&\mathcal{L}(u, \lambda, q)\\
=&J(q) - \left\langle \lambda, \frac{\partial u}{\partial n} \right\rangle + \left\langle \frac{\partial \lambda}{\partial n}, u \right\rangle - (\Delta\lambda, u) - (\lambda, k^2(1+q)u) - (\lambda, k^2 q u_0)\\
=&J(q) + \left\langle \frac{\partial \lambda}{\partial n} + \mathrm{i}k\lambda, u \right\rangle - (\Delta\lambda + k^2(1+q)\lambda, u) - (\lambda, k^2 q u_0),
\end{aligned} \tag{31}$$

where $\langle f, g\rangle = \mathrm{Re}\int_\Omega f\overline{g}$ denotes the real part of the inner product in $L^2(\partial\Omega)$. Therefore,

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial q}(q) = &\left(T^*(Tu - d), \frac{\partial u}{\partial q}(q)\right) + \left\langle \frac{\partial \lambda}{\partial n} + \mathrm{i}k\lambda, \frac{\partial u}{\partial q}(q) \right\rangle\\
&- \left(\Delta\lambda + k^2(1+q)\lambda, \frac{\partial u}{\partial q}(q)\right) - (k^2\lambda, u) - (\lambda, k^2 u_0).
\end{aligned} \tag{32}$$

To eliminate $\frac{\partial u}{\partial q}(q)$, we choose λ to satisfy

$$\begin{aligned}
&\Delta\lambda + k^2(1+q)\lambda = T^*(Tu - d), \quad \text{in } \Omega,\\
&\frac{\partial \lambda}{\partial n} + \mathrm{i}k\lambda = 0, \quad \text{on } \partial\Omega,
\end{aligned} \tag{33}$$

or equivalently we solve the conjugate of λ satisfying

$$\begin{aligned} &\Delta\overline{\lambda} + k^2(1+q)\overline{\lambda} = \overline{T^*(Tu-d)}, \quad \text{in } \Omega, \\ &\frac{\partial\overline{\lambda}}{\partial n} = \mathrm{i}k\overline{\lambda}, \quad \text{on } \partial\Omega. \end{aligned} \tag{34}$$

The gradient of J with respect to q can then be obtained by

$$\frac{\partial J(q)}{\partial q} = \frac{\partial \mathcal{L}(q)}{\partial q} = -k^2(\overline{\lambda}, u + u_0). \tag{35}$$

The process to compute the gradient of J with respect to q is summarized as follows, where the forward solver is called twice:

- Solve u satisfying (28) and (29).
- Solve $\overline{\lambda}$ satisfying (34).
- Compute the gradient by (35).