A SUPPLEMENTS FOR SECTION 3

The following are general assumptions across our theories:

Assumption A.1. The problem domain Ω is an open, bounded, and nonempty subset of $\mathbb{R}^d$, where $d \in \mathbb{N}^+$ is the spatial(-temporal) dimensionality. And

Assumption A.2. The boundary value problem (BVP) considered in Eq. (1) is well-posed, which means the solution exists and is unique, and $\mathcal{F}^{-1}$ is well-defined.

Assumption A.3. $\|u\| \neq 0$ and $\|f\| \neq 0$.

Remark. This assumption assures that the *relative* conditional number is well-defined. If it is not satisfied, we could define the *absolute* conditional number by removing the zero terms.

Assumption A.4. For any continuous function v defined on Ω (i.e., $v \in C(\Omega)$), it holds that $\inf_{\boldsymbol{\theta}\in\Theta} \|u_{\boldsymbol{\theta}} - v\| = 0$.

Remark. We assume that the neural network has sufficient approximation capability and ignore the corresponding error.

A.1 PROOF FOR THEOREM 3.2

Under Assumption A.1 – A.4, the proof of Theorem 3.2 is given as follows.

Proof. According to the local Lipschitz continuity of $\mathcal{F}^{-1}$, there exists $r > 0$ such that:

$$\left\|\mathcal{F}^{-1}[w_1] - \mathcal{F}^{-1}[w_2]\right\| \leq K\|w_1 - w_2\|, \tag{15}$$

holds for any $w_1, w_2 \in W$ which satisfy that $\|w_1 - f\| < r$ and $\|w_2 - f\| < r$.

Taking an $\epsilon < r$, we can derive that:

$$\begin{aligned}
&\sup_{0<\|\delta f\|\leq\epsilon} \frac{\|\delta u\|/\|u\|}{\|\delta f\|/\|f\|} \\
&= \frac{\|f\|}{\|u\|} \sup_{0<\|\mathcal{F}[u_{\boldsymbol{\theta}}]-f\|\leq\epsilon} \frac{\|u_{\boldsymbol{\theta}} - u\|}{\|\mathcal{F}[u_{\boldsymbol{\theta}}] - f\|} \\
&= \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\leq\epsilon} \frac{\|\mathcal{F}^{-1}[f+h] - \mathcal{F}^{-1}[f]\|}{\|h\|} \quad (\text{let } \mathcal{F}[u_{\boldsymbol{\theta}}] - f = h) \\
&\leq \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\leq\epsilon} \frac{K\,\|h\|}{\|h\|} \\
&= \frac{\|f\|}{\|u\|} K.
\end{aligned} \tag{16}$$

Finally, let $\epsilon \to 0^+$, we can prove the theorem:

$$\operatorname{cond}(\mathcal{P}) = \lim_{\epsilon\to 0^+} \sup_{0<\|\delta f\|\leq\epsilon} \frac{\|\delta u\|/\|u\|}{\|\delta f\|/\|f\|} \leq \frac{\|f\|}{\|u\|} K. \tag{17}$$

□

A.2 THE EXISTENCE OF CONDITION NUMBER IN SPECIAL CASES

Proposition A.5. *Considering a well-posed $\mathcal{P} : \{\mathcal{F}[u] = f \text{ in } \Omega, u = g \text{ in } \partial\Omega\}$, we assert that:*

1. *If $\mathcal{F}$ is linear (i.e., a linear PDE) and $g = 0$ (homogeneous BC), then $\mathcal{F}^{-1}$ is a bounded linear operator and $\operatorname{cond}(\mathcal{P}) = \frac{\|f\|}{\|u\|}\|\mathcal{F}^{-1}\| < \infty$.*
2. *Define $\mathcal{P}_1 : \{\mathcal{F}[u] = 0 \text{ in } \Omega, u = g \text{ in } \partial\Omega\}$. If $\mathcal{F}$ is linear and $\mathcal{P}_1$ is well-posed, then $\operatorname{cond}(\mathcal{P}) < \infty$.*

3. *If $\mathcal{F}^{-1}$ is Fréchet differentiable at f, then* $\operatorname{cond}(\mathcal{P}) = \frac{\|f\|}{\|u\|}\|D\mathcal{F}^{-1}[f]\| < \infty$*, where* $D\mathcal{F}^{-1}[f]\colon S \to V$ *is a bounded linear operator, the Fréchet derivative of* $\mathcal{F}^{-1}$ *at* f*.*

We divide the Proposition A.5 into the following theorems and prove them one by one.

Theorem A.6. *If $\mathcal{F}$ is linear and $g = 0$, then $\mathcal{F}^{-1}$ is a bounded linear operator and:*

$$\operatorname{cond}(\mathcal{P}) = \frac{\|f\|}{\|u\|}\left\|\mathcal{F}^{-1}\right\| < \infty. \tag{18}$$

Proof. Firstly, it is easy to show the linearity. Considering $k_1, k_2 \in \mathbb{K}, w_1, w_2 \in S$, there exists $u_1, u_2 \in V$ such that $\mathcal{F}[u_1] = w_1 \wedge u_1|_{\partial\Omega} = 0$ and $\mathcal{F}[u_2] = w_2 \wedge u_2|_{\partial\Omega} = 0$. Then, we have:

$$\mathcal{F}^{-1}[k_1w_1 + k_2w_2] = k_1u_1 + k_2u_2 = k_1\mathcal{F}^{-1}[w_1] + k_2\mathcal{F}^{-1}[w_2], \tag{19}$$

where the first equation holds because $\mathcal{F}[k_1u_1 + k_2u_2] = k_1\mathcal{F}[u_1] + k_2\mathcal{F}[u_2] = k_1w_1 + k_2w_2$ and $k_1u_1 + k_2u_2 = 0$ in $\partial\Omega$.

Secondly, according to the well-posedness, $\mathcal{F}^{-1}$ is continuous and thus bounded.

Finally, we have:

$$\begin{aligned}
&\sup_{0<\|\delta f\|\le\epsilon} \frac{\|\delta u\|/\|u\|}{\|\delta f\|/\|f\|} \\
&= \frac{\|f\|}{\|u\|} \sup_{0<\|\mathcal{F}[u_{\boldsymbol{\theta}}]-f\|\le\epsilon} \frac{\|u_{\boldsymbol{\theta}} - u\|}{\|\mathcal{F}[u_{\boldsymbol{\theta}}] - f\|} \\
&= \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\le\epsilon} \frac{\|\mathcal{F}^{-1}[f+h] - \mathcal{F}^{-1}[f]\|}{\|h\|} \quad (\text{let } \mathcal{F}[u_{\boldsymbol{\theta}}] - f = h) \\
&= \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\le\epsilon} \frac{\left\|\mathcal{F}^{-1}[h]\right\|}{\|h\|} \\
&= \frac{\|f\|}{\|u\|}\left\|\mathcal{F}^{-1}\right\|.
\end{aligned} \tag{20}$$

Therefore, let $\epsilon \to 0^+$, $\operatorname{cond}(\mathcal{P}) = \frac{\|f\|}{\|u\|}\left\|\mathcal{F}^{-1}\right\| < \infty$.

□

Theorem A.7. *Define $\mathcal{P}_1 : \{\mathcal{F}[u] = 0$ in $\Omega, u = g$ in $\partial\Omega\}$. If $\mathcal{F}$ is linear and $\mathcal{P}_1$ is well-posed, then:*

$$\operatorname{cond}(\mathcal{P}) < \infty. \tag{21}$$

Proof. Since $\mathcal{P}_1$ is well-posed, there exists a unique solution $u_1 \in V$ to it. We define $\mathcal{G} : S \to V$ as $\mathcal{G}[w] = \mathcal{F}^{-1}[w] - u_1$. Then we show that $\mathcal{G}$ is linear. Consider $k_1, k_2 \in \mathbb{K}, w_1, w_2 \in S$,

$$\begin{aligned}
\mathcal{G}[k_1w_1 + k_2w_2] &= \mathcal{F}^{-1}[k_1w_1 + k_2w_2] - u_1, \\
k_1\mathcal{G}[w_1] + k_2\mathcal{G}[w_2] &= k_1\left(\mathcal{F}^{-1}[w_1] - u_1\right) + k_2\left(\mathcal{F}^{-1}[w_2] - u_1\right).
\end{aligned} \tag{22}$$

We have to show that:

$$\begin{aligned}
&\mathcal{F}^{-1}[k_1w_1 + k_2w_2] - u_1 &&= k_1\left(\mathcal{F}^{-1}[w_1] - u_1\right) + k_2\left(\mathcal{F}^{-1}[w_2] - u_1\right) \\
\iff\quad &\mathcal{F}^{-1}[k_1w_1 + k_2w_2] &&= k_1\left(\mathcal{F}^{-1}[w_1] - u_1\right) + k_2\left(\mathcal{F}^{-1}[w_2] - u_1\right) + u_1.
\end{aligned} \tag{23}$$

Apply $\mathcal{F}$ on both sides:

$$\begin{aligned}
k_1w_1 + k_2w_2 &= \mathcal{F}\left(\mathcal{F}^{-1}[k_1w_1 + k_2w_2]\right) \\
&= \mathcal{F}\left(k_1\left(\mathcal{F}^{-1}[w_1] - u_1\right) + k_2\left(\mathcal{F}^{-1}[w_2] - u_1\right) + u_1\right) \\
&= k_1w_1 + k_2w_2.
\end{aligned} \tag{24}$$

And consider the value on the boundary:

$$\begin{aligned} g &= \left(\mathcal{F}^{-1}[k_1 w_1 + k_2 w_2]\right)\Big|_{\partial\Omega} \\ &= \left(k_1\left(\mathcal{F}^{-1}[w_1] - u_1\right) + k_2\left(\mathcal{F}^{-1}[w_2] - u_1\right) + u_1\right)\Big|_{\partial\Omega} \\ &= k_1(g-g) + k_2(g-g) + g = g. \end{aligned} \tag{25}$$

Then, according to the well-defineness of $\mathcal{F}^{-1}$, we can prove that Eq. (23) holds and thus $\mathcal{G}$ is linear. Besides, since $\mathcal{F}^{-1}$ is continuous, $\mathcal{G}$ is a bounded linear operator.

Finally, we have:

$$\begin{aligned} &\sup_{0<\|\delta f\|\le\epsilon} \frac{\|\delta u\|/\|u\|}{\|\delta f\|/\|f\|} \\ &= \frac{\|f\|}{\|u\|} \sup_{0<\|\mathcal{F}[u_{\boldsymbol{\theta}}]-f\|\le\epsilon} \frac{\|u_{\boldsymbol{\theta}} - u\|}{\|\mathcal{F}[u_{\boldsymbol{\theta}}] - f\|} \\ &= \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\le\epsilon} \frac{\|\mathcal{F}^{-1}[f+h] - \mathcal{F}^{-1}[f]\|}{\|h\|} \quad (\text{let } \mathcal{F}[u_{\boldsymbol{\theta}}] - f = h) \\ &= \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\le\epsilon} \frac{\|\mathcal{G}[f+h] - \mathcal{G}[f]\|}{\|h\|} \\ &= \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\le\epsilon} \frac{\|\mathcal{G}[h]\|}{\|h\|} \\ &= \frac{\|f\|}{\|u\|} \|\mathcal{G}\|. \end{aligned} \tag{26}$$

Therefore, let $\epsilon \to 0^+$, $\mathrm{cond}(\mathcal{P}) = \frac{\|f\|}{\|u\|}\|\mathcal{G}\| < \infty$.

□

Theorem A.8. *If $\mathcal{F}^{-1}$ is Fréchet differentiable at f, we have that:*

$$\mathrm{cond}(\mathcal{P}) = \frac{\|f\|}{\|u\|}\left\|D\mathcal{F}^{-1}[f]\right\| < \infty, \tag{27}$$

where $D\mathcal{F}^{-1}[f]\colon S \to V$ is a bounded linear operator, the Fréchet derivative of $\mathcal{F}^{-1}$ at f.

Proof. Since $\mathcal{F}^{-1}$ is Fréchet differentiable at f, it is true that:

$$\begin{aligned} &\lim_{\epsilon\to 0^+} \sup_{0<\|h\|\le\epsilon} \frac{\left\|\mathcal{F}^{-1}[f+h] - \mathcal{F}^{-1}[f] - D\mathcal{F}^{-1}[f][h]\right\|}{\|h\|} \\ &= \lim_{\|h\|\to 0^+} \frac{\left\|\mathcal{F}^{-1}[f+h] - \mathcal{F}^{-1}[f] - D\mathcal{F}^{-1}[f][h]\right\|}{\|h\|} = 0. \end{aligned} \tag{28}$$

We can find that $W \neq \{0\}$ since $u \in V$, $\mathcal{F}[u] = f \in W$, and $\|f\| \neq 0$. Therefore, we have that:

$$\begin{aligned} &\lim_{\epsilon\to 0^+} \sup_{0<\|h\|\le\epsilon} \frac{\left\|D\mathcal{F}^{-1}[f][h]\right\|}{\|h\|} \\ &= \lim_{\epsilon\to 0^+} \sup_{0<\|h\|\le\epsilon} \left\|D\mathcal{F}^{-1}[f]\left[\frac{h}{\|h\|}\right]\right\| = \left\|D\mathcal{F}^{-1}[f]\right\|, \end{aligned} \tag{29}$$

which holds due to the fact that $D\mathcal{F}^{-1}[f]$ is a bounded linear operator.

Then, we have that:

$$\begin{aligned}
&\sup_{0<\|\delta f\|\leq\epsilon} \frac{\|\delta u\|/\|u\|}{\|\delta f\|/\|f\|} \\
&= \frac{\|f\|}{\|u\|} \sup_{0<\|\mathcal{F}[u_{\boldsymbol{\theta}}]-f\|\leq\epsilon} \frac{\|u_{\boldsymbol{\theta}}-u\|}{\|\mathcal{F}[u_{\boldsymbol{\theta}}]-f\|} \\
&= \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\leq\epsilon} \frac{\|\mathcal{F}^{-1}[f+h]-\mathcal{F}^{-1}[f]\|}{\|h\|} \qquad \text{(let } \mathcal{F}[u_{\boldsymbol{\theta}}]-f=h\text{)} \\
&\leq \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\leq\epsilon} \frac{\left\|\mathcal{F}^{-1}[f+h]-\mathcal{F}^{-1}[f]-D\mathcal{F}^{-1}[f][h]\right\|}{\|h\|} \\
&\quad + \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\leq\epsilon} \frac{\left\|D\mathcal{F}^{-1}[f][h]\right\|}{\|h\|} \to 0 + \frac{\|f\|}{\|u\|} \left\|D\mathcal{F}^{-1}[f]\right\|,
\end{aligned} \tag{30}$$

when $\epsilon \to 0^+$.

As for the left-hand side, it follows that:

$$\begin{aligned}
&\frac{\|f\|}{\|u\|} \sup_{0<\|h\|\leq\epsilon} \frac{\|\mathcal{F}^{-1}[f+h]-\mathcal{F}^{-1}[f]\|}{\|h\|} \\
&\geq \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\leq\epsilon} \left(\frac{\left\|D\mathcal{F}^{-1}[f][h]\right\|}{\|h\|} \right. \\
&\quad \left. - \frac{\left\|\mathcal{F}^{-1}[f+h]-\mathcal{F}^{-1}[f]-D\mathcal{F}^{-1}[f][h]\right\|}{\|h\|} \right) \\
&\geq \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\leq\epsilon} \left(\frac{\left\|D\mathcal{F}^{-1}[f][h]\right\|}{\|h\|} \right. \\
&\quad \left. - \sup_{0<\|h\|\leq\epsilon} \frac{\left\|\mathcal{F}^{-1}[f+h]-\mathcal{F}^{-1}[f]-D\mathcal{F}^{-1}[f][h]\right\|}{\|h\|} \right) \\
&= \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\leq\epsilon} \frac{\left\|D\mathcal{F}^{-1}[f][h]\right\|}{\|h\|} \\
&\quad - \frac{\|f\|}{\|u\|} \sup_{0<\|h\|\leq\epsilon} \frac{\left\|\mathcal{F}^{-1}[f+h]-\mathcal{F}^{-1}[f]-D\mathcal{F}^{-1}[f][h]\right\|}{\|h\|} \\
&\to \frac{\|f\|}{\|u\|} \left\|D\mathcal{F}^{-1}[f]\right\| - 0,
\end{aligned} \tag{31}$$

when $\epsilon \to 0^+$.

According to the squeeze theorem, we have proven the theorem:

$$\operatorname{cond}(\mathcal{P}) = \lim_{\epsilon\to 0^+} \sup_{0<\|\delta f\|\leq\epsilon} \frac{\|\delta u\|/\|u\|}{\|\delta f\|/\|f\|} = \frac{\|f\|}{\|u\|} \left\|D\mathcal{F}^{-1}[f]\right\| < \infty. \tag{32}$$

□

A.3 Proof for Theorem 3.3

Firstly, we define the inner product in $L^2((0, 2\pi/P))$ as:

$$\langle f, g\rangle = \frac{P}{2\pi} \int_0^{2\pi} f(x)g(x)\mathrm{d}x. \tag{33}$$

With the inner product defined above, $L^2((0, 2\pi/P))$ forms a Hilbert space. As $f \in L^2$, we can have a Fourier series representation of f:

$$f = 2c + \sum_{k\geq 1} a_k \sin(kPx) + \sum_{k\geq 1} b_k \cos(kPx). \tag{34}$$

It is then easy to obtain $u = \mathcal{F}^{-1}[f]$ from the series:

$$u = cx(x - 2\pi/P) - \sum_{k\geq 1} \frac{a_k}{k^2 P^2} \sin(kPx) - \sum_{k\geq 1} \frac{b_k}{k^2 P^2} (\cos(kPx) - 1). \tag{35}$$

By definition, $\|\mathcal{F}^{-1}\|$ can be rewrite as $\|\mathcal{F}^{-1}\| = \sup_{\|f\|=1} \|\mathcal{F}^{-1}[f]\|$. Therefore, the original problem is equivalent to the following constrained optimizing problem:

$$\begin{aligned}
&\max \ \|u\|^2 \\
&s.t. \ \|f\|^2 = 1 \\
&\text{where } \|f\|^2 = 4c^2 + \frac{1}{2}\sum_{k\geq 1} a_k^2 + \frac{1}{2}\sum_{k\geq 1} b_k^2 \\
&\qquad \|u\|^2 = \frac{1}{P^4}(\frac{8\pi^4}{15}c^2 - \frac{4\pi^2}{3}c\sum_{k\geq 1}\frac{b_k}{k^2} - 4c\sum_{k\geq 1}\frac{b_k}{k^4} + \frac{1}{2}\sum_{k\geq 1}\frac{a_k^2}{k^4} + \frac{1}{2}\sum_{k\geq 1}\frac{b_k^2}{k^4} + (\sum_{k\geq 1}\frac{b_k}{k^2})^2).
\end{aligned} \tag{36}$$

We then prove the following lemma.

Lemma A.9. *When $\|u\|^2$ reaches its maximum, we have $a_k = 0, \forall k \geq 1$.*

Proof. Firstly, it is obvious that $a_k = 0, \forall k \geq 2$. This is because the only term for a_k is $\sum_{k\geq 1} \frac{a_k^2}{k^4}$. Thus, when $\exists k \geq 2, a_k \neq 0$, then it is better to move the value from a_k to a_1.

Now we suppose $a_1 \neq 0$. Since $\|f\|^2 = 4c^2 + \frac{1}{2}\sum_{k\geq 1} a_k^2 + \frac{1}{2}\sum_{k\geq 1} b_k^2 = 1$, we can replace a_1^2 by $2 - \sum_{k\geq 1} b_k^2 - 8c^2$. So we get the following problem:

$$\begin{aligned}
&\max \ \|u\|^2 = P^{-4}((\frac{8\pi^4}{15} - 4)c^2 - \frac{4\pi^2}{3}c\sum_{k\geq 1}\frac{b_k}{k^2} - 4c\sum_{k\geq 1}\frac{b_k}{k^4} + 1 - \frac{1}{2}\sum_{k\geq 1} b_k^2 + \frac{1}{2}\sum_{k\geq 1}\frac{b_k^2}{k^4} + (\sum_{k\geq 1}\frac{b_k}{k^2})^2) \\
&s.t. \ 1 - \frac{1}{2}\sum_{k\geq 1} b_k^2 - 4c^2 > 0.
\end{aligned} \tag{37}$$

To simplify the expression, we define $B = \sum_{k\geq 1} \frac{b_k}{k^2}$. When $\|u\|^2$ reaches its maximum, it must satisfy $\frac{\partial}{\partial b_j}\|u\|^2 = 0$:

$$\frac{\partial}{\partial b_j}\|u\|^2 = P^{-4}(-\frac{4\pi^2}{3}c\frac{1}{j^2} - 4c\frac{1}{j^4} - b_k + \frac{b_k}{j^4} + 2B\frac{1}{j^2}) = 0. \tag{38}$$

When $j = 1$, we get $B = 2c(1 + \frac{\pi^2}{3})$. When $j \geq 2$, we can solve b_j from the equation that $b_j = \frac{\frac{4\pi^2}{3}cj^2 + 4c - 2Bj^2}{1 - j^4} = \frac{4c}{1+j^2}$. Therefore, we can solve $b_1 = B - \sum_{k\geq 2}\frac{b_k}{k^2} = 2c(1 + \pi\coth(\pi))$.

Now we define $d_k = b_k/c$, which are constants satisfying $d_1 = 2(1 + \pi\coth(\pi))$ and $d_j = \frac{4}{1+j^2}, \forall j \geq 2$. Then $\|u\|^2$ can be reformulized as:

$$\begin{aligned}
\|u\|^2 &= P^{-4}(1 + c^2(\frac{8\pi^4}{15} - 4 - \frac{4\pi^2}{3}\sum_{k\geq 1}\frac{d_k}{k^2} - 4\sum_{k\geq 1}\frac{d_k}{k^4} - \frac{1}{2}\sum_{k\geq 1} d_k^2 + \frac{1}{2}\sum_{k\geq 1}\frac{d_k^2}{k^4} + (\sum_{k\geq 1}\frac{d_k}{k^2})^2)) \\
&= P^{-4}(1 + c^2 S).
\end{aligned} \tag{39}$$

Where $S > 0$. From the constraint that $1 - \frac{1}{2}\sum_{k\geq 1} b_k^2 - 4c^2 = 1 - c^2(\frac{1}{2}\sum_{k\geq 1} d_k^2 + 4) > 0$, we can get the feasible interval of c: $c \in (-\sqrt{1/(\frac{1}{2}\sum_{k\geq 1} d_k^2 + 4)}, \sqrt{1/(\frac{1}{2}\sum_{k\geq 1} d_k^2 + 4)})$. In this way, $\|u\|^2$ has no maximum, leading to a contradiction. Therefore, we proved that a_1 should be zero. □

Finally, we provide a proof for Theorem 3.3.

Proof. Given the conclusion in the Lemma A.9, we will focus on b_k and c only. Now assume $c \neq 0$ and replace b_k by $d_k = b_k/c$.

$$\begin{aligned} \|f\|^2 &= c^2(4 + \frac{1}{2}\sum_{k\geq 1} d_k^2) = 1, \\ \|u\|^2 &= P^{-4}c^2(\frac{8\pi^4}{15} - \frac{4\pi^2}{3}\sum_{k\geq 1}\frac{d_k}{k^2} - 4\sum_{k\geq 1}\frac{d_k}{k^4} + \frac{1}{2}\sum_{k\geq 1}\frac{d_k^2}{k^4} + (\sum_{k\geq 1}\frac{d_k}{k^2})^2). \end{aligned} \tag{40}$$

By doing this, we can remove the constraint $\|f\|^2 = 1$ by replacing $c^2 = 2/(8 + \sum_{k\geq 1} e_k^2 + \sum_{k\geq 1} d_k^2)$. Now our objective is simply maximizing:

$$\|u\|^2 = \frac{\frac{8\pi^4}{15} - \frac{4\pi^2}{3}\sum_{k\geq 1}\frac{d_k}{k^2} - 4\sum_{k\geq 1}\frac{d_k}{k^4} + \frac{1}{2}\sum_{k\geq 1}\frac{d_k^2}{k^4} + (\sum_{k\geq 1}\frac{d_k}{k^2})^2}{P^4(8 + \sum_{k\geq 1} d_k^2)}. \tag{41}$$

To simplify the long expression, we define $B = \sum_{k\geq 1}\frac{d_k}{k^2}$, $C = \sum_{k\geq 1} d_k^2$, $D = \sum_{k\geq 1}\frac{d_k}{k^4}$ and $E = \sum_{k\geq 1}\frac{d_k^2}{k^4}$ in the following proof.

When $\|u\|^2$ reaches its maximum, it must satisfy $\frac{\partial}{\partial d_j}\|u\|^2 = 0$. Thus we can get the following equation:

$$\frac{\partial}{\partial d_j}\|u\|^2 = \frac{(8+C)(-\frac{4\pi^2}{3j^2} - \frac{4}{j^4} + \frac{d_j}{j^4} + 2B\frac{1}{j^2}) - 2d_j(\frac{8\pi^4}{15} - \frac{4\pi^2}{3}B - 4D + \frac{1}{2}E + B^2)}{P^4(8+C)^2} = 0. \tag{42}$$

From the equation we can solve for d_k:

$$d_k = \frac{((2B - \frac{4\pi^2}{3})k^2 - 4)(8+C)}{(\frac{16\pi^4}{15} - \frac{8\pi^2}{3}B - 8D + E + 2B^2)k^4 - 8 - C}. \tag{43}$$

Now we learn that d_k can be determined by B, C, D, E. We denote $d_k = g_k(B, C, D, E)$ and we can now solve B, C, D, E from the 4 equations below:

$$\begin{aligned} B &= \sum_{k\geq 1}\frac{g_k(B,C,D,E)}{k^2}, \\ C &= \sum_{k\geq 1} g_k^2(B,C,D,E), \\ D &= \sum_{k\geq 1}\frac{g_k(B,C,D,E)}{k^4}, \\ E &= \sum_{k\geq 1}\frac{g_k^2(B,C,D,E)}{k^4}. \end{aligned} \tag{44}$$

Where we get $B = \frac{2\pi^2}{3} - 8, C = \pi^2 - 8, D = \frac{2(-720+60\pi^2+\pi^4)}{45}, E = \frac{8(-2160+210\pi^2+\pi^4)}{45}$.

Thus, we get $d_k = -\frac{4}{4k^2-1}$ and $\|u\|^2 = 16P^{-4}$ for maximum value. So $\|\mathcal{F}^{-1}\| = \|u\| = 4P^{-2}$

□

A.4 Proof for Corollary 3.4

Proof. Since $\mathrm{cond}(\mathcal{P}) < \infty$, we arbitrarily take $M > 0$, then there exists $\xi > 0$ such that:

$$\left|\sup_{0<\|\delta f\|\leq\epsilon}\frac{\|\delta u\|/\|u\|}{\|\delta f\|/\|f\|} - \mathrm{cond}(\mathcal{P})\right| < M, \tag{45}$$

which holds for any $\epsilon \in (0, \xi)$.

Thus, we can defined $\alpha\colon (0, \xi) \to \mathbb{R}$ as:

$$\alpha(x) = \sup_{0<\|\delta f\|\leq x} \frac{\|\delta u\|/\|u\|}{\|\delta f\|/\|f\|} - \mathrm{cond}(\mathcal{P}), \tag{46}$$

which satisfies that $\lim_{x\to 0^+} \alpha(x) = 0$.

It follows that:

$$\sup_{0<\|\delta f\|\leq \epsilon} \frac{\|\delta u\|/\|u\|}{\|\delta f\|/\|f\|} = \mathrm{cond}(\mathcal{P}) + \alpha(\epsilon), \quad \forall \epsilon \in (0, \xi), \tag{47}$$

which is equivalent to the statement that for any $\epsilon \in (0, \xi)$, when $0 < \sqrt{\mathcal{L}(\boldsymbol{\theta})} \leq \epsilon$:

$$\frac{\|u_{\boldsymbol{\theta}} - u\|}{\|u\|} \leq (\mathrm{cond}(\mathcal{P}) + \alpha(\epsilon)) \frac{\sqrt{\mathcal{L}(\boldsymbol{\theta})}}{\|f\|}, \quad \forall \boldsymbol{\theta} \in \Theta. \tag{48}$$

If $\sqrt{\mathcal{L}(\boldsymbol{\theta})} = 0$, then $u_{\boldsymbol{\theta}} = u$ since the BVP is well-posed, and thus Eq. (48) still holds. □

A.5 Proof for Theorem 3.5

Firstly, we state the following assumptions.

Assumption A.10. Let $B(\boldsymbol{\theta}^*, r) = \{\boldsymbol{\theta} \mid \boldsymbol{\theta} \in \Theta \wedge \|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_1 < r\}, r > 0$, where $\|\cdot\|_1$ is the L^1 vector norm. We assume that $\boldsymbol{\theta}^{(k)} \in B(\boldsymbol{\theta}^*, r), \forall k \geq 0$ and that $\xi > 1$.

Remark. We posit that the loss function can be optimized to a relatively small value (r is typically a small constant), allowing concentrating on the subsequent optimization starting from a point near the minimum. Empirically, optimizing the loss function near the minimum constitutes a significant portion of the entire process, lending credence to the relevance and practical utility of our assumption. Besides, we additionally assume that $\xi > 1$ to make $\alpha(1/\sqrt{k})$ well-defined.

Assumption A.11. $\mathcal{L}(\boldsymbol{\theta}) \in C^2(B(\boldsymbol{\theta}^*, r))$, and $\boldsymbol{\theta}^*$ is the (unique and correct) local minimum of $\mathcal{L}(\boldsymbol{\theta})$ (thus, $\mathcal{L}(\boldsymbol{\theta}^*) \approx 0$). Moreover, in $B(\boldsymbol{\theta}^*, r)$, $\mathcal{L}(\boldsymbol{\theta})$ is convex and L-smooth with constant $L > 0$, i.e., $\|\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_1) - \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_2)\| \leq L\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|$ for any $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in B(\boldsymbol{\theta}^*, r)$.

Remark. We reasonably assume the convexity of $\mathcal{L}(\boldsymbol{\theta})$, as it can be well approximated by a truncated local Taylor expansion that is convex within $B(\boldsymbol{\theta}^*, r)$. We refer readers to Appendix A.6 for a detailed discussion on the approximation error of the local Taylor expansion.

Assumption A.12. The learning rate η is sufficiently small, i.e., $\eta \leq 1/L$.

Secondly, we introduce two lemmas as follows.

Lemma A.13. *If $f\colon X \to \mathbb{R}$ is L-smooth with constant $L > 0$ then:*

$$f\left(x - \frac{1}{L}\nabla f(x)\right) - f(x) \leq -\frac{1}{2L}\|\nabla f(x)\|^2, \tag{49}$$

holds for all $x \in X \subset \mathbb{R}^n$.

Proof. We refer readers to Gower. (2018). □

Lemma A.14. *Suppose $f\colon X \to \mathbb{R}$ is convex, differentiable, and is L-smooth with constant $L > 0$. Then if we run gradient descent for k iterations with a fixed step size $\eta \leq 1/L$, given an initial $x^{(0)}$, it will yield a sequence $x^{(1)}, \ldots, x^{(k)}$. Assuming that $x^{(i)} \in X, 0 \leq i \leq k$, we have that:*

$$f(x^{(k)}) - f(x^*) \leq \frac{\|x^{(0)} - x^*\|^2}{2\eta k}, \tag{50}$$

where $x^* \in X$ and $f(x^*)$ is the optimal value.

Proof. We refer readers to Ryan Tibshirani (2013). □

Finally, we provide proof for Theorem 3.5 under Assumption A.10, A.11, A.12, and A.1 – A.4.

Proof. Starting from $\boldsymbol{\theta}^{(0)}$, we run gradient descent until k steps. Based on Lemma A.14, we have that:

$$\mathcal{L}(\boldsymbol{\theta}^{(k)}) \leq \frac{\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^*\|^2}{2\eta k} + \mathcal{L}(\boldsymbol{\theta}^*) \approx \frac{\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^*\|^2}{2\eta k}. \tag{51}$$

According to Corollary 3.4 (where we let $\epsilon = \|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^*\|/\sqrt{2\eta k}$), there exists a function $\alpha' \colon (0, \xi') \to \mathbb{R}, \xi' > 0$ with $\lim_{x \to 0^+} \alpha'(x) = 0$, such that:

$$\frac{\|u_{\boldsymbol{\theta}^{(k)}} - u\|}{\|u\|} \leq \left(\mathrm{cond}(\mathcal{P}) + \alpha'\left(\frac{\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^*\|}{\sqrt{2\eta k}}\right)\right) \frac{\sqrt{\mathcal{L}(\boldsymbol{\theta}^{(k)})}}{\|f\|}, \tag{52}$$

which can be rewritten as:

$$\frac{\|u_{\boldsymbol{\theta}^{(k)}} - u\|}{\|u\|} \leq \left(\mathrm{cond}(\mathcal{P}) + \alpha\left(\frac{1}{\sqrt{k}}\right)\right) \frac{\sqrt{\mathcal{L}(\boldsymbol{\theta}^{(k)})}}{\|f\|}, \tag{53}$$

where $\alpha \colon (0, \xi) \to \mathbb{R}, \xi > 0$ and $\alpha(x) = \alpha'((\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^*\|/\sqrt{2\eta})x)$.

Plugging Eq. (51) into the above equation, we can conclude the following:

$$\frac{\|u_{\boldsymbol{\theta}^{(k)}} - u\|}{\|u\|} \lessapprox \left(\mathrm{cond}(\mathcal{P}) + \alpha\left(\frac{1}{\sqrt{k}}\right)\right) \frac{\|\boldsymbol{\theta}^{(0)} - \boldsymbol{\theta}^*\|}{\sqrt{2\eta k}\|f\|}, \tag{54}$$

which is our target in this proof. □

A.6 Discussion on Approximation Error of the Truncated Taylor Expansion

In this subsection, we will analyze the approximation error of the second-order local Taylor expansion $\widetilde{\mathcal{L}}(\boldsymbol{\theta})$ w.r.t. $\mathcal{L}(\boldsymbol{\theta})$ in $B(\boldsymbol{\theta}^*, r)$, which is given by:

$$\widetilde{\mathcal{L}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}^*) + \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^*)^\top(\boldsymbol{\theta} - \boldsymbol{\theta}^*) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^\top \nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^*)(\boldsymbol{\theta} - \boldsymbol{\theta}^*), \tag{55}$$

where the Hessian matrix $\nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^*)$ is symmetric positive semidefinite since $\boldsymbol{\theta}^*$ is a local minimum and $\mathcal{L} \in C^2(B(\boldsymbol{\theta}^*, r))$ (under Assumption A.10 and A.11), and thus $\widetilde{\mathcal{L}}(\boldsymbol{\theta})$ is convex.

Theorem A.15. *Suppose $S \supset B(\boldsymbol{\theta}^*, r)$ is an open convex set where $\mathcal{L}$ is of class C^3 in S and all of its third-order partial derivatives can be bounded by $M > 0$, i.e., $|\partial^\alpha \mathcal{L}(\boldsymbol{\theta})| \leq M$ for any $\boldsymbol{\theta} \in S$ and any multi-index α with $|\alpha| = 3$, it follows that:*

$$|\mathcal{L}(\boldsymbol{\theta}) - \widetilde{\mathcal{L}}(\boldsymbol{\theta})| \leq \frac{M}{3!} r^3, \tag{56}$$

which holds for any $\boldsymbol{\theta} \in B(\boldsymbol{\theta}^*, r)$.

Proof. According to Taylor's theorem, we have that:

$$\begin{aligned} |\mathcal{L}(\boldsymbol{\theta}) - \widetilde{\mathcal{L}}(\boldsymbol{\theta})| &\leq \frac{M}{3!}\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_1^3 \\ &\leq \frac{M}{3!} r^3. \end{aligned} \tag{57}$$

□

Remark. This theorem unveils that the approximation error bound is proportional to the third power of the radius r. When our optimization is pushed really close to the minimum, such an error is completely negligible.

A.7 PROOF FOR THEOREM 3.6

We discretize the loss function $\mathcal{L}(\boldsymbol{\theta})$ on a set of collocation points $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$:

$$\mathcal{L}(\boldsymbol{\theta}) \propto \hat{\mathcal{L}}(\boldsymbol{\theta}) = \frac{1}{2}\|\mathcal{F}[u_{\boldsymbol{\theta}}](\boldsymbol{X}) - f(\boldsymbol{X})\|^2, \tag{58}$$

where $\boldsymbol{X} \in \mathbb{R}^{N\times d} = [\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}]^\top$.

The NTK $\boldsymbol{K}(t) \in \mathbb{R}^{N\times N}$ of PINNs is defined to be:

$$\boldsymbol{K}_{ij}(t) = \frac{\partial \mathcal{F}[u_{\boldsymbol{\theta}(t)}](\boldsymbol{x}^{(i)})}{\partial \boldsymbol{\theta}} \cdot \frac{\partial \mathcal{F}[u_{\boldsymbol{\theta}(t)}](\boldsymbol{x}^{(j)})}{\partial \boldsymbol{\theta}}, \quad 1 \le i, j \le N, t \in [0, +\infty). \tag{59}$$

From Jacot et al. (2018); Wang et al. (2022c), we can obtain that:

$$\frac{\partial \mathcal{F}[u_{\boldsymbol{\theta}(t)}](\boldsymbol{X})}{\partial t} = -\boldsymbol{K}(t)(\mathcal{F}[u_{\boldsymbol{\theta}(t)}](\boldsymbol{X}) - f(\boldsymbol{X})), \tag{60}$$

According to NTK's conclusion (Jacot et al., 2018; Wang et al., 2022c), $\boldsymbol{K}(t)$ nearly stays invariant during the training process (in the infinite-width limit):

$$\boldsymbol{K}(t) \approx \boldsymbol{K}^{\infty}, \quad \forall t \in [0, +\infty), \tag{61}$$

where $\boldsymbol{K}^{\infty}$ is a fixed kernel. Therefore, Eq. (60) can be further rewritten as:

$$\mathcal{F}[u_{\boldsymbol{\theta}(t)}](\boldsymbol{X}) \approx \left(\boldsymbol{I} - e^{-\boldsymbol{K}(t)t}\right) f(\boldsymbol{X}). \tag{62}$$

Since $\boldsymbol{K}(t)$ is positive semi-definite (Wang et al., 2022c) and is nearly time-invariant, we can take its spectral decomposition and make the orthogonal part time-invariant: $\boldsymbol{K}(t) \approx \boldsymbol{Q}^\top \Lambda(t) \boldsymbol{Q}$, where $\boldsymbol{Q}$ is a time-invariant orthogonal matrix and $\Lambda(t)$ is a diagonal matrix with entries being the eigenvalues $\lambda_i(t) \ge 0$ of $\boldsymbol{K}(t)$. Consequently, we can further derive that:

$$\mathcal{F}[u_{\boldsymbol{\theta}(t)}](\boldsymbol{X}) - f(\boldsymbol{X}) \approx -\boldsymbol{Q}^\top e^{-\Lambda(t)t} \boldsymbol{Q} f(\boldsymbol{X}), \tag{63}$$

which is equivalent to:

$$\boldsymbol{Q}\left(\mathcal{F}[u_{\boldsymbol{\theta}(t)}](\boldsymbol{X}) - f(\boldsymbol{X})\right) \approx -e^{-\Lambda(t)t} \boldsymbol{Q} f(\boldsymbol{X}). \tag{64}$$

The equation suggests that the i-th element of the left-hand side will diminish approximately at the rate of $e^{-\lambda_i(t)t}$. Put differently, the eigenvalues of the kernel serve as critical indicators, characterizing the rate at which the training loss declines. This motivates us to define the average convergence rate $c(t)$ as in Wang et al. (2022c):

$$c(t) = \frac{1}{N}\sum_{i=1}^{N} \lambda_i(t) = \frac{1}{N}\mathrm{tr}(\boldsymbol{K}). \tag{65}$$

Let $f_{\boldsymbol{\theta}} = \mathcal{F}[u_{\boldsymbol{\theta}}]$. Delving into the expression of $c(t)$, we have that:

$$\begin{aligned}
c(t) &= \frac{1}{N}\sum_{i=1}^{N} \left\| \frac{\partial \mathcal{F}[u_{\boldsymbol{\theta}(t)}]}{\partial \boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) \right\|^2 \\
&= \frac{1}{N}\sum_{i=1}^{N} \left\| \left(\frac{\partial \mathcal{F}[u_{\boldsymbol{\theta}(t)}]}{\partial u} \circ \frac{\partial u_{\boldsymbol{\theta}(t)}}{\partial \boldsymbol{\theta}} \right)(\boldsymbol{x}^{(i)}) \right\|^2 \\
&\approx \frac{1}{|\Omega|} \left\| \frac{\partial \mathcal{F}[u_{\boldsymbol{\theta}(t)}]}{\partial u} \circ \frac{\partial u_{\boldsymbol{\theta}(t)}}{\partial \boldsymbol{\theta}} \right\|^2 && (L^2 \text{ function norm}) \\
&= \frac{1}{|\Omega|} \left\| \left(D\mathcal{F}^{-1}[f_{\boldsymbol{\theta}(t)}]\right)^{-1} \circ \frac{\partial u_{\boldsymbol{\theta}(t)}}{\partial \boldsymbol{\theta}} \right\|^2 \\
&\ge \frac{1/|\Omega|}{\|D\mathcal{F}^{-1}[f_{\boldsymbol{\theta}(t)}]\|^2} \left\| \frac{\partial u_{\boldsymbol{\theta}(t)}}{\partial \boldsymbol{\theta}} \right\|^2 && (\text{operator norm of } D\mathcal{F}^{-1}[f_{\boldsymbol{\theta}(t)}]) \\
&= \frac{\|f\|^2/(\|u\|^2|\Omega|)}{(\mathrm{cond}(\mathcal{P}))^2 + \alpha(\|f_{\boldsymbol{\theta}(t)} - f\|^2)} \left\| \frac{\partial u_{\boldsymbol{\theta}(t)}}{\partial \boldsymbol{\theta}} \right\|^2 \\
&= \frac{\|f\|^2/(\|u\|^2|\Omega|)}{(\mathrm{cond}(\mathcal{P}))^2 + \alpha(\mathcal{L}(\boldsymbol{\theta}(t)))} \left\| \frac{\partial u_{\boldsymbol{\theta}(t)}}{\partial \boldsymbol{\theta}} \right\|^2,
\end{aligned} \tag{66}$$

where $D\mathcal{F}^{-1}[w]\colon \mathcal{F}(U) \to V$ is the Fréchet derivative of $\mathcal{F}^{-1}$ at w.

B SUPPLEMENTS FOR SECTION 4

B.1 DETAILED DERIVATION FOR EQ. (13)

Lemma B.1. *Supposing that $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is invertible, we have:*

$$\lim_{\epsilon\to 0^+} \sup_{\substack{0<\|\boldsymbol{v}\|\leq\epsilon \\ \boldsymbol{v}\in\mathbb{R}^N}} \frac{\|\boldsymbol{A}\boldsymbol{v}\|}{\|\boldsymbol{v}\|} = \|\boldsymbol{A}\|. \tag{67}$$

Proof. For any $\epsilon > 0$, we firstly prove that:

$$\left\{ \frac{\|\boldsymbol{A}\boldsymbol{v}\|}{\|\boldsymbol{v}\|} \colon 0 < \|\boldsymbol{v}\| \leq \epsilon \wedge \boldsymbol{v} \in \mathbb{R}^N \right\} = \left\{ \frac{\|\boldsymbol{A}\boldsymbol{v}\|}{\|\boldsymbol{v}\|} \colon \|\boldsymbol{v}\| \neq 0 \wedge \boldsymbol{v} \in \mathbb{R}^N \right\}. \tag{68}$$

We only need to prove that:

$$\left\{ \frac{\|\boldsymbol{A}\boldsymbol{v}\|}{\|\boldsymbol{v}\|} \colon 0 < \|\boldsymbol{v}\| \leq \epsilon \wedge \boldsymbol{v} \in \mathbb{R}^N \right\} \supseteq \left\{ \frac{\|\boldsymbol{A}\boldsymbol{v}\|}{\|\boldsymbol{v}\|} \colon \|\boldsymbol{v}\| \neq 0 \wedge \boldsymbol{v} \in \mathbb{R}^N \right\}, \tag{69}$$

because the other direction is obvious. For any $a \in \left\{ \|\boldsymbol{A}\boldsymbol{v}\|/\|\boldsymbol{v}\| \colon \|\boldsymbol{v}\| \neq 0 \wedge \boldsymbol{v} \in \mathbb{R}^N \right\}$, there exists $\boldsymbol{v}$ with $\|\boldsymbol{v}\| \neq 0$ such that $a = \|\boldsymbol{A}\boldsymbol{v}\|/\|\boldsymbol{v}\|$. We consider $\boldsymbol{v}' = \epsilon\boldsymbol{v}/\|\boldsymbol{v}\|$. It is clear that $\|\boldsymbol{v}'\| = \epsilon$ and that:

$$\frac{\|\boldsymbol{A}\boldsymbol{v}'\|}{\|\boldsymbol{v}'\|} = \frac{\epsilon/\|\boldsymbol{v}\|\|\boldsymbol{A}\boldsymbol{v}\|}{\epsilon/\|\boldsymbol{v}\|\|\boldsymbol{v}\|} = \frac{\|\boldsymbol{A}\boldsymbol{v}\|}{\|\boldsymbol{v}\|} = a. \tag{70}$$

Then, we have that $a \in \left\{ \|\boldsymbol{A}\boldsymbol{v}\|/\|\boldsymbol{v}\| \colon 0 < \|\boldsymbol{v}\| \leq \epsilon \wedge \boldsymbol{v} \in \mathbb{R}^N \right\}$. Therefore, Eq. (68) holds and thus:

$$\sup \left\{ \frac{\|\boldsymbol{A}\boldsymbol{v}\|}{\|\boldsymbol{v}\|} \colon 0 < \|\boldsymbol{v}\| \leq \epsilon \wedge \boldsymbol{v} \in \mathbb{R}^N \right\} = \sup \left\{ \frac{\|\boldsymbol{A}\boldsymbol{v}\|}{\|\boldsymbol{v}\|} \colon \|\boldsymbol{v}\| \neq 0 \wedge \boldsymbol{v} \in \mathbb{R}^N \right\} = \|\boldsymbol{A}\|. \tag{71}$$

Let $\epsilon \to 0^+$, we finally prove that this lemma.

□

We now start our derviation. Let $\boldsymbol{u}_{\boldsymbol{\theta}}$ denote the predictions of the neural network at the mesh locations: $\boldsymbol{u}_{\boldsymbol{\theta}} = (u_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))_{i=1}^N$. From Definition 3.1, we have:

$$\begin{aligned}
\operatorname{cond}(\mathcal{P}) &= \lim_{\epsilon\to 0^+} \sup_{\substack{0<\|\delta f\|\leq\epsilon \\ \boldsymbol{\theta}\in\Theta}} \frac{\|\delta u\|/\|u\|}{\|\delta f\|/\|f\|} \\
&= \frac{\|f\|}{\|u\|} \lim_{\epsilon\to 0^+} \sup_{\substack{0<\|\mathcal{F}[u_{\boldsymbol{\theta}}]-f\|\leq\epsilon \\ \boldsymbol{\theta}\in\Theta}} \frac{\|u_{\boldsymbol{\theta}} - u\|}{\|\mathcal{F}[u_{\boldsymbol{\theta}}] - f\|} \\
&\approx \frac{\|\boldsymbol{b}\|}{\|\boldsymbol{u}\|} \lim_{\epsilon\to 0^+} \sup_{\substack{0<\|\boldsymbol{A}\boldsymbol{u}_{\boldsymbol{\theta}}-\boldsymbol{b}\|\leq\epsilon \\ \boldsymbol{\theta}\in\Theta}} \frac{\|\boldsymbol{u}_{\boldsymbol{\theta}} - \boldsymbol{u}\|}{\|\boldsymbol{A}\boldsymbol{u}_{\boldsymbol{\theta}} - \boldsymbol{b}\|} \\
&= \frac{\|\boldsymbol{b}\|}{\|\boldsymbol{u}\|} \lim_{\epsilon\to 0^+} \sup_{\substack{0<\|\boldsymbol{A}(\boldsymbol{u}_{\boldsymbol{\theta}}-\boldsymbol{u})\|\leq\epsilon \\ \boldsymbol{\theta}\in\Theta}} \frac{\|\boldsymbol{u}_{\boldsymbol{\theta}} - \boldsymbol{u}\|}{\|\boldsymbol{A}(\boldsymbol{u}_{\boldsymbol{\theta}} - \boldsymbol{u})\|},
\end{aligned} \tag{72}$$

where the approximate equality holds because we discretize the BVP. Because of the assumption that the neural network has sufficient approximation capability (see Assumption A.4) and the fact that $\|\boldsymbol{A}\boldsymbol{v}\| \leq \|\boldsymbol{A}\|\|\boldsymbol{v}\|, \forall \boldsymbol{v} \in \mathbb{R}^N$, Eq. (72) can be further rewritten as:

$$\frac{\|\boldsymbol{b}\|}{\|\boldsymbol{u}\|} \lim_{\epsilon\to 0^+} \sup_{\substack{0<\|\boldsymbol{v}\|\leq\epsilon \\ \boldsymbol{v}\in\mathbb{R}^N}} \frac{\|\boldsymbol{v}\|}{\|\boldsymbol{A}\boldsymbol{v}\|} = \frac{\|\boldsymbol{b}\|}{\|\boldsymbol{u}\|}\|\boldsymbol{A}^{-1}\|, \tag{73}$$

where the equality holds according to Lemma B.1.

When we apply the precondition number $\boldsymbol{P}$ satisfying that $\boldsymbol{P} \approx \boldsymbol{A}$ ($\boldsymbol{P}^{-1} \approx \boldsymbol{A}^{-1}$, also), the linear system transfers from $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}$ to $\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{u} = \boldsymbol{P}^{-1}\boldsymbol{b}$. Equivalently, we have $\boldsymbol{A} \to \boldsymbol{P}^{-1}\boldsymbol{A}$ and $\boldsymbol{b} \to \boldsymbol{P}^{-1}\boldsymbol{b}$. Then, Eq. (73) becomes:

$$\frac{\|\boldsymbol{b}\|}{\|\boldsymbol{u}\|}\|\boldsymbol{A}^{-1}\| \longrightarrow \frac{\|\boldsymbol{P}^{-1}\boldsymbol{b}\|}{\|\boldsymbol{u}\|}\|\boldsymbol{A}^{-1}\boldsymbol{P}\| \approx \frac{\|\boldsymbol{A}^{-1}\boldsymbol{b}\|}{\|\boldsymbol{u}\|}\|\boldsymbol{A}^{-1}\boldsymbol{A}\| = 1. \tag{74}$$

B.2 Enforcing Boundary Conditions via Discretized Losses

In this subsection, we will introduce how to enforce the boundary conditions (BCs) by our discretized loss function.

Dirichlet BCs. We consider the following 1D Poisson equation:

$$\begin{aligned} \Delta u(x) &= 0, \quad x \in \Omega = (0,1), \\ u(x) &= c, \quad x \in \partial\Omega = \{0,1\}, \end{aligned} \tag{75}$$

where $u = u(x)$ is the unknown and $c \in \mathbb{R}$. We discretize the interval $[0,1]$ into five points $\{0, 0.25, 0.5, 0.75, 1\}$ and construct the following discretized equation by the FDM:

$$\frac{u(x+h) - 2u(x) + u(x-h)}{h^2} = 0, \quad x = \{0.25, 0.5, 0.75\}, \tag{76}$$

where $h = 0.25$ and $u(0) = u(1) = c$. This can be reformulated as the following linear system:

$$\begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} u(0.75) \\ u(0.5) \\ u(0.25) \end{bmatrix} = \begin{bmatrix} -c \\ 0 \\ -c \end{bmatrix}. \tag{77}$$

Now, we can see that the BC is enforced by substituting its values into the equation. Similar strategies can also be applied to other numerical schemes such as the FEM.

Neumann BCs and Robin BCs. Such types of BCs are typically enforced via the weak form of the PDEs. We consider the following Poisson equation with a Robin BC:

$$\begin{aligned} -\Delta u(\boldsymbol{x}) &= f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \\ \alpha u(\boldsymbol{x}) + \beta \frac{\partial u}{\partial n}(\boldsymbol{x}) &= g(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{78}$$

where $\alpha, \beta \in \mathbb{R}$, $\frac{\partial u}{\partial n}(\boldsymbol{x})$ is the normal derivative. The weak form is derived as:

$$-\int_\Omega v \Delta u \, \mathrm{d}\boldsymbol{x} = \int_\Omega f v \, \mathrm{d}\boldsymbol{x}, \tag{79}$$

where $v \in H^1$ is the test function. Then, we perform integration by parts:

$$\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\boldsymbol{x} - \int_{\partial\Omega} \frac{\partial u}{\partial n} v \, \mathrm{d}\boldsymbol{x} = \int_\Omega f v \, \mathrm{d}\boldsymbol{x}. \tag{80}$$

We plug in the Robin BC to obtain:

$$\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\boldsymbol{x} + \frac{\alpha}{\beta} \int_{\partial\Omega} uv \, \mathrm{d}\boldsymbol{x} = \int_\Omega f v \, \mathrm{d}\boldsymbol{x} + \frac{1}{\beta} \int_{\partial\Omega} g v \, \mathrm{d}\boldsymbol{x}. \tag{81}$$

Finally, we assemble the above equation by the FEM and can obtain the loss that incorporates the BC. For other numerical schemes like FDM, we can plug in the finite difference formula of the derivative term to enforce the BC, which is similar to the cases of Dirichlet BCs.

Other BCs. For other forms of BCs, enforcement is usually implemented by substitution. For example, when dealing with left-right periodic BCs, we often substitute the values in the left boundary with the values in the right boundary. Or equivalently, we reduce the degrees of freedom of the left and right boundaries by half.

B.3 Handling Time-Dependent & Nonlinear Problems

We now introduce our strategies to handle time-dependent and nonlinear problems.

Algorithm 2 Preconditoning PINNs for time-dependent problems (sequential)

Input: number of iterations K, mesh size N, learning rate η, time steps $\{t_i\}_{i=1}^n$, initial condition $u_0(\boldsymbol{x})$, and initial parameters $\boldsymbol{\theta}^{(0)}$
Output: solutions at each time steps $u_i(\boldsymbol{x}), i = 1, \ldots, n$
for $i = 1, \ldots, n$ **do**
  Generate a mesh $\{\boldsymbol{x}^{(j)}\}_{j=1}^N$ for current time step
  Evaluate $u_{i-1}(\boldsymbol{x})$ on the mesh to obtain $\boldsymbol{u}_{i-1}$
  Assemble the linear system $\boldsymbol{A}' = (\boldsymbol{I} + \boldsymbol{A}(t_i)), \boldsymbol{b}' = (\boldsymbol{b}(t_i) + \boldsymbol{u}_{i-1})$ according to Eq. (86)
  Compute the preconditioner for $\boldsymbol{A}'$: $\boldsymbol{P} = \widehat{\boldsymbol{L}}\widehat{\boldsymbol{U}}$ via ILU
  for $k = 1, \ldots, K$ **do**
    Evaluate the neural network $u_{\boldsymbol{\theta}^{(k-1)}}$ on mesh points: $\boldsymbol{u}_{\boldsymbol{\theta}^{(k-1)}} = (u_{\boldsymbol{\theta}^{(k-1)}}(\boldsymbol{x}^{(j)}))_{j=1}^N$
    Compute the loss function $\mathcal{L}^\dagger(\boldsymbol{\theta}^{(k-1)})$ by:

$$\mathcal{L}^\dagger(\boldsymbol{\theta}) = \left\| \boldsymbol{P}^{-1}(\boldsymbol{A}'\boldsymbol{u}_{\boldsymbol{\theta}} - \boldsymbol{b}') \right\|^2 \tag{82}$$

    Update the parameters via gradient descent: $\boldsymbol{\theta}^{(k)} \leftarrow \boldsymbol{\theta}^{(k-1)} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}^\dagger(\boldsymbol{\theta}^{(k-1)})$
  end for
  Let $u_i(\boldsymbol{x}) \leftarrow u_{\boldsymbol{\theta}^{(K)}}(\boldsymbol{x})$
  Let $\boldsymbol{\theta}^{(0)} \leftarrow \boldsymbol{\theta}^{(K)}$ (transfer learning)
end for
Note:
(a) If the mesh $\{\boldsymbol{x}^{(j)}\}_{j=1}^N$, the matrix $\boldsymbol{A}$, and the bias $\boldsymbol{b}$ do not vary with time, we can only generate them once at the beginning instead of regeneration at each time step.
(b) We use transfer learning to migrate the neural network from the previous time step to the next time step since the solution varies little for most physical problems (if the number of time steps n is sufficiently large).

Time-Dependent Problems. For problems with time dependencies, one straightforward approach is to treat time as an additional spatial dimension, resulting in a unified spatial-temporal equation. For instance, supposing that we are dealing with a problem defined in a 2D square $[0, 1]^2$ and a time interval $[0, 1]$, we can consider it as a problem defined in a 3D cube $[0, 1]^3$, where we build the mesh and assemble the equation system. However, this approach can necessitate extremely fine meshing to ensure adequate accuracy, particularly for problems with high temporal frequencies.

An alternative approach involves discretizing the time dimension into specific time steps and subsequently solving the spatial equation iteratively for each step. For example, we consider the following abstraction of time-dependent PDEs:

$$\frac{\partial u}{\partial t}(\boldsymbol{x}, t) + \mathcal{F}[u](\boldsymbol{x}, t) = f(\boldsymbol{x}, t), \quad \forall \boldsymbol{x} \in \Omega, t \in (0, T], \tag{84}$$

with the initial condition of $\boldsymbol{u}(\boldsymbol{x}, 0) = h(\boldsymbol{x}), \forall \boldsymbol{x} \in \Omega$ and proper boundary conditions, where t denotes the time coordinate, $T \in \mathbb{R}^+$, and u is the unknown. We now discretize the time interval into n time $t_0, t_1, \ldots, t_n$ ($t_0 = 0, t_n = T$). Let $u_i(\boldsymbol{x})$ denote $u(\boldsymbol{x}, t_i)$. Starting from $u_0(\boldsymbol{x}) = h(\boldsymbol{x})$, we can construct the following iterative systems ($i = 1, 2, 3, \ldots,$):

$$u_i(\boldsymbol{x}) + (t_i - t_{i-1})\mathcal{F}[u_i](\boldsymbol{x}, t_i) = (t_i - t_{i-1})f(\boldsymbol{x}, t_i) + u_{i-1}(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \Omega. \tag{85}$$

Then, we perform discretization in the spatial dimension with a mesh $\{\boldsymbol{x}^{(i)}\}_{i=1}^N$:

$$(\boldsymbol{I} + \boldsymbol{A}(t_i))\boldsymbol{u}_i = \boldsymbol{b}(t_i) + \boldsymbol{u}_{i-1}, \tag{86}$$

where $\boldsymbol{A}(t_i), \boldsymbol{b}(t_i)$ are matrices at time t_i and $\boldsymbol{u}_i = (u_i(\boldsymbol{x}^{(j)}))_{j=1}^N$. It is noted that the specific form of Eq. (86) depends on the numerical schemes employed. For example, when using the FEM, Eq. (86) should become:

$$(\boldsymbol{K} + \boldsymbol{A}(t_i))\boldsymbol{u}_i = \boldsymbol{b}(t_i) + \boldsymbol{K}\boldsymbol{u}_{i-1}, \tag{87}$$

where $\boldsymbol{K}$ is the mass matrix which simply integrates the trial and test functions.

Now, we can iteratively solve Eq. (86) with a PINN to obtain the solution at each time step. Specifically, we can sequentially solve each time step at one time, as described by Algorithm 2, or divide the time interval into several sub-intervals and train in parallel within sub-intervals (see Algorithm 3).

Algorithm 3 Preconditoning PINNs for time-dependent problems (parallelized)

Input: number of iterations K, mesh size N, learning rate η, time steps for m sub-intervals $S_1 = \{t_i^1\}_{i=1}^n, \ldots, S_m = \{t_i^m\}_{i=1}^n$ (each sub-interval has n steps), initial condition $u_0(\boldsymbol{x})$, and initial parameters $\boldsymbol{\theta}_i^{(0)}, i = 1, \ldots, n$
Output: solutions at each time steps within each sub-interval $u_i^s(\boldsymbol{x}), i = 1, \ldots, n, s = 1, \ldots, m$
Initialize: $u_0^1(\boldsymbol{x}) \leftarrow u_0(\boldsymbol{x})$
for $s = 1, \ldots, m$ **do**
  Generate a mesh $\{\boldsymbol{x}^{(j)}\}_{j=1}^N$ for current time step
  Evaluate $u_0^s(\boldsymbol{x})$ on the mesh to obtain $\boldsymbol{u}_0^s$
  Assemble the matrix $\boldsymbol{A}_i' = (\boldsymbol{I} + \boldsymbol{A}(t_i^s)), i = 1, \ldots, n$
  Compute the preconditioner for $\boldsymbol{A}_i'$: $\boldsymbol{P}_i = \widehat{\boldsymbol{L}}_i \widehat{\boldsymbol{U}}_i$ via ILU, $i = 1, \ldots, n$
  for $k = 1, \ldots, K$ **do**
    Evaluate the neural network $u_{\boldsymbol{\theta}_i^{(k-1)}}$ on mesh points: $\boldsymbol{u}_{\boldsymbol{\theta}_i^{(k-1)}} = (u_{\boldsymbol{\theta}_i^{(k-1)}}(\boldsymbol{x}^{(j)}))_{j=1}^N$, $i = 1, \ldots, n$
    Assemble the bias $\boldsymbol{b}_1' = (\boldsymbol{b}(t_1^s) + \boldsymbol{u}_0^s)$ and $\boldsymbol{b}_i' = (\boldsymbol{b}(t_i^s) + \boldsymbol{u}_{\boldsymbol{\theta}_{i-1}^{(k-1)}})$, where $i = 2, \ldots, n$
    Compute the loss function $\mathcal{L}^\dagger(\boldsymbol{\theta}_1^{(k-1)}, \ldots, \boldsymbol{\theta}_n^{(k-1)})$ by:

$$\mathcal{L}^\dagger(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n) = \sum_{i=1}^n w_i \left\| \boldsymbol{P}_i^{-1}(\boldsymbol{A}_i' \boldsymbol{u}_{\boldsymbol{\theta}_i} - \boldsymbol{b}_i') \right\|^2, \tag{83}$$

    where w_i is the reweighting parameters of causality (Wang et al., 2022a), satisfying that $\sum_{i=1}^n w_i = 1$
    Update the parameters via gradient descent:
    $\boldsymbol{\theta}_i^{(k)} \leftarrow \boldsymbol{\theta}_i^{(k-1)} - \eta \nabla_{\boldsymbol{\theta}_i} \mathcal{L}^\dagger(\boldsymbol{\theta}_1^{(k-1)}, \ldots, \boldsymbol{\theta}_n^{(k-1)}), i = 1, \ldots, n$
  end for
  Let $u_i^s(\boldsymbol{x}) \leftarrow u_{\boldsymbol{\theta}_i^{(K)}}, i = 1, \ldots, n$
  if $s < m$ **then**
    Let $u_0^{s+1}(\boldsymbol{x}) \leftarrow u_n^s(\boldsymbol{x})$
  end if
  Let $\boldsymbol{\theta}_i^{(0)} \leftarrow \boldsymbol{\theta}_i^{(K)}$ (transfer learning), $i = 1, \ldots, n$
end for
Note:

(a) In our approach, we employ multiple neural networks, denoted as $u_{\boldsymbol{\theta}_i}, i = 1, \ldots, n$, to predict the solution at each time step. During implementation, these networks share all their weights except for the final linear layer. This design choice ensures efficient memory usage without compromising the distinctiveness of each network's predictions.

Nonlinear Problems. In the context of nonlinear problems, a strategy is to transfer the nonlinear components to the right-hand side and only precondition the linear portion. For example, we consider the following equation:

$$\Delta u(\boldsymbol{x}) + \sin u(\boldsymbol{x}) = f(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \Omega. \tag{89}$$

We can simply move the nonlinear term $\sin u(\boldsymbol{x})$ to the right-hand-side and assemble:

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b} - \sin \boldsymbol{u}. \tag{90}$$

Then, we can compute the preconditioner for the linear part $\boldsymbol{A}$ and the loss function becomes $\mathcal{L}^\dagger(\boldsymbol{\theta}) = \|\boldsymbol{P}^{-1}(\boldsymbol{A}\boldsymbol{u}_{\boldsymbol{\theta}} - \boldsymbol{b} + \sin \boldsymbol{u}_{\boldsymbol{\theta}})\|^2$. Nonetheless, this might lead to convergence issues in cases of highly nonlinearity.

To address this, we employ the Newton-Raphson method, allowing us to linearize the problem and then solve the associated linear tangent equation during each Newton iteration. Specifically, assembling a nonlinear problem results in a system of nonlinear equations:

$$\boldsymbol{F}(\boldsymbol{u}) = \boldsymbol{0}, \quad \boldsymbol{F}(\boldsymbol{u}) = (F_1(\boldsymbol{u}), \ldots, F_m(\boldsymbol{u})), \tag{91}$$

Algorithm 4 Preconditoning PINNs for non-linear problems

Input: number of iterations K, number of newton iteration T, mesh size N, learning rate η, initial guess $u_0(\boldsymbol{x})$, and initial parameters $\boldsymbol{\theta}^{(0)}$
Output: solution $u_T(\boldsymbol{x})$
Generate a mesh $\{\boldsymbol{x}^{(j)}\}_{j=1}^N$ for the problem domain Ω
Assemble the nonlinear system $\boldsymbol{F}$
for $i = 1, \ldots, T$ **do**
  Evaluate $u_{i-1}(\boldsymbol{x})$ on the mesh to obtain $\boldsymbol{u}_{i-1}$
  Compute the Jacobian matrix $J_{\boldsymbol{F}}(\boldsymbol{u}_{i-1})$
  Compute the preconditioner for $J_{\boldsymbol{F}}(\boldsymbol{u}_{i-1})$: $\boldsymbol{P} = \widehat{\boldsymbol{L}}\widehat{\boldsymbol{U}}$ via ILU
  for $k = 1, \ldots, K$ **do**
    Evaluate the neural network $u_{\boldsymbol{\theta}^{(k-1)}}$ on mesh points: $\boldsymbol{u}_{\boldsymbol{\theta}^{(k-1)}} = (u_{\boldsymbol{\theta}^{(k-1)}}(\boldsymbol{x}^{(j)}))_{j=1}^N$
    Compute the loss function $\mathcal{L}^\dagger(\boldsymbol{\theta}^{(k-1)})$ by:

$$\mathcal{L}^\dagger(\boldsymbol{\theta}) = \left\| \boldsymbol{P}^{-1}(J_{\boldsymbol{F}}(\boldsymbol{u}_{i-1})\boldsymbol{u}_{\boldsymbol{\theta}} - J_{\boldsymbol{F}}(\boldsymbol{u}_{i-1})\boldsymbol{u}_{i-1} + \boldsymbol{F}(\boldsymbol{u}_{i-1})) \right\|^2 \tag{88}$$

    Update the parameters via gradient descent: $\boldsymbol{\theta}^{(k)} \leftarrow \boldsymbol{\theta}^{(k-1)} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}^\dagger(\boldsymbol{\theta}^{(k-1)})$
  end for
  Let $u_i(\boldsymbol{x}) \leftarrow u_{\boldsymbol{\theta}^{(K)}}(\boldsymbol{x})$
  Let $\boldsymbol{\theta}^{(0)} \leftarrow \boldsymbol{\theta}^{(K)}$ (transfer learning)
end for
Note:
(a) Here, we only present the vanilla Newton method, while a lot of advanced techniques could be applied, which include line search, relaxation, specific stopping criteria, and so on.

where m is the number of nonlinear equations. The Newton-Raphson method solves the above equation with the following iterations ($i = 1, 2, 3 \ldots,$):

$$\boldsymbol{u}_i = \boldsymbol{u}_{i-1} - J_{\boldsymbol{F}}(\boldsymbol{u}_{i-1})^{-1}\boldsymbol{F}(\boldsymbol{u}_{i-1}), \tag{92}$$

where $J_{\boldsymbol{F}}(\boldsymbol{u}_{i-1})^{-1}$ the Jacobian matrix of $\boldsymbol{F}$ at $\boldsymbol{u}_{i-1}$. Now, we can use the neural network to solve the linear equation $J_{\boldsymbol{F}}(\boldsymbol{u}_{i-1})\boldsymbol{u}_i = J_{\boldsymbol{F}}(\boldsymbol{u}_{i-1})\boldsymbol{u}_{i-1} - \boldsymbol{F}(\boldsymbol{u}_{i-1})$ for $\boldsymbol{u}_i$ and proceed the iteration. We provide a detailed description in Algorithm 4.

C Supplements for Section 5.2

C.1 Environment and Global Settings

Environment. We employ PyTorch (Paszke et al., 2019) as our deep-learning backend and base our physics-informed learning experiment on DeepXDE (Lu et al., 2021a). All models are trained on an NVIDIA TITAN Xp 12GB GPU in the operating system of Ubuntu 18.04.5 LTS. When analytical solutions are not available, we utilize the Finite Difference Method (FDM) to produce ground truth solutions for the PDEs.

Global Settings. Unless otherwise stated, all the neural networks used are MLP of 5 hidden layers with 100 neurons in each layer. Besides, $\tanh$ is used for the activation function and Glorot normal (Glorot & Bengio, 2010) is used for trainable parameter initialization. The networks are all trained with an Adam optimizer (Kingma & Ba, 2014) (where the learning rate is 10^{-3} and $\beta_1 = \beta_2 = 0.99$) for 20000 iterations.

C.2 Details of Wave, Burgers', and Helmholtz Equations

The specific definitions of the PDEs are shown below.

Wave Equation. The governing PDE is:

$$u_{tt} - C^2 u_{xx} = \left(\frac{\pi}{8}\right)^2 (C^2 - 1) \sin\left(\frac{\pi}{8}x\right) \cos\left(\frac{\pi}{8}t\right), \tag{93}$$

with the boundary condition:

$$u(0,t) = u(8,t) = 0, \tag{94}$$

and initial condition:

$$\begin{aligned} u(x,0) &= \sin\left(\frac{\pi}{8}x\right) + \frac{1}{2}\sin\left(\frac{\pi}{2}x\right), \\ u_t(x,0) &= 0, \end{aligned} \tag{95}$$

defined on the domain $\Omega \times T = [0,8] \times [0,8]$, where $u = u(x,t)$ is the unknown.

The reference solution is:

$$u(x,t) = \sin\left(\frac{\pi}{8}x\right)\cos\left(\frac{\pi}{8}t\right) + \frac{1}{2}\sin\left(\frac{\pi}{2}x\right)\cos\left(\frac{C\pi}{2}t\right). \tag{96}$$

In the experiment, we uniformly sample the value of parameter C with a step of 0.1 within the range $[1.1, 5]$.

Helmholtz Equation. The governing PDE is:

$$\Delta u + u = (1 - 2\pi^2 A^2)\sin(A\pi x_1)\sin(A\pi x_2), \tag{97}$$

with the boundary condition:

$$u(x_1, 0) = u(x_1, 1) = u(0, x_2) = u(1, x_2) = 0, \tag{98}$$

defined on $\Omega = [0,1]^2$, where $u = u(\boldsymbol{x}) = u(x_1, x_2)$ is the unknown.

The reference solution is:

$$u(x,y) = \sin(A\pi x_1)\sin(A\pi x_2). \tag{99}$$

In the experiment, we vary A as integers between 1 and 20.

Burgers' Equation. The governing PDE on domain $\Omega \times T = [-1,1] \times [0,1]$ is:

$$u_t + uu_x - \nu u_{xx} = \sin(\pi x), \tag{100}$$

with the boundary condition:

$$u(-1,t) = u(1,t) = 0, \tag{101}$$

and initial condition:

$$u(0,x) = -\sin(\pi x), \tag{102}$$

where $u = u(x,t)$ is the unknown.

In the experiment, we uniformly sample 21 values of ν on a logarithmic scale (base 10) ranging from 10^{-2} to 1. The reference solution is generated by the FDW with a mesh of 501×21, where the nonlinear algebra equation is solved by 10-step Newton iterations.

C.3 Experimental Details

Implementation Details. Firstly, we introduce how we numerically estimate the condition number:

1. **FDM Approach:** We assemble the matrix $\boldsymbol{A}$ with a specified uniform mesh. For linear PDEs, according to Eq. (13), we have that $\mathrm{cond}(\mathcal{P}) \approx \frac{\|\boldsymbol{b}\|}{\|\boldsymbol{u}\|}\|\boldsymbol{A}^{-1}\|$. Therefore, we could approximate the condition number by calculating the norm of $\boldsymbol{A}^{-1}$. For nonlinear PDEs, in light of Proposition A.5, we have $\mathrm{cond}(\mathcal{P}) = \frac{\|f\|}{\|u\|}\|D\mathcal{F}^{-1}[f]\|$ by assuming its Fréchet differentiablity. Then, we could approximate the condition number by the norm of the inverse of the Jacobian matrix of the discretized nonlinear equations.

2. **Neural Network Approach:** According to the definition of the condition number, we can directly train a neural network to maximize:

$$\frac{\|\delta u\|/\|u\|}{\|\delta f\|/\|f\|}. \tag{103}$$

where $\|\delta f\|$ are confined to a small value. For linear PDEs, we can simplify the problem to be computing this equation: $\|\mathcal{F}^{-1}\| = \sup_{\|f\|=1} \frac{\|\mathcal{F}^{-1}[f]\|}{\|f\|} = \sup_{\|f\|=1} \frac{\|u_{\boldsymbol{\theta}}\|}{\|f\|}$. Since the operator is linear, we can further remove the constraint $\|f\| = 1$ and optimize $\frac{\|u_{\boldsymbol{\theta}}\|}{\|f\|} = \frac{\|u_{\boldsymbol{\theta}}\|}{\|\mathcal{F}(u_{\boldsymbol{\theta}})\|}$ over the parameter space to find the maximum, which will be minimizing its reciprocal or its opposite.

Hyper-parameters. Secondly, we introduce the hyper-parameters of computing solution or the condition number for each problem:

- **1D Poisson Equation:** We employ a mesh of the size 100 for FDM. The hard-constraint ansatz for the PINN is: $x(2\pi/P - x)/(\pi/P)^2 u_{\boldsymbol{\theta}}$. We use 2048 collocation points and 128 boundary points to train the PINN for 5000 epochs to compute the condition number.
- **Wave Equation:** We employ a mesh of the size 50×50 for FDM. The hard-constraint ansatz for the PINN is: $u_0 + x(8-x)/16 \cdot (t(12-t))^2/256 \cdot u_{\boldsymbol{\theta}}$, where t is time and u_0 is the initial condition. We use 8192 collocation points and 2048 boundary points to train the PINN with the learning rate of 10^{-4}.
- **Helmholtz Equation:** We employ a mesh of the size 50×50 for FDM. The hard-constraint ansatz for the PINN is: $\alpha u_{\boldsymbol{\theta}} + (1-\alpha)\sin(A\pi x)\sin(A\pi y)$, where $\alpha = 16x(1-x)y(1-y)$. We use 8192 collocation points and 2048 boundary points to train the PINN.
- **Burgers' Equation:** We employ a mesh of the size 500×20 for FDM. The hard-constraint ansatz for the PINN is: $\alpha(1-\beta)u_{\boldsymbol{\theta}} - \beta\sin(\pi x)$, where $\alpha = (1+x)(1-x), \beta = \exp(-t)$. We use 8192 collocation points and 2048 boundary points to train the PINN.

Nomralization of the Condition Number. For Burgers equation and Wave equation, we set:

$$\text{normalized cond}(\mathcal{P}) = \text{MinMax}(\log(\text{cond}(\mathcal{P}) + c)) \tag{104}$$

where $c = 0$ for Wave equation. For the Helmholtz equation, we select

$$\text{normalized cond}(\mathcal{P}) = \text{MinMax}(\sqrt{\text{cond}(\mathcal{P})}) \tag{105}$$

as the normalizer. Here, $\text{MinMax}(\cdot)$ denotes a min-max normalization for the given sequence to ensure the final values living in $[0, 1]$.

C.4 Physical Interpretation for Correlation Between PINN Error and Condition Number

Figure 2b unveils a robust linear association between the normalized condition number and the log-scaled L2 relative error (L2RE). This correlation can be expressed as:

$$\log(\text{L2RE}) \lesssim \text{normalized cond}(\mathcal{P}),$$

where, for simplicity, we omit the bias term (similarly in subsequent derivations).

To demystify this pronounced correlation, we first investigate the spectral behaviors of PINNs in approximating functions. When a neural network mimics the solutions of PDEs, it might exhibit a spectral bias. This implies that networks are more adept at capturing low-frequency components than their high-frequency counterparts (Rahaman et al., 2019). Recent studies have empirically demonstrated an exponential preference of neural networks towards frequency Xu et al. (2019). This leads to the inference that the error could be exponentially influenced by the system's frequency. Hence, it is plausible to represent this relationship as:

$$\log(\text{L2RE}) \lesssim \text{Frequency}.$$

In what follows, we explore how Frequency correlates with $\text{cond}(\mathcal{P})$. Using Frequency as a bridge, we will model the relationship between $\log(\text{L2RE})$ and $\text{cond}(\mathcal{P})$.

- **Helmholtz Equation:** Here, $\mathcal{F}^{-1}$ remains constant with the parameter A. This implies that $\mathrm{cond}(\mathcal{P}) \propto \frac{\|f\|}{\|u\|} = |1 - 2\pi^2 A^2|$. Given that A determines the solution's frequency, we infer that $\sqrt{\mathrm{cond}(\mathcal{P})} \underset{\sim}{\propto} \mathrm{Frequency}$. This leads to the conclusion that $\log(\mathrm{L2RE}) \underset{\sim}{\propto} \sqrt{\mathrm{cond}(\mathcal{P})}$, aligning with our experimental findings.
- **Wave & Burgers' Equation:** For these equations, the parameters C and ν influence the frequency of both the solution and the operator $\mathcal{F}$. Given their similar roles, we use the wave equation to elucidate the relationship between the condition number and the parameter. This relationship is found to be *at least exponential*. Based on Proposition A.5, we define $\mathcal{P}_1$ as:

$$u_{tt} - C^2 u_{xx} = 0, \tag{106}$$

maintaining the initial and boundary conditions. Assuming $\mathcal{P}_1$ is well-posed, we introduce $\mathcal{G}[w] = \mathcal{F}^{-1}[w] - u_1$ for every w in S, where u_1 is the solution to $\mathcal{P}_1$. Chossing a particular $f_0(x,t) = C^4(-e^{C^2 t}(1 + Kx) + e^{Cx}(1 + C^2 t))$ with $K = \frac{e^{8C}-1}{8}$, we derive $\mathcal{G}[f_0](x,t) = (e^{C^2 t} - 1 - C^2 t)(e^{Cx} - 1 - Kx)$. Consequently, we obtain:

$$\mathrm{cond}(\mathcal{P}) = \frac{\|f\|}{\|u\|}\|\mathcal{G}\| \geq \frac{\|f\|}{\|u\|}\frac{\|\mathcal{G}[f_0]\|}{\|f_0\|} \underset{\sim}{\propto} \frac{e^{kC}}{C^n}, \tag{107}$$

where k, n are constants independent of C. In summary, we deduce $\log(\mathrm{cond}(\mathcal{P})) \underset{\sim}{\propto} \mathrm{Frequency}$, leading to $\log(\mathrm{L2RE}) \underset{\sim}{\propto} \log(\mathrm{cond}(\mathcal{P}))$.

D Supplements for Section 5.3

D.1 Environment and Global Settings

Environment. The environment settings are basically consistent with that in Appendix C.1, except that:

- The model in NS2d-CG is trained on an Tesla V100-PCIE 16GB GPU. If you want to in a GPU with lower memory, you can specify `Use Sparse Solver = True` in the configuration to save memory.
- The reference data are generated by the work of Hao et al. (2023).
- We employ the finite element method (FEM) for discretization, utilizing FEniCS (Alnæs et al., 2015) as the platform.

Global Settings. Unless otherwise stated, we adopt the following settings:

- For 2D problems (including the time dimension), we employ the MLP of 3 hidden layers with 64 neurons in each layer. For 3D problems (including the time dimension), we employ the MLP of 5 hidden layers with 128 neurons in each layer. Besides, `SiLU` is used for the activation function. The initialization method is the default one in PyTorch. And we employ 10-dimensional Fourier features, as detailed in (Tancik et al., 2020), uniformly sampled on a logarithmic scale (base 2) spanning $2\pi \times [2^{-5}, 2^5]$.
- The networks are all trained with an Adam optimizer (Kingma & Ba, 2014) (where the learning rate is 10^{-3} and $\beta_1 = 0.9, \beta_2 = 0.99$) for 20000 iterations.
- The results of baselines are from the paper (Hao et al., 2023), except the computation time results, which are re-evaluated in the same environment as our method.

Baselines Introduction. We redirect readers to the Section 3.3.1 in the paper (Hao et al., 2023).

D.2 PDE Problems' Introduction and Implementation Details

In this section, we briefly describe PDE problems considered in PINNacle Hao et al. (2023) used in our experiment, as well as the implementation and hyper-parameters for our method. We refer to the original paper (Hao et al., 2023) for the problem details such as initial conditions and boundary conditions.

Burgers1d-C. The equation is given by:

$$\frac{\partial u}{\partial t} + uu_x = \nu u_{xx}, \tag{108}$$

define on $\Omega \times T = [-1, 1] \times [0, 1]$, where $u = u(x, t)$ is the unknown, Ω is the spatial domain whereas T is the temporal domain (the same below). In this and subsequent PDE problems, initial conditions and boundary conditions are omitted for clarity unless specified otherwise. Let $\Omega' = \Omega \times T, x' = (x, t)$. The weak form is expressed as:

$$\int_{\Omega'} \frac{\partial u}{\partial t} \cdot v \, \mathrm{d}x' + \int_{\Omega'} (uu_x) \cdot v \, \mathrm{d}x' + \nu \int_{\Omega'} u_x \cdot v_x \, \mathrm{d}x' = 0, \tag{109}$$

where v is the test function. We employ the FEniCS to discretize the problem with a mesh of size 500×20. Given that the matrix size remains within the memory constraints, we utilize a dense matrix implementation for faster matrix computations. The drop tolerance of the ILU is 10^{-4}. We solve the problem with 10-step Newton iterations (see Algorithm 4) and train the neural model for 2000 iterations in each Newton step.

Burgers2d-C. The equation is given by:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} = 0, \tag{110}$$

defined on $\Omega \times T = [0, 4]^2 \times [0, 1]$, where $\boldsymbol{u} = (u_1(\boldsymbol{x}, t), u_2(\boldsymbol{x}, t))$ is the unknown. We solve this problem by an (implicit) time-stepping scheme (see Algorithm 3). The number of sub-time intervals is 50, with each interval having 10 steps. The weak form is expressed as:

$$\int_\Omega \boldsymbol{u}_1 \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} + \delta t \nu \int_\Omega \nabla \boldsymbol{u}_1 \cdot \nabla \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} + \delta t \int_\Omega (\boldsymbol{u}_1 \cdot \nabla \boldsymbol{u}_1) \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} = \int_\Omega \boldsymbol{u}_0 \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x}, \tag{111}$$

where $\boldsymbol{u}_0 = \boldsymbol{u}_0(\boldsymbol{x})$ is the solution at the previous time step, $\boldsymbol{u}_1 = \boldsymbol{u}_1(\boldsymbol{x})$ is the solution at current time step, $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x})$ is the test function, and $\delta t = 1/500$ is the time step length. We employ the FEniCS to discretize the problem with an external mesh including 12657 nodes generated by COMSOL Multiphysics (commercial software for FEM (COMSOL AB, 2022)). It is noted that we do not employ a Newton method to solve the discretized nonlinear equations since the time overhead is too high. Instead, we only precondition the linear portion (see Appendix B.3) and let the neural model find the correct solution by gradient descent. Besides, we utilize a sparse matrix implementation since the matrix size exceeds the memory constraint. The drop tolerance of the ILU is 10^{-1}. We train the model for 2000 iterations in each sub-time interval while 40000 iterations in the first interval (i.e., cold-start training). Finally, in this problem, we employ an MLP of 5 layers with 128 neurons in each layer as our neural model.

Poisson2d-C. The equation is given by:

$$-\Delta u = 0, \tag{112}$$

defined on a 2D irregular domain Ω, a rectangular domain $[-0.5, 0.5]^2$ with four circular voids of the same size, where $u = u(\boldsymbol{x})$ is the unknown. The weak form is expressed as:

$$\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\boldsymbol{x} = 0, \tag{113}$$

where v is the test function. We employ the FEniCS to discretize the problem with an external mesh including 10602 nodes generated by the Gmsh (Geuzaine & Remacle, 2009). Given that the matrix size remains within the memory constraints, we utilize a dense matrix implementation for faster matrix computations. The drop tolerance of the ILU is 10^{-3}.

Poisson2d-CG. The equation is given by:

$$-\Delta u + k^2 u = f, \tag{114}$$

defined on a 2D irregular domain Ω, a rectangular domain $[-1, 1]^2$ with four circular voids of different sizes, where $u = u(\boldsymbol{x})$ is the unknown, $k = 8$, and $f = f(\boldsymbol{x})$ is given. The weak form is expressed as:

$$\int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\boldsymbol{x} + k^2 \int_\Omega u \cdot v \, \mathrm{d}\boldsymbol{x} = \int_\Omega f \cdot v \, \mathrm{d}\boldsymbol{x}, \tag{115}$$

where v is the test function. We employ the FEniCS to discretize the problem with an external mesh including 9382 nodes generated by the Gmsh. Given that the matrix size remains within the memory constraints, we utilize a dense matrix implementation for faster matrix computations. The drop tolerance of the ILU is 10^{-3}.

Poisson3d-CG. The equation is given by:

$$-\mu_i \Delta u + k_i^2 u = f \quad \text{in } \Omega_i, \quad i = 1, 2, \tag{116}$$

defined on a 3D irregular domain Ω, a cubic domain $[0, 1]^3$ with four spherical voids of different sizes, where $u = u(\boldsymbol{x})$ is the unknown, $\Omega_1 = \Omega \cap \{\boldsymbol{x} = (x_1, x_2, x_3) \mid x_3 < 0.5\}$, $\Omega_2 = \Omega \cap \{\boldsymbol{x} = (x_1, x_2, x_3) \mid x_3 \geq 0.5\}$, $\mu_1 = \mu_2 = 1$, $k_1 = 8, k_2 = 10$, and $f = f(\boldsymbol{x})$ is given. The weak form is expressed as:

$$\mu_1 \int_{\Omega_1} \nabla u \cdot \nabla v \, \mathrm{d}\boldsymbol{x} + k_1^2 \int_{\Omega_1} u \cdot v \, \mathrm{d}\boldsymbol{x} + \mu_2 \int_{\Omega_2} \nabla u \cdot \nabla v \, \mathrm{d}\boldsymbol{x} + k_2^2 \int_{\Omega_2} u \cdot v \, \mathrm{d}\boldsymbol{x} = \int_{\Omega} f \cdot v \, \mathrm{d}\boldsymbol{x}, \tag{117}$$

where v is the test function. We employ the FEniCS to discretize the problem with an external mesh including 13680 nodes generated by the Gmsh. Given that the matrix size remains within the memory constraints, we utilize a dense matrix implementation for faster matrix computations. The drop tolerance of the ILU is 10^{-3}.

Poisson2d-MS. The equation is given by:

$$\begin{aligned} -\nabla(a \nabla u) &= f \quad \text{in } \Omega, \\ \frac{\partial u}{\partial n} + u &= 0 \quad \text{in } \partial\Omega, \end{aligned} \tag{118}$$

defined on $\Omega = [-10, 10]^2$, where $u = u(\boldsymbol{x})$ is the unknown and $a = a(\boldsymbol{x})$ denotes a predefined function. Notably, Ω is partitioned into a 5×5 grid of uniform cells. Within each cell, a takes a piecewise linear form, introducing discontinuities at the cell boundaries. We define the weak form to be:

$$\int_{\Omega} a(\nabla u \cdot \nabla v) \, \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} a(u \cdot v) \, \mathrm{d}\boldsymbol{x} = \int_{\Omega} f \cdot v \, \mathrm{d}\boldsymbol{x}, \tag{119}$$

where v is the test function. We employ the FEniCS to discretize the problem with a mesh of size 100×100. Given that the matrix size remains within the memory constraints, we utilize a dense matrix implementation for faster matrix computations. The drop tolerance of the ILU is 10^{-3}. Finally, in this problem, we employ a Fourier MLP of 5 layers with 128 neurons in each layer as our neural model, where the Fourier features have a dimension of 128 and are sampled in $\mathcal{N}(0, \pi)$.

Heat2d-VC. The equation is given by:

$$\frac{\partial u}{\partial t} - \nabla(a \nabla u) = f, \tag{120}$$

define on $\Omega \times T = [0, 1]^2 \times [0, 5]$, where $u = u(\boldsymbol{x}, t)$ is the unknown and $a = a(\boldsymbol{x})$ denotes a predefined function with multi-scale frequencies. Let $\Omega' = \Omega \times T$, $\boldsymbol{x}' = (\boldsymbol{x}, t)$. We define the weak form to be:

$$\int_{\Omega'} \frac{\partial u}{\partial t} \cdot v \, \mathrm{d}\boldsymbol{x}' + \int_{\Omega'} a(\nabla u \cdot \nabla v) \, \mathrm{d}\boldsymbol{x}' = \int_{\Omega'} f \cdot v \, \mathrm{d}\boldsymbol{x}', \tag{121}$$

where v is the test function. We employ the FEniCS to discretize the problem with a mesh of size $20 \times 100 \times 100$. Besides, we utilize a sparse matrix implementation since the matrix size exceeds the memory constraint. The drop tolerance of the ILU is 10^{-1}. Finally, in this problem, we employ a Fourier MLP of 5 layers with 128 neurons in each layer as our neural model, where the Fourier features have a dimension of 128 and are sampled in $\mathcal{N}(0, \pi)$.

Heat2d-MS. The equation is given by:

$$\frac{\partial u}{\partial t} - \nabla \cdot \left(\left(\frac{1}{(500\pi)^2}, \frac{1}{(\pi)^2} \right) \odot \nabla u \right) = 0, \tag{122}$$

define on $\Omega \times T = [0,1]^2 \times [0,5]$, where $u = u(\boldsymbol{x}, t)$ is the unknown and $\odot$ denotes an element-wise multiplication. Let $\Omega' = \Omega \times T, \boldsymbol{x}' = (\boldsymbol{x}, t)$. We define the weak form to be:

$$\int_{\Omega'} \frac{\partial u}{\partial t} \cdot v \, \mathrm{d}\boldsymbol{x}' + \int_{\Omega'} \left(\left(\frac{1}{(500\pi)^2}, \frac{1}{(\pi)^2} \right) \odot \nabla u \right) \cdot \nabla v \, \mathrm{d}\boldsymbol{x}' = 0, \tag{123}$$

where v is the test function. We employ the FEniCS to discretize the problem with a mesh of size $500 \times 20 \times 20$. Besides, we utilize a sparse matrix implementation since the matrix size exceeds the memory constraint. The drop tolerance of the ILU is 10^{-1}. Finally, in this problem, we employ an MLP of 5 layers with 128 neurons in each layer as our neural model. The model is trained for 50000 iterations.

Heat2d-CG. The equation is given by:

$$\begin{aligned} \frac{\partial u}{\partial t} - \Delta u &= 0 && \text{in } \Omega \times T, \\ \frac{\partial u}{\partial n} &= 5 - u && \text{in } \partial\Omega_{\text{large}} \times T, \\ \frac{\partial u}{\partial n} &= 1 - u && \text{in } \partial\Omega_{\text{small}} \times T, \\ \frac{\partial u}{\partial n} &= 0.1 - u && \text{in } \partial\Omega_{\text{outer}} \times T, \end{aligned} \tag{124}$$

define on $\Omega \times T$, where $T = [0,3]$, Ω is a rectangular domain $[-8,8] \times [-12,12]$ with eleven large circular voids and six small circular voids, and $u = u(\boldsymbol{x}, t)$ is the unknown. Here, $\partial\Omega_{\text{large}}$ denotes the inner large circular boundary, $\partial\Omega_{\text{small}}$ the inner small circular boundary, $\partial\Omega_{\text{outer}}$ the outer rectangular boundary, and $\partial\Omega_{\text{large}} \cup \partial\Omega_{\text{small}} \cup \partial\Omega_{\text{outer}} = \partial\Omega$. We let:

$$\begin{aligned} \Omega' &= \Omega \times T, \\ \partial\Omega'_{\text{large}} &= \partial\Omega_{\text{large}} \times T, \\ \partial\Omega'_{\text{small}} &= \partial\Omega_{\text{small}} \times T, \\ \partial\Omega'_{\text{outer}} &= \partial\Omega_{\text{outer}} \times T, \end{aligned} \tag{125}$$

and $\boldsymbol{x}' = (\boldsymbol{x}, t)$. We define the weak form to be:

$$\begin{aligned} \int_{\Omega'} \frac{\partial u}{\partial t} \cdot v \, \mathrm{d}\boldsymbol{x}' + \int_{\Omega'} \nabla u \cdot \nabla v \, \mathrm{d}\boldsymbol{x}' - \int_{\partial\Omega'_{\text{large}}} (5 - u) \cdot v \, \mathrm{d}\boldsymbol{x}' \\ - \int_{\partial\Omega'_{\text{small}}} (1 - u) \cdot v \, \mathrm{d}\boldsymbol{x}' - \int_{\partial\Omega'_{\text{outer}}} (0.1 - u) \cdot v \, \mathrm{d}\boldsymbol{x}' = 0, \end{aligned} \tag{126}$$

where v is the test function. We employ the FEniCS to discretize the problem with an external mesh including 255946 nodes generated by the Gmsh. Besides, we utilize a sparse matrix implementation since the matrix size exceeds the memory constraint. The drop tolerance of the ILU is 10^{-1}.

Heat2d-LT. The equation is given by:

$$\frac{\partial u}{\partial t} = 0.001 \Delta u + 5 \sin(u^2) f, \tag{127}$$

define on $\Omega \times T = [0,1]^2 \times [0,100]$, where $u = u(\boldsymbol{x}, t)$ is the unknown and $f = f(\boldsymbol{x}, t)$ is given. We solve this problem by an (implicit) time-stepping scheme (see Algorithm 3). The number of sub-time intervals is 2000, with each interval having 1 step. We define the weak form to be:

$$\int_{\Omega} u_1 \cdot v \, \mathrm{d}\boldsymbol{x} + 0.001\delta t \int_{\Omega} \nabla u_1 \cdot \nabla v \, \mathrm{d}\boldsymbol{x} - \delta t \int_{\Omega} \left(5 \sin(u_1^2) f \right) \cdot v \, \mathrm{d}\boldsymbol{x}' = \int_{\Omega} u_0 \cdot v \, \mathrm{d}\boldsymbol{x}, \tag{128}$$

where $u_0 = u_0(\boldsymbol{x})$ is the solution at the previous time step, $u_1 = u_1(\boldsymbol{x})$ is the solution at current time step, $v = v(\boldsymbol{x})$ is the test function, and $\delta t = 1/2000$ is the time step length. We employ the FEniCS to discretize the problem with a mesh of size 20×20. It is noted that we do not employ a Newton method to solve the discretized nonlinear equations since the time overhead is too high. Instead, we

only precondition the linear portion (see Appendix B.3) and let the neural model find the correct solution by gradient descent. Given that the matrix size remains within the memory constraints, we utilize a dense matrix implementation for faster matrix computations. The drop tolerance of the ILU is 10^{-4}. We train the model for 1000 iterations in each sub-time interval while 100000 iterations in the first interval (i.e., cold-start training). Finally, in this problem, we employ an MLP of 5 layers with 128 neurons in each layer as our neural model.

NS2d-C. The equation is given by:

$$\begin{aligned} \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \frac{1}{Re} \Delta \boldsymbol{u} &= 0, \\ \nabla \cdot \boldsymbol{u} &= 0, \end{aligned} \tag{129}$$

defined on $\Omega = [0, 1]^2$, where $\boldsymbol{u} = (u_1(\boldsymbol{x}), u_2(\boldsymbol{x}))$ and p are the unknown velocity and pressure, respectively, and Re is the Reynolds number. The weak form is expressed as:

$$\frac{1}{Re} \int_\Omega \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} + \int_\Omega (\boldsymbol{u} \cdot \nabla \boldsymbol{u}) \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} - \int_\Omega p \nabla \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} - \int_\Omega q \nabla \boldsymbol{u} \, \mathrm{d}\boldsymbol{x} = 0, \tag{130}$$

where $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x})$ and $q = q(\boldsymbol{x})$ are, respectively, the test functions corresponding to $\boldsymbol{u}$ and p. We employ the FEniCS to discretize the problem with a mesh of size 50×50. Given that the matrix size remains within the memory constraints, we utilize a dense matrix implementation for faster matrix computations. The drop tolerance of the ILU is 10^{-4}. We solve the problem with 20-step Newton iterations (see Algorithm 4) and train the neural model for 1000 iterations in each Newton step.

NS2d-CG. The equation is given by:

$$\begin{aligned} \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \frac{1}{Re} \Delta \boldsymbol{u} &= 0, \\ \nabla \cdot \boldsymbol{u} &= 0, \end{aligned} \tag{131}$$

defined on $\Omega = [0, 4] \times [0, 2] \setminus ([0, 2] \times [1, 2])$, where $\boldsymbol{u} = (u_1(\boldsymbol{x}), u_2(\boldsymbol{x}))$ and p are the unknown velocity and pressure, respectively, and Re is the Reynolds number. The weak form is expressed as:

$$\frac{1}{Re} \int_\Omega \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} + \int_\Omega (\boldsymbol{u} \cdot \nabla \boldsymbol{u}) \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} - \int_\Omega p \nabla \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} - \int_\Omega q \nabla \boldsymbol{u} \, \mathrm{d}\boldsymbol{x} = 0, \tag{132}$$

where $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x})$ and $q = q(\boldsymbol{x})$ are, respectively, the test functions corresponding to $\boldsymbol{u}$ and p. We employ the FEniCS to discretize the problem with an external mesh including 2907 nodes generated by the Gmsh. Given that the matrix size remains within the memory constraints, we utilize a dense matrix implementation for faster matrix computations. The drop tolerance of the ILU is 10^{-4}. We solve the problem with 20-step Newton iterations (see Algorithm 4) and train the neural model for 1000 iterations in each Newton step.

NS2d-LT. The equation is given by:

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \frac{1}{Re} \Delta \boldsymbol{u} &= f, \\ \nabla \cdot \boldsymbol{u} &= 0, \end{aligned} \tag{133}$$

defined on $\Omega \times T = ([0, 2] \times [0, 1]) \times [0, 5]$, where $\boldsymbol{u} = (u_1(\boldsymbol{x}), u_2(\boldsymbol{x}))$ and p are the unknown velocity and pressure, respectively, Re is the Reynolds number, and $f = f(\boldsymbol{x}, t)$ is predefined. We solve this problem by an (implicit) time-stepping scheme (see Algorithm 3). The number of sub-time intervals is 50, with each interval having 1 step. The weak form is expressed as:

$$\begin{aligned} \int_\Omega \boldsymbol{u}_1 \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} + \delta t \frac{1}{Re} \int_\Omega \nabla \boldsymbol{u}_1 \cdot \nabla \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} + \delta t \int_\Omega (\boldsymbol{u}_1 \cdot \nabla \boldsymbol{u}_1) \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} \\ - \delta t \int_\Omega p_1 \nabla \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} - \delta t \int_\Omega q \nabla \boldsymbol{u}_1 \, \mathrm{d}\boldsymbol{x} = \int_\Omega \boldsymbol{u}_0 \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x}, \end{aligned} \tag{134}$$

where $\boldsymbol{u}_0 = \boldsymbol{u}_0(\boldsymbol{x})$ is the velocity at the previous time step, $\boldsymbol{u}_1 = \boldsymbol{u}_1(\boldsymbol{x})$ and $p_1 = p_1(\boldsymbol{x})$ are the velocity and pressure at current time step, $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x}), q = q(\boldsymbol{x})$ are the test functions corresponding

to velocity and pressure, and $\delta t = 1/50$ is the time step length. We employ the FEniCS to discretize the problem with a mesh of size 60×30. It is noted that we do not employ a Newton method to solve the discretized nonlinear equations since the time overhead is too high. Instead, we only precondition the linear portion (see Appendix B.3) and let the neural model find the correct solution by gradient descent. Given that the matrix size remains within the memory constraints, we utilize a dense matrix implementation for faster matrix computations. The drop tolerance of the ILU is 10^{-4}. We train the model for 1000 iterations in each sub-time interval while 100000 iterations in the first interval (i.e., cold-start training).

Wave1d-C. The equation is given by:

$$\frac{\partial^2 u}{\partial t^2} - 4\frac{\partial^2 u}{\partial x^2} = 0, \tag{135}$$

defined on $\Omega \times T = [0, 1] \times [0, 1]$, where $u = u(x, t)$ is the unknown. Let $\Omega' = \Omega \times T, x' = (x, t)$. The weak form is expressed as:

$$-\int_{\Omega'} \frac{\partial u}{\partial t} \cdot \frac{\partial v}{\partial t} \,\mathrm{d}x' + 4\int_{\Omega'} \frac{\partial u}{\partial x} \cdot \frac{\partial v}{\partial x} \,\mathrm{d}x' = 0, \tag{136}$$

where v is the test function. We employ the FEniCS to discretize the problem with a mesh of size 100×100. Given that the matrix size remains within the memory constraints, we utilize a dense matrix implementation for faster matrix computations. The drop tolerance of the ILU is 10^{-3}.

Wave2d-CG. The equation is given by:

$$\frac{1}{c}\frac{\partial^2 u}{\partial t^2} - \Delta u = 0, \tag{137}$$

define on $\Omega \times T = [-1, 1]^2 \times [0, 5]$, where $u = u(\boldsymbol{x}, t)$ is the unknown and $c = c(\boldsymbol{x})$ is a parameter function with high frequencies, generated by the Gaussian random field. We solve this problem by an (implicit) time-stepping scheme (see Algorithm 3). The number of sub-time intervals is 50, with each interval having 5 steps. We define the weak form to be:

$$\int_\Omega u_1 \cdot v \,\mathrm{d}\boldsymbol{x} + \delta t^2 \int_\Omega c\,(\nabla u_1 \cdot \nabla v) \,\mathrm{d}\boldsymbol{x} = \int_\Omega (2u_0 - u_{-1}) \cdot v \,\mathrm{d}\boldsymbol{x}, \tag{138}$$

where $u_{-1} = u_{-1}(\boldsymbol{x})$ is the solution at the time step before the previous time step, $u_0 = u_0(\boldsymbol{x})$ is the solution at the previous time step, $u_1 = u_1(\boldsymbol{x})$ is the solution at current time step, $v = v(\boldsymbol{x})$ is the test function, and $\delta t = 1/250$ is the time step length. We employ the FEniCS to discretize the problem with a mesh of size 40×40. Given that the matrix size remains within the memory constraints, we utilize a dense matrix implementation for faster matrix computations. The drop tolerance of the ILU is 10^{-4}. We train the model for 1000 iterations in each sub-time interval while 500000 iterations in the first interval (i.e., cold-start training).

Wave2d-MS. The equation is given by:

$$\frac{\partial^2 u}{\partial t^2} + \nabla \cdot \left(\left(1, a^2\right) \odot \nabla u\right) = 0, \tag{139}$$

defined on $\Omega \times T = [0, 1]^2 \times [0, 100]$, where $u = u(\boldsymbol{x}, t)$ is the unknown and a is a given parameter. Let $\Omega' = \Omega \times T, \boldsymbol{x}' = (\boldsymbol{x}, t)$. The weak form is expressed as:

$$\int_{\Omega'} \frac{\partial u}{\partial t} \cdot \frac{\partial v}{\partial t} \,\mathrm{d}\boldsymbol{x}' + \int_{\Omega'} \left(\left(1, a^2\right) \odot \nabla u\right) \cdot \nabla v \,\mathrm{d}\boldsymbol{x}' = 0, \tag{140}$$

where v is the test function. We employ the FEniCS to discretize the problem with a mesh of size $10 \times 10 \times 1000$. Besides, we utilize a sparse matrix implementation since the matrix size exceeds the memory constraint. The drop tolerance of the ILU is 10^{-1}. Finally, in this problem, we employ a Fourier MLP of 5 layers with 128 neurons in each layer as our neural model, where the Fourier features have a dimension of 128 and are sampled in $\mathcal{N}(0, \pi)$.

GS. The equation is given by:

$$\begin{aligned}\frac{\partial u_1}{\partial t} &= \varepsilon_1 \Delta u_1 + b(1-u_1) - u_1 u_2^2, \\ \frac{\partial u_2}{\partial t} &= \varepsilon_2 \Delta u_2 - d u_2 + u_1 u_2^2,\end{aligned} \tag{141}$$

defined on $\Omega \times T = [-1,1]^2 \times [0,200]$, where $\boldsymbol{u} = (u_1(\boldsymbol{x},t), u_2(\boldsymbol{x},t))$ is the unknown and $b, d, \epsilon_1, \epsilon_2$ are given. We solve this problem by an (implicit) time-stepping scheme (see Algorithm 3). The number of sub-time intervals is 200, with each interval having 1 step. The weak form is expressed as:

$$\begin{aligned}\int_\Omega \boldsymbol{u}_1 \cdot \boldsymbol{v} \,\mathrm{d}\boldsymbol{x} + \delta t \int_\Omega (\epsilon_1 \nabla u_{1,1} \cdot \nabla v_1 + \epsilon_2 \nabla u_{1,2} \cdot \nabla v_2) \,\mathrm{d}\boldsymbol{x} \\ + \delta t \int_\Omega \left((u_{1,1} u_{1,2}^2) \cdot v_1 - (u_{1,1} u_{1,2}^2) \cdot v_2 \right) \mathrm{d}\boldsymbol{x} \\ + \delta t \int_\Omega (-b(1-u_{1,1}) \cdot v_1 + d u_{1,2} \cdot v_2)) \,\mathrm{d}\boldsymbol{x} = \int_\Omega \boldsymbol{u}_0 \cdot \boldsymbol{v} \,\mathrm{d}\boldsymbol{x},\end{aligned} \tag{142}$$

where $\boldsymbol{u}_0 = \boldsymbol{u}_0(\boldsymbol{x})$ is the solution at the previous time step, $\boldsymbol{u}_1 = \boldsymbol{u}_1(\boldsymbol{x}) = (u_{1,1}(\boldsymbol{x}), u_{1,2}(\boldsymbol{x}))$ is the solution at current time step, $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x})$ is the test function, and $\delta t = 1/200$ is the time step length. We employ the FEniCS to discretize the problem with a mesh of size 128×128. It is noted that we do not employ a Newton method to solve the discretized nonlinear equations since the time overhead is too high. Instead, we only precondition the linear portion (see Appendix B.3) and let the neural model find the correct solution by gradient descent. Besides, we utilize a sparse matrix implementation since the matrix size exceeds the memory constraint. The drop tolerance of the ILU is 10^{-1}. We train the model for 1000 iterations in each sub-time interval while 20000 iterations in the first interval (i.e., cold-start training). Finally, in this problem, we employ an MLP of 5 layers with 128 neurons in each layer as our neural model.

KS. The equation is given by:

$$\frac{\partial u}{\partial t} + \alpha u \frac{\partial u}{\partial x} + \beta \frac{\partial^2 u}{\partial x^2} + \gamma \frac{\partial^4 u}{\partial x^4} = 0, \tag{143}$$

define on $\Omega \times T = [0, 2\pi] \times [0,1]$, where $u = u(x,t)$ is the unknown and α, β, γ are multi-scale co-efficients. We solve this problem by an (implicit) time-stepping scheme (see Algorithm 3). The number of sub-time intervals is 1, with each interval having 250 steps. We define the weak form to be:

$$\int_\Omega u_1 v \,\mathrm{d}x + \alpha \delta t \int_\Omega u_1 \frac{\partial u_1}{\partial x} v \,\mathrm{d}x - \beta \delta t \int_\Omega \frac{\partial u_1}{\partial x} \frac{\partial v}{\partial x} \,\mathrm{d}x - \gamma \delta t \int_\Omega \frac{\partial^3 u_1}{\partial x^3} \frac{\partial v}{\partial x} \,\mathrm{d}x = \int_\Omega u_0 v \,\mathrm{d}x, \tag{144}$$

where $u_0 = u_0(\boldsymbol{x})$ is the solution at the previous time step, $u_1 = u_1(\boldsymbol{x})$ is the solution at current time step, $v = v(\boldsymbol{x})$ is the test function, and $\delta t = 1/250$ is the time step length. We employ the FEniCS to discretize the problem with a mesh of size 500. It is noted that we do not employ a Newton method to solve the discretized nonlinear equations since the time overhead is too high. Instead, we only precondition the linear portion (see Appendix B.3) and let the neural model find the correct solution by gradient descent. Given that the matrix size remains within the memory constraints, we utilize a dense matrix implementation for faster matrix computations. The drop tolerance of the ILU is 10^{-4}. We train the model for 15000 iterations in each sub-time interval. Finally, in this problem, we employ an MLP of 5 layers with 128 neurons in each layer as our neural model.

Poisson Inverse Problem (PInv). The equation is given by:

$$-\nabla(a \nabla u) = f, \tag{145}$$

define on $\Omega = [0,1]^2$, where $u = u(\boldsymbol{x})$ is the unknown solution, $a = a(\boldsymbol{x})$ denotes the unknown parameter function, and $f = f(\boldsymbol{x})$ is predefined. Given 2500 uniformly distributed samples $\{u(\boldsymbol{x}^{(i)})\}$ with Gaussian noise of $\mathcal{N}(0, 0.1)$, our target is to reconstruct the unknown solution u and infer the unknown parameter function a. We define the weak form to be:

$$\int_\Omega a(\nabla u \cdot \nabla v) \,\mathrm{d}\boldsymbol{x} = \int_\Omega f \cdot v \,\mathrm{d}\boldsymbol{x}, \tag{146}$$

Table 3: Comparison between our method, SOTA PINN baseline, and the adjoint method over 5 trials. The best results are in **bold**.

Problem	L2RE (mean ± std)			Average Running Time (s)		
	Ours	SOTA	Adjoint	Ours	SOTA	Adjoint
PInv	**1.80e-2 ± 9.30e-3**	2.45e-2 ± 1.03e-2	7.82e+2 ± 0.00e+0	1.87e+2	4.90e+2	1.40e+0
HInv	**9.04e-3 ± 2.34e-3**	5.09e-2 ± 4.34e-3	1.50e+3 ± 0.00e+0	3.21e+2	3.39e+3	1.07e+1

where v is the test function. We employ the FEniCS to discretize the problem with a mesh of size 100×100. Besides, we utilize a sparse matrix implementation. For fast speed, we employ the Jacobi preconditioner since the preconditioner needs updating every iteration. Finally, in this problem, we employ an MLP of 3 layers with 64 neurons in each layer for u and an MLP of 5 layers with 128 neurons in each layer for a. The models are trained for 11000 iterations, where 10000 iterations are warm-up iterations. In warm-up iterations, only data loss is involved while physics loss is included in the rest of iterations.

Heat Inverse Problem (HInv). The equation is given by:

$$\frac{\partial u}{\partial t} - \nabla(a \nabla u) = f, \tag{147}$$

define on $\Omega \times T = [-1, 1]^2 \times [0, 1]$, where $u = u(\boldsymbol{x}, t)$ is the unknown solution, $a = a(\boldsymbol{x})$ denotes the unknown parameter function, and $f = f(\boldsymbol{x}, t)$ is predefined. Given 2500 uniformly distributed samples $\{u(\boldsymbol{x}^{(i)}, t^{(i)})\}$ with Gaussian noise of $\mathcal{N}(0, 0.1)$, our target is to reconstruct the unknown solution u and infer the unknown parameter function a. Let $\Omega' = \Omega \times T, \boldsymbol{x}' = (\boldsymbol{x}, t)$. We define the weak form to be:

$$\int_{\Omega'} \frac{\partial u}{\partial t} \cdot v \, \mathrm{d}\boldsymbol{x}' + \int_{\Omega'} a(\nabla u \cdot \nabla v) \, \mathrm{d}\boldsymbol{x}' = \int_{\Omega'} f \cdot v \, \mathrm{d}\boldsymbol{x}', \tag{148}$$

where v is the test function. We employ the FEniCS to discretize the problem with a mesh of size $40 \times 40 \times 10$. Besides, we utilize a sparse matrix implementation. For fast speed, we employ the Jacobi preconditioner since the preconditioner needs updating every iteration. Finally, in this problem, we employ an MLP of 3 layers with 64 neurons in each layer for u and an MLP of 3 layers with 64 neurons in each layer for a. The models are trained for 5000 iterations, where 4000 iterations are warm-up iterations. In warm-up iterations, only data loss is involved while physics loss is included in the rest of iterations.

D.3 Benchmark of Inverse Problems

Here, we consider two inverse problems, the Poisson Inverse Problem (PInv) and Heat Inverse Problem (HInv), from the benchmark Hao et al. (2022). In such problems, our target is to reconstruct the unknown solution from 2500 noisy samples and infer the unknown parameter function. We compare our method with the SOTA PINN baseline in Hao et al. (2022) and the traditional adjoint method designed for PDE-constrained optimization. We report the results in Table 3.

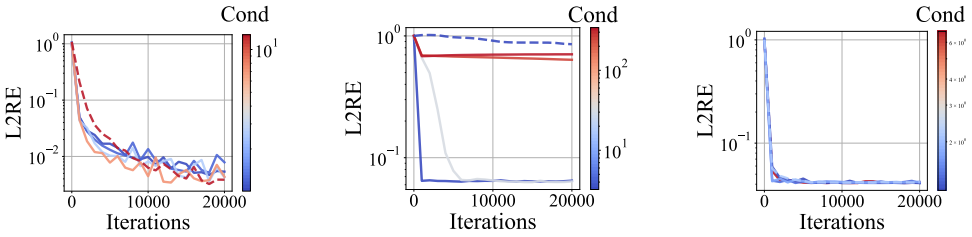
From the results, we can conclude that our method achieves state-of-the-art performance in both accuracy and running time. Although the adjoint method converges very fast, it fails to approach the correct solution. This is because the numerical method does not impose any continuous prior on the ansatz and can overfit the noise in the solution samples.

D.4 Experimental Results of Ablation Study

We provide the comprehensive results of the four Poisson problems in this subsection. Table 4 presents the convergence results of L2RE as well as some metrics to measure the precision of the preconditioner for different cases. For example, "$\boldsymbol{P}^{-1}f$ Error" measures the L2RE between the $\boldsymbol{P}^{-1}f$ and the $\boldsymbol{A}^{-1}f$. Besides, Figure 4 shows the convergence history of different cases. We can find that although preconditioning (ILU) cannot ensure that the condition number decreases, it can often promote convergence.

Table 4: Comprehensive results of the ablation study.

Poisson		Drop Tolerance				No Preconditioner
		1.00e-4	**1.00e-3**	**1.00e-2**	**1.00e-1**	
2d-C	L2RE	1.70e-3	2.74e-3	4.07e-3	2.18e-3	3.54e-2
	Cond	1.10e+0	2.82e+0	1.52e+1	6.03e+1	1.13e+2
	$\boldsymbol{P}^{-1}f$ Error	2.04e-2	2.08e-1	5.51e-1	7.67e-1	–
2d-CG	L2RE	5.38e-3	7.87e-3	4.27e-3	4.36e-3	3.86e-3
	Cond	1.01e+0	1.19e+0	2.55e+0	7.22e+0	1.27e+1
	$\boldsymbol{P}^{-1}f$ Error	2.84e-3	4.05e-2	3.50e-1	7.00e-1	–
3d-CG	L2RE	4.18e-2	4.11e-2	4.11e-2	4.23e-2	4.19e-2
	Cond	6.77e+0	1.17e+0	1.38e+0	1.77e+0	2.20e+0
	$\boldsymbol{P}^{-1}f$ Error	4.63e-1	2.05e-1	5.84e-1	8.73e-1	–
2d-MS	L2RE	6.48e-2	6.38e-2	6.37e-1	7.06e-1	8.55e-1
	Cond	3.23e+0	3.25e+1	2.47e+2	3.42e+2	3.39e+0
	$\boldsymbol{P}^{-1}f$ Error	3.74e-1	6.42e-1	8.13e-1	9.58e-1	–



(a) Poisson2d-CG (b) Poisson2d-MS (c) Poisson3d-CG

Figure 4: The training L2 relative error (L2RE) in ablation study. The dashed line marks the trajectory corresponding to the one without the preconditioner.

E Supplementary Experimental Results

In the next few pages, we display the detailed experiment results in different metrics, including L2RE, L1RE, MSE, and the standard deviation of these metrics over 5 runs.

Table 5: Mean (std) of L2RE for main experiments.

L2RE	Name	Ours	Vanilla		Loss Reweighting/Sampling			Optimizer	Loss functions		Architecture		
–			PINN	PINN-w	LRA	NTK	RAR	MultiAdam	gPINN	vPINN	LAAF	GAAF	FBPINN
Burgers	1d-C	**1.42E-2(1.62E-4)**	1.45E-2(1.59E-3)	2.63E-2(4.68E-3)	2.61E-2(1.18E-2)	1.84E-2(3.66E-3)	3.32E-2(2.14E-2)	4.85E-2(1.61E-2)	2.16E-1(3.34E-2)	3.47E-1(3.49E-2)	**1.43E-2(1.44E-3)**	5.20E-2(2.08E-2)	2.32E-1(9.14E-2)
	2d-C	5.23E-1(7.52E-2)	3.24E-1(7.54E-4)	2.70E-1(3.93E-3)	**2.60E-1(5.78E-3)**	2.75E-1(4.78E-3)	3.45E-1(4.56E-5)	3.33E-1(8.65E-3)	3.27E-1(1.25E-4)	6.38E-1(1.47E-2)	2.77E-1(1.39E-2)	2.95E-1(1.17E-2)	–
Poisson	2d-C	**3.98E-3(3.70E-3)**	6.94E-1(8.78E-3)	3.49E-2(6.91E-3)	1.17E-1(1.26E-1)	1.23E-2(7.37E-3)	6.99E-1(7.46E-3)	2.63E-2(6.57E-3)	6.87E-1(1.87E-2)	4.91E-1(1.55E-2)	7.68E-1(4.70E-2)	6.04E-1(7.52E-2)	4.49E-2(7.91E-3)
	2d-CG	**5.07E-3(1.93E-3)**	6.36E-1(2.57E-3)	6.08E-2(4.88E-3)	4.34E-2(7.95E-3)	1.43E-2(4.31E-3)	6.48E-1(7.87E-3)	2.76E-1(1.03E-1)	7.92E-1(4.56E-3)	2.86E-1(2.00E-3)	4.80E-1(1.43E-2)	8.71E-1(2.67E-1)	2.90E-2(3.92E-3)
	3d-CG	**4.16E-2(7.53E-4)**	5.60E-1(2.84E-2)	3.74E-1(3.23E-2)	1.02E-1(3.16E-2)	9.47E-1(4.94E-4)	5.76E-1(5.40E-2)	3.63E-1(7.81E-2)	4.85E-1(5.70E-2)	7.38E-1(6.47E-4)	5.79E-1(2.65E-2)	5.02E-1(7.47E-2)	7.39E-1(7.24E-2)
	2d-MS	**6.40E-2(1.12E-3)**	6.30E-1(1.07E-2)	7.60E-1(6.96E-3)	7.94E-1(6.51E-2)	7.48E-1(9.94E-3)	6.44E-1(2.13E-2)	5.90E-1(4.06E-2)	6.16E-1(1.74E-2)	9.72E-1(2.23E-2)	5.93E-1(1.18E-1)	9.31E-1(7.12E-2)	1.04E+0(6.13E-5)
Heat	2d-VC	**3.11E-2(6.17E-3)**	1.01E+0(6.34E-2)	2.35E-1(1.70E-2)	2.12E-1(8.61E-4)	2.14E-1(5.82E-3)	9.66E-1(1.86E-2)	4.75E-1(8.44E-2)	2.12E+0(5.51E-1)	9.40E-1(1.73E-1)	6.42E-1(6.32E-2)	8.49E-1(1.06E-1)	9.52E-1(2.29E-3)
	2d-MS	**2.84E-2(1.30E-2)**	6.21E-2(1.38E-2)	2.42E-1(2.67E-2)	8.79E-2(2.56E-2)	4.40E-2(4.81E-3)	7.49E-2(1.05E-2)	2.18E-1(9.26E-2)	1.13E-1(3.08E-3)	9.30E-1(2.06E-2)	7.40E-2(1.92E-2)	9.85E-1(1.04E-1)	8.20E-2(4.87E-3)
	2d-CG	**1.50E-2(1.17E-4)**	3.64E-2(8.82E-3)	1.45E-1(4.77E-3)	1.25E-1(4.30E-3)	1.16E-1(1.21E-2)	2.72E-2(3.22E-3)	7.12E-2(1.30E-2)	9.38E-2(1.45E-2)	1.67E+0(3.62E-3)	2.39E-2(1.39E-3)	4.61E-1(2.63E-1)	9.16E-2(3.29E-2)
	2d-LT	**2.11E-1(1.00E-2)**	9.99E-1(1.05E-5)	9.99E-1(8.01E-5)	9.99E-1(7.37E-5)	1.00E+0(2.82E-4)	9.99E-1(1.56E-4)	1.00E+0(3.85E-5)	1.00E+0(9.82E-5)	1.00E+0(0.00E+0)	9.99E-1(4.49E-4)	9.99E-1(2.20E-4)	1.01E+0(1.23E-4)
NS	2d-C	**1.28E-2(2.44E-3)**	4.70E-2(1.12E-3)	1.45E-1(1.21E-2)	NA	1.98E-1(2.60E-2)	4.69E-1(1.16E-2)	7.27E-1(1.95E-1)	7.70E-2(2.99E-3)	2.92E-1(8.24E-2)	3.60E-2(3.87E-3)	3.79E-2(4.32E-3)	8.45E-2(2.26E-2)
	2d-CG	**6.62E-2(1.26E-3)**	1.19E-1(5.46E-3)	3.26E-1(7.69E-3)	3.32E-1(7.60E-3)	2.93E-1(2.02E-2)	3.34E-1(6.52E-4)	4.31E-1(6.95E-2)	1.54E-1(5.89E-3)	9.94E-1(3.80E-3)	8.24E-2(8.21E-3)	1.74E-1(7.00E-2)	8.27E+0(3.68E-5)
	2d-LT	**9.09E-1(4.00E-4)**	9.96E-1(1.19E-3)	1.00E+0(3.34E-4)	1.00E+0(4.05E-4)	9.99E-1(6.04E-4)	1.00E+0(3.35E-4)	1.00E+0(2.19E-4)	9.95E-1(7.19E-4)	1.73E+0(1.00E-5)	9.98E-1(3.42E-3)	9.99E-1(1.10E-3)	1.00E+0(2.07E-3)
Wave	1d-C	**1.28E-2(1.20E-4)**	5.88E-1(9.63E-2)	2.85E-1(8.97E-3)	3.61E-1(1.95E-2)	9.79E-2(7.72E-3)	5.39E-1(1.77E-2)	1.21E-1(1.76E-2)	5.56E-1(1.67E-2)	8.39E-1(5.94E-2)	4.54E-1(1.08E-2)	6.77E-1(1.05E-1)	5.91E-1(4.74E-2)
	2d-CG	**5.85E-1(9.05E-3)**	1.84E+0(3.40E-1)	1.66E+0(7.39E-2)	1.48E+0(1.03E-1)	2.16E+0(1.01E-1)	1.15E+0(1.06E-1)	1.09E+0(1.24E-1)	8.14E-1(1.18E-2)	7.99E-1(4.31E-2)	8.19E-1(2.67E-2)	7.94E-1(9.33E-3)	1.06E+0(7.54E-2)
	2d-MS	**5.71E-2(5.68E-3)**	1.34E+0(2.34E-1)	1.02E+0(1.16E-2)	1.02E+0(1.36E-2)	1.04E+0(3.11E-2)	1.35E+0(2.43E-1)	1.01E+0(5.64E-3)	1.02E+0(4.00E-3)	9.82E-1(1.23E-3)	1.06E+0(1.71E-2)	1.06E+0(5.35E-2)	1.03E+0(6.68E-3)
Chaotic	GS	**1.44E-2(2.53E-3)**	3.19E-1(3.18E-1)	1.58E-1(9.10E-2)	9.37E-2(4.42E-5)	2.16E-1(7.73E-2)	9.46E-2(9.46E-4)	9.37E-2(1.21E-5)	2.48E-1(1.10E-1)	1.16E+0(1.43E-1)	9.47E-2(7.07E-5)	9.46E-2(1.15E-4)	7.99E-2(1.69E-2)
	KS	**9.52E-1(2.94E-3)**	1.01E+0(1.28E-3)	9.86E-1(2.24E-2)	9.57E-1(2.85E-3)	9.64E-1(4.94E-3)	1.01E+0(8.63E-4)	9.61E-1(4.77E-3)	9.94E-1(3.83E-3)	9.72E-1(5.80E-4)	1.01E+0(2.12E-3)	1.00E+0(1.24E-2)	1.02E+0(2.31E-2)

Table 6: Mean (std) of L1RE for main experiments.

L1RE	Name	Ours	Vanilla		Loss Reweighting/Sampling			Optimizer	Loss functions		Architecture		
–			PINN	PINN-w	LRA	NTK	RAR	MultiAdam	gPINN	vPINN	LAAF	GAAF	FBPINN
Burgers	1d-C	**9.05E-3(1.45E-4)**	9.55E-3(6.42E-4)	1.88E-2(4.05E-3)	1.35E-2(2.57E-3)	1.30E-2(1.73E-3)	1.35E-2(4.66E-3)	2.64E-2(5.69E-3)	1.42E-1(1.98E-2)	4.02E-2(6.41E-3)	1.40E-2(3.68E-3)	1.95E-2(8.30E-3)	3.75E-2(9.70E-3)
	2d-C	4.14E-1(2.24E-2)	2.96E-1(7.40E-4)	2.43E-1(2.98E-3)	**2.31E-1(7.16E-3)**	2.48E-1(5.33E-3)	3.27E-1(3.73E-5)	3.12E-1(1.15E-2)	3.01E-1(3.55E-4)	6.56E-1(3.01E-2)	2.57E-1(2.06E-2)	2.67E-1(1.22E-2)	–
Poisson	2d-C	**4.43E-3(4.69E-3)**	7.40E-1(5.49E-3)	3.08E-2(5.13E-3)	7.82E-2(7.47E-2)	1.30E-2(8.23E-3)	7.48E-1(1.01E-2)	2.47E-2(6.38E-3)	7.35E-1(2.08E-2)	4.60E-1(1.39E-2)	7.67E-1(1.36E-2)	6.57E-1(3.99E-2)	5.01E-2(4.71E-3)
	2d-CG	**4.76E-3(1.92E-3)**	5.45E-1(4.71E-3)	4.54E-2(6.42E-3)	2.63E-2(5.50E-3)	1.33E-2(4.96E-3)	5.60E-1(8.19E-3)	2.46E-1(1.07E-1)	7.31E-1(2.77E-3)	2.45E-1(5.14E-3)	4.04E-1(1.03E-2)	7.09E-1(2.12E-1)	3.21E-2(6.23E-3)
	3d-CG	**3.82E-2(1.26E-3)**	4.51E-1(3.35E-2)	3.33E-1(2.64E-2)	7.76E-2(1.63E-2)	9.93E-1(2.91E-4)	4.61E-1(4.46E-2)	3.55E-1(7.75E-2)	4.57E-1(5.07E-2))	7.96E-1(3.57E-4)	4.60E-1(1.13E-2)	3.82E-1(4.89E-2)	6.91E-1(7.52E-2)
	2d-MS	**4.84E-2(1.52E-3)**	7.60E-1(1.06E-2)	7.49E-1(1.12E-2)	7.93E-1(7.62E-2)	7.26E-1(1.46E-2)	7.84E-1(2.42E-2)	6.94E-1(5.61E-2)	7.41E-1(2.01E-2)	9.61E-1(5.67E-2)	6.31E-1(5.42E-2)	9.04E-1(1.01E-1)	9.94E-1(9.67E-5)
Heat	2d-VC	**2.81E-2(6.46E-3)**	1.12E+0(5.79E-2)	2.41E-1(1.73E-2)	2.07E-1(1.04E-3)	2.03E-1(1.12E-2)	1.06E+0(5.13E-2)	5.45E-1(1.07E-1)	2.41E+0(5.27E-1)	8.79E-1(2.57E-1)	7.49E-1(8.54E-2)	9.91E-1(1.37E-1)	9.44E-1(1.75E-3)
	2d-MS	**3.22E-2(1.42E-2)**	9.30E-2(2.27E-2)	2.90E-1(2.43E-2)	1.13E-1(3.57E-2)	6.69E-2(8.24E-3)	1.19E-1(2.16E-2)	3.00E-1(1.14E-1)	1.80E-1(1.12E-2)	9.25E-1(3.90E-2)	1.14E-1(4.98E-2)	1.08E+0(2.02E-1)	5.33E-2(3.92E-3)
	2d-CG	**8.42E-3(2.71E-4)**	3.05E-2(8.47E-3)	1.37E-1(7.70E-3)	1.12E-1(2.57E-3)	1.07E-1(1.44E-2)	2.21E-2(3.42E-3)	5.88E-2(1.02E-2)	8.20E-2(1.32E-2)	3.09E+0(1.86E-2)	1.94E-2(1.98E-3)	3.77E-1(2.17E-1)	6.77E-1(3.93E-2)
	2d-LT	**1.36E-1(4.34E-3)**	9.98E-1(6.00E-5)	9.98E-1(1.42E-4)	9.98E-1(1.47E-4)	9.99E-1(1.01E-3)	9.98E-1(2.28E-4)	9.99E-1(5.69E-5)	9.98E-1(8.62E-4)	9.98E-1(0.00E+0)	9.98E-1(1.27E-4)	9.98E-1(8.58E-5)	1.01E+0(7.75E-4)
NS	2d-C	**6.90E-3(7.17E-4)**	5.08E-2(3.06E-3)	1.84E-1(1.52E-2)	NA	2.44E-1(3.05E-2)	5.54E-1(1.24E-2)	9.86E-1(3.16E-1)	9.43E-2(3.24E-3)	1.98E-1(7.81E-2)	4.42E-2(7.38E-3)	3.78E-2(8.71E-3)	1.18E-1(3.10E-2)
	2d-CG	**9.62E-2(1.06E-3)**	1.77E-1(1.00E-2)	4.22E-1(8.72E-3)	4.12E-1(6.93E-3)	3.69E-1(2.46E-2)	4.65E-1(4.44E-3)	6.23E-1(8.86E-2)	2.36E-1(1.15E-2)	9.95E-1(3.50E-4)	1.25E-1(1.42E-2)	2.40E-1(8.01E-2)	5.92E+0(5.65E-4)
	2d-LT	**8.51E-1(8.00E-4)**	9.88E-1(1.86E-3)	9.98E-1(4.68E-4)	9.97E-1(3.64E-4)	9.95E-1(6.66E-4)	1.00E+0(2.46E-4)	9.99E-1(9.27E-4)	9.90E-1(3.60E-4)	1.00E+0(1.40E-4)	9.90E-1(3.78E-3)	9.96E-1(2.68E-3)	1.00E+0(1.38E-3)
Wave	1d-C	**1.11E-2(2.87E-4)**	5.87E-1(9.20E-2)	2.78E-1(8.86E-3)	3.49E-1(2.02E-2)	9.42E-2(9.13E-3)	5.40E-1(1.74E-2)	1.15E-1(1.91E-2)	5.60E-1(1.69E-2)	1.41E+0(1.30E-1)	4.38E-1(1.40E-2)	6.82E-1(1.08E-1)	6.55E-1(4.86E-2)
	2d-CG	**4.95E-1(1.23E-2)**	1.96E+0(3.83E-1)	1.78E+0(8.89E-2)	1.58E+0(1.15E-1)	2.34E+0(1.14E-1)	1.16E+0(1.16E-1)	1.09E+0(1.54E-1)	7.22E-1(1.63E-2)	1.08E+0(1.25E-1)	7.45E-1(2.15E-2)	7.08E-1(9.13E-3)	1.15E+0(1.03E-1)
	2d-MS	**7.46E-2(8.35E-3)**	2.04E+0(7.38E-1)	1.10E+0(4.25E-2)	1.08E+0(6.01E-2)	1.13E+0(4.91E-2)	2.08E+0(7.45E-1)	1.07E+0(1.40E-2)	1.11E+0(1.91E-2)	1.05E+0(1.00E-2)	1.17E+0(4.66E-2)	1.12E+0(8.62E-2)	1.29E+0(2.81E-2)
Chaotic	GS	**4.18E-3(6.93E-4)**	3.45E-1(4.57E-1)	1.29E-1(1.54E-1)	2.01E-2(5.99E-5)	1.11E-1(4.79E-2)	2.98E-2(6.44E-3)	2.00E-2(6.12E-5)	2.72E-1(1.79E-1)	1.04E+0(3.04E-1)	2.07E-2(9.19E-4)	1.16E-1(1.31E-1)	5.06E-2(1.87E-2)
	KS	8.70E-1(8.52E-3)	9.44E-1(8.57E-4)	8.95E-1(2.99E-2)	**8.60E-1(3.48E-3)**	8.64E-1(3.31E-3)	9.42E-1(8.75E-4)	8.73E-1(8.40E-3)	9.36E-1(6.12E-3)	8.88E-1(9.92E-3)	9.39E-1(3.25E-3)	9.44E-1(9.86E-3)	9.85E-1(3.35E-2)

Table 7: Mean (std) of MSE for main experiments.

MSE	Name	Ours	Vanilla		Loss Reweighting/Sampling			Optimizer	Loss functions		Architecture		
–			PINN	PINN-w	LRA	NTK	RAR	MultiAdam	gPINN	vPINN	LAAF	GAAF	FBPINN
Burgers	1d-C	**7.52E-5(1.53E-6)**	7.90E-5(1.78E-5)	2.64E-4(8.69E-5)	3.03E-4(2.62E-4)	1.30E-4(5.19E-5)	5.78E-4(6.31E-4)	9.68E-4(5.51E-4)	1.77E-2(5.58E-3)	5.13E-3(1.90E-3)	1.80E-4(1.35E-4)	3.00E-4(1.56E-4)	1.53E-2(1.03E-2)
	2d-C	2.31E-1(7.11E-2)	1.69E-1(7.86E-4)	1.17E-1(3.41E-3)	**1.09E-1(4.84E-3)**	1.22E-1(4.22E-3)	1.92E-1(5.07E-5)	1.79E-1(9.36E-3)	1.72E-1(1.31E-4)	7.08E-1(5.16E-2)	1.26E-1(1.54E-2)	1.41E-1(1.12E-2)	–
Poisson	2d-C	**7.22E-6(1.03E-5)**	1.17E-1(2.98E-3)	3.09E-4(1.25E-4)	7.24E-3(9.95E-3)	5.00E-5(5.33E-5)	1.19E-1(2.55E-3)	1.79E-4(8.84E-5)	1.15E-1(6.22E-3)	4.86E-2(4.43E-3)	1.39E-1(5.67E-3)	9.38E-2(1.91E-2)	7.89E-4(2.17E-4)
	2d-CG	**9.29E-6(7.92E-6)**	1.28E-1(1.03E-3)	1.17E-3(1.83E-4)	6.13E-4(2.31E-4)	6.99E-5(3.50E-5)	1.32E-1(3.23E-3)	2.73E-2(1.92E-2)	1.98E-1(2.28E-3)	2.50E-2(3.80E-4)	7.67E-2(2.73E-3)	1.77E-1(8.70E-2)	4.84E-4(9.87E-5)
	3d-CG	**1.46E-4(5.29E-6)**	2.64E-2(2.67E-3)	1.18E-2(1.97E-3)	9.51E-4(6.51E-4)	7.54E-2(7.86E-5)	2.81E-2(5.15E-3)	1.16E-2(4.42E-3)	2.01E-2(4.93E-3)	4.58E-2(8.04E-5)	2.82E-2(2.62E-3)	2.16E-2(5.87E-3)	4.63E-2(9.28E-3)
	2d-MS	**2.75E-2(9.75E-4)**	2.67E+0(9.04E-2)	3.90E+0(7.16E-2)	4.28E+0(6.83E-1)	3.77E+0(9.98E-2)	2.80E+0(1.87E-1)	2.36E+0(3.15E-1)	2.56E+0(1.43E-1)	6.09E+0(5.46E-1)	1.83E+0(3.00E-1)	5.87E+0(8.72E-1)	6.68E+0(8.23E-4)
Heat	2d-VC	**3.95E-5(1.54E-5)**	4.00E-2(4.94E-3)	2.19E-3(3.21E-4)	1.76E-3(1.43E-5)	1.79E-3(9.80E-5)	3.67E-2(1.42E-3)	9.14E-3(3.13E-3)	1.89E-1(9.44E-2)	3.23E-2(2.26E-2)	1.74E-2(4.35E-3)	2.93E-2(7.12E-3)	3.56E-2(1.71E-4)
	2d-MS	**2.59E-5(1.80E-5)**	1.09E-4(4.94E-5)	1.60E-3(3.35E-4)	2.25E-4(1.22E-4)	5.27E-5(1.18E-5)	1.54E-4(4.17E-5)	1.51E-3(1.25E-3)	3.43E-4(1.87E-5)	2.57E-2(2.22E-3)	1.57E-4(8.06E-5)	3.10E-2(1.15E-2)	2.17E-4(2.47E-5)
	2d-CG	**3.34E-4(5.02E-6)**	2.09E-3(9.69E-4)	3.15E-2(2.08E-3)	2.32E-2(1.59E-3)	2.02E-2(4.15E-3)	1.12E-3(2.65E-4)	7.79E-3(2.63E-3)	1.34E-2(4.13E-3)	1.16E+1(9.04E-2)	8.53E-4(9.74E-5)	3.94E-1(2.71E-1)	5.61E-1(5.96E-2)
	2d-LT	**5.09E-2(4.88E-3)**	1.14E+0(2.38E-5)	1.13E+0(1.82E-4)	1.14E+0(1.67E-4)	1.14E+0(6.41E-4)	1.14E+0(3.55E-4)	1.14E+0(8.74E-5)	1.14E+0(2.23E-4)	1.14E+0(0.00E+0)	1.14E+0(2.20E-4)	1.14E+0(3.27E-4)	1.16E+0(2.83E-4)
NS	2d-C	**3.22E-6(1.23E-6)**	4.19E-5(2.00E-6)	4.03E-4(6.45E-5)	NA	7.56E-4(1.90E-4)	4.18E-3(2.05E-4)	1.07E-2(5.67E-3)	1.13E-4(8.77E-6)	5.30E-4(3.50E-4)	2.33E-5(4.71E-6)	2.67E-5(4.71E-6)	1.37E-4(7.24E-5)
	2d-CG	**2.15E-4(8.21E-6)**	6.94E-4(6.45E-5)	5.19E-3(2.43E-4)	5.40E-3(2.49E-4)	4.22E-3(5.82E-4)	5.45E-3(2.13E-5)	9.32E-3(3.09E-3)	1.16E-3(8.97E-5)	1.06E+0(1.61E-2)	3.37E-4(6.60E-5)	1.72E-3(1.33E-3)	3.34E+0(2.97E-5)
	2d-LT	**4.30E+2(4.00E-1)**	5.06E+2(1.21E+0)	5.10E+2(3.40E-1)	5.10E+2(4.13E-1)	5.09E+2(6.15E-1)	5.10E+2(3.42E-1)	5.10E+2(2.23E-1)	5.05E+2(7.30E-1)	5.11E+2(1.76E-2)	5.06E+2(1.82E+0)	5.11E+2(2.99E+0)	5.15E+2(1.77E+0)
Wave	1d-C	**5.08E-5(1.16E-6)**	1.11E-1(3.66E-2)	2.54E-2(1.61E-3)	4.08E-2(4.31E-3)	3.01E-3(4.82E-4)	9.07E-2(6.02E-3)	4.68E-3(1.28E-3)	9.66E-2(5.85E-3)	6.17E-1(1.19E-1)	6.03E-2(2.87E-3)	1.48E-1(4.44E-2)	1.39E-1(1.97E-2)
	2d-CG	**1.59E-2(5.16E-4)**	1.64E-1(6.13E-2)	1.28E-1(1.13E-2)	1.03E-1(1.46E-2)	2.17E-1(2.05E-2)	6.25E-2(1.17E-2)	5.59E-2(1.29E-2)	3.09E-2(8.98E-4)	5.24E-2(9.01E-3)	3.49E-2(3.38E-3)	2.99E-2(4.68E-4)	5.78E-2(7.99E-3)
	2d-MS	**2.20E+3(4.38E+2)**	1.30E+5(4.25E+4)	7.35E+4(1.68E+3)	7.34E+4(1.97E+3)	7.69E+4(4.55E+3)	1.33E+5(4.47E+4)	7.15E+4(8.04E+2)	7.27E+4(5.47E+2)	1.13E+2(1.46E+2)	7.91E+4(2.55E+3)	7.98E+4(8.00E+3)	8.95E+5(1.15E+4)
Chaotic	GS	**1.04E-4(3.69E-5)**	1.00E-1(1.35E-1)	1.64E-2(1.70E-2)	4.32E-3(4.07E-6)	2.59E-2(1.44E-2)	4.40E-3(8.83E-5)	4.32E-3(1.11E-6)	3.62E-2(2.28E-2)	4.00E-1(2.33E-1)	4.32E-3(4.71E-6)	1.69E-2(1.79E-2)	5.16E-3(1.64E-3)
	KS	**1.03E+0(4.00E-3)**	1.16E+0(2.95E-3)	1.11E+0(5.07E-2)	1.04E+0(6.20E-3)	1.06E+0(1.09E-2)	1.16E+0(1.98E-3)	1.05E+0(1.04E-2)	1.12E+0(8.67E-3)	1.05E+0(2.50E-3)	1.16E+0(4.50E-3)	1.14E+0(2.33E-2)	1.16E+0(5.28E-2)