A Brief Introduction to HINTS

Dinghang Tan

2025-03-04

Outline

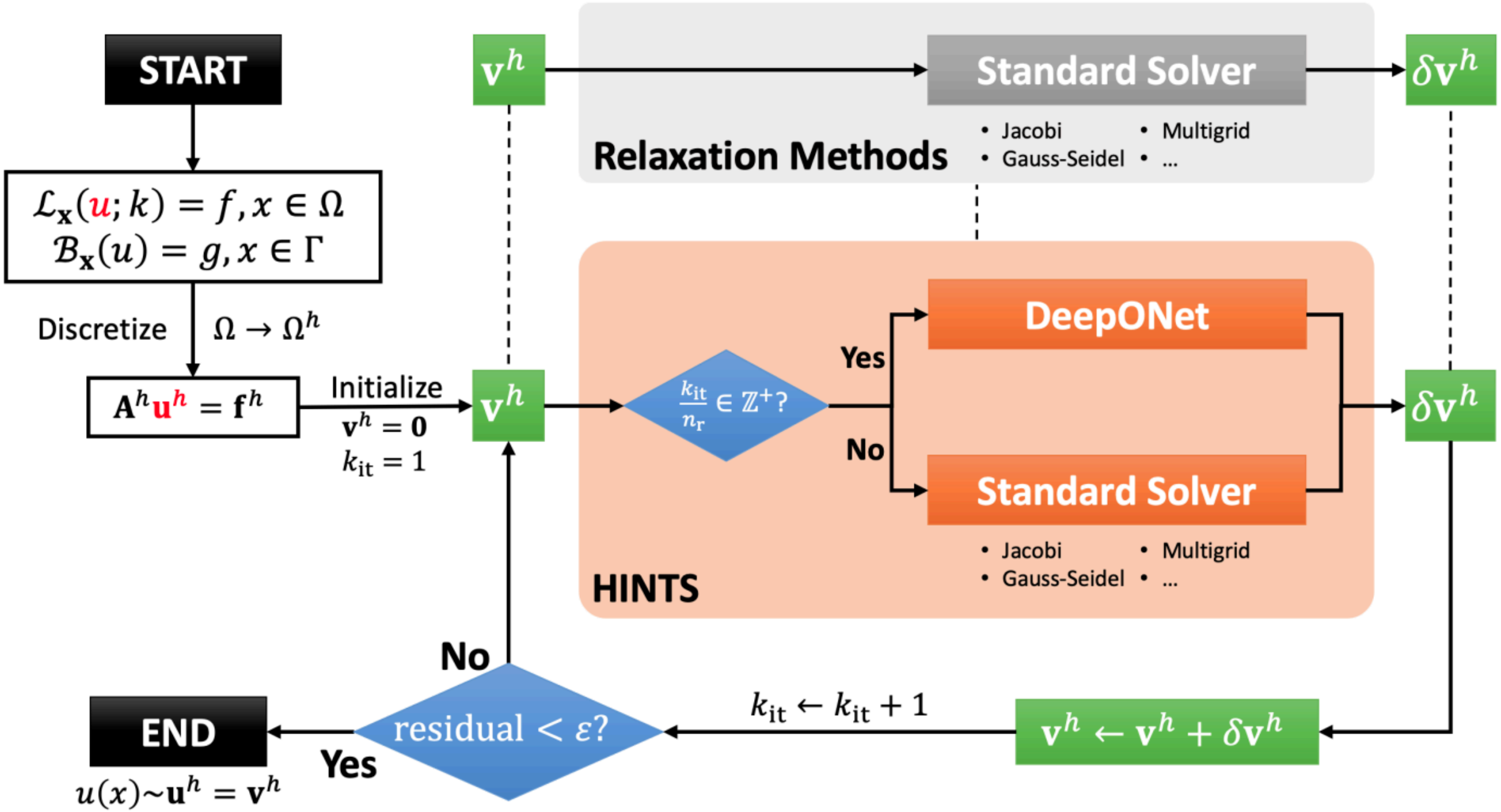
1. Overview

1.1 What is HINTS

HINTS[1] is a hybrid, iterative, numerical, and transferable solver for differential equations which combines standard relaxation methods and the Deep Operator Network (DeepONet).

[1]E. Zhang, A. Kahana, E. Turkel, R. Ranade, J. Pathak, and G. E. Karniadakis, “A hybrid iterative numerical transferable solver (hints) for pdes based on deep operator network and relaxation methods,” *arXiv preprint arXiv:2208.13273*, p. 198, 2022.

1.1 What is HINTS



2. Basic Principles

2.1 Smoothing Principles

1. A basic residual-correction method to solve $Au = f$ for a given initial guess u_0.
    1. residual $r = f - Au_k$.
    2. correction $e = Br$ with $B \approx A^{-1}$.
    3. $u_{k+1} = u_k + e$.
2. Amplification matrix.

$$e_{k+1} := u - u_{k+1} = (I - BA)(u - u_k) = (I - BA)e_k$$

Theorem 2.1.1 The residual-correction shceme converges iff

2.1 Smoothing Principles

$$\rho(I - BA) < 1.$$

3. Jacobi iteration $B = \operatorname{diag}(A)^{-1}$.
4. $-u'' = 0, x \in (0,1)$ with $u(0) = u(1) = 0$.

$$\int_0^1 \varphi_i'(x)\varphi_j'(x)\,\mathrm{d}x = \frac{1}{h}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} =: \frac{1}{h}A$$

2.1 Smoothing Principles

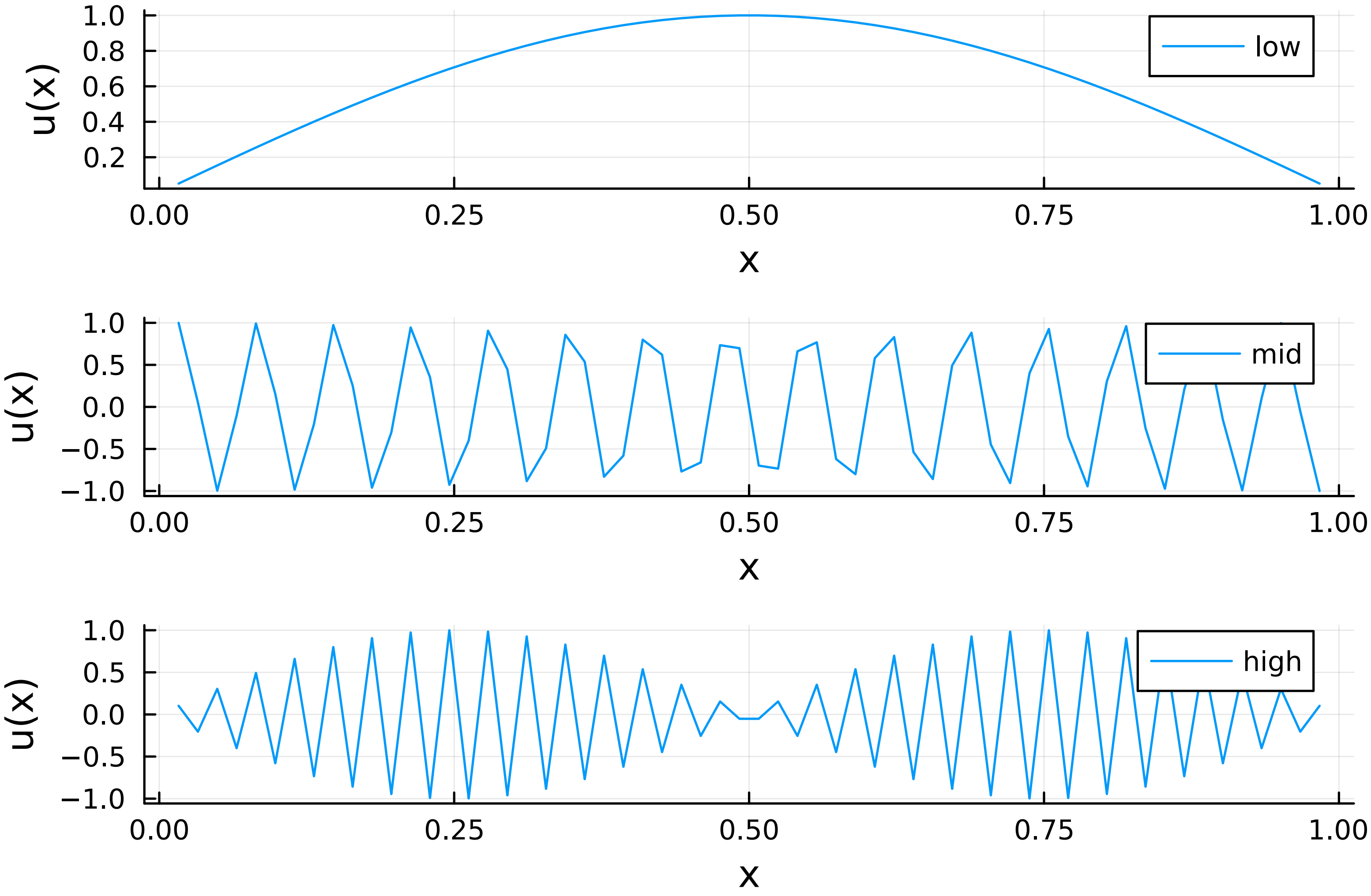
where $\varphi_i(x)(i = 1, 2, ..., n)$ are hat functions and $h = \frac{1}{n+1}$. The eigenpairs (λ_k, v_k) of A are

$$\lambda_k(A) = 2(1 - \cos(hk\pi)), v_{k,j}(A) = \sqrt{2h}\sin(jkh\pi).$$

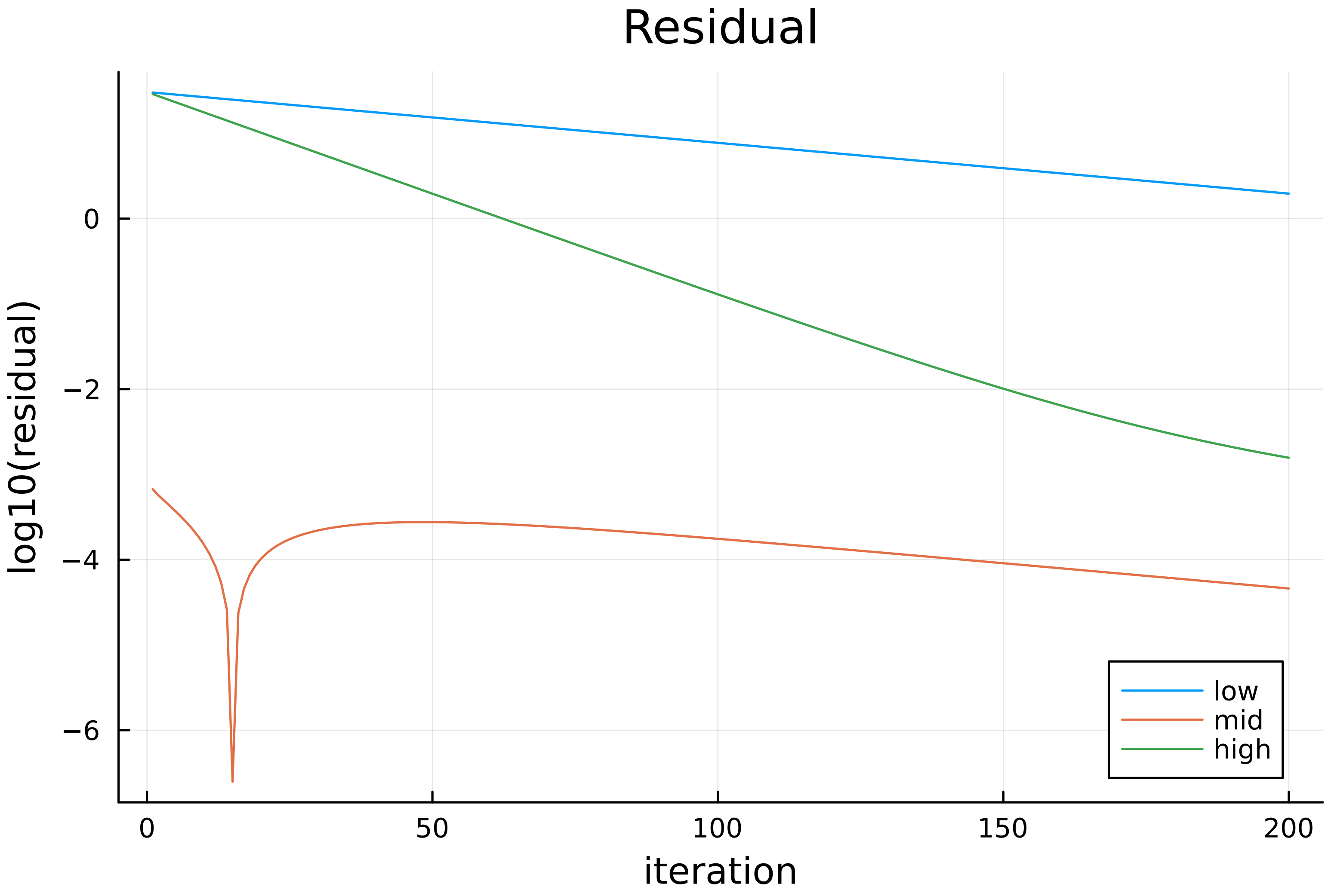
Note that $B = \frac{1}{2}I$,

$$\lambda_k(I - BA) = \cos(hk\pi) = \cos\left(\frac{k\pi}{n+1}\right).$$

2.1 Smoothing Principles



2.1 Smoothing Principles



2.2 Frequency Principle

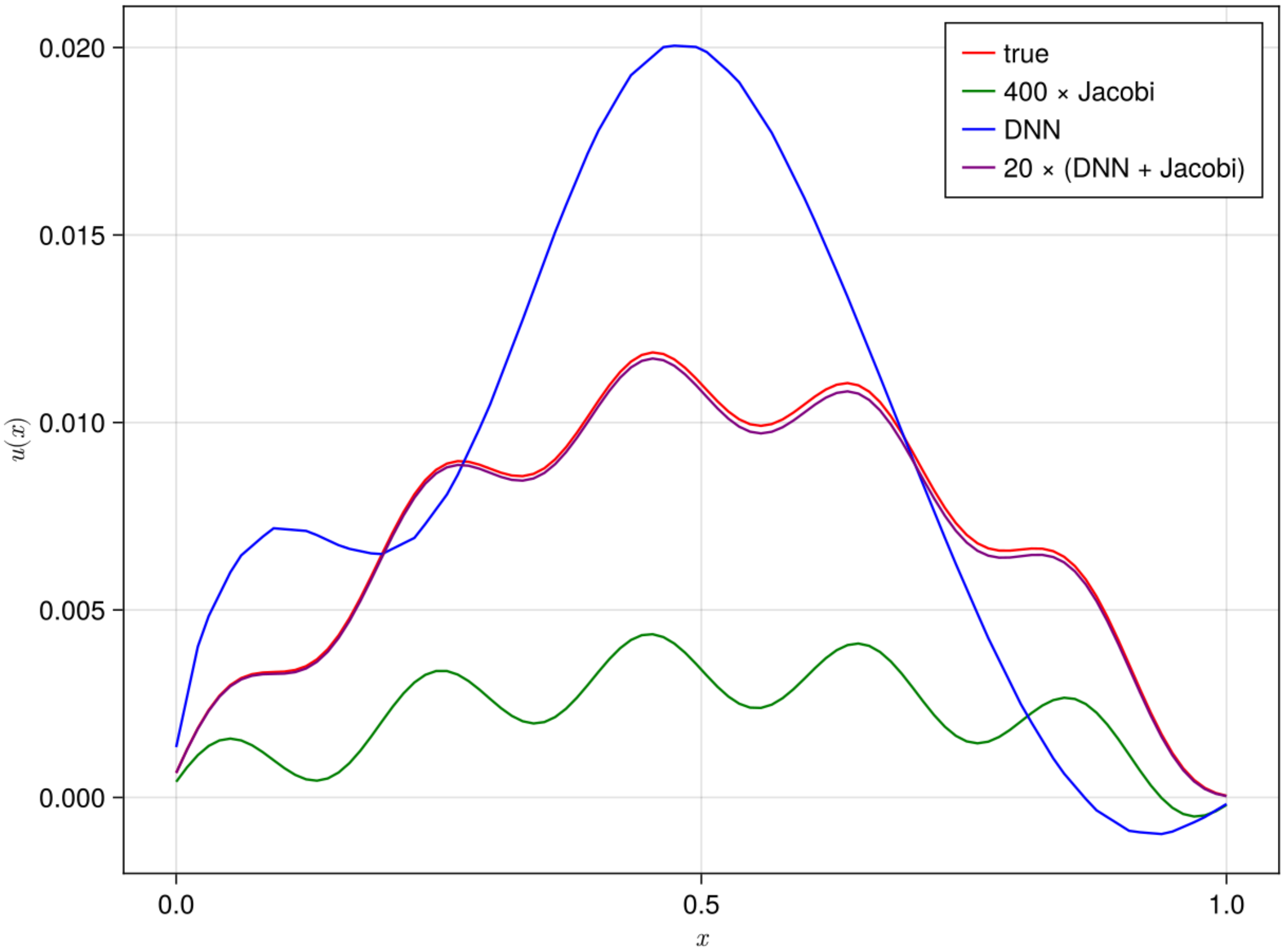
Frequency principle[1] indicates that deep neural networks always approximate the low-frequency part of the target function first, and then approach the high-frequency part. Here is an example.[2]

$$
\begin{aligned}
&-u'' = \frac{1}{10}\sin(\pi x) + \sin(1000\pi x), x \in (0,1), \\
&u(0) = u(1) = 0.
\end{aligned}
$$

[1]Z.-Q. J. X. Zhi-Qin John Xu, Y. Z. Yaoyu Zhang, T. L. Tao Luo, Y. X. Yanyang Xiao, and Z. M. Zheng Ma, "Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks," *Communications in Computational Physics*, vol. 28, no. 5, pp. 1746–1767, 2020, doi: 10.4208/cicp.OA-2020-0085.

[2]see https://github.com/dhtantoy/HintsDemo.jl.git for more details.

2.2 Frequency Principle



3. Algorithms

3.1 Residual Correction

function ITERATION(A, $\boldsymbol{b}$, epochs, ε)

    $\boldsymbol{v} \leftarrow \boldsymbol{0}$

    $i \leftarrow 0$

    $\boldsymbol{r} \leftarrow \boldsymbol{b} - A\boldsymbol{v}$

    while $\|\boldsymbol{r}\| \leq \varepsilon$ and $i \leq$ epochs **do**

        solve $A\boldsymbol{\delta v} = \boldsymbol{r}$

        $\boldsymbol{v} \leftarrow \boldsymbol{v} + \boldsymbol{\delta v}$

        $i \leftarrow i + 1$

    return $\boldsymbol{v}$

3.2 Residual Equations

function DEEPONET_JACOBI(A, $\boldsymbol{r}$, p)

    $\boldsymbol{\delta v} \leftarrow \mathrm{DeepONet}(\boldsymbol{r})$

    $\boldsymbol{\delta v^0} \leftarrow \boldsymbol{\delta v}$

    $s \leftarrow 0$

    while $s < p$ **do**

        solve $A\boldsymbol{\delta v} = \boldsymbol{r}$

        $\boldsymbol{\delta v}_i^{s+1} \leftarrow \frac{1}{a_{ii}}\left(\boldsymbol{r} - \sum_{j=1}^{n} a_{ij}\boldsymbol{\delta v}_j^s\right)$

        $s \leftarrow s + 1$

    return $\boldsymbol{\delta v}^p$

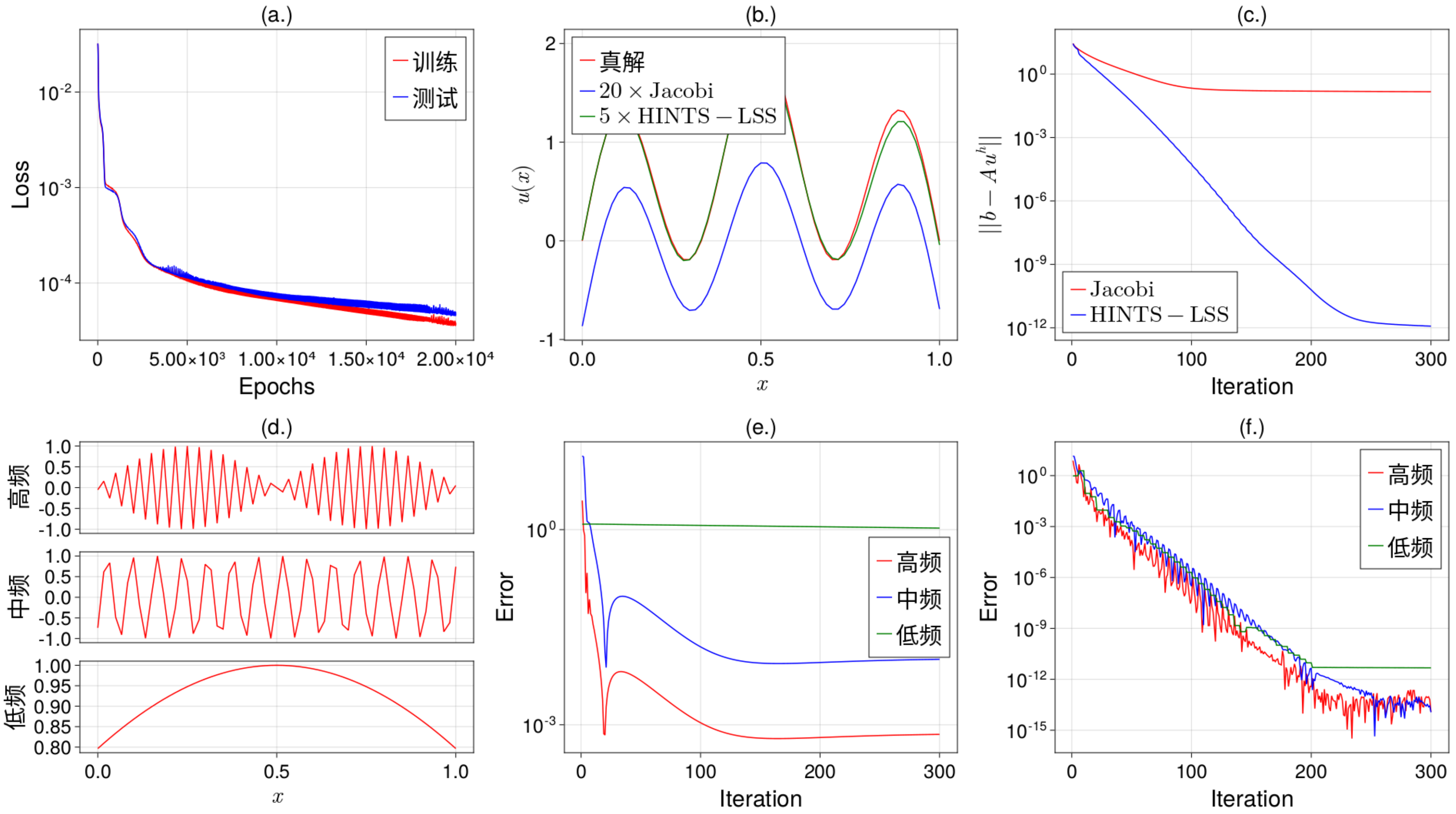
4. Numerical Experiments

4.1 1D Poisson Equation

$$-u'' = f, x \in (0,1),$$
$$u'(0) + u(0) = c_0,$$
$$u'(1) + u(1) = c_1.$$

Here $f = \pi^2 \sin(\pi x) + 25\pi^2 \sin(5\pi x)$ and $c_0 = c_1 = 0$. The exact solution is $u = \sin(\pi x) + \sin(5\pi x)$.

4.1 1D Poisson Equation



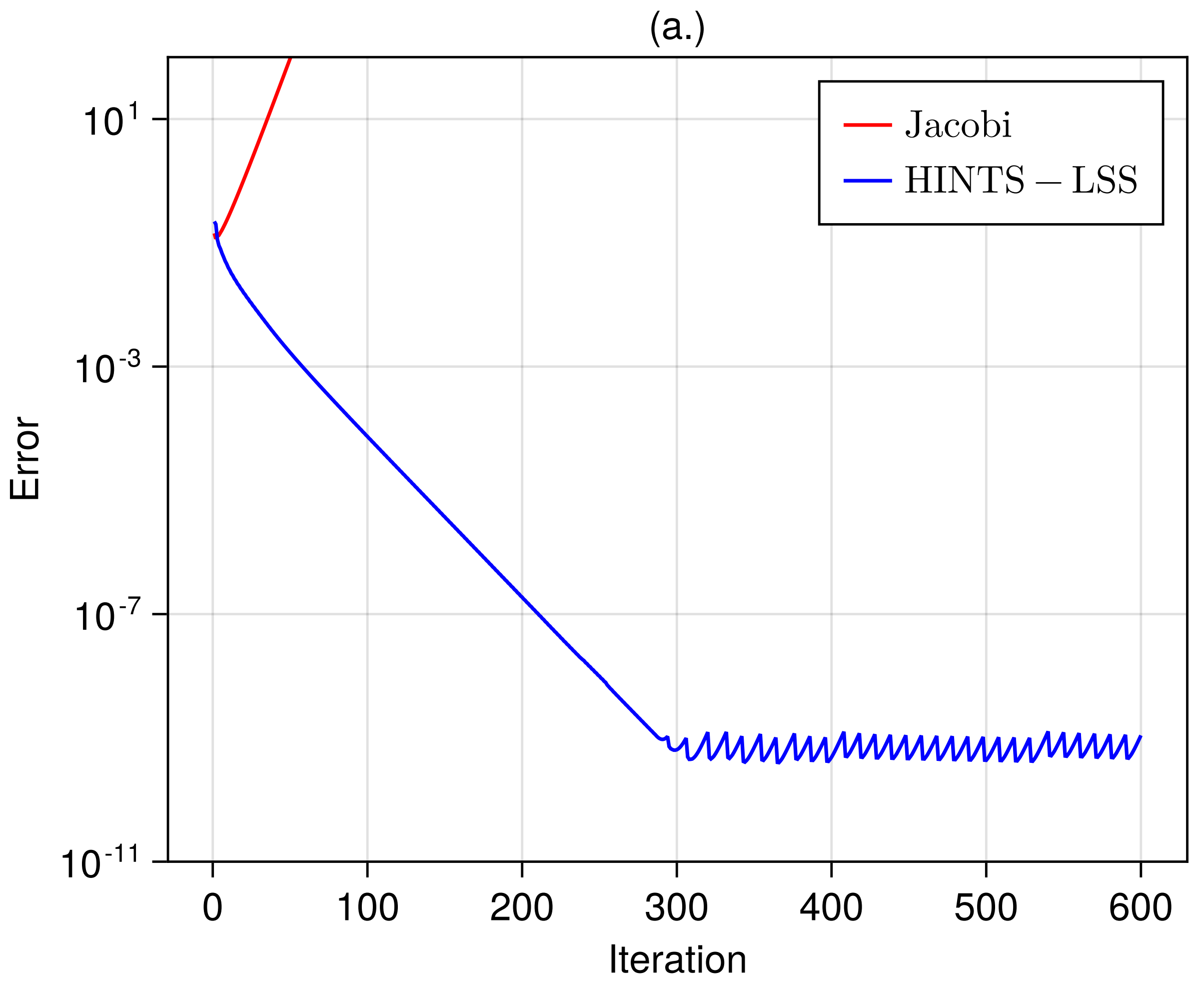
4.2 2D Helmholtz Equation

$$-\Delta u - k^2 u = f, \quad \text{in } \Omega,$$
$$\nabla u \cdot \boldsymbol{n} + u = g, \text{ on } \partial\Omega,$$

where $\Omega = (0,1)^2$, $f(x,y) = -x^2 - y^2 - \exp(-y)$, $g(x,y) = \sin(x)\sin(y)$, and

$$k(x,y) = \begin{cases} 10 \text{ if } (x,y) \in [0,0.5]^2 \cup [0.5,1]^2, \\ \\ 20 \text{ if } (x,y) \in [0,0.5) \times (0.5,1] \cup (0.5,1] \times [0,0.5). \end{cases}$$

4.2 2D Helmholtz Equation



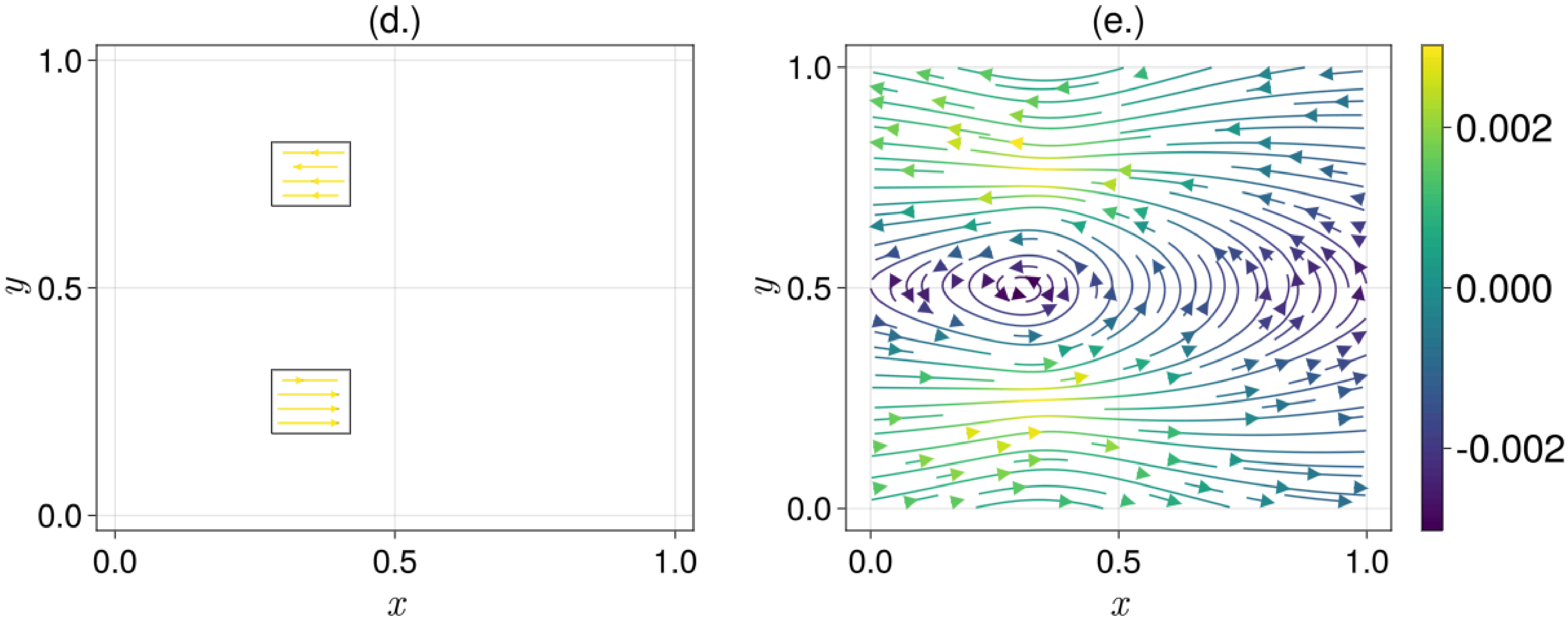
4.3 2D Stokes Equations

$$\begin{aligned} -\nabla \cdot (\mu \nabla \boldsymbol{u}) + \nabla p = \boldsymbol{f}, &\quad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} = 0, &\quad \text{in } \Omega, \\ (\nabla \boldsymbol{u} - pI) \cdot \boldsymbol{n} + \boldsymbol{u} = \boldsymbol{0}, &\text{ on } \partial\Omega, \end{aligned}$$

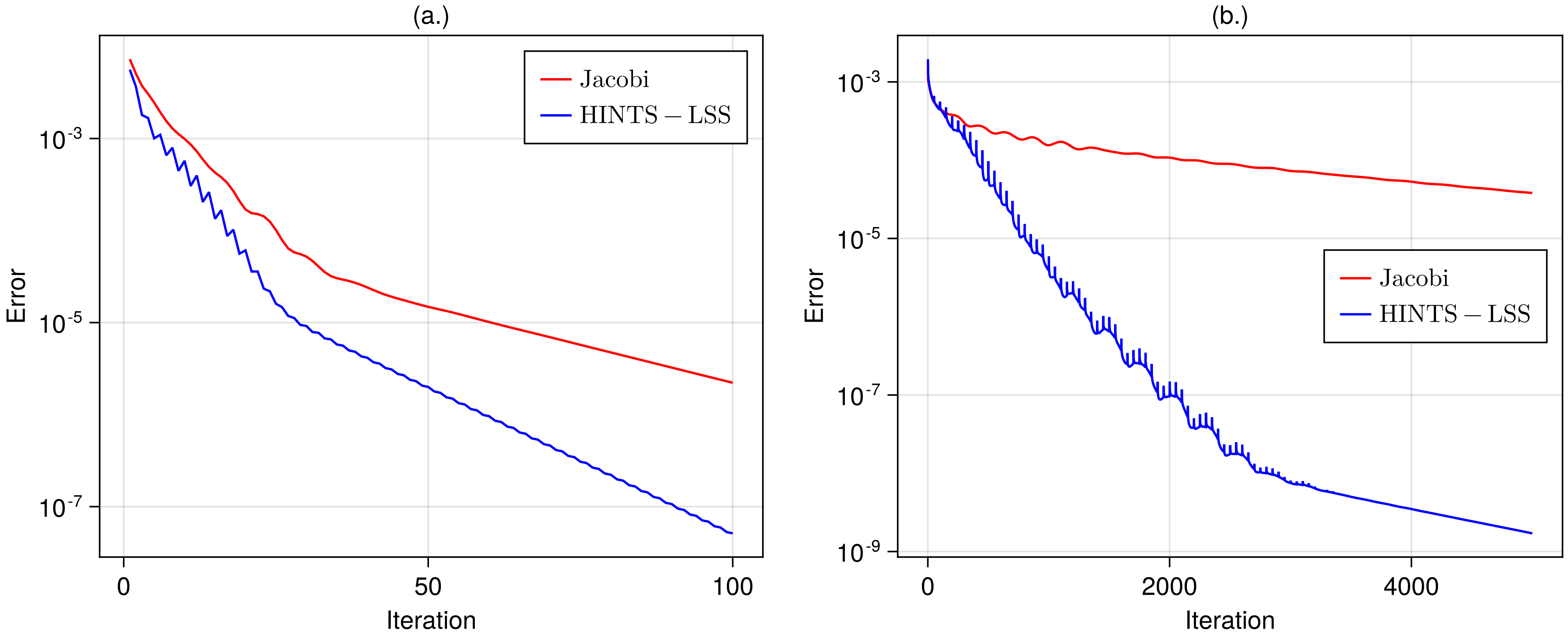
where $\Omega = (0,1)^2, \mu = 1$ and

$$\boldsymbol{f}(x,y) = \begin{cases} \begin{pmatrix} 1 & 0 \end{pmatrix}^\top \text{ if } (x,y) \in [0.3, 0.4] \times [0.2, 0.3], \\ \begin{pmatrix} -1 & 0 \end{pmatrix}^\top \text{ if } (x,y) \in [0.3, 0.4] \times [0.7, 0.8], \\ \begin{pmatrix} 0 & 0 \end{pmatrix}^\top \text{ otherwise.} \end{cases}$$

4.3 2D Stokes Equations



4.3 2D Stokes Equations



5. Conclusions

5.1 Pros and Cons

- Retains the features of operator learning, allowing the same model to solve equations with different boundary conditions.
- Effectively accelerates the convergence speed of stationary iteration, especially for Jacobi iteration.
- Can use models trained on coarse grids to solve discrete problems on fine grids.
- Can use models trained with low-order finite elements to solve problems discretized with high-order finite elements.
- The algorithm only involves matrix-vector multiplication.
- Training of DeepONet is costly.

5.1 Pros and Cons

- Difficult to approximate high-frequency target functions.