# A Method for Computing Inverse Parametric PDEs with Randomized Neural Networks

Steven Suchuan Dong

Center for Computational & Applied Mathematics

Department of Mathematics

Purdue University

(Joint with: Yiran Wang)

Scientific Machine Learning Workshop

Brown University ICERM, June 2023

Randomized Neural Network:

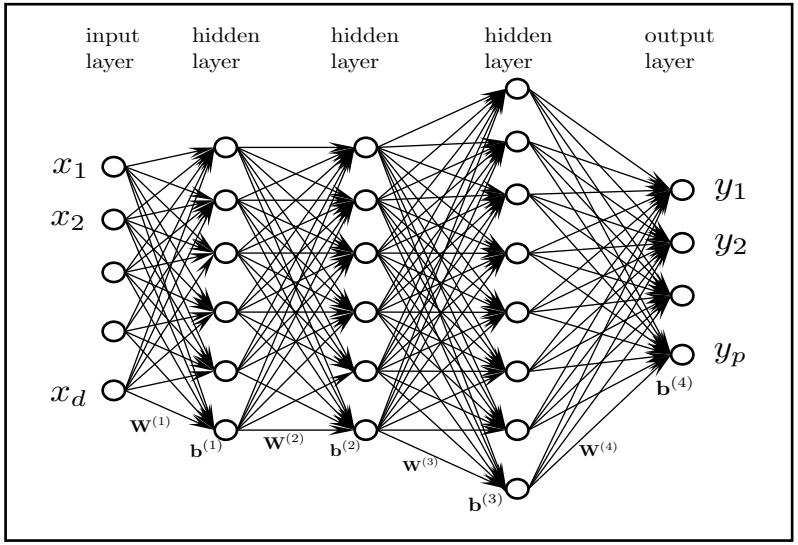
- A subset of NN parameters is assigned to random values and fixed (not-trainable);
- Only the rest of NN parameters are adjustable/trainable.

Strategy Underlying Randomized NNs:

- Full set of NN parameters extremely hard and costly to train;
- Randomized NN can make the optimization task of NN training simpler, (hopefully) without severely sacrificing achievable approximation capacity.

# ELM: One Type of Randomized NN

ELM – Extreme Learning Machine




Feed-forward NN

Extreme Learning Machine (ELM): (Huang et al, 2006)

- All hidden-layer coefficients are randomly assigned and fixed (non-trainable);
- Only the output-layer coefficients are trainable.

Also known as:
RVFL (Random Vector Functional Link Network, Pao et al. 1994)

# Randomized Neural Networks Are Universal Function Approximators

- Universal approximation theorem (Igelnik & Pao, 1995; Huang et al, 2006)

**Theorem (Huang et al 2006):** Given any bounded non-constant piecewise continuous function $g : \mathbb{R} \to \mathbb{R}$, for any continuous target function $f$, there exists a sequence of single-hidden-layer feedforward neural networks with $g$ as the activation function, with its hidden-layer coefficients randomly generated, and with its output-layer coefficients appropriately chosen, such that $\lim_{n\to\infty} \|f - f_n\| = 0$, where $n$ is the number of hidden-layer nodes and $f_n$ is the output of the neural network.

Hidden-layer coefficients randomly set (not trained);
Output-layer coefficients trainable/adjustable

# What Are Inverse PDE Problems?

- PDE contains unknown parameters (as constants or field distributions).
- Some measurement data (sparse, noisy) available.
- Goal: determine the unknown PDE parameters and the unknown solution field simultaneously

Given:

Sparse measurement of concentration data at certain points

Want:

Diffusion coefficient, and concentration field in space/time

# Inverse Parametric PDE Problem

Unknown (inverse) parameters

Unknown solution field

$$\alpha_1 \mathcal{L}_1(u) + \alpha_2 \mathcal{L}_2(u) + \cdots + \alpha_n \mathcal{L}_n(u) + \mathcal{F}(u) = f(\mathbf{x}), \qquad \text{(PDE)}$$

$$\mathcal{B}u(\mathbf{x}) = g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \qquad \text{(BC)}$$

$$\underbrace{\mathcal{M}u}(\boldsymbol{\xi}) = S(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in \Omega_s \subset \Omega. \qquad \text{(measurement)}$$

Measured variable

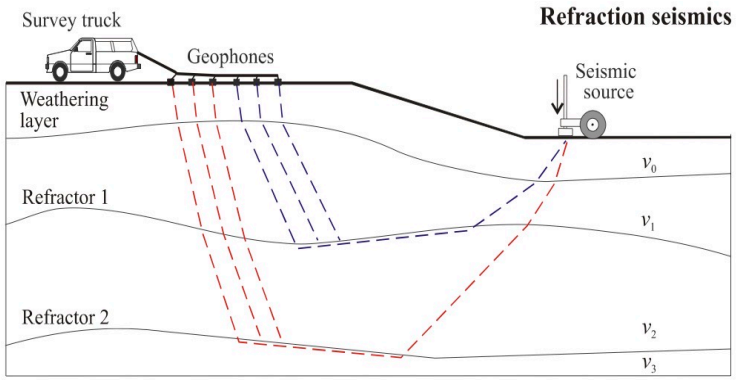
Measurement data (noisy)

Random sparse measurement points

Known: measurement data $S(\boldsymbol{\xi})$, source terms, BC
Unknown: $\alpha_i$ $(1 \leqslant i \leqslant n)$, $u(\mathbf{x})$

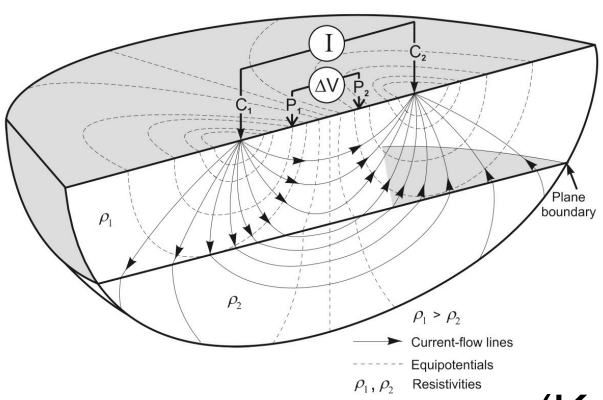
# Why Inverse PDE Problems

- widespread in environmental, medical, and geological applications



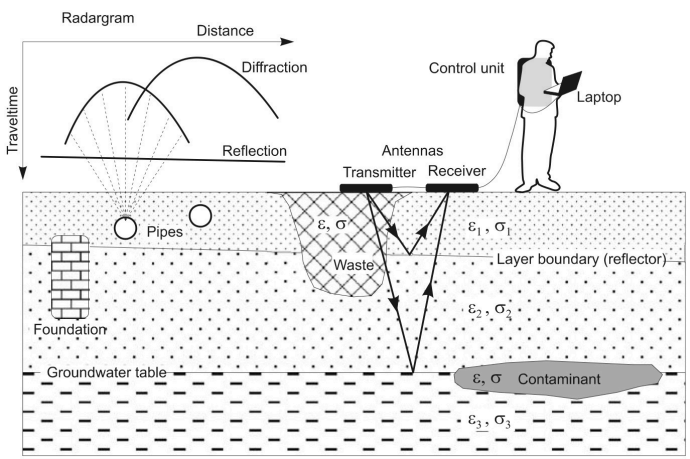

Seismic method

Direct current resistivity method




(Knodel et al. 2007)

Ground penetrating radar



# NN-Related Works on Inverse PDEs

- Bongard & Lipton (2007); Schmidt & Lipton (2009)
- Brunton et al (2016); Rudy et al (2017,2019); Schaeffer (2017); Zhang & Lin (2018);
- Raissi & Karniadakis (2018, 2020);
- Raissi et al (2019);
- Berg & Nystrom (2018, 2019); Long et al (2019); Both et al (2021)
- Jagtap & Karniadakis (2020); Jagtap et al (2020, 2022);
- Karniadakis et al (2021) ← (Review Article)
- Lu et al (2021); Mao et al (2020); Chen et al (2020); Meng & Karniadakis (2020); Yang et al (2021); Cai et al (2021);
- Tartakovsky et al (2020); Li et al (2020); Mathews et al (2020);
- Patel et al (2022); Dwivedi et al (2021); Yuan et al (2022);
- … …

(Incomplete list of contributions)

# LocELM / ELM for Forward PDEs

LocELM Method:

- Based on ELM (randomized NN);
- Combines domain decomposition, $C^k$ continuity conditions, and local neural networks;
- Network training by linear or nonlinear least squares methods (not by gradient-descent type algorithms);
- For forward PDE problems (linear or nonlinear).

- Spectral accuracy (exponential convergence w.r.t. number of collocation points and number of training parameters)
- Outperform traditional numerical methods (classical and high-order FEM) in terms of accuracy and computational cost (network training time)

(Dong & Li, CMAME, 2021; Dong & Yang, JCP, 2022)

- Can we extend this approach to inverse PDE problems to gain similar performance benefits (high accuracy, competitive computational cost)?

# Main Complication

$$\alpha_1 \mathcal{L}_1(u) + \alpha_2 \mathcal{L}_2(u) + \cdots + \alpha_n \mathcal{L}_n(u) + \mathcal{F}(u) = f(\mathbf{x})$$

- Unknown PDE parameters and unknown solution field are coupled;
- Problem is fully nonlinear.

# **Three Algorithms:**
# (training NN, solving inverse problem)

NLLSQ $\Longrightarrow$

- Solve $\alpha_i$ and solution field $u$ altogether by nonlinear least squares method with perturbations (NLLSQ-perturb).

VarPro-F1 $\Longrightarrow$

- Eliminate $\alpha_i$ from original problem using variable projection (VarPro) $\Longrightarrow$ Reduced problem about $u$ only.
- Solve reduced problem for $u$ by NLLSQ-perturb.
- Compute $\alpha_i$ by linear least squares method.

VarPro-F2 $\Longrightarrow$

- Eliminate solution field $u$ from original problem using VarPro $\Longrightarrow$ Reduced problem about $\alpha_i$ only.
- Solve reduced problem for $\alpha_i$ by NLLSQ-perturb.
- Compute $u$ based on $\alpha_i$.

# NLLSQ Algorithm

(nonlinear least squares algorithm)

# Main Procedure

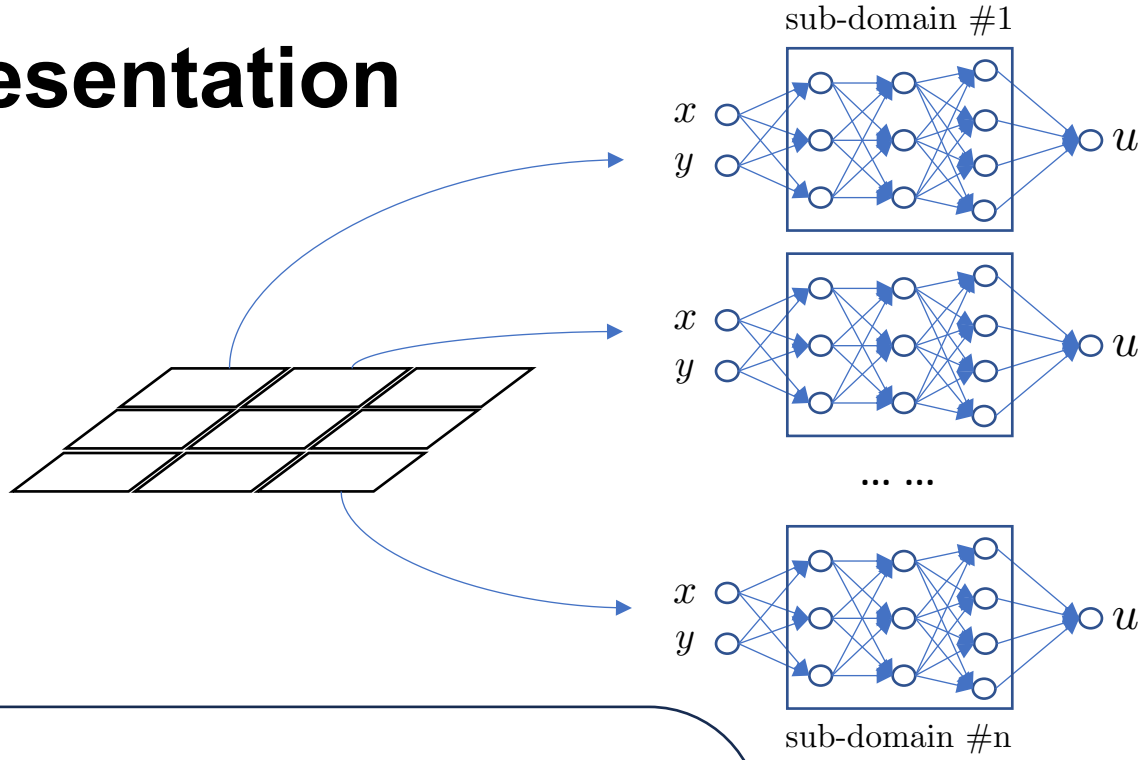
Given: PDE, BC, Measurement data

Procedure:

- Represent solution field $u$ by local ELM (locELM) representation;
- Formulate residual function in terms of local ELM representation;
- Enforce residual function to be zero on training collocation points and random measurement points
  $\Longrightarrow$ nonlinear algebraic system about $\alpha_i$ and ELM training parameters.
- Seek least squares solution to algebraic system, and solve this system by nonlinear least squares method with perturbations (NLLSQ-perturb).
- Set ELM training parameters by the least squares solution.

# Local ELM Representation of Functions

(Dong & Li, CMAME, 2021)




locELM Representation:

- Partition domain into sub-domains.
- Impose $C^k$ continuity conditions across sub-domain boundaries.
- Represent function on each sub-domain by a local randomized NN.
- Local NN: hidden-layer coefficients set to random numbers on $[-R_m, R_m]$ and fixed.
- Local NN: Output layer linear, whose coefficients are trainable.
- Local NNs are coupled due to $C^k$ continuity, trained in coupled fashion.

# LocELM Representation

NN Logic $\Longrightarrow$ Expansion Relation

On sub-domain $\Omega_s$, $(1 \leq s \leq N)$:

$$u_s(\mathbf{x}) = \sum_{j=1}^{M} \beta_{sj} \phi_{sj}(\mathbf{x})$$

output field on $\Omega_s$

output-layer coefficients (trainable)

output field of last hidden layer (fixed, random)

Function on each sub-domain is expanded in terms of a set of random basis functions

# Residual Functions

$$u_i(\mathbf{x}) = \sum_{j=1}^{M} \beta_{ij}\phi_{ij}(\mathbf{x}) = \boldsymbol{\Phi}_i(\mathbf{x})\boldsymbol{\beta}_i$$

$$\boldsymbol{\alpha} = (\dots, \alpha_i, \dots), \quad \boldsymbol{\beta} = (\dots, \boldsymbol{\beta}_i, \dots)$$

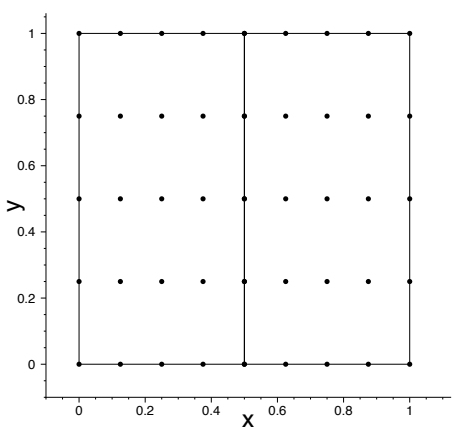
$$\mathbf{R}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{x}, \boldsymbol{\xi}) = \begin{bmatrix} \alpha_1\mathcal{L}_1(u_i) + \alpha_2\mathcal{L}_2(u_i) + \cdots + \alpha_n\mathcal{L}_n(u_i) + \mathcal{F}(u_i) - f(\mathbf{x}),\ \mathbf{x} \in \Omega_i,\ 1 \leqslant i \leqslant N \\ \mathcal{B}u_i(\mathbf{x}) - g(\mathbf{x}),\ \mathbf{x} \in \partial\Omega \cap \Omega_i,\ 1 \leqslant i \leqslant N \\ \mathcal{M}u_i(\boldsymbol{\xi}) - S(\boldsymbol{\xi}),\ \boldsymbol{\xi} \in \Omega_s \cap \Omega_i,\ 1 \leqslant i \leqslant N \\ \mathcal{C}u_i(\mathbf{x}) - \mathcal{C}u_j(\mathbf{x}),\ \mathbf{x} \in \partial\Omega_i \cap \partial\Omega_j,\ \text{for all adjacent } (\Omega_i, \Omega_j),\ 1 \leqslant i, j \leqslant N \end{bmatrix}$$

(PDE)
(BC)
(measurement)
(C^k continuity)

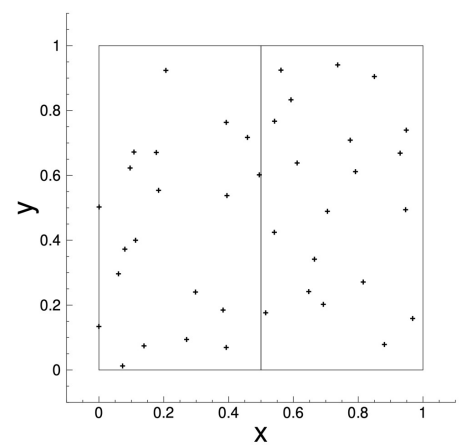
Set/choose:

- training collocation points: $\mathbf{x}_p^s$, on each sub-domain $\Omega_s$
- measurement points: $\boldsymbol{\xi}_q^s$, on each sub-domain $\Omega_s$

Random measurement points




Regular grid points as collocation points



# From Residual Function to Algebraic System

Enforce residual function to zero on collocation and measurement points

Nonlinear algebraic system about $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$

Compute $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by nonlinear least squares method with perturbations (NLLSQ-perturb)

Algebraic System:

$$\mathbf{0} = \mathbf{R}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{x}_p^s, \boldsymbol{\xi}_q^s) = \mathbf{R}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \begin{bmatrix} \mathbf{R}^{\mathrm{pde}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \\ \mathbf{R}^{\mathrm{bc}}(\boldsymbol{\beta}) \\ \mathbf{R}^{\mathrm{mea}}(\boldsymbol{\beta}) \\ \mathbf{R}^{\mathrm{ck}}(\boldsymbol{\beta}) \end{bmatrix}$$

NLLSQ Algorithm

# VarPro-F1 Algorithm

# Main Procedure

VarPro: Variable Projection
(Golub & Pereyra, 1973)

Idea:

- Eliminate $\boldsymbol{\alpha}$ from original problem using variable projection (VarPro) $\Longrightarrow$ Reduced problem about $\boldsymbol{\beta}$ only.
- Solve reduced problem for $\boldsymbol{\beta}$ by nonlinear least squares method with perturbations (NLLSQ-perturb).
- Compute $\boldsymbol{\alpha}$ by linear least squares method.

Eliminate inverse parameters from original problem $\Longrightarrow$ Reduced problem about solution field

# VarPro Reformulation

Start with Algebraic System:

$$\mathbf{R}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{x}_p^s, \boldsymbol{\xi}_q^s) = \mathbf{R}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \begin{bmatrix} \mathbf{R}^{\mathrm{pde}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \\ \mathbf{R}^{\mathrm{bc}}(\boldsymbol{\beta}) \\ \mathbf{R}^{\mathrm{mea}}(\boldsymbol{\beta}) \\ \mathbf{R}^{\mathrm{ck}}(\boldsymbol{\beta}) \end{bmatrix} = \mathbf{0}$$

Rewrite the System:

$$\mathbf{R}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathbf{H}(\boldsymbol{\beta})\boldsymbol{\alpha} - \mathbf{b}(\boldsymbol{\beta}) = \mathbf{0}$$

$$\text{or } \mathbf{H}(\boldsymbol{\beta})\boldsymbol{\alpha} = \mathbf{b}(\boldsymbol{\beta})$$

(noting $\mathbf{R}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is linear w.r.t. $\boldsymbol{\alpha}$)

For arbitrary given \beta, the optimal \alpha is:

$$\boldsymbol{\alpha}^* = \mathbf{H}^+(\boldsymbol{\beta})\mathbf{b}(\boldsymbol{\beta})$$

Substitute Back:

$$\begin{aligned} \mathbf{r}(\boldsymbol{\beta}) &= \mathbf{R}(\boldsymbol{\alpha}^*, \boldsymbol{\beta}) \\ &= \mathbf{H}(\boldsymbol{\beta})\mathbf{H}^+(\boldsymbol{\beta})\mathbf{b}(\boldsymbol{\beta}) - \mathbf{b}(\boldsymbol{\beta}) \\ &= \mathbf{0} \end{aligned}$$

Reduced Problem about $\boldsymbol{\beta}$

# VarPro-F1 Algorithm

Procedure:

- Solve reduced problem for $\boldsymbol{\beta}$ by nonlinear least squares method with perturbations (NLLSQ-perturb).

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{r}(\boldsymbol{\beta})\|^2 = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{H}(\boldsymbol{\beta})\mathbf{H}^+(\boldsymbol{\beta})\mathbf{b}(\boldsymbol{\beta}) - \mathbf{b}(\boldsymbol{\beta})\|^2$$

- Solve for $\boldsymbol{\alpha}$ by linear least squares method based on $\boldsymbol{\beta}^*$.

$$\boldsymbol{\alpha}^* = \mathbf{H}^+(\boldsymbol{\beta}^*)\mathbf{b}(\boldsymbol{\beta}^*)$$

VarPro-F1 solution: $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$

Equivalent to: $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) = \arg\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{R}(\boldsymbol{\alpha}, \boldsymbol{\beta})\|^2$

# VarPro-F2 Algorithm

# VarPro-F2 Algorithm
## (When Forward PDE Problem Is Linear)

Idea:

- Eliminate $\boldsymbol{\beta}$ from original problem using VarPro
  $\Longrightarrow$ Reduced problem about $\boldsymbol{\alpha}$ only.
- Solve reduced problem for $\boldsymbol{\alpha}$ by nonlinear least squares method with perturbations (NLLSQ-perturb).
- Compute $\boldsymbol{\beta}$ by linear least squares method.

Reciprocal to VarPro-F1 in some sense

# VarPro Reformulation
## (Linear Forward PDE Problem)

Start with Algebraic System:

$$\mathbf{R}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{x}_p^s, \boldsymbol{\xi}_q^s) = \mathbf{R}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \begin{bmatrix} \mathbf{R}^{\mathrm{pde}}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \\ \mathbf{R}^{\mathrm{bc}}(\boldsymbol{\beta}) \\ \mathbf{R}^{\mathrm{mea}}(\boldsymbol{\beta}) \\ \mathbf{R}^{\mathrm{ck}}(\boldsymbol{\beta}) \end{bmatrix} = \mathbf{0}$$

Rewrite the System:

$$\mathbf{R}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathbf{H}(\boldsymbol{\alpha})\boldsymbol{\beta} - \mathbf{b} = \mathbf{0}$$

$$\text{or } \mathbf{H}(\boldsymbol{\alpha})\boldsymbol{\beta} = \mathbf{b}$$

(noting $\mathbf{R}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is linear w.r.t. $\boldsymbol{\beta}$)

For arbitrary given \alpha, the optimal \beta is:

$$\boldsymbol{\beta}^* = \mathbf{H}^+(\boldsymbol{\alpha})\mathbf{b}$$

Substitute Back:

$$\begin{aligned} \mathbf{r}(\boldsymbol{\alpha}) &= \mathbf{R}(\boldsymbol{\alpha}, \boldsymbol{\beta}^*) \\ &= \mathbf{H}(\boldsymbol{\alpha})\mathbf{H}^+(\boldsymbol{\alpha})\mathbf{b} - \mathbf{b} \\ &= \mathbf{0} \end{aligned}$$

Reduced Problem about $\boldsymbol{\alpha}$

# VarPro-F2 Algorithm
## (Linear Forward PDE Problem)

Procedure:

- Solve reduced problem for $\boldsymbol{\alpha}$ by nonlinear least squares method with perturbations (NLLSQ-perturb).

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\mathbf{r}(\boldsymbol{\alpha})\|^2 = \arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\|\mathbf{H}(\boldsymbol{\alpha})\mathbf{H}^+(\boldsymbol{\alpha})\mathbf{b} - \mathbf{b}\|^2$$

- Solve for $\boldsymbol{\beta}$ by linear least squares method based on $\boldsymbol{\alpha}^*$.

$$\boldsymbol{\beta}^* = \mathbf{H}^+(\boldsymbol{\alpha}^*)\mathbf{b}$$

VarPro-F2 solution: $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$

# VarPro-F2 Algorithm
## (When Forward PDE Problem is Nonlinear)

(Newton method + VarPro-F2 algorithm)

Idea: (nonlinear forward PDE problem)

- Use Newton method to linearize the problem w.r.t. the solution field.
- Within each Newton iteration, use VarPro-F2 algorithm to solve the linearized problem for $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, as in previous slides.
- Upon convergence of Newton iteration, attain solution to original inverse problem.

# Numerical Examples

# Inverse Poisson Equation

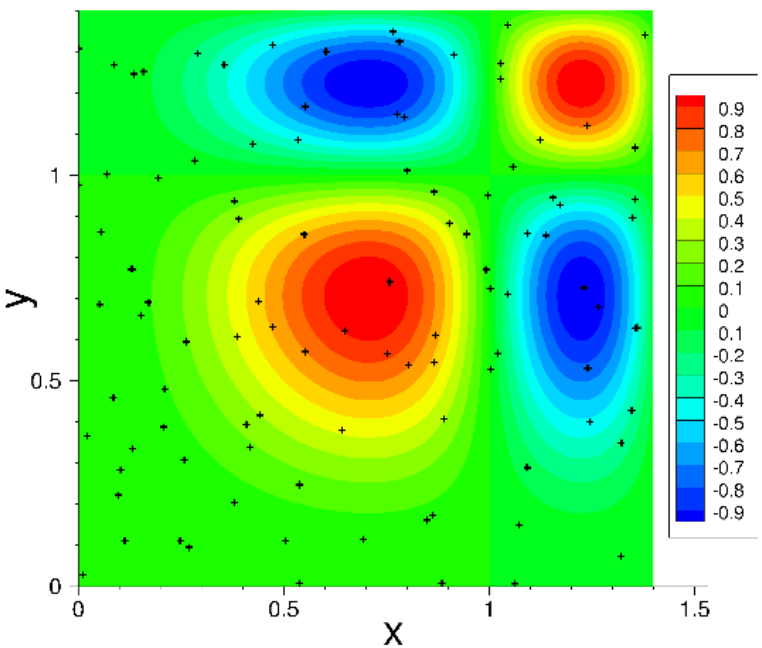
$$\frac{\partial^2 u}{\partial x^2} + \alpha \frac{\partial^2 u}{\partial y^2} = f(x,y)$$

$$u(0,y) = g_1(y), \quad u(1.4,y) = g_2(y)$$

$$u(x,0) = g_3(x), \quad u(x,1.4) = g_4(x)$$

$$u(\xi_i,\eta_i) = S(\xi_i,\eta_i) \qquad \text{(measurement)}$$

Solution field
Random measurement points




Known:
source term and BC: $f$, $g_i$
measurement data: $S(\xi_i,\eta_i)$

Unknown:
$\alpha$, $u(x,y)$

Noisy measurement data

$$S(\xi_i,\eta_i) = u_{ex}(\xi_i,\eta_i)(1+\epsilon\zeta_i)$$

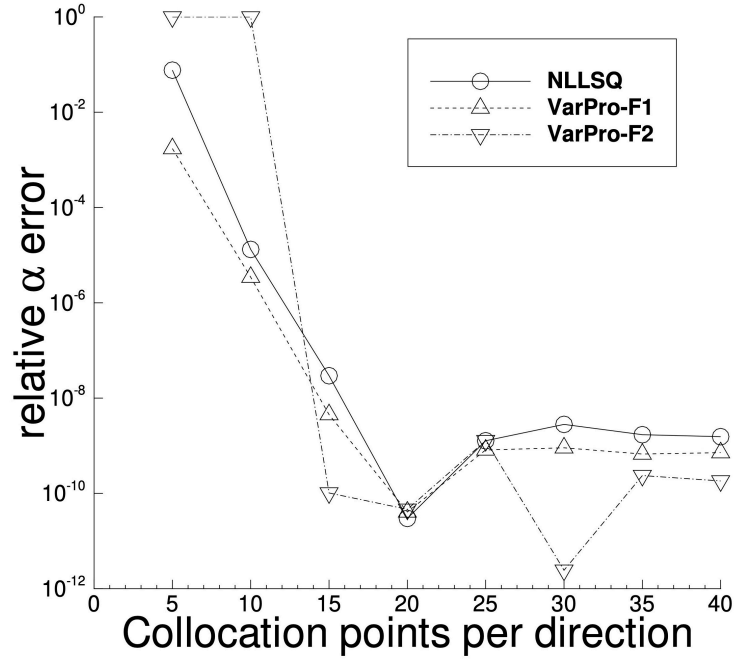
Noise level

Random number [-1,1]

# Error of Inverse Parameters

(Noise-free measurement data)

Error decreases exponentially w.r.t. collocation points




Alpha error vs. collocation points

Computed alpha vs. collocation points

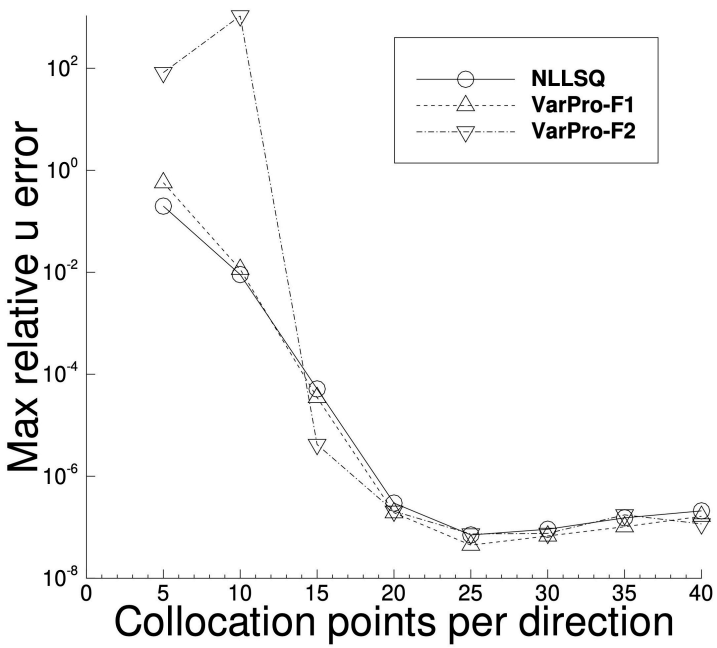
| $Q$ | $\alpha$ (NLLSQ) | $\alpha$ (VarPro-F1) | $\alpha$ (VarPro-F2) |
|---|---|---|---|
| 5×5 | 1.076466245043E+0 | 9.982719409724E-1 | 0.000000000000E+0 |
| 10×10 | 9.999867935849E-1 | 9.999965494049E-1 | -3.188390321381E-5 |
| 15×15 | 1.000000029498E+0 | 9.999999954822E-1 | 9.999999998978E-1 |
| 20×20 | 9.999999999701E-1 | 9.999999999592E-1 | 9.999999999536E-1 |
| 25×25 | 9.999999987249E-1 | 1.000000000817E+0 | 1.000000001279E+0 |
| 30×30 | 1.000000002811E+0 | 1.000000000906E+0 | 1.000000000002E+0 |
| 35×35 | 1.000000001708E+0 | 1.000000000670E+0 | 1.000000000237E+0 |
| 40×40 | 1.000000001552E+0 | 1.000000000717E+0 | 1.000000000183E+0 |

$\alpha_{exact} = 1$
NN: [2,600,1]
Gaussian activation
100 measurement points

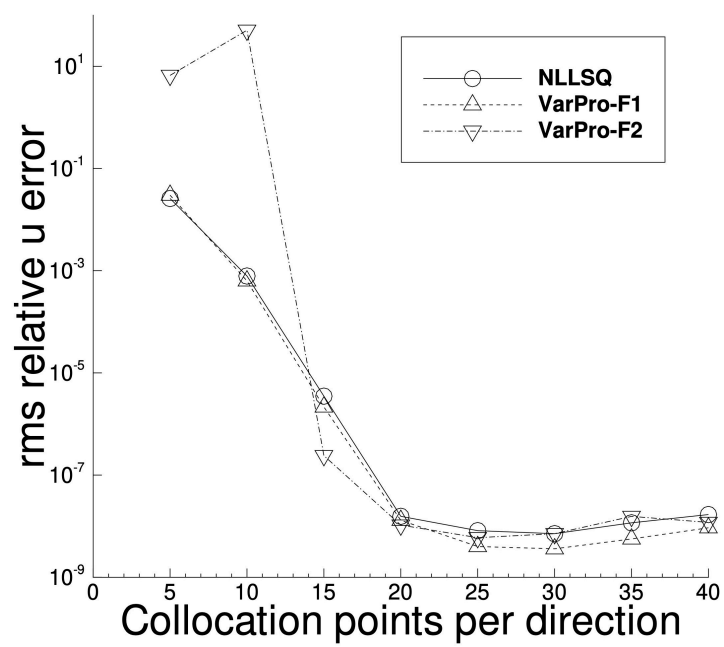
# Error in Solution Field vs. Collocation Points

(Noise-free measurement data)

Maximum error in domain




Rms error in domain



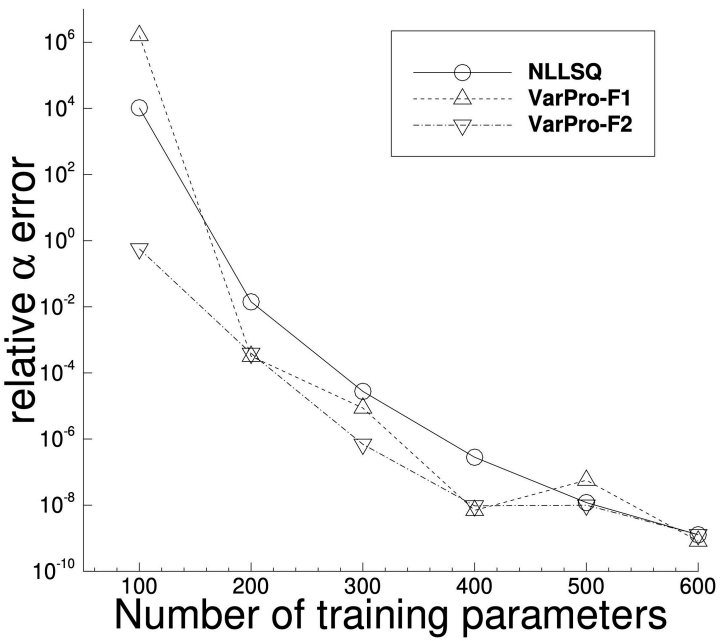

Errors decrease exponentially w.r.t. number of collocation points

# Errors vs. Training Parameters

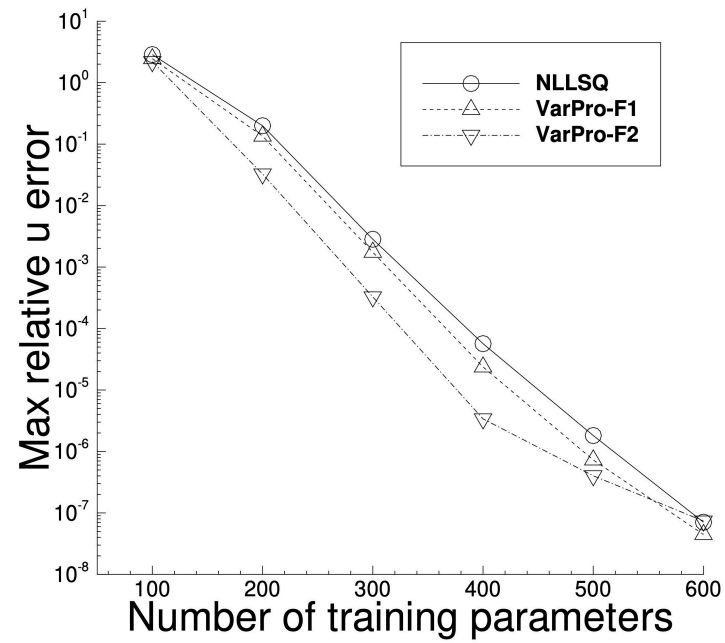
(Noise-free measurement data)

NN: $[2, M, 1]$
$25 \times 25$ collocation points
100 measurement points

Alpha error vs. training parameters




u error vs. training parameters



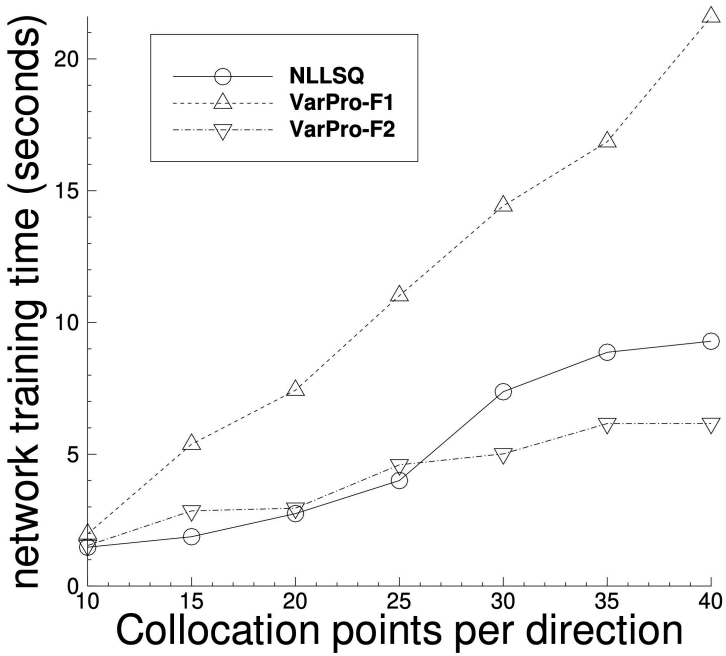

Errors decrease (approximately) exponentially w.r.t. number of training parameters

# Network Training Cost

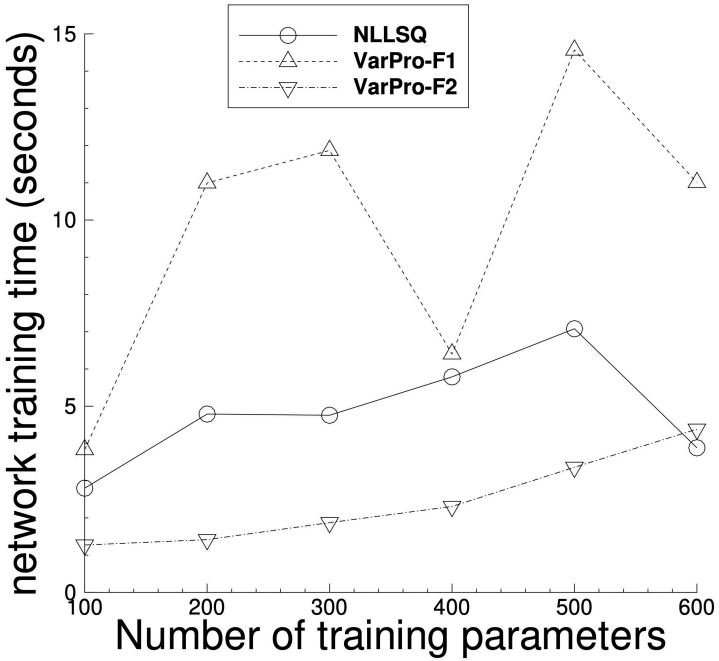
Timing collected on a MAC computer

(Noise-free measurement data)

Training time vs. collocation points




Training time vs. training parameters




Network training time grows roughly linearly w.r.t. number of collocation points and number of training parameters

# Noisy Data:
# Inverse Parameters

$$S(\boldsymbol{\xi}_i) = u_{ex}(\boldsymbol{\xi}_i)(1 + \epsilon\zeta_i)$$

Noise level

Random number [-1,1]

NN: [2,500,1]
25×25 collocation points
50 random measurement points
$\alpha_{ex} = 1$

Accuracy degrades with increasing noise level

| | $\epsilon$ | computed-$\alpha$ | $\epsilon$ | computed-$\alpha$ | |
|---|---|---|---|---|---|
| No noise | 0.0 | 9.99999993208E-1 | 0.1 | 9.8779390E-1 | 10% noise |
| | 0.001 | 9.9987537E-1 | 0.2 | 9.7602056E-1 | |
| | 0.005 | 9.9937764E-1 | 0.5 | 9.4329282E-1 | |
| 1% noise | 0.01 | 9.9875752E-1 | 0.7 | 9.2316247E-1 | |
| | 0.05 | 9.9383633E-1 | 1.0 | 8.9557261E-1 | |

Computed by NLLSQ Algorithm

# Noisy Data: Errors

NN: [2,500,1]
$25\times25$ collocation points
50 random measurement points
$\alpha_{ex} = 1$

Relative errors

| | | NLLSQ | | | VarPro-F1 | | | VarPro-F2 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\epsilon$ | $e_\alpha$ | $l^\infty$-u | $l^2$-u | $e_\alpha$ | $l^\infty$-u | $l^2$-u | $e_\alpha$ | $l^\infty$-u | $l^2$-u |
| | 0.0 | 6.79E-9 | 1.81E-6 | 1.93E-7 | 5.93E-8 | 7.59E-7 | 1.38E-7 | 4.20E-10 | 4.50E-7 | 2.31E-8 |
| | 0.001 | 1.25E-4 | 2.79E-4 | 8.04E-5 | 1.33E-4 | 2.82E-4 | 8.50E-5 | 1.23E-4 | 2.81E-4 | 7.85E-5 |
| | 0.005 | 6.22E-4 | 1.39E-3 | 4.01E-4 | 6.73E-4 | 1.42E-3 | 4.29E-4 | 6.10E-4 | 1.41E-3 | 3.92E-4 |
| 1% noise | 0.01 | 1.24E-3 | 2.79E-3 | 8.02E-4 | 1.35E-3 | 2.85E-3 | 8.61E-4 | 1.22E-3 | 2.81E-3 | 7.82E-4 |
| | 0.05 | 6.16E-3 | 1.39E-2 | 4.00E-3 | 6.63E-3 | 1.42E-2 | 4.26E-3 | 6.49E-3 | 1.41E-2 | 4.07E-3 |
| 10% noise | 0.1 | 1.22E-2 | 2.79E-2 | 7.98E-3 | 1.33E-2 | 2.86E-2 | 8.58E-3 | 1.19E-2 | 2.81E-2 | 7.75E-3 |
| | 0.5 | 5.67E-2 | 1.42E-1 | 3.90E-2 | 6.08E-2 | 1.44E-1 | 4.17E-2 | 5.52E-2 | 1.43E-1 | 3.78E-2 |
| | 1.0 | 1.04E-1 | 2.88E-1 | 7.63E-2 | 1.11E-1 | 2.93E-1 | 8.14E-2 | 1.08E-1 | 2.90E-1 | 7.62E-2 |

Inverse parameters

Solution field

Method remains quite accurate with noisy data

# Comparison with PINN

PINN – Physics-informed neural network (Raissi et al 2019)

Both:
30×30 collocation points
100 random measurement points
Gaussian activation function

PINN:
NN: [2,30,30,30,1]
20,000 epochs
learning rate: 0.01 → 1E-4 for first 10K epochs, then fixed

Current (NLLSQ):
NN: [2,500,1]

Current method produces more accurate results with less training time

No noise 1% noise

| method | $\epsilon$=0 | | | | $\epsilon$=0.01 | | | |
|---|---|---|---|---|---|---|---|---|
| | $e_\alpha$ | $l^\infty$-u | $l^2$-u | time(sec) | $e_\alpha$ | $l^\infty$-u | $l^2$-u | time(sec) |
| PINN (Adam) | 6.31E-3 | 1.08E-2 | 3.56E-3 | 134.5 | 5.53E-3 | 1.03E-2 | 3.30E-3 | 130.9 |
| current (NLLSQ) | 1.66E-8 | 3.66E-6 | 2.62E-7 | 11.5 | 9.72E-4 | 1.76E-3 | 5.26E-4 | 10.4 |

Network training time

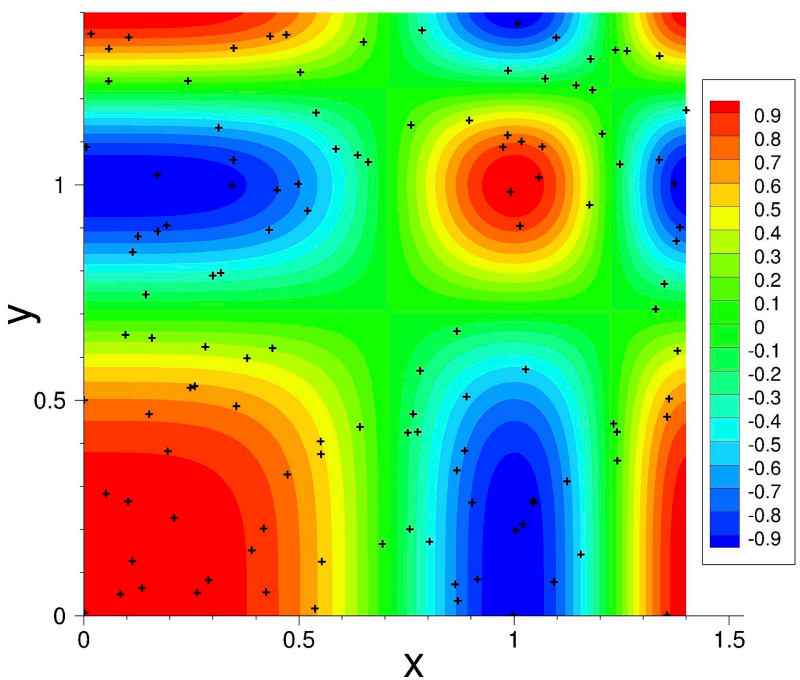
Network training time

# Inverse Nonlinear Helmholtz Equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} - \alpha_1 u + \alpha_2 \cos(2u) = f(x,y),$$
$$u(0,y) = g_1(y), \quad u(1.4,y) = g_2(y),$$
$$u(x,0) = g_3(x), \quad u(x,1.4) = g_4(x),$$
$$u(\xi_i,\eta_i) = S(\xi_i,\eta_i) \quad \leftarrow \text{(measurement)}$$

<u>Unknowns:</u>

$\alpha_1,\ \alpha_2,\ u(x,y)$




<u>Exact solution:</u>

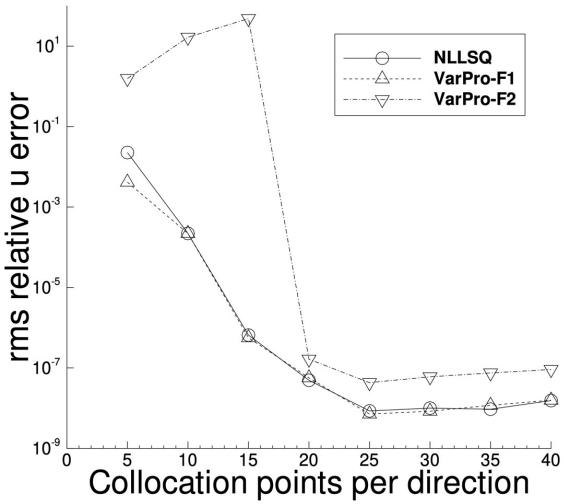
$$\alpha_1^{ex} = 100$$
$$\alpha_2^{ex} = 5$$
$$u_{ex}(x,y) = \cos(\pi x^2)\cos(\pi y^2)$$

Solution field

Random measurement points

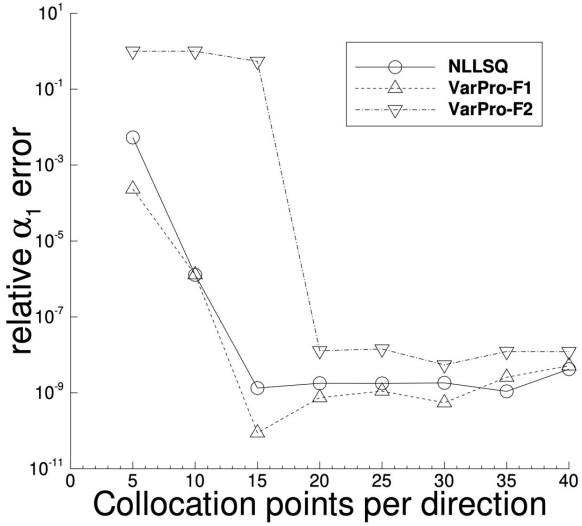
# Error vs. Collocation Points (noise-free data)

NN: [2,500,1]
100 measurement points

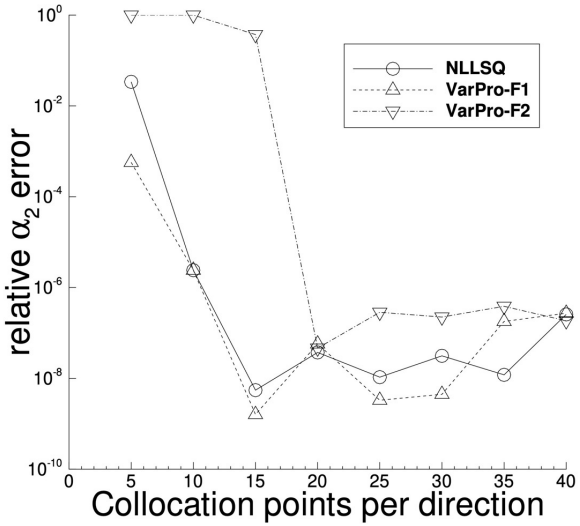



Solution field error

$\alpha_1$ error

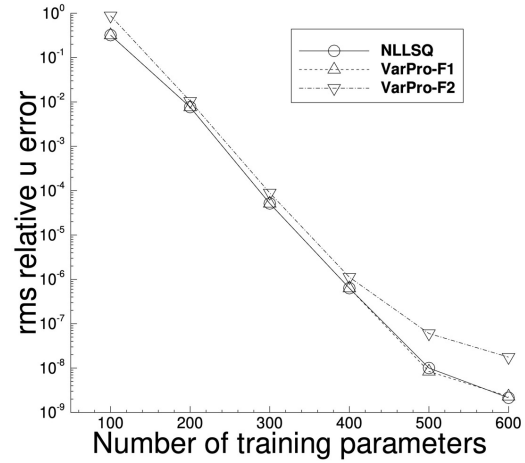



$\alpha_2$ error



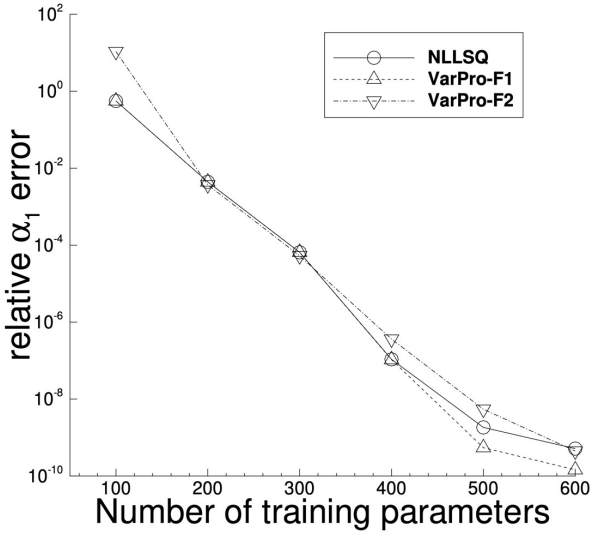

Exponential convergence

# Error vs. Training Parameters (noise-free data)

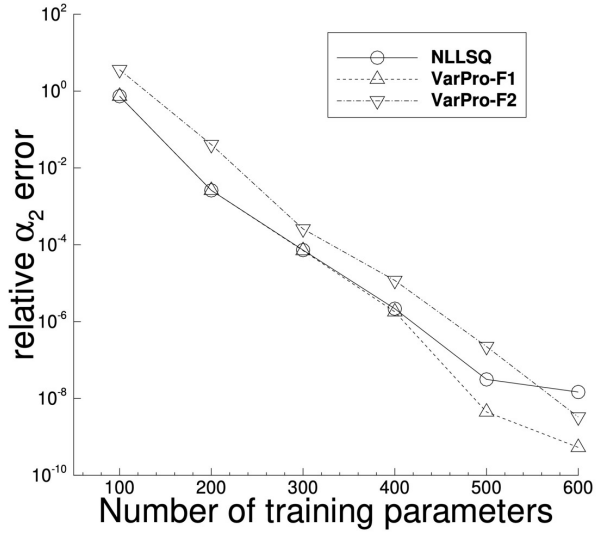



$u$ error

$\alpha_1$ error




$\alpha_2$ error




Exponential convergence w.r.t. number of training parameters

NN: [2,M,1]
30×30 collocation points
100 measurement points

# Noisy Data: Errors

NN: [2,500,1]
30×30 collocation points
50 measurement points

Method remains quite accurate with noisy data

Relative errors

| | | NLLSQ | | | VarPro-F1 | | | VarPro-F2 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\epsilon$ | $e_{\alpha_1}$ | $e_{\alpha_2}$ | $l^2$-u | $e_{\alpha_1}$ | $e_{\alpha_2}$ | $l^2$-u | $e_{\alpha_1}$ | $e_{\alpha_2}$ | $l^2$-u |
| | 0.0 | 1.99E-9 | 1.32E-7 | 1.05E-8 | 1.96E-11 | 1.76E-8 | 7.85E-9 | 7.18E-9 | 4.77E-7 | 5.95E-8 |
| | 0.001 | 4.31E-4 | 1.12E-4 | 2.46E-4 | 4.31E-4 | 1.04E-4 | 2.46E-4 | 4.31E-4 | 1.45E-4 | 2.46E-4 |
| | 0.002 | 8.62E-4 | 2.19E-4 | 4.92E-4 | 8.62E-4 | 2.40E-4 | 4.92E-4 | 8.62E-4 | 2.60E-4 | 4.92E-4 |
| | 0.005 | 2.16E-3 | 5.21E-4 | 1.23E-3 | 2.16E-3 | 5.67E-4 | 1.23E-3 | 2.16E-3 | 6.00E-4 | 1.23E-3 |
| 1% noise | 0.01 | 4.32E-3 | 9.64E-4 | 2.46E-3 | 4.32E-3 | 9.54E-4 | 2.46E-3 | 4.32E-3 | 1.26E-3 | 2.46E-3 |
| | 0.02 | 8.67E-3 | 1.41E-3 | 4.92E-3 | 8.67E-3 | 1.61E-3 | 4.92E-3 | 8.67E-3 | 1.84E-3 | 4.92E-3 |
| 5% noise | 0.05 | 2.19E-2 | 3.72E-4 | 1.23E-2 | 2.19E-2 | 6.72E-4 | 1.23E-2 | 2.19E-2 | 2.29E-3 | 1.23E-2 |
| | 0.1 | 4.44E-2 | 8.72E-3 | 2.45E-2 | 4.44E-2 | 8.77E-3 | 2.45E-2 | 4.44E-2 | 7.37E-3 | 2.45E-2 |

$\alpha_1$ error

$\alpha_2$ error

$u$ error

# Comparison with PINN

Both:
30×30 collocation points
100 random measurement points
Gaussian activation function

PINN:
NN: [2,30,30,30,30,30,30,1]
200,000 epochs
learning rate: 0.01 → 1E-4 for first 10K epochs, then fixed

Current (NLLSQ):
NN: [2,500,1]

| | noise level | method | $e_{\alpha_1}$ | $e_{\alpha_2}$ | $l^\infty$-u | $l^2$-u | training-time(sec) |
|---|---|---|---|---|---|---|---|
| No noise | $\epsilon = 0$ | PINN (Adam) | 7.08E-1 | 2.68E-1 | 1.48E+0 | 5.65E-1 | 3049.2 |
| | | current (NLLSQ) | 5.71E-9 | 3.05E-7 | 5.98E-8 | 1.49E-8 | 10.3 |
| 1% noise | $\epsilon = 0.01$ | PINN (Adam) | 6.74E-1 | 7.76E-1 | 1.56E+0 | 6.79E-1 | 2742.9 |
| | | current (NLLSQ) | 4.34E-3 | 7.13E-4 | 5.25E-3 | 2.38E-3 | 10.0 |

Current method produces more accurate results with less training time

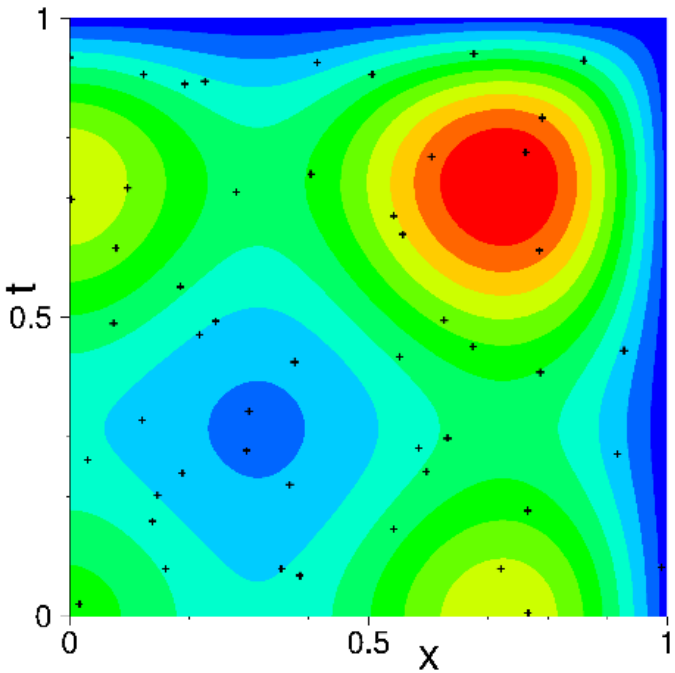
# Inverse Sine-Gordan Equation

$$\frac{\partial^2 u}{\partial t^2} - \alpha_1 \frac{\partial^2 u}{\partial x^2} + \alpha_2 u + \alpha_3 \sin(u) = f(x,t)$$
$$u(0,t) = g_1(t), \quad u(1,t) = g_2(t),$$
$$u(x,0) = g_3(x), \quad \frac{\partial u}{\partial t}(x,0) = g_4(x)$$
$$u(\xi_i, \eta_i) = S(\xi_i, \eta_i)$$

<u>Unknown:</u>

$\alpha_1,\ \alpha_2,\ \alpha_3,\ u(x,t)$

$$\alpha_1^{ex} = \alpha_2^{ex} = \alpha_3^{ex} = 1$$

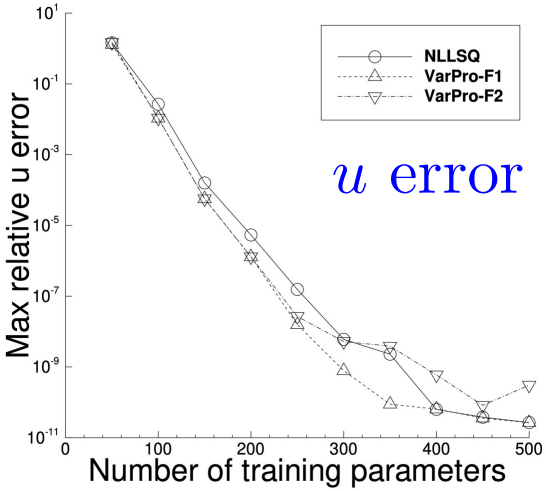



Solution field

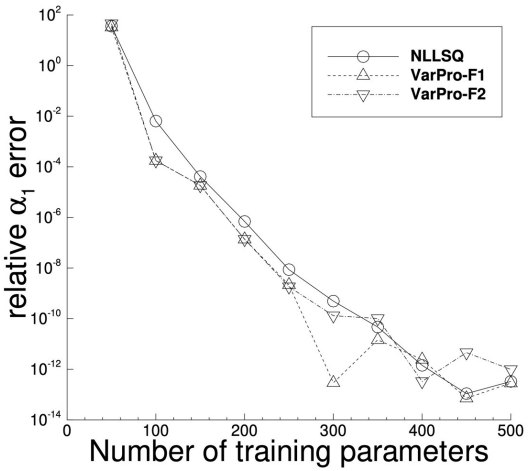
Random measurement points

# Error vs. Training Parameters

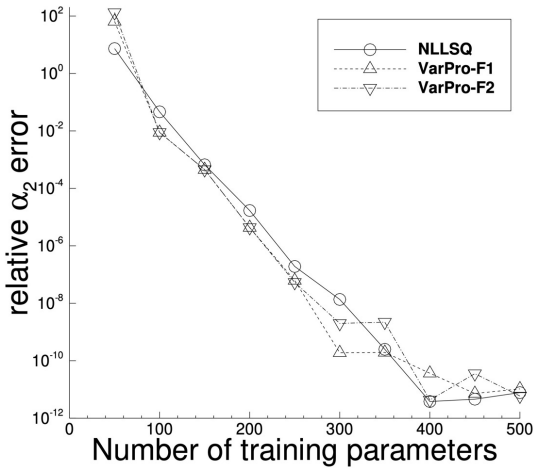
(noise-free data)



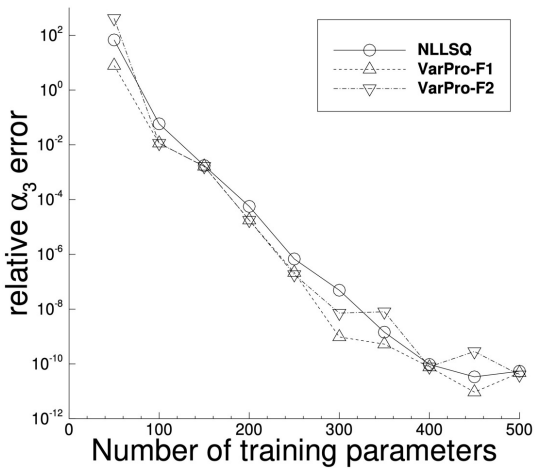

$\alpha_1$ error




$\alpha_2$ error




$\alpha_3$ error




Exponential convergence w.r.t. number of training parameters

# Noisy Data: Errors

$u$ relative errors

| $\epsilon$ | NLLSQ $l^\infty$-u | NLLSQ $l^2$-u | VarPro-F1 $l^\infty$-u | VarPro-F1 $l^2$-u | VarPro-F2 $l^\infty$-u | VarPro-F2 $l^2$-u |
|---|---|---|---|---|---|---|
| 0.0 | 3.44E-11 | 4.16E-12 | 7.03E-11 | 5.01E-12 | 7.73E-10 | 1.57E-10 |
| 0.001 | 8.76E-4 | 3.71E-4 | 8.49E-4 | 3.70E-4 | 8.51E-4 | 3.69E-4 |
| 0.005 | 4.39E-3 | 1.86E-3 | 4.26E-3 | 1.85E-3 | 4.25E-3 | 1.85E-3 |
| 1% noise 0.01 | 8.77E-3 | 3.71E-3 | 8.50E-3 | 3.70E-3 | 8.51E-3 | 3.70E-3 |
| 0.05 | 4.35E-2 | 1.87E-2 | 4.25E-2 | 1.87E-2 | 4.24E-2 | 1.86E-2 |
| 0.1 | 8.65E-2 | 3.78E-2 | 8.46E-2 | 3.77E-2 | 8.43E-2 | 3.76E-2 |
| 0.5 | 4.10E-1 | 1.95E-1 | 4.08E-1 | 1.95E-1 | 4.07E-1 | 1.95E-1 |
| 1.0 | 8.90E-1 | 3.83E-1 | 8.97E-1 | 3.84E-1 | 9.07E-1 | 3.86E-1 |

$\alpha_1$, $\alpha_2$, $\alpha_3$ relative errors

| $\epsilon$ | NLLSQ $e_{\alpha_1}$ | NLLSQ $e_{\alpha_2}$ | NLLSQ $e_{\alpha_3}$ | VarPro-F1 $e_{\alpha_1}$ | VarPro-F1 $e_{\alpha_2}$ | VarPro-F1 $e_{\alpha_3}$ | VarPro-F2 $e_{\alpha_1}$ | VarPro-F2 $e_{\alpha_2}$ | VarPro-F2 $e_{\alpha_3}$ |
|---|---|---|---|---|---|---|---|---|---|
| 0.0 | 1.93E-12 | 4.90E-11 | 1.35E-10 | 1.50E-12 | 4.42E-12 | 2.23E-11 | 3.61E-11 | 9.72E-10 | 3.02E-9 |
| 0.001 | 6.90E-4 | 2.66E-3 | 3.00E-3 | 6.88E-4 | 2.64E-3 | 3.09E-3 | 6.86E-4 | 2.63E-3 | 3.08E-3 |
| 0.005 | 3.44E-3 | 1.31E-2 | 1.45E-2 | 3.44E-3 | 1.32E-2 | 1.54E-2 | 3.43E-3 | 1.32E-2 | 1.54E-2 |
| 1% noise 0.01 | 6.88E-3 | 2.63E-2 | 2.93E-2 | 6.86E-3 | 2.63E-2 | 3.08E-2 | 6.84E-3 | 2.62E-2 | 3.04E-2 |
| 0.05 | 3.38E-2 | 1.27E-1 | 1.39E-1 | 3.38E-2 | 1.30E-1 | 1.53E-1 | 3.39E-2 | 1.32E-1 | 1.60E-1 |
| 0.1 | 6.65E-2 | 2.49E-1 | 2.76E-1 | 6.65E-2 | 2.55E-1 | 3.05E-1 | 6.65E-2 | 2.57E-1 | 3.12E-1 |
| 0.5 | 2.67E-1 | 8.07E-1 | 5.90E-1 | 2.65E-1 | 7.69E-1 | 4.39E-1 | 2.66E-1 | 7.95E-1 | 5.41E-1 |
| 1.0 | 4.09E-1 | 1.01E+0 | 2.18E-1 | 4.12E-1 | 1.07E+0 | 5.43E-1 | 4.15E-1 | 1.10E+0 | 6.27E-1 |

# Comparison with PINN

Both:
30×30 collocation points
100 random measurement points
Gaussian activation function

PINN:
NN: [2,30,30,30,30,1]
200,000 epochs
learning rate: 0.01 → 1E-4 for first 10K epochs, then fixed

Current (NLLSQ):
NN: [2,400,1]

## Relative Errors

| noise level | method | $e_{\alpha_1}$ | $e_{\alpha_2}$ | $e_{\alpha_3}$ | $l^\infty$-u | $l^2$-u | training-time(sec) |
|---|---|---|---|---|---|---|---|
| $\epsilon = 0$ | PINN (Adam) | 9.21E-3 | 2.30E-1 | 7.33E-1 | 1.86E-2 | 3.32E-3 | 1853.3 |
| | current (NLLSQ) | 7.65E-10 | 4.49E-9 | 6.60E-9 | 7.97E-10 | 3.50E-10 | 23.6 |
| $\epsilon = 0.01$ | PINN (Adam) | 1.16E-2 | 1.35E-1 | 3.51E-1 | 1.18E-2 | 2.97E-3 | 1833.2 |
| | current (NLLSQ) | 5.45E-3 | 2.59E-2 | 5.09E-3 | 5.76E-3 | 2.63E-3 | 30.1 |

Current method produces more accurate results with less training time

# Inverse Helmholtz Equation with Variable Coefficient

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} - \gamma(x,y)u = f(x,y),$$

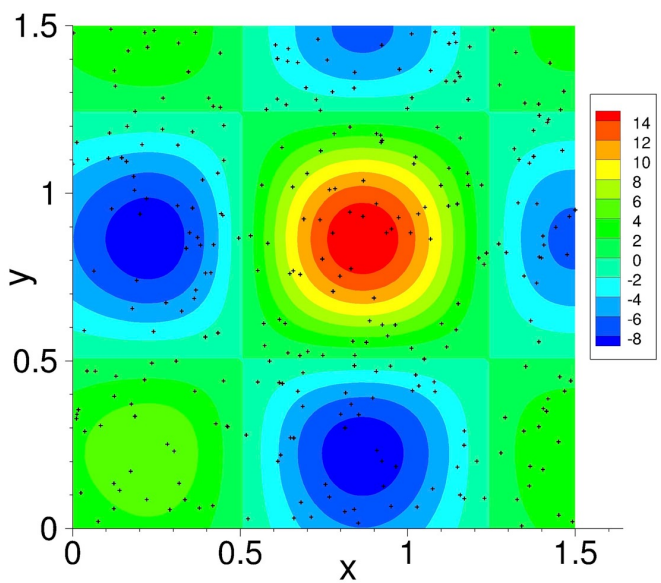
$$u(a_1, y) = g_1(y), \quad u(b_1, y) = g_2(y),$$

$$u(x, a_2) = g_3(x), \quad u(x, b_2) = g_4(x),$$

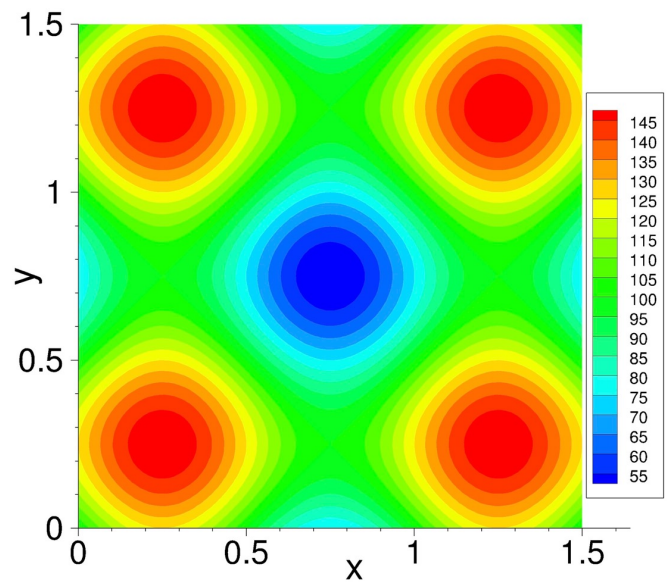
$$u(\xi_i, \eta_i) = S(\xi_i, \eta_i), \quad \text{(measurement)}$$

<u>Unknowns:</u> $\gamma(x,y)$, $u(x,y)$

solution field $u(x,y)$




coefficient field $\gamma(x,y)$



# Convergence Tests (noise-free data)

Computed by NLLSQ

| | | | $l^\infty$-$\gamma$ | $l^2$-$\gamma$ | $l^\infty$-u | $l^2$-u |
|---|---|---|---|---|---|---|
| collocation point test | $Q =$ | 5×5 | 9.84E-2 | 3.15E-2 | 7.94E-3 | 4.38E-4 |
| | | 10×10 | 4.05E-3 | 2.90E-4 | 1.63E-4 | 1.02E-5 |
| | | 15×15 | 1.15E-4 | 4.94E-6 | 3.38E-6 | 2.99E-7 |
| | | 20×20 | 4.99E-6 | 4.84E-7 | 2.65E-7 | 2.70E-8 |
| | | 25×25 | 4.42E-6 | 3.57E-7 | 3.34E-7 | 2.90E-8 |
| | | 30×30 | 1.56E-6 | 1.46E-7 | 1.19E-7 | 1.08E-8 |
| | | 35×35 | 1.61E-6 | 1.73E-7 | 1.26E-7 | 1.15E-8 |
| training parameter test | $M =$ | 50 | 6.79E+0 | 1.67E+0 | 1.63E+0 | 5.67E-1 |
| | | 100 | 8.64E-2 | 1.10E-2 | 8.44E-3 | 2.00E-3 |
| | | 200 | 1.98E-4 | 2.08E-5 | 6.46E-6 | 6.42E-7 |
| | | 300 | 2.07E-6 | 1.26E-7 | 3.13E-8 | 2.46E-9 |
| | | 400 | 5.08E-7 | 2.95E-8 | 5.76E-9 | 5.29E-10 |
| | | 500 | 1.61E-7 | 1.18E-8 | 1.95E-9 | 1.91E-10 |

Exponential convergence w.r.t. number of collocation points and number of training parameters

# Noisy Data: Errors

NN: [2,400,2]
30×30 collocation points
300 random measurement points
with regularization

## Relative errors

| | noise | level | $l^\infty$-$\gamma$ | $l^2$-$\gamma$ | $l^\infty$-u | $l^2$-u |
|---|---|---|---|---|---|---|
| NLLSQ | $\epsilon =$ | 0.0 | 1.39E-4 | 7.11E-6 | 2.57E-6 | 2.24E-7 |
| | | 0.0005 | 1.47E-2 | 1.42E-3 | 7.48E-4 | 2.17E-4 |
| | | 0.001 | 2.92E-2 | 2.85E-3 | 1.50E-3 | 4.34E-4 |
| | | 0.005 | 1.27E-1 | 1.36E-2 | 7.33E-3 | 2.17E-3 |
| | | 0.01 | 2.14E-1 | 2.58E-2 | 1.48E-2 | 4.32E-3 |
| | | 0.05 | 7.38E-1 | 1.02E-1 | 7.35E-2 | 2.14E-2 |
| | | 0.1 | 8.43E-1 | 1.84E-1 | 1.48E-1 | 4.26E-2 |
| VarPro-F1 | $\epsilon =$ | 0.0 | 3.16E-4 | 3.02E-5 | 7.75E-6 | 1.11E-6 |
| | | 0.0005 | 1.65E-2 | 1.93E-3 | 7.23E-4 | 2.18E-4 |
| | | 0.001 | 3.21E-2 | 3.81E-3 | 1.44E-3 | 4.35E-4 |
| | | 0.005 | 1.30E-1 | 1.78E-2 | 7.29E-3 | 2.16E-3 |
| | | 0.01 | 2.31E-1 | 3.34E-2 | 1.48E-2 | 4.31E-3 |
| | | 0.05 | 9.30E-1 | 1.16E-1 | 7.19E-2 | 2.14E-2 |
| | | 0.1 | 1.02E+0 | 2.08E-1 | 1.42E-1 | 4.29E-2 |
| VarPro-F2 | $\epsilon =$ | 0.0 | 5.71E-4 | 4.93E-5 | 1.28E-5 | 1.29E-6 |
| | | 0.0005 | 2.18E-2 | 2.46E-3 | 7.19E-4 | 2.23E-4 |
| | | 0.001 | 4.33E-2 | 4.90E-3 | 1.44E-3 | 4.46E-4 |
| | | 0.005 | 1.88E-1 | 2.30E-2 | 6.97E-3 | 2.21E-3 |
| | | 0.01 | 3.37E-1 | 4.42E-2 | 1.37E-2 | 4.39E-3 |
| | | 0.05 | 1.60E+0 | 1.93E-1 | 6.91E-2 | 2.18E-2 |
| | | 0.1 | 2.66E+0 | 3.11E-1 | 1.38E-1 | 4.31E-2 |

1% noise

# Summary

- Developed a method based on randomized NN for inverse PDE problems
- Presented three algorithms for training NN to solve inverse problem
  - NLLSQ,
  - VarPro-F1,
  - VarPro-F2

- For noise-free data, method exhibits spectral accuracy w.r.t. number of collocation points and number training parameters
- Noise degrades the accuracy of method. But method remains quite accurate in the presence of noise.

# References

(Inverse PDEs)

1. S. Dong & Y. Wang, "A method for computing inverse parametric PDE problems with random-weight neural networks". *Journal of Computational Physics*, accepted, 2023. (also arXiv:2210.04338)

(Local ELM for Forward PDEs)

2. S. Dong & Z. Li, "Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations". *Computer Methods in Applied Mechanics and Engineering*, 387, 114129, 2021.

3. S. Dong & J. Yang, "On computing the hyperparameter of extreme learning machines: Algorithm and application to computational PDEs, and comparison with classical and high-order finite elements". *Journal of Computational Physics*, 463, 111290, 2022.

# Nonlinear Least Squares Method

Gauss-Newton Method + Trust Region Strategy

Newton Method with an approximate Hessian matrix (retaining only the contribution from Jacobian matrix)

To improve convergence of Gauss-Newton iterations

No need to compute 2nd derivatives of residual

(Bjorck, 1996)