SAND2022-16343C

Sandia National Laboratories

Exceptional service in the national interest

Polynomial-Spline Networks with Exact Integrals and Convergence Rates

Jonas A. Actor, Andy Huang, Nat Trask
Center for Computing Research
Sandia National Laboratories

IEEE SSCI 2022, Singapore, 4-7 December 2022

U.S. DEPARTMENT OF ENERGY NNSA

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Contact: jaactor@sandia.gov

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

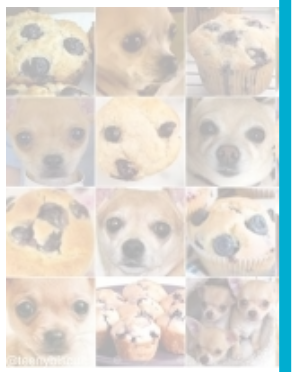
Why Scientific ML?

Traditional ML

- Lots of data
- Structured data: images, text, video
- Model invariant to translation, noise
- Goal is accuracy
- Classification, correlation, segmentation

Scientific ML

- Small amounts of data
- Unstructured data
- Model invariants of conservation laws
- Goal is accuracy, UQ, V+V
- Simulation for multiscale/multiphysics



However…

- NNs used for SciML don't have **convergence rates**, or converge only with the number of training points
- NNs used for SciML commit **variational crimes**, complicating error analysis

Images: freeCodeCamp, TensorFlow, SNL (CSandia/EIMS), SNL (CSandia/EIMS)

Goal

Construct a trainable approximation scheme that

1. Trains as well as NNs
2. Provides convergence rates
3. Maintains closed-form expressions for integrals

Polynomial-Spline Networks

$$y(x) = \sum_{\alpha=1}^{N_{cells}} \left(\sum_{\gamma=0}^{N_{splines}} w_{\alpha,\gamma} \phi_\gamma(x) \right) \left(\sum_{\beta=1}^{d_P} c_{\alpha,\beta}\, p_\beta(x) \right)$$

where

- p_β are **polynomial** basis functions of degree B_P
- ϕ_γ are **free-knot B-spline** basis functions of spline degree B_S
- $w_{\alpha,\gamma}$ are coefficients where for all γ, $\sum_\alpha w_{\alpha,\gamma} = 1$ and entrywise $w_{\alpha,\gamma} \geq 0$
- $c_{\alpha,\beta}$ are polynomial coefficients

Comparison to DNNs

DNNs as max-affine splines

Balestriero, R. “A spline theory of deep learning”, ICML, 2018.

Gupta, K., et al. “Calibration of Neural Networks using Splines”, ICLR, 2021 (poster).

DNNs as P1 finite element hat functions

He, J., et al. *ReLU Deep Neural Networks and Linear Finite Elements*, J Comp Math, 28(3): 502-527, 2020.

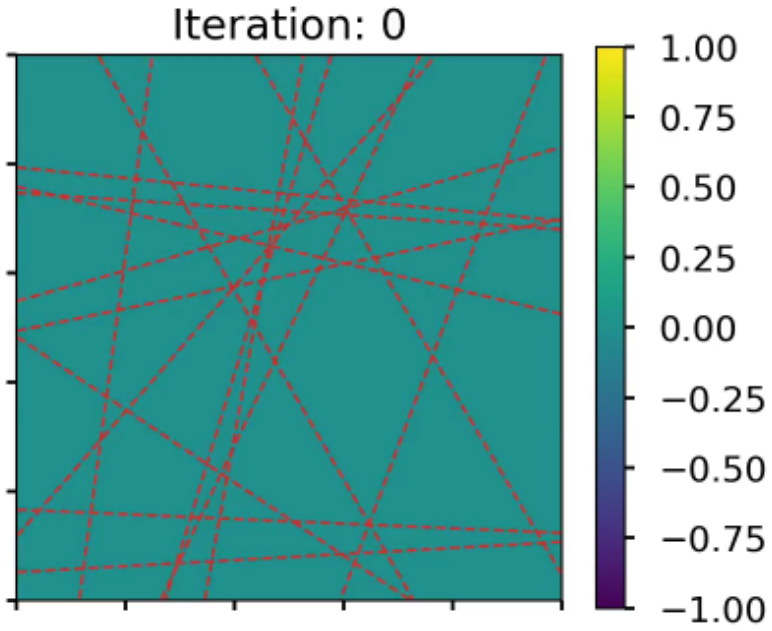
Opschoor, JA., et al. *Deep ReLU networks and high-order finite element methods*. Analysis and Applications, 18(05): 715-770, 2020.

DNNs as piecewise linear approximations

Arora, R., et al. “Understanding Deep Neural Networks with Rectified Linear Units”, COLT, 2018.

Khalife, S., and Basu, A. *Neural networks with linear threshold activations: structure and algorithms,* arXiv:2111.08117, 2021.

Comparison to DNNs



We take a shortcut:
Instead of ReLU DNN, we construct our splines explicitly and move the knots during training as in **free-spline interpolation**.

Video courtesy of Nat Trask, 01442

Relation to Free-Knot Interpolation

If $\mathrm{B_P} = 0$ and $N_{cells} = N_{splines}$,

$$y(x) = \sum_{\alpha=1}^{N_{cells}} \left(\sum_{\gamma=0}^{N_{splines}} w_{\alpha,\gamma}\phi_\gamma(x) \right)(c_\alpha) = \sum_{\alpha=1}^{N_{cells}} \sum_{\gamma=0}^{N_{splines}} c_\alpha w_{\alpha,\gamma}\phi_\gamma(x)$$

and taking $w_{\alpha,\gamma} = \delta_{\alpha,\gamma}$ yields

$$y(x) = \sum_{\alpha=1}^{N_{splines}} c_\alpha \phi_\alpha(x)$$

i.e. we perform **free-knot** spline interpolation.

Relation to Polynomial Approximation

If $N_{cells} = 1$,

$$y(x) = \left(\sum_{\gamma=0}^{N_{splines}} w_\gamma \phi_\gamma(x) \right) \left(\sum_{\beta=1}^{d_P} c_\beta \, p_\beta(x) \right) = 1 \cdot \left(\sum_{\beta=1}^{d_P} c_\beta \, p_\beta(x) \right) = \sum_{\beta=1}^{d_P} c_\beta \, p_\beta(x)$$

i.e. we perform polynomial approximation.

Model architecture guarantees at least ***hp*-convergence**, since:

- Architecture generalizes of polynomial approximation → *p*-convergence
- Architecture generalizes free-knot spline interpolation → *h*-convergence

Mixture-of-Experts Models

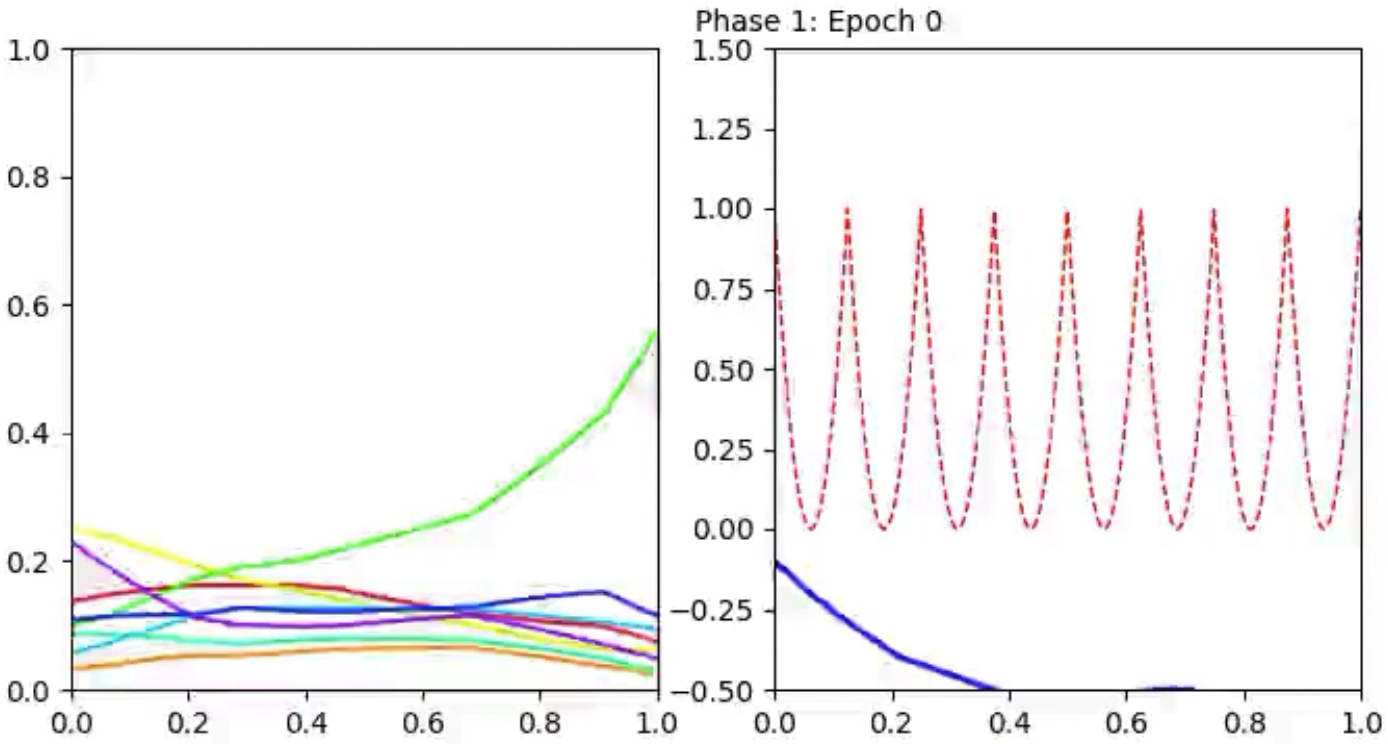
View polynomial-spline networks as a Mixture-of-Experts (MOE) model, where:

- Gating functions form a Partition of Unity (POU) from *convex combinations* of B-splines

$$\left\{ \varphi_\alpha(x) = \sum_{\gamma=1}^{N_{splines}} w_{\alpha,\gamma}\, \phi_\gamma(x) \right\}$$

- Local experts are polynomials of degree B_P

Approximation with Partitions of Unity



Left. POUs constructed using softmax, as part of a Mixture of Experts model; each expert's model is a quadratic polynomial:

$$y(x) = \sum_{\alpha} \varphi_\alpha(x)\, p_\alpha(x)$$

Right. Mixture of Experts approximation of piecewise quadratic function.

Defining our Partition of Unity

1. Define fine-scale trainable **spline knots** $\{t_\gamma\}_{\gamma=1,\dots,N_{splines}}$

2. Define B1-splines $\phi_\gamma(x) = \begin{cases} \frac{x - t_{\gamma-1}}{t_\gamma - t_{\gamma-1}} & \text{if } x \in [t_{\gamma-1}, t_\gamma] \\ 1 - \frac{x - t_\gamma}{t_{\gamma+1} - t_\gamma} & \text{if } x \in [t_\gamma, t_{\gamma+1}] \\ 0 & \text{else} \end{cases}$

3. Define trainable **convex combination matrix** $W \in \mathbb{R}^{N_{splines} \times N_{cells}}$, constrained so that

$$0 \leq w_{\alpha,\gamma} \leq 1 \quad \forall\, \alpha, \gamma$$

$$\sum_\alpha w_{\alpha,\gamma} = 1 \quad \forall\, \gamma$$

4. Define POU functions $\varphi_\alpha(x) = \sum_\gamma w_{\alpha,\gamma}\ \phi_\gamma(x)$

Exact Integration

On each cell $[t_\gamma, t_{\gamma+1}]$, the polynomial-spline network $y(x)$ is a polynomial of degree $B_S + B_P$

$$y(x) = \sum_{\alpha=1}^{N_{cells}} \left(\sum_{\gamma=0}^{N_{splines}} w_{\alpha,\gamma} \phi_\gamma(x) \right) \left(\sum_{\beta=1}^{d_P} c_{\alpha,\beta}\, p_\beta(x) \right) = \sum_k d_k\, q_i(x) \quad \text{for } x \in [t_\gamma, t_{\gamma+1}]$$

where $q_k(x) = \phi_\gamma(x) p_\beta(x)$ span the product space of Polynomial Experts × POUs.

Choose a polynomial basis and evaluate closed-form expressions for integrals! E.g. for monomial basis $q_k(x) = x^k$,

$$\int_\Omega y(x)\, dx = \sum_k d_k \int_{t_\gamma}^{t_{\gamma+1}} q_k(x)\, dx = \sum_k \frac{d_k}{k} \left(t_{\gamma+1}^k - t_\gamma^k \right).$$

Test Problems

- Regression Problems
  - (R1) $f(x) = \sin(2\pi x)$
  - (R2) $f(x) = |\sin(3\pi x^2)| + |\cos(5\pi x^2)|$
  - Loss: MSE
- Variational Problems
  - (V1) $-d^2u = 2$ on $\Omega = [0,1]$ with Dirichlet BC $u = 0$ on $\partial\Omega$
  - (V2) $-\Delta u = 0$ on $\Omega = [-1,1]^2$ with Dirichlet BC $u = g(r,\theta) = \sqrt{r}\sin\left(\frac{\theta}{2}\right)$ on $\Gamma_D = \partial\Omega \cup [0,1] \times \{0\}$
  - Loss: Euler-Lagrange Functional → **no data necessary!**

$$\text{(V1)} \quad L(u) = \int_\Omega \frac{1}{2}\|\nabla u\|^2 \, dx + \beta(u(0)^2 + u(1)^2)$$

$$\text{(V2)} \quad L(u) = \int_\Omega \frac{1}{2}\|\nabla u\|^2 \, dx + \beta \int_{\Gamma_D} \left(u - g(r,\theta)\right)^2 ds$$

Regression Formulation

Trainable Variables

- **spline knots** $\{t_\gamma\}_{\gamma=1,\ldots,N_{cells}}$
- **convex combination matrix** $W \in \mathbb{R}^{N_{splines} \times N_{cells}}$

Use LSGD[1] to obtain polynomial coefficients $c_{\alpha,\beta}$ at each training step:

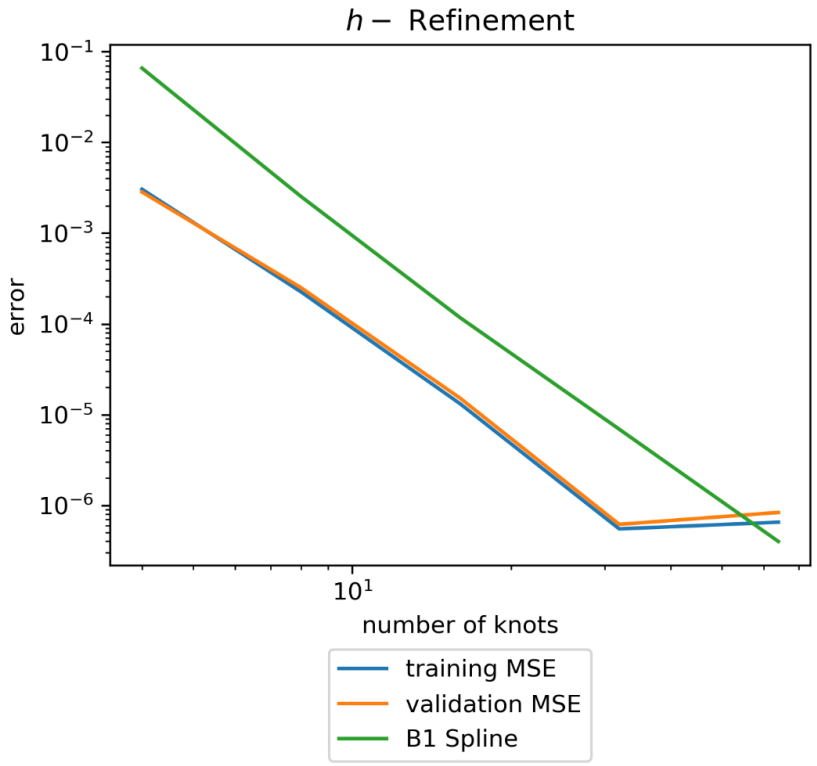
$$y(x) = c^T \Phi(x) \quad \text{where} \quad \Phi_{\alpha,\beta}(x) = \left(\sum_\gamma w_{\alpha,\gamma} \phi_\gamma(x) \right) p_{\beta(x)}$$

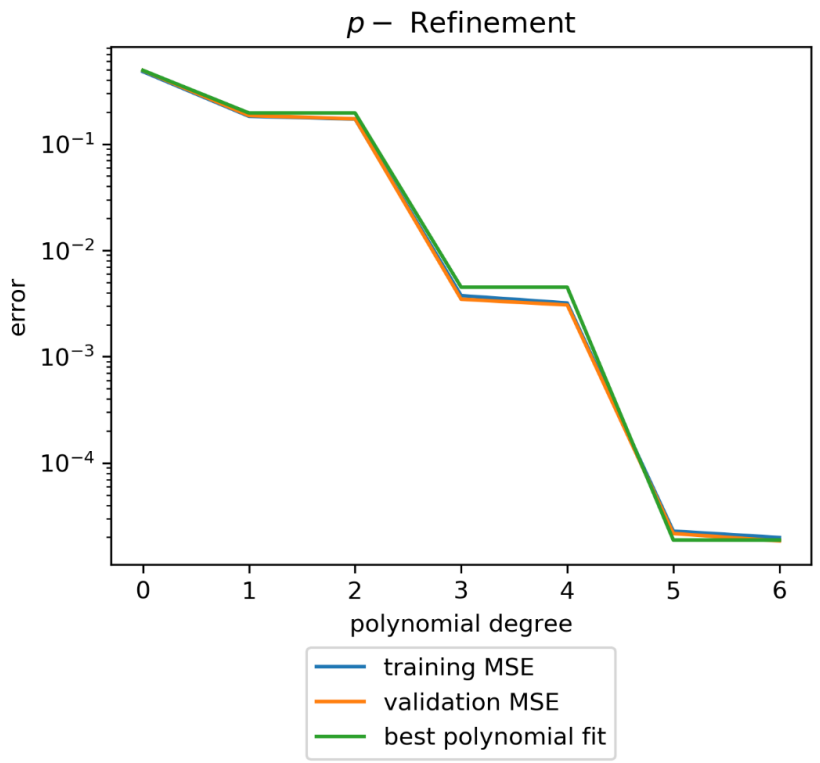
Then solve least-squares problem for c :

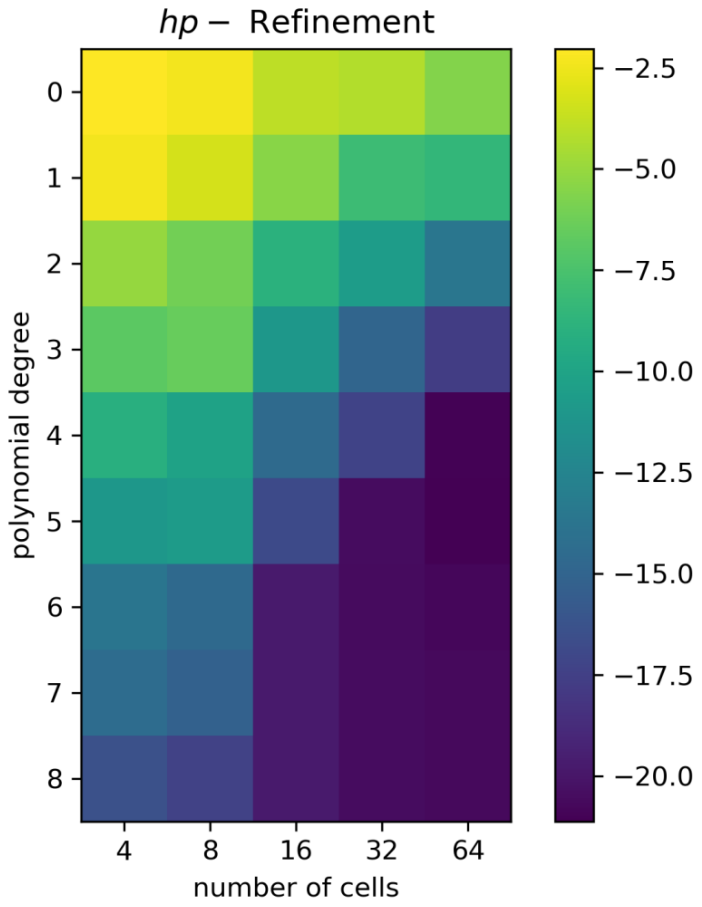
$$c \leftarrow \min_c \| y - c^T \Phi \|_F^2$$

1. Cyr, et al., “Robust training and initialization of deep neural networks: An adaptive basis viewpoint,” MSML, 2020.

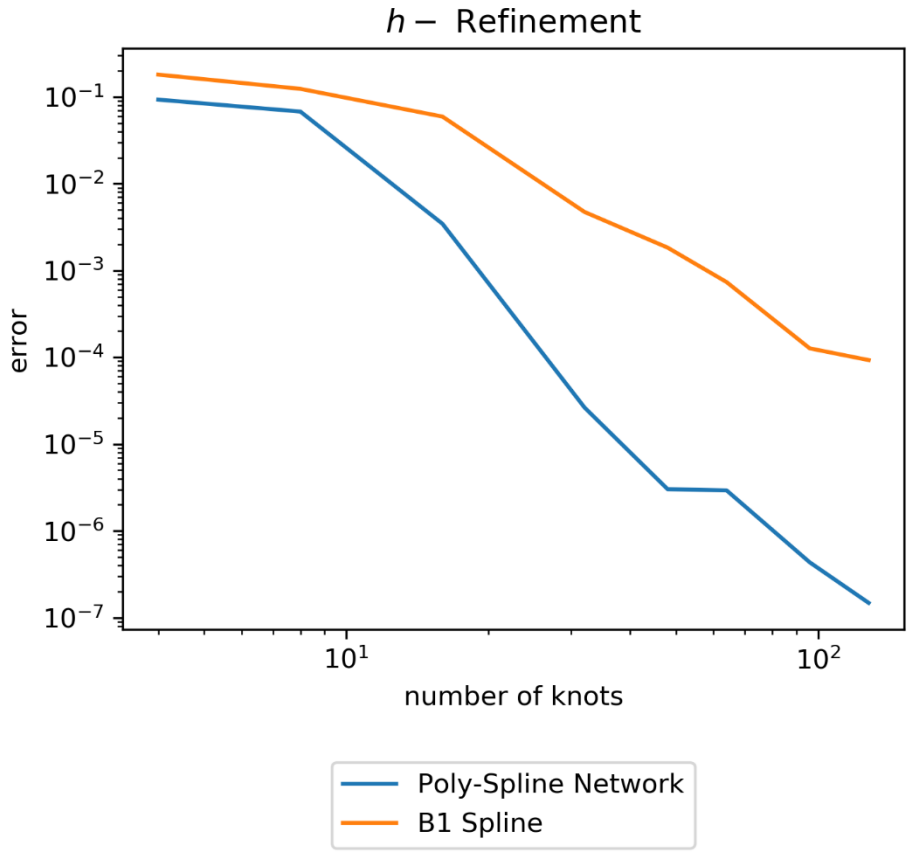
(R1) Smooth Regression Problem

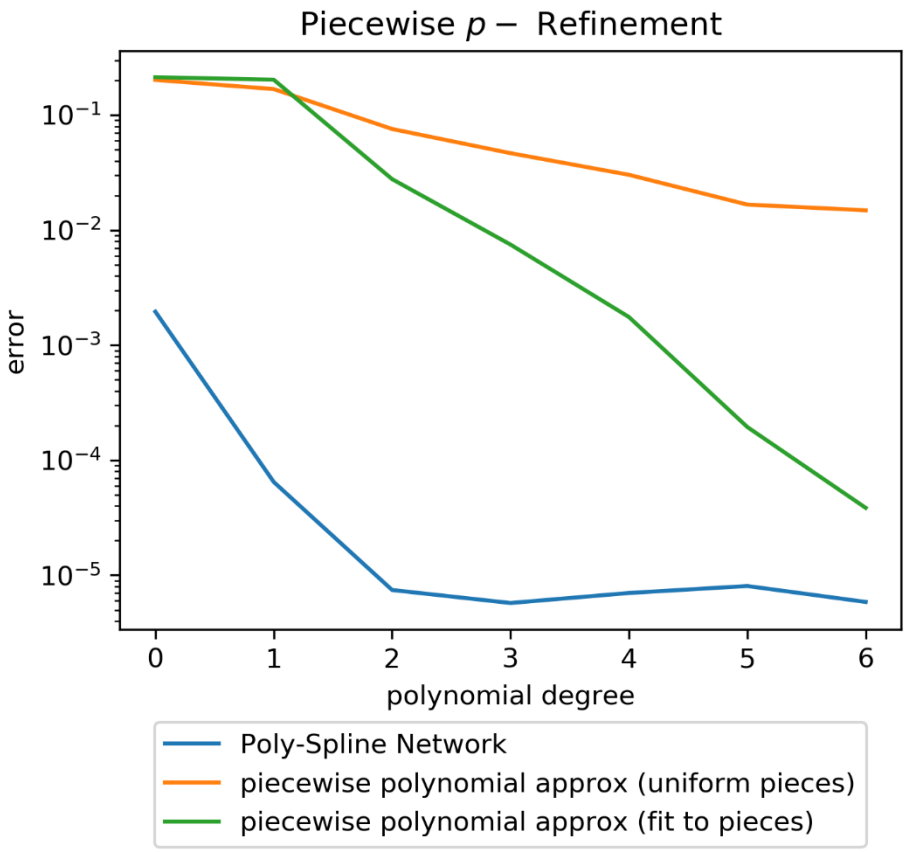






(R2) Piecewise Regression Problem





Variational Formulation

Trainable Variables

- **spline knots** $\{t_\gamma\}_{\gamma=1,\dots,N_{cells}}$
- **convex combination matrix** $W \in \mathbb{R}^{N_{splines} \times N_{cells}}$

Use LSGD to obtain polynomial coefficients $c_{\alpha,\beta}$ at each training step:

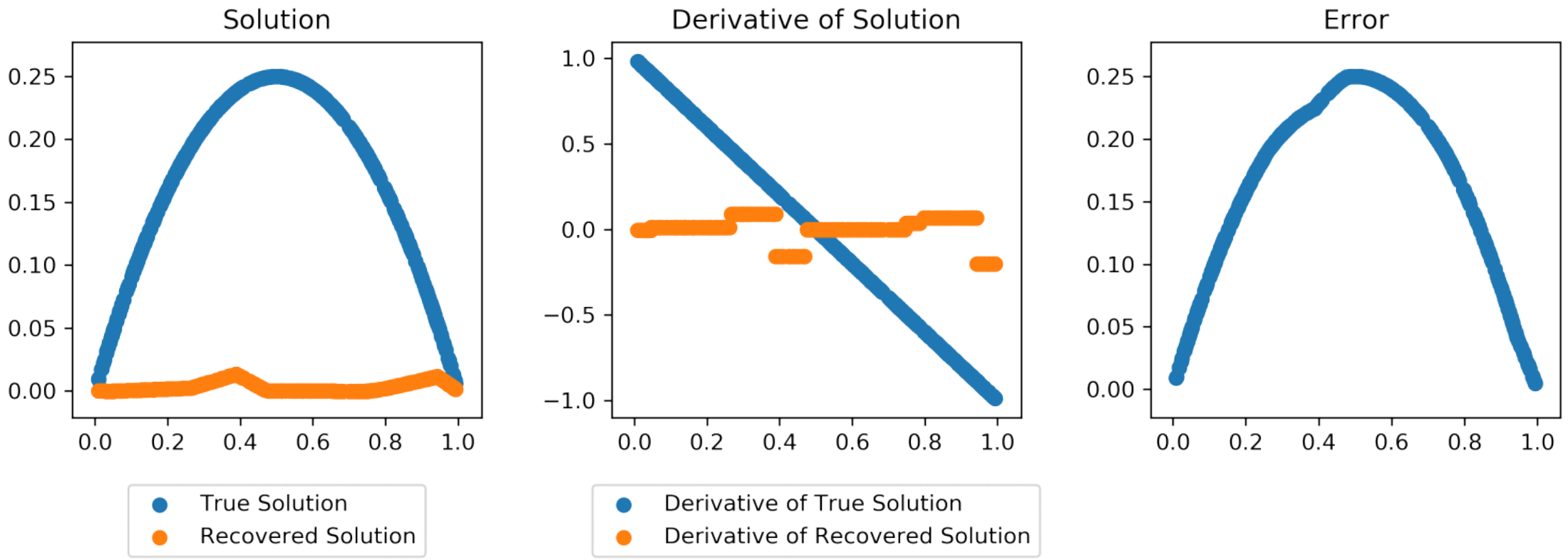
$$u(x) = c^T \Phi(x) \quad \text{where} \quad \Phi_{\alpha,\beta}(x) = \left(\sum_\gamma w_{\alpha,\gamma} \phi_\gamma(x) \right) p_{\beta(x)}$$

Then solve linear system for $c = A^{-1} b$, where e.g. for (V2), we have

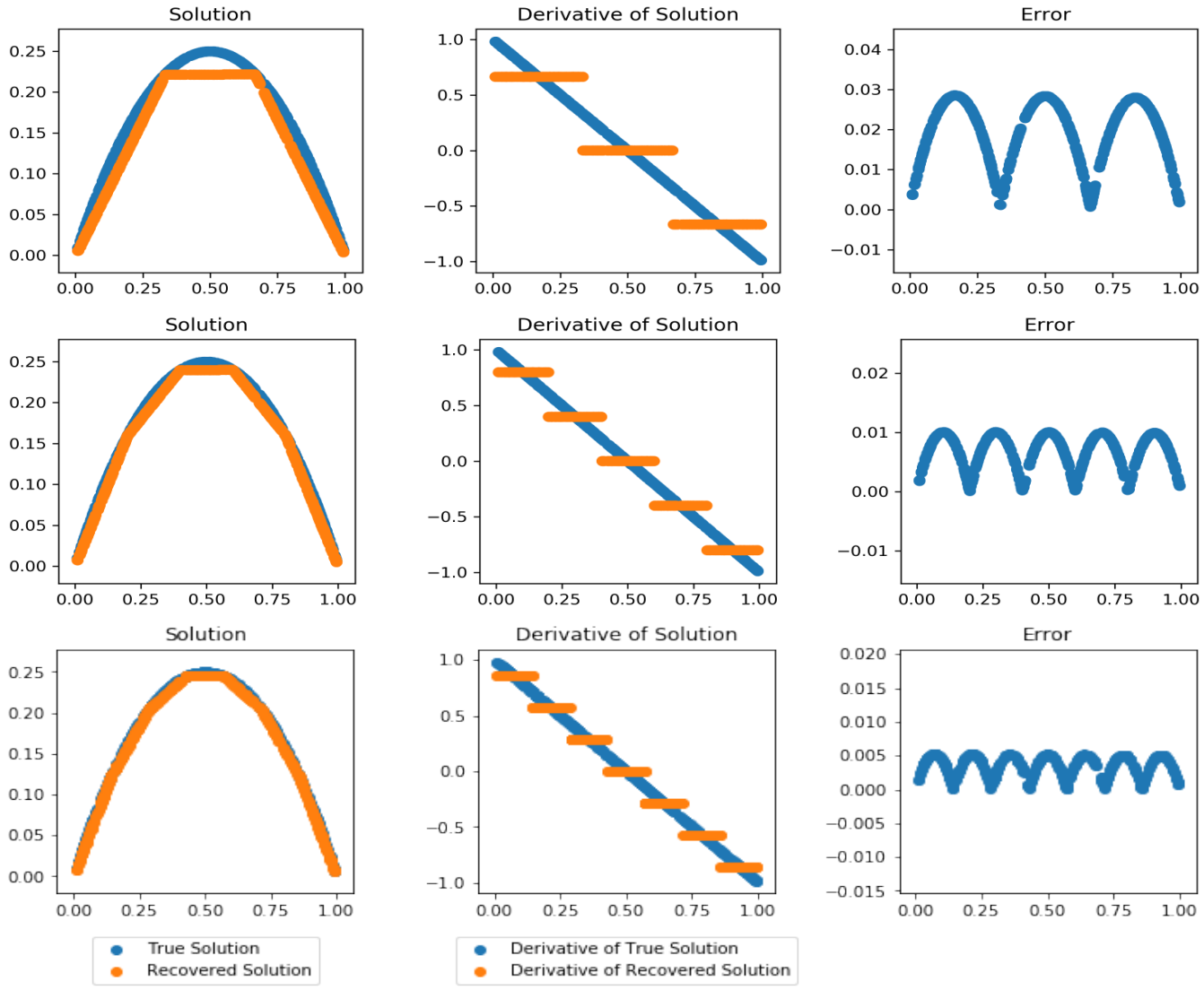
$$A = \int_\Omega \nabla\Phi \, \nabla\Phi^{\mathrm{T}} \, dx + \beta \int_{\partial\Omega} \Phi\Phi^T \, ds$$

$$b = \int g \, \Phi \, ds$$

(V1) Smooth Variational Problem



(V1) Smooth Variational Problem



Maintains *h*-refinement with number of splines, POUs, just like for the regression problem!

(V2) Nonsmooth Variational Problem

Method	Mesh Type	# Cells	Solve Size	L_2 Error
Poly-Spline Network	Adaptive	16	48×48	0.0262
FEM U3	Uniform	18	16×16	0.0362
FEM U6	Uniform	72	49×49	0.0231
FEM A	Adaptive	120	72×72	0.0242

- More accurate than FEM on uniform mesh with same size linear system
- Similar accuracy to FEM on uniform mesh with same number of basis elements
- Much smaller linear system than FEM solution using adaptive mesh refinement to obtain comparable accuracy

Moving Forward: SciML Applications

- Learn chain complexes, finite element exterior calculus operators
  - Structure-preserving NNs for inverse problems
  - SciML models for conservation laws, Maxwell's Equations without sampling error, inexact quadrature
- Build efficient formulations for higher order splines for POU construction
- Use VAEs to learn PDE solutions on low-dimensional manifold embeddings

Further reading

- Actor, JA., Huang, A., Trask, N. "Polynomial-Spline Neural Networks with Exact Integrals." *arXiv:*2110.14055, 2021.
- Actor, JA., et al. "Data-Driven Whitney Forms for Structure-Preserving Control Volume Analysis." https://www.researchgate.net/publication/364325736_Data-Driven_Whitney_Forms_for_Structure-Preserving_Control_Volume_Analysis, 2022.