MURI Presentation
Tuesday, February 1, 2022

COMPUTATIONAL APPLIED SCIENCE LABORATORY

Machine-Learning Tools for Curvature Computation in the Level-Set Method

Luis Ángel Larios Cárdenas

Advisor
Prof. Frédéric Gibou

Outline

- Fundamentals
  - The level-set method.
- Curvature estimation
  - Importance.
  - Numerical improvements.
  - Preliminary machine learning research.
- Error-modeling schemes
  - Error-correcting neural networks.
- Future work

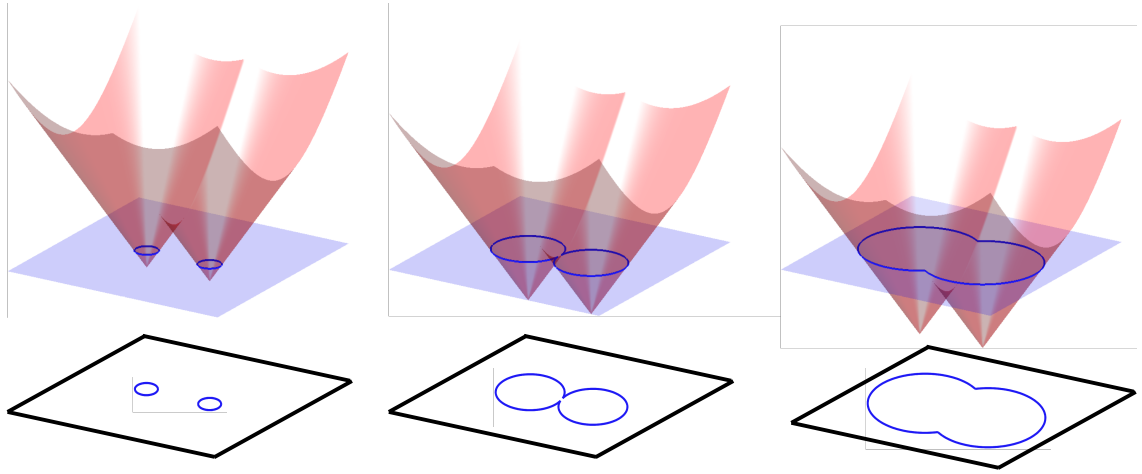
Fundamentals

Fundamentals

- The level-set method (LSM) is an implicit formulation for advecting an interface, Γ, captured as the **zero isocontour** of the **level-set (LS) function**, $\phi(\mathbf{x})$: $\mathbb{R}^n \rightarrow \mathbb{R}$, [Osher and Sethian 88] [Osher and Fedkiw 02][Gibou *et al.* 18].

- $\phi(\mathbf{x})$ is usually the signed normal distance function to the interface Γ:

$$\phi(\mathbf{x}) = \begin{cases} -d, & \mathbf{x} \in \Omega^-, \\ +d, & \mathbf{x} \in \Omega^+, \\ 0, & \mathbf{x} \in \Gamma. \end{cases}$$



LS representation of a free boundary in 2D
(Courtesy of [Gibou *et al.* 18]).

- If $\mathbf{v}(\mathbf{x})$ is a velocity field defined for all $\mathbf{x} \in \Omega$, the evolution of $\phi(\mathbf{x})$ satisfies the Hamilton-Jacobi **level-set equation**:

$$\phi_t + \mathbf{v} \cdot \nabla\phi = 0$$

Fundamentals

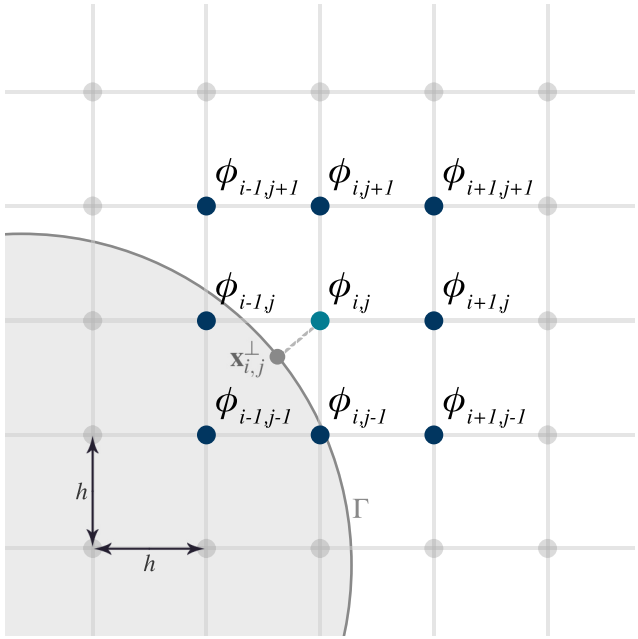
- It is straightforward to compute normal vectors and mean curvature [Min and Gibou 07][Du Chéné *et al.* 08]:

$$\hat{\mathbf{n}} = \frac{\nabla\phi}{\|\nabla\phi\|}, \qquad \kappa = \nabla \cdot \frac{\nabla\phi}{\|\nabla\phi\|}$$

- To maintain the signed-distance property, we **reinitialize** $\phi(\mathbf{x})$ periodically by solving the **reinitialization equation** [Sussman *et al.* 94] to steady state (or up to ν iterations):

$$\phi_\tau + S(\phi^0)(\|\nabla\phi\| - 1) = 0$$

- Keeping $\phi(\mathbf{x})$ as a signed distance function is important because:
  - $\|\nabla\phi\| = 1$ simplifies computations and helps maintain the physical thickness of Γ in multiphase flow simulations [Salih and Ghosh 13],
  - Removes noisy features that might get amplified when computing $\partial\phi/\partial x_i$, and
  - Removes unstructured patterns that affect the performance and training of deep learning models [LaLC and Gibou 21a & 21b].



Stencil of vertices and level-set values used in the numerical computation of curvature at node (i, j) next to Γ.

Curvature Estimation

Why Is Curvature Important?

- Curvature is a fundamental attribute in FBPs for its relation to surface tension, $\mathbf{f}_\sigma$, in physics and its regularization property in optimization.

Navier-Stokes equations for incompressible flow

$$\partial_t \rho + \mathbf{u} \cdot \nabla \rho = 0$$
$$\rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}) = \nabla \cdot [\mu(\nabla \mathbf{u} + \nabla^T \mathbf{u})] - \nabla p + \mathbf{f}_\sigma$$
$$\nabla \cdot \mathbf{u} = 0$$

Setting $\mathbf{u} = \mathbf{0}$

Equilibrium condition

$$-\nabla p + \sigma \kappa \mathbf{n} \delta_s = 0,$$

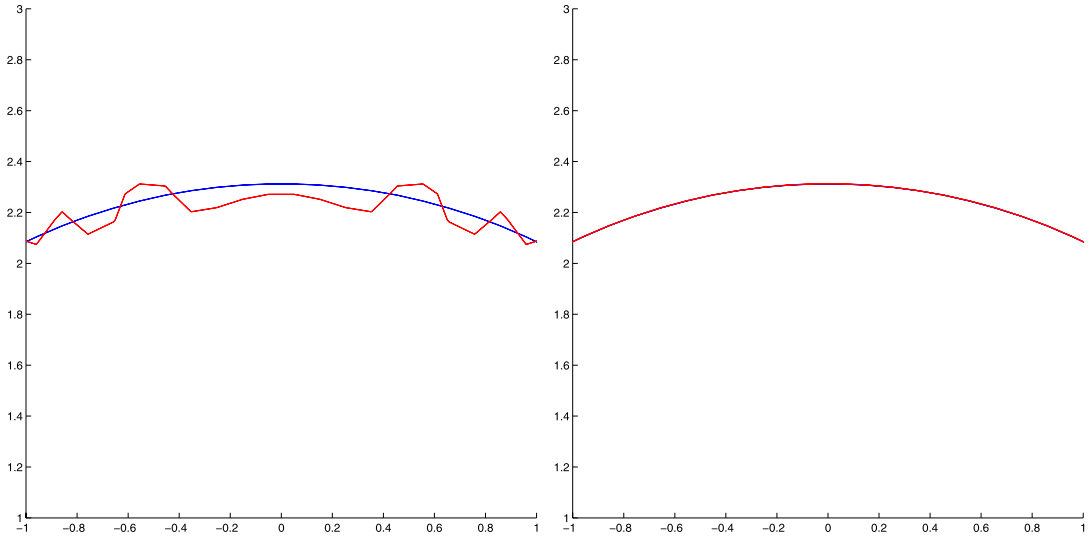
which is verified for

$$[p] = \sigma\kappa, \qquad \kappa = constant$$

- It is critical to approximate curvature accurately ***at*** the interface ($\phi = 0$)
  - To recover the continuous equilibrium solution [Popinet 18], and
  - To avoid erroneous pressure jumps, $[p]$, that affect breakup and coalescence [Lervåg 14].
- Computing κ accurately in LSM becomes challenging
  - When $\phi(\mathbf{x})$ develops kinks close to Γ, or
  - Along steep portions in poorly resolved regions.

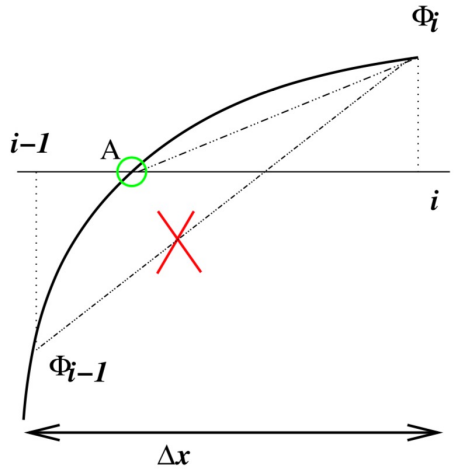
Accurate Numerical Curvature Computation

- To compute the second-order accurate derivatives used for curvature, $\phi(\mathbf{x})$ must be guaranteed to have some smoothness [Du Chéné *et al.* 08][Popinet 18].
- Reinitialization recovers the signed distance property and smooths out $\phi(\mathbf{x})$ for downstream calculations.
  - We must consider Γ's location and the fact that information propagates along the characteristics, in Γ's normal direction.



Standard schemes
Approximating $D^{\pm}$ with (W)ENO schemes from [Jiang and Peng 00].

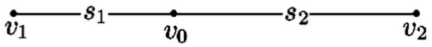
Fixed schemes
Sub-cell correction of [Russo and Smereka 00].



Using Γ's location (A) explicitly in the approximation of one-sided derivatives (Courtesy of [Du Chéné *et al.* 08]).

Second-order accurate reinitialized LS functions [Russo and Smereka 00]:

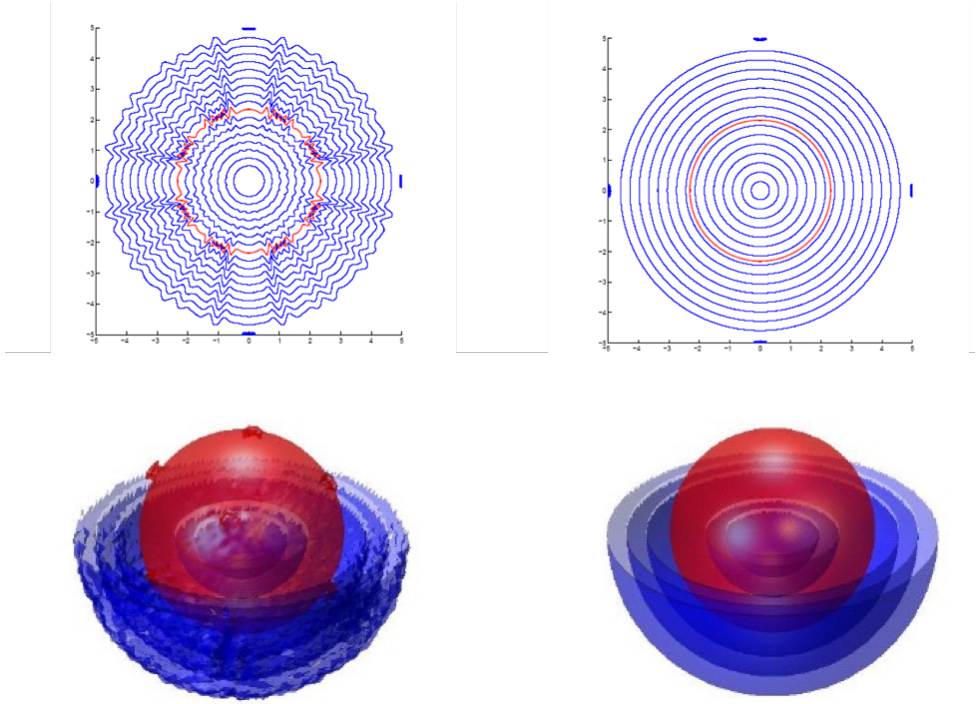
$$D_x^+\phi_0^n = \frac{0-\phi_0^n}{s_I} - \frac{s_I}{2}\,\mathrm{minmod}(D_{xx}^0\phi_0^n, D_{xx}^0\phi_2^n)$$



Accurate Numerical Curvature Computation

- Reinitialization affects not only mass preservation but also curvature accuracy.

Curvature isosurfaces in 2 and 3 dimensions.



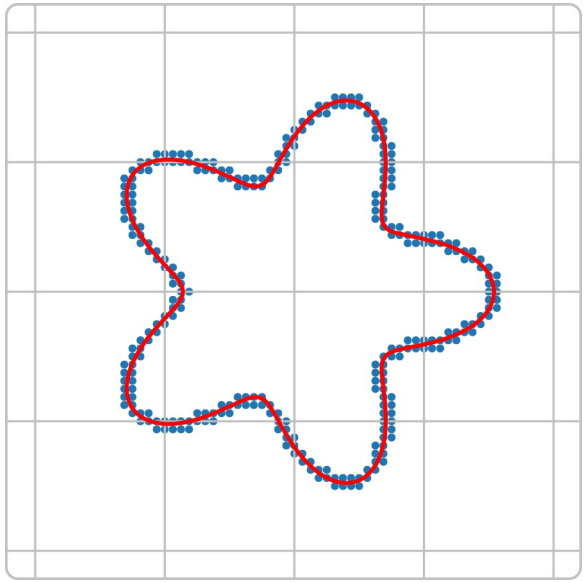
Standard schemes of [Jiang and Peng oo].

Fourth-order accurate reinitialization of [Du Chéné *et al.* o8].

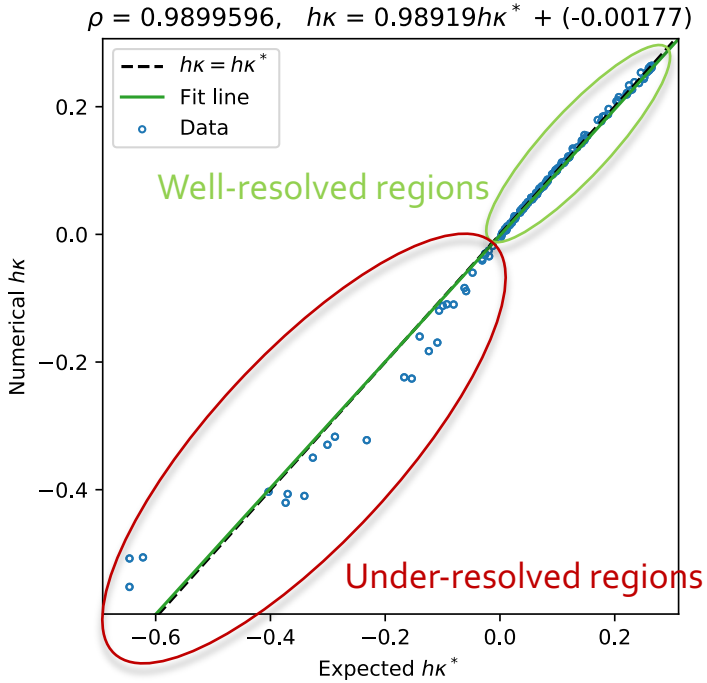
- Even second-order accurate reinitialization is prone to producing noisy curvature values.
- For this reason, [Du Chéné *et al.* o8] extended [Russo and Smereka oo]'s idea and designed fourth-order accurate reinitialization schemes and obtained ***second-order accurate mean curvature***.

Standing Challenges in Curvature Computation

- Despite these advancements:
  - Poorly resolved regions make it difficult to estimate large curvature values.
  - It may be difficult/expensive to build higher-order WENO schemes for numerical approximation in adaptive grids.



Under-resolved, concave interface regions in an irregular interface



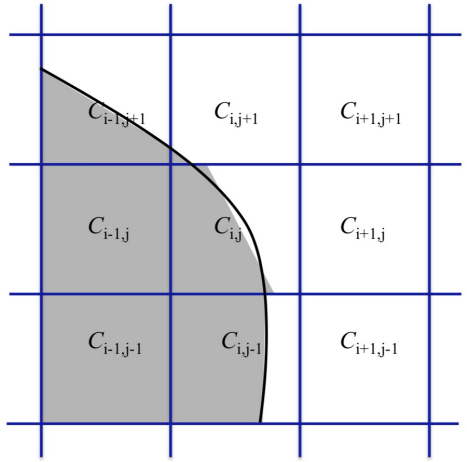
Dimensionless curvature bilinearly interpolated at the interface.

Preliminary Research: **A Neural-Network-only Approach to Estimate κ**

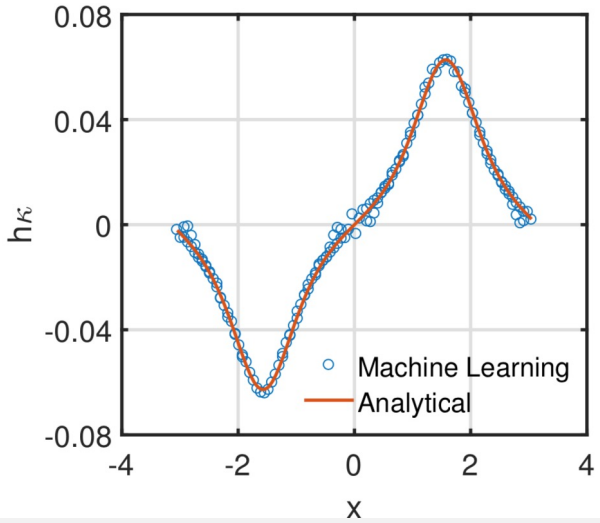
- Inspired by the neural network (NN) of [Qi *et al.* 19] to estimate $h\kappa$ in VOF:

$$h\kappa_{i,j} \approx f_{VOF}\left(\begin{bmatrix} C_{i-1,j+1} & C_{i,j+1} & C_{i+1,j+1} \\ C_{i-1,j} & C_{i,j} & C_{i+1,j} \\ C_{i-1,j-1} & C_{i,j-1} & C_{i+1,j-1} \end{bmatrix}\right)$$

  - Trained a *two-layered model* on samples from **circles** of varying sizes.
  - Tested $f_{VOF}(\cdot)$ on a static sine curve and in a surface-tension-driven oscillating-drop simulation with *reasonably good results*.

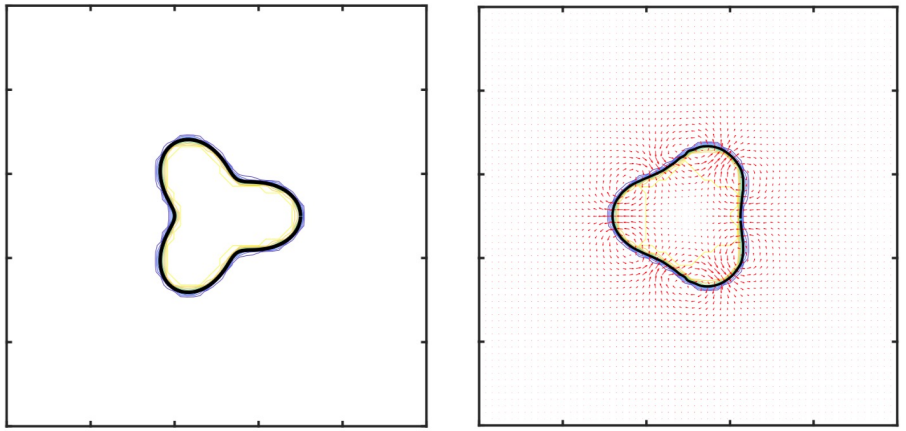


Interface cells and volume fractions constructed by the PLIC method.



Comparing exact versus ML curvature estimation for a sinusoidal curve.
(All figures are courtesy of [Qi *et al.* 19]).

Flow simulation: initial conditions and state at $t = 0.02$.



Preliminary Research: **A Neural-Network-only Approach to Estimate κ**

- We adapted [Qi *et al.* 19]'s idea to estimate $h\kappa$ in LSM [LaLC and Gibou 21a]:

$$h\kappa_{i,j} \approx f_{LSM}\left(\begin{bmatrix} \phi_{i-1,j+1} & \phi_{i,j+1} & \phi_{i+1,j+1} \\ \phi_{i-1,j} & \phi_{i,j} & \phi_{i+1,j} \\ \phi_{i-1,j-1} & \phi_{i,j-1} & \phi_{i+1,j-1} \end{bmatrix}\right)$$

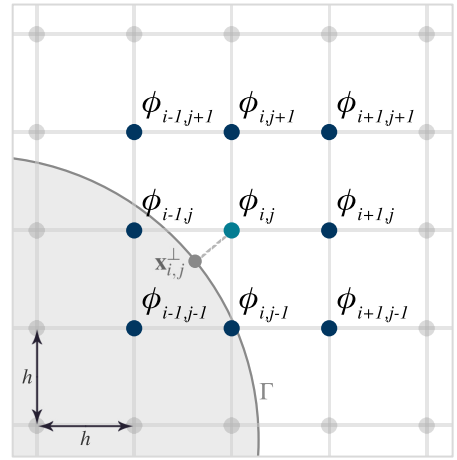
  - Trained *five-layered NNs* on samples of the form $(\boldsymbol{\phi}, h\kappa^*) \in \mathbb{R}^{10}$ from **circles** with $r \in [1.6h, 0.5 - 2h]$ uniformly spaced out with N_r values:

$$N_r(\varrho) = \left\lceil \frac{\varrho - 8.2}{2} \right\rceil + 1, \quad \varrho = 256, 266, 276 \text{ (← nodes per unit length)}$$

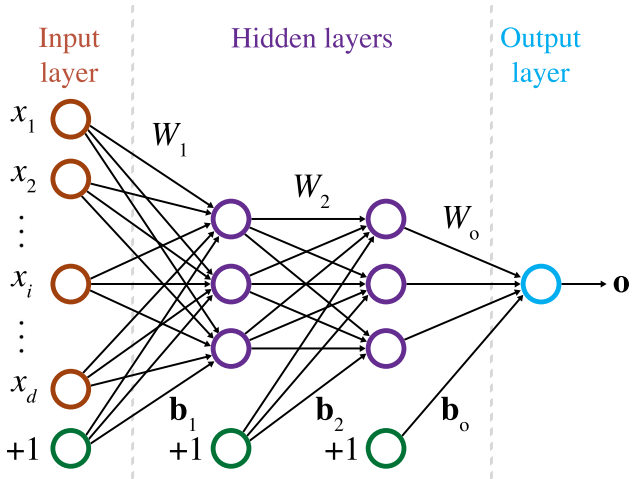
  - **One model per grid resolution**, optimized for exact signed-distance and reinitialized LS functions:

$$\phi_c^{sdf}(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_c\| - r, \qquad \phi_c^{rls}(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_c\|^2 - r^2.$$

  - Built $\mathcal{D}$ with tuples from $\phi_c^{rls}(\mathbf{x})$ with $\nu = 5, 10, 15, 20$ redistancing steps.
  - Randomized $\mathbf{x}_c$ five times for each r.
  - Used our parallel adaptive LSM implementation [Min and Gibou 07][Mirzadeh *et al.* 16].
  - Split $\mathcal{D}$ into *training* (70%), *testing* (15%), and *validation* (15%) subsets.



Nine-point-stencil LS values and $\mathbf{x}^{\perp}_{i,j}$, where $h\kappa_{i,j}$ is computed.



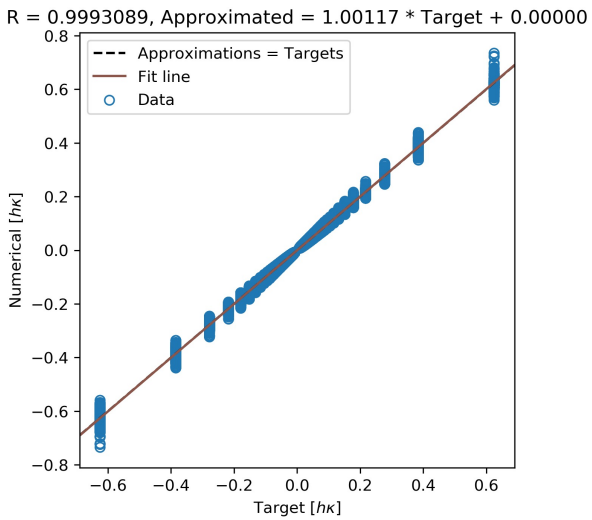
Example three-layered NN.

Preliminary Research: **A Neural-Network-only Approach to Estimate κ**

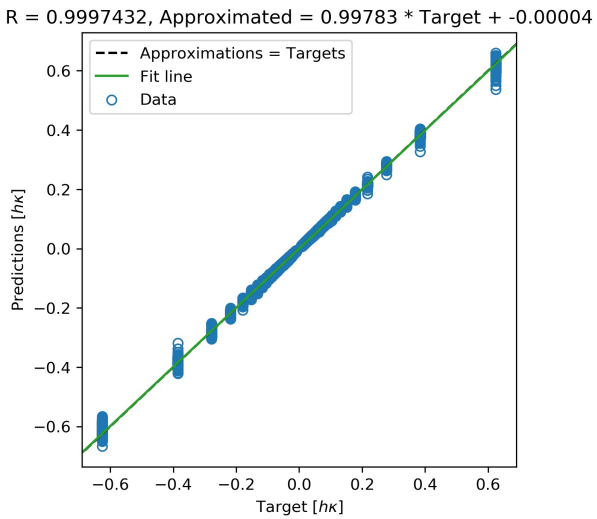
- We used **TensorFlow** and **Keras** to design and train our simple multilayer perceptrons (MLPs) with
  - *Z-scores* to normalize $\boldsymbol{\phi}$ with SciKit-Learn's `StandardScaler` class.
  - 140 ***ReLU*** neurons in each hidden layer for $h = 1/265$.
  - *MSE* loss function for optimization.
  - *Adam* optimizer with a learning rate of 0.00015.
  - *Early-stopping*, which monitored the validation *MAE* (30-epoch patience).

Standardization or z-scoring

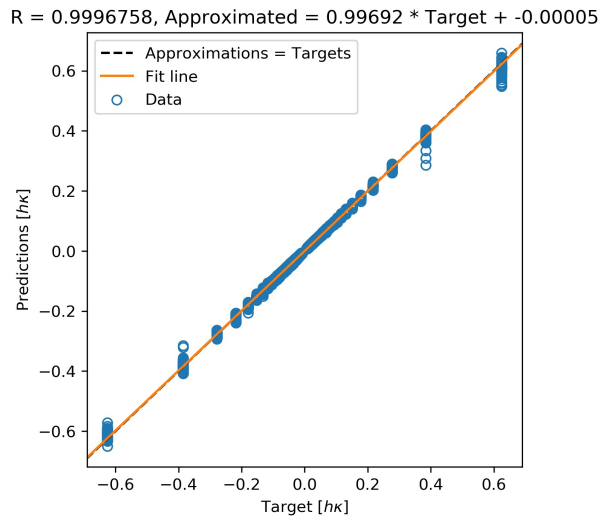
$$\psi_i \leftarrow \frac{\psi_i - \mu_\psi}{\sigma_\psi}$$



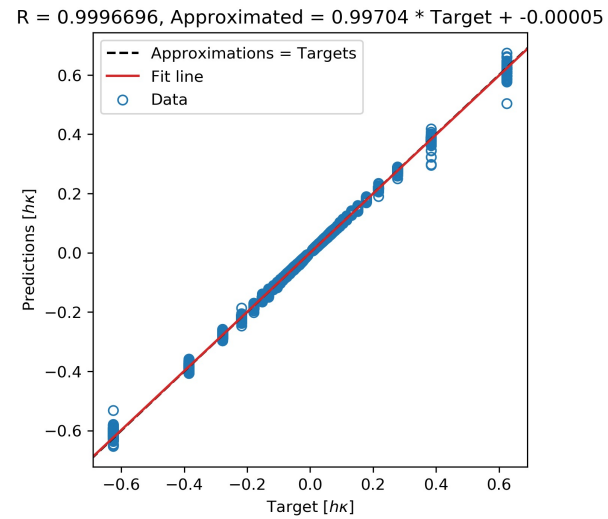
(a) Numerical method on full $\mathcal{D}$.



(b) $f_{LSM}(\cdot)$ on training subset.



(c) $f_{LSM}(\cdot)$ on testing subset.



(d) $f_{LSM}(\cdot)$ on validation subset.

Fit quality for our MLP trained for $h = 1/265$. Agreement between numerical and neural solutions.

Preliminary Research: **A Neural-Network-only Approach to Estimate κ**

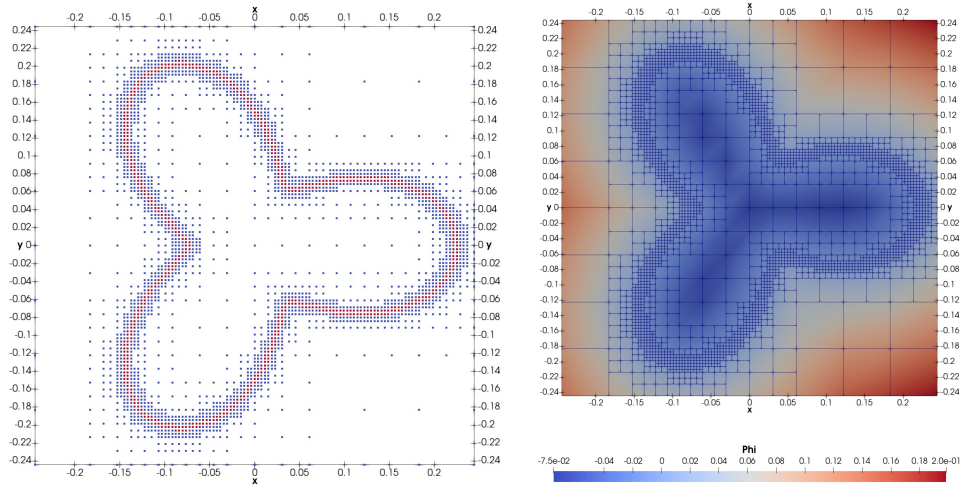
Testing $f_{LSM}(\cdot)$ trained for $h = 1/265$ on the *polar-rose* LS function

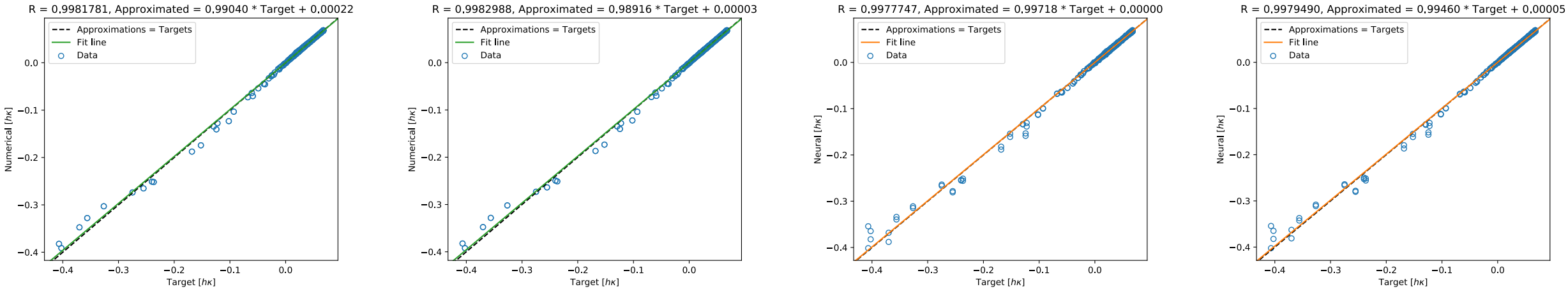
$$\phi_{rose}(\mathbf{x}) = \|\mathbf{x}\| - a\cos(p\theta) - b\,,$$

whose zero-isocontour is defined by

$$\gamma(\theta) = a\cos(p\theta) - b\,.$$

We have chosen $p = 3$, $a = 0.075$, and $b = 0.15$ in a domain discretized with one ***quadtree*** with $\eta = 7$ max level of refinement.





(a) Numerical at 10 iterations. (b) Numerical at 20 iterations. (c) $f_{LSM}(\cdot)$ at 10 iterations. (d) $f_{LSM}(\cdot)$ at 20 iterations.

Fit quality for the numerical- and $f_{LSM}(\cdot)$-based methods.

Preliminary Research: **A Hybrid Inference System to Estimate κ**

- *Can we do better than a network-only approach?*

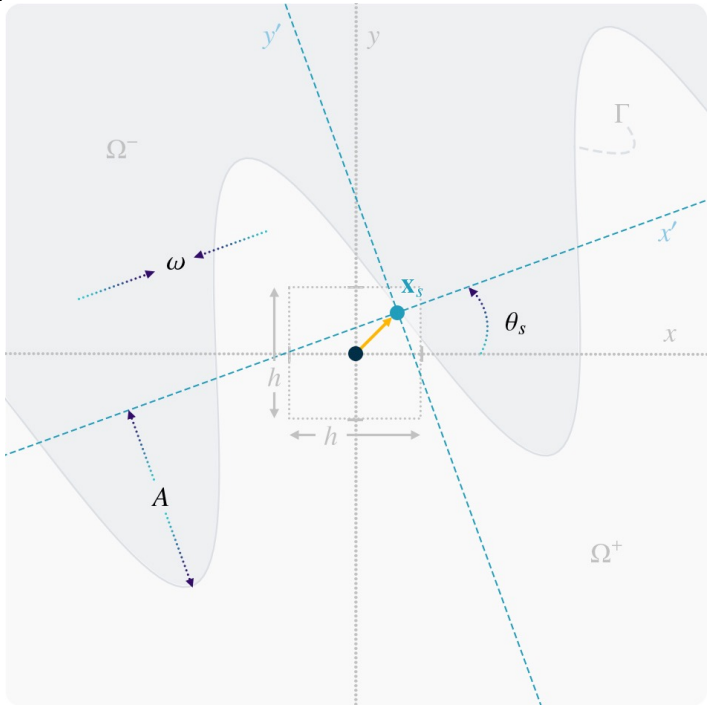
  Yes! By hybridization.

- First, we have presented **enhanced NNs** [LaLC and Gibou 21b], $F_h(\cdot)\colon \mathbb{R}^9 \mapsto \mathbb{R}$, that output $h\kappa$ for smooth two-dimensional interfaces, given a preprocessed nine-point stencil, $\boldsymbol{\phi}$, of LS values.

  - Added **affine-transformed sinusoidal interfaces** for training,

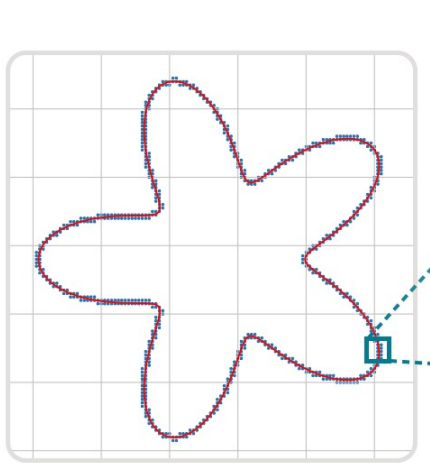
    $$f_s(t) = A\sin(\omega t)\,,$$

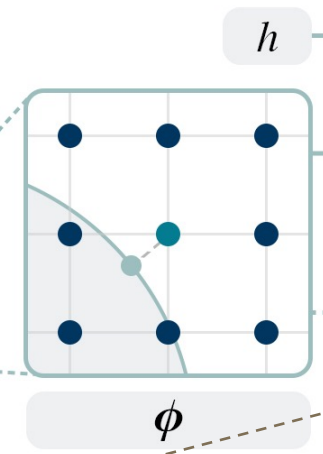
    with their corresponding $\phi_s^{sdf}(x)$ and $\phi_s^{rls}(x)$ LS functions. Varying $f_s(t)$ combinatorically in A, ω, and θ_s with random translations ($\mathbf{x}_s$).
    - Ease-off and histogram-based probabilistic subsampling.
    - Data augmentation by rotating stencils by $90°$ three times.
  - Reduced degrees of freedom by considering only ***half the curvature spectrum***. Exploited **problem's intrinsic symmetry**.
  - Used $-\kappa^* \in [0.5, 85\frac{1}{3}]$ for all grid resolutions.
  - Redistributed circular interfaces uniformly on κ^* instead of their radii.
  - Redistanced both $\phi_s^{rls}(\mathbf{x})$ and $\phi_c^{rls}(\mathbf{x})$ only with $\nu = 10$ iterations.
  - Preprocessed $\boldsymbol{\phi}$ with **PCA and whitening**.

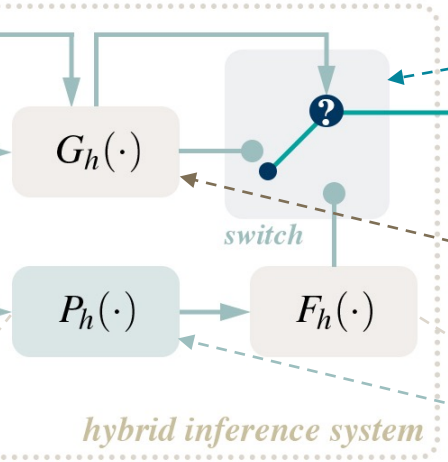


Transforming $f_s(t)$ by $T(x_s)$ and $R(\theta_s)$.
Domain partitioned into Ω^- and Ω^+ regions.

Preliminary Research: **A Hybrid Inference System to Estimate κ**







Decides whether we should keep $G_h(\cdot)$'s estimation or use ML

Computes $h\kappa$ at the nodes and linearly interpolates it at

$$\mathbf{x}_{i,j}^{\Gamma} = \mathbf{x}_{i,j} - \phi_{i,j} \frac{\nabla\phi(\mathbf{x}_{i,j})}{\|\nabla\phi(\mathbf{x}_{i,j})\|} \approx \mathbf{x}_{i,j}^{\perp}$$

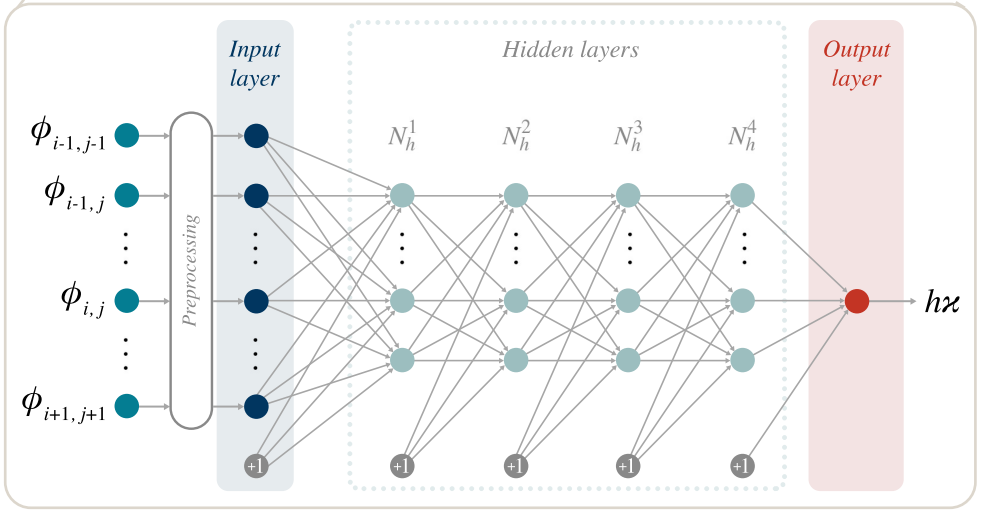
Performs PCA and whitening

Hybrid inference system

$h\kappa \leftarrow G_h(\mathcal{G}, \widehat{N}, \boldsymbol{\phi})$;
if $|h\kappa| < h\kappa_{flat}$ **then**
  return $h\kappa$;
else
  $\varsigma \leftarrow \text{sign}(h\kappa)$;
  if ς is $+$ **then** $\boldsymbol{\phi} \leftarrow -\boldsymbol{\phi}$; **end**
  $h\kappa \leftarrow F_h(P_h(\boldsymbol{\phi}))$;
  if ς is $+$ **then** $h\kappa \leftarrow -h\kappa$; **end**
  return $h\kappa$;
end

$\kappa_{flat} = 5$

Inference threshold



Five-layered MLP trained only on the negative curvature spectrum data set $\mathcal{D}$.

Preliminary Research: **A Hybrid Inference System to Estimate κ**

- We followed the same training approach as [LaLC and Gibou 21a] except for:
  - *PCA + whitening* to **speed up convergence** [LeCun *et al.* 12] using SciKit-Learn's `PCA` class.
  - 180 ***ReLU*** neurons in each hidden layer for $h = 1/128$ (i.e., for *unit-square quadtrees* with $\eta = 7$ maximum level of refinement).
  - *Halving the learning rate* from 1.5×10^{-4} to 1.5×10^{-5} if validation MAE does not improve for 15 epochs.
  - Delayed *early-stopping* with a patience of 60 epochs.

Principal component analysis (PCA) is a change of basis, where the new axes are the first k singular vectors of the **covariance matrix**

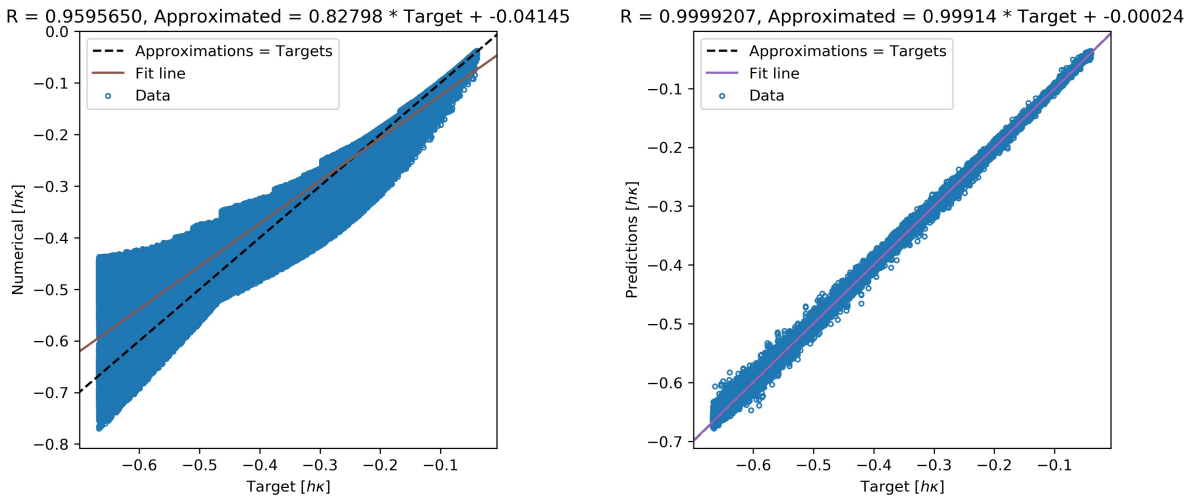
$$C = \frac{1}{n-1}(D-M)^T(D-M).$$

If $X = D - M$ and $C = USV^T$, then

$$X^{(k)} = XV^{(k)}$$

is the n-by-k transformed data set, where $V^{(k)}$ are the first k cols in V.

Whitening: $X_w^{(k)} = X^{(k)}\left(S^{(k)}\right)^{-\frac{1}{2}}$



Fit quality on $\mathcal{D}^{rls}$ for (left) $G_h(\cdot)$ and (right) $F_h(\cdot)$ trained for $\eta = 7$.

	MAE	Max absolute error	MSE
Neural network	1.815692×10^{-1}	7.242649	8.291462×10^{-2}
Numerical method	4.511332	2.926552×10^{1}	5.141757×10^{1}

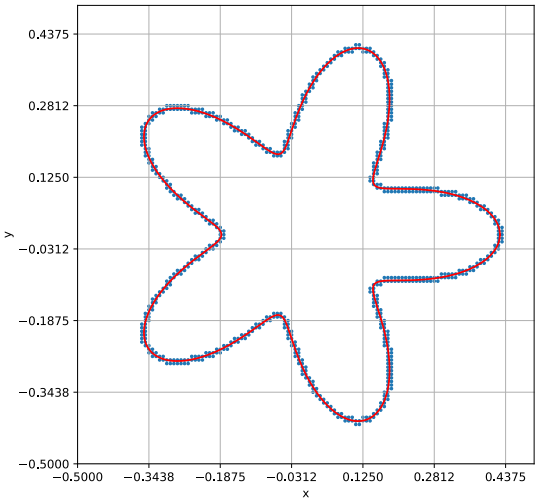
Training statistics on $\mathcal{D}^{rls}$ in terms of κ for $\eta = 7$.

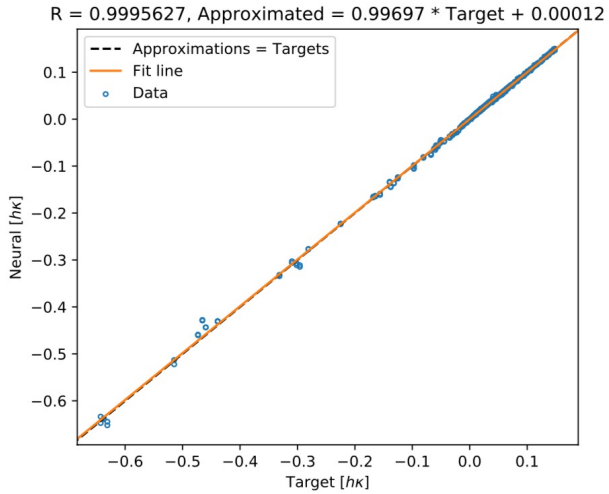
Preliminary Research: **A Hybrid Inference System to Estimate κ**

Testing our hybrid system for $h = 2^{-7}$ on $\phi_{rose}(\mathbf{x})$ with $p = 5$, $a = 0.12$, and $b = 0.305$.

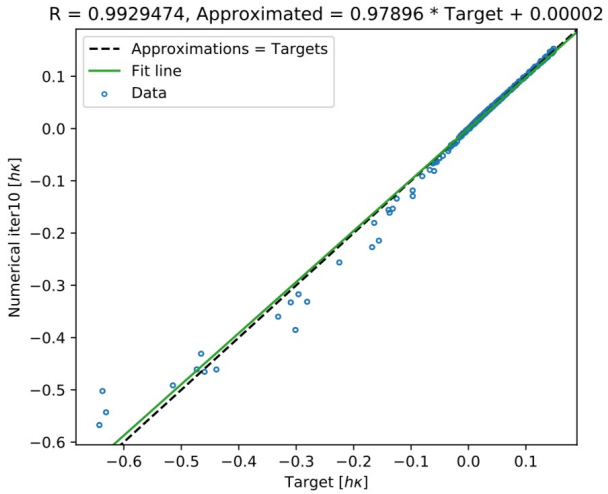
Method	Mean Absolute Error	Max Absolute Error	Mean Squared Error
Hybrid approach	1.778957×10^{-1}	4.841265	1.601629×10^{-1}
Numerical method, 10 iters.	4.441347×10^{-1}	1.726560×10^{1}	2.570065
Numerical method, 20 iters.	4.046386×10^{-1}	1.745362×10^{1}	2.554465

Comparing κ results for $h = 1/128$ with a model with four 180-ReLU-neuron hidden layers.

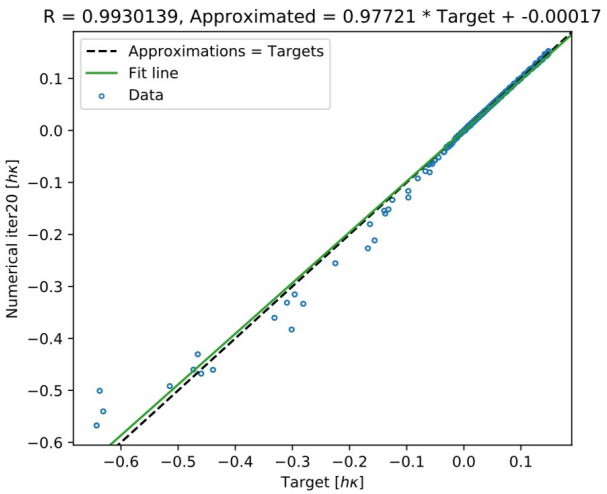




(a) Hybrid approach ($\nu = 10$).



(b) Numerical $G_h(\cdot)$ ($\nu = 10$).

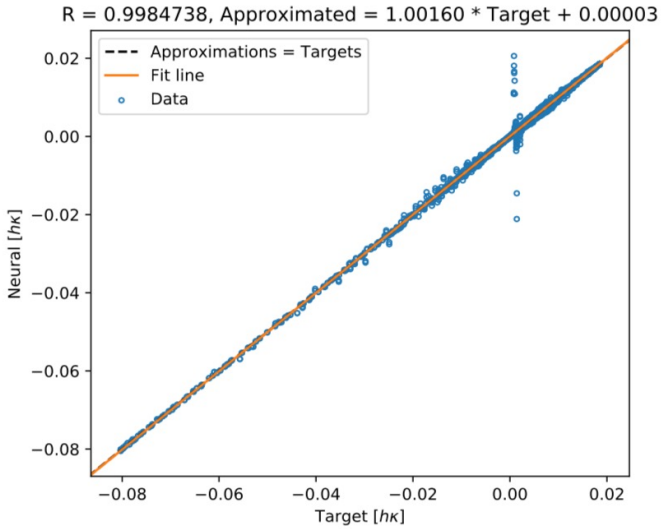


(c) Numerical $G_h(\cdot)$ ($\nu = 20$).

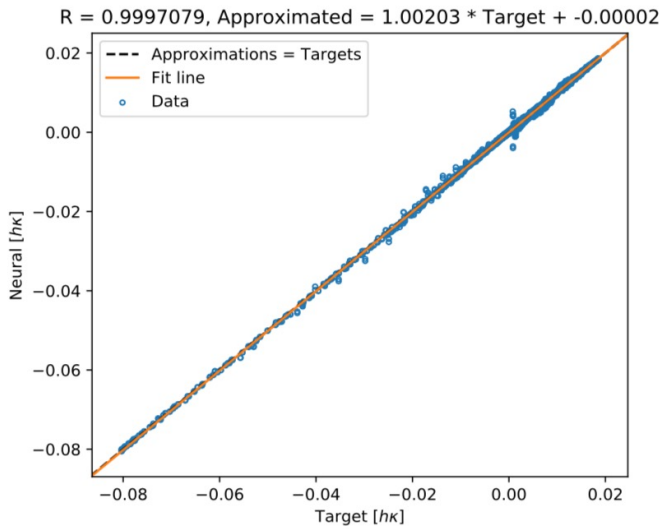
Fit quality for $h\kappa$ using the hybrid approach and $G_h(\cdot)$.

Can we do better than that?

- PCA and whitening are good for convergence but leads to overfitting as $h \to 0$ (despite early stopping).
  - Can we increase generalization?
- The process for building $\mathcal{D}$ and optimizing $F_h(\cdot)$ does not scale well for large η.
  - Can we exploit other properties besides the problem's intrinsic symmetry?
  - Can we just not throw away the numerical $h\kappa$ approximation during inference?



(a) Using PCA and whitening.



(b) Using z-scores.

Generalization of a network-only model for $\eta = 10$ when inferring $h\kappa$ along previous $\phi_{rose}(\mathbf{x})$.

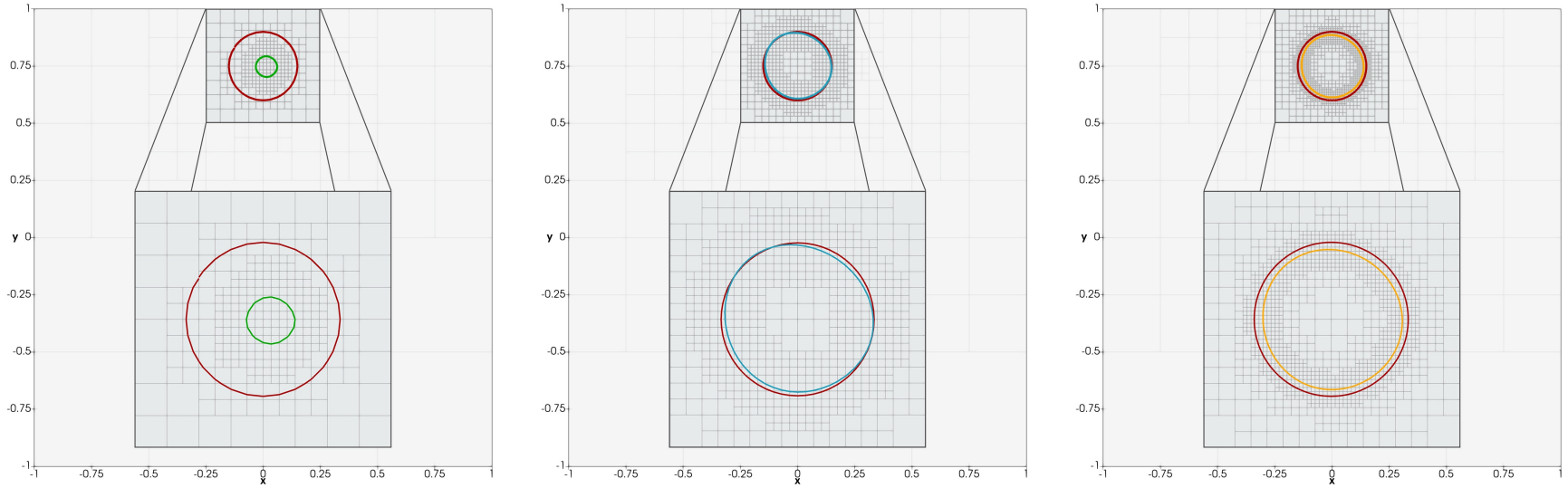
η	Approach	$\lvert\mathcal{D}\rvert$	Params.	Units per hidden layer
7	Network-only	4'442,728	200,193	256
	Hybrid	2'330,785	99,721	180
8	Network-only	7'258,628	159,145	228
	Hybrid	3'629,297	89,081	170
9	Network-only	9'315,620	148,281	220
	Hybrid	5,205,788	52,521	130
10	Network-only	12'592,164	159,145	228
	Hybrid	6,273,339	31,401	100

Topological analysis. Data-set size grows too much as η increases.

Error neural modeling schemes

Introducing **Error-Correcting Neural Networks** (ECNets)

- Drawing inspiration from ML-augmented scalar advection [Pathak et al. 20], ECNets for semi-Lagrangian transport [LaLC and Gibou 21c] using quadtrees [Strain 99], and image-super resolution technologies [Dong et al. 14].
  - **Goal**: To compute on-the-fly corrections to a coarse grid trajectory, so that it follows that of a much finer grid.
- In [LaLC and Gibou 21c], we introduced the MLSemiLagrangian() routine: a hybrid advection solver for FBPs.
  - MLSemiLagrangian() couples numerical advection with the ECNet $\mathcal{F}_{c,f}(\cdot)$ to improve LS accuracy next to Γ^n.
  - Exploited advection and curvature symmetry invariance.



Zero-contours at the end of the 15th revolution. Left and right: numerical baseline for $h = 2^{-6}$ and $h = 2^{-7}$. Center: MLSemiLagrangian() solution for $h = 2^{-6}$. Analytical disk appears in red.

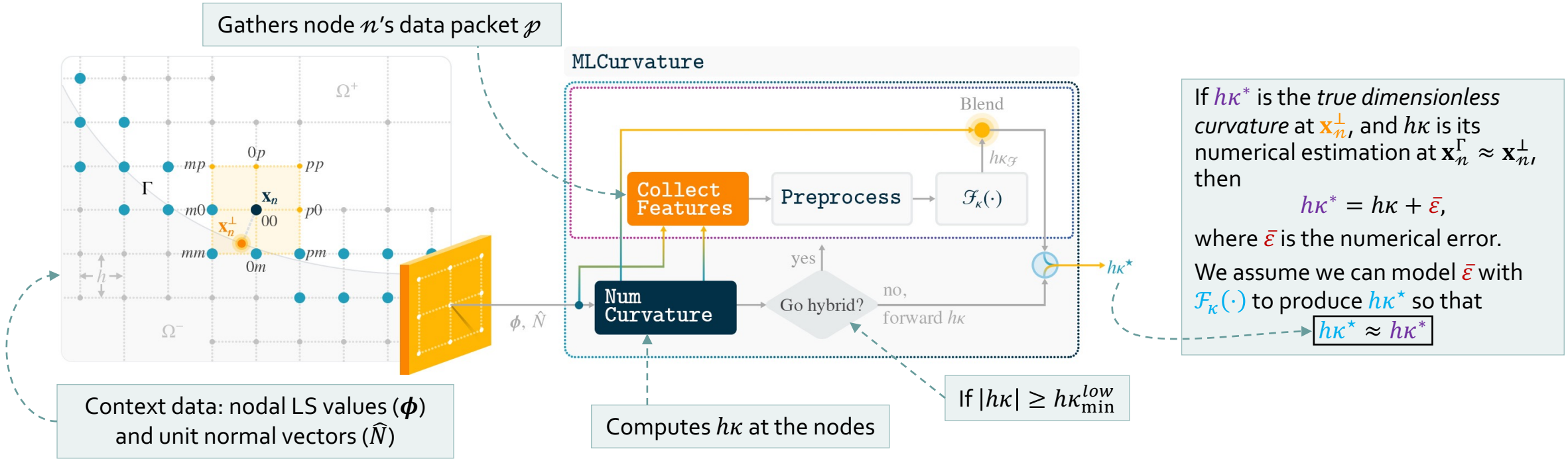
Use case: Rotating disk of radius 0.15, initially placed at $(0, 0.75)$, subject to

$$\mathbf{u}(x, y) = \frac{1}{\sqrt{2}} \begin{bmatrix} -y \\ x \end{bmatrix}$$

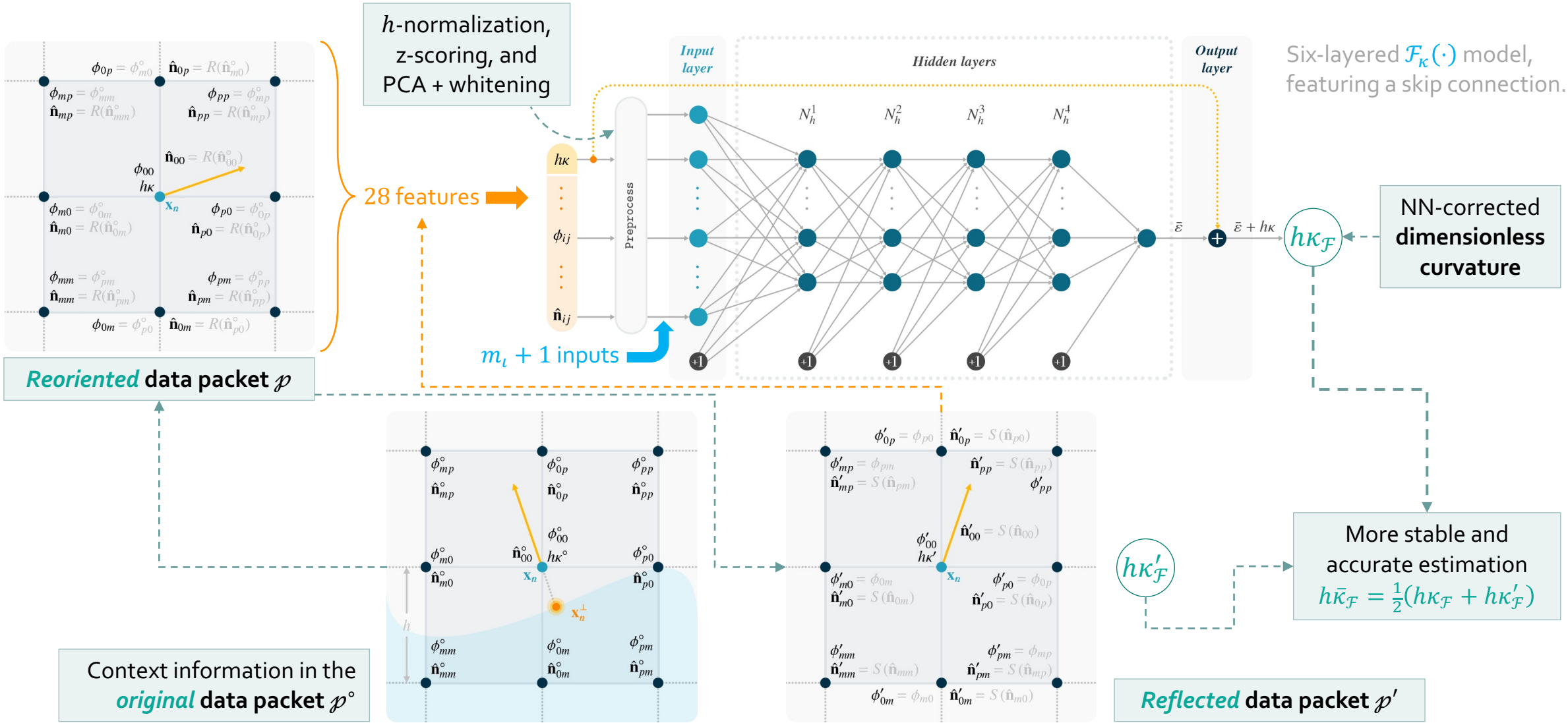
within $\Omega \equiv [-1, +1]^2$.

Novel **Error-Correcting Neural Network for 2D Curvature Computation**

- We have proposed an MLCurvature() hybrid routine in [LaLC and Gibou 22] that uses an ECNet to correct the numerical error in $h\kappa$ by:
  - Leveraging ***context information*** like ϕ values, gradient, and $h\kappa$ itself.
  - Enabling $\mathcal{F}_\kappa(\cdot)$ prediction only when necessary (and its output with $h\kappa$ near flat regions).
  - Exploiting κ ***rotation/reflection invariance*** to simplify the problem and improve stability through double predicting.
  - Incorporating ***dimensionality reduction***, ***well-balanced*** data sets, and ***regularization*** to increase generalization.



Novel **Error-Correcting Neural Network for 2D Curvature Computation**



Novel **Error-Correcting Neural Network for 2D Curvature Computation**

Algorithm 1: $h\kappa^\star \leftarrow$ `MLCurvature`$(n, \mathcal{F}_\kappa(\cdot), \boldsymbol{\phi}, \hat{N}, h, h\kappa^{low}_{\min}, h\kappa^{up}_{\min})$: Compute the dimensionless curvature for the interface node n using the standard scheme with error correction provided by $\mathcal{F}_\kappa(\cdot)$.

Input: n: node object; $\mathcal{F}_\kappa(\cdot)$: error-correcting neural network; $\boldsymbol{\phi}$: nodal level-set values; $\hat{N}$: nodal unit normal vectors; h: mesh size; $h\kappa^{low}_{\min}$: minimum $|h\kappa|$ to enable neural inference; $h\kappa^{up}_{\min}$: $|h\kappa|$'s upper bound for blending numerical with neurally corrected approximation.

Result: $h\kappa^\star$: dimensionless curvature at $\mathbf{x}^\Gamma_n$.

`// Numerical computation`

$K \leftarrow$ `NumCurvature(`n`.stencil,` $\boldsymbol{\phi}$`,` $\hat{N}$`)`;

$\mathbf{x}^\Gamma_n \leftarrow n.\mathbf{x} - \boldsymbol{\phi}[n]\hat{N}[n]$;

$h\kappa \leftarrow h\cdot$`Interpolate(`$n$`.stencil,` K`,` $\mathbf{x}^\Gamma_n$`)`;

Computing (numerical) $h\kappa$ at $\mathbf{x}^\Gamma_n$

`// Selectively enabling neural correction`

if $|h\kappa| \geqslant h\kappa^{low}_{\min}$ **then**

  $p \leftarrow$ `CollectFeatures(`n`.stencil,` $\boldsymbol{\phi}$`,` $\hat{N}$`)`;

  $p.h\kappa \leftarrow h\kappa$;

Populating n's original data packet

  `// Produce two samples for` n

  transform p so that $p.h\kappa$ is negative;

  reorient p so that the angle of $p.\hat{\mathbf{n}}_{00}$ lies between 0 and $\pi/2$;

  $h\kappa_{\mathcal{F}} \leftarrow \mathcal{F}_\kappa$`([Preprocess(`$p$`,` h`),` $p.h\kappa$`])`;

Computing $h\kappa_{\mathcal{F}}$ for p

  let p' be the reflected data packet about the line $y = x + \beta$ going through $n.\mathbf{x}$;

  $h\kappa'_{\mathcal{F}} \leftarrow \mathcal{F}_\kappa$`([Preprocess(`$p'$`,` h`),` $p'.h\kappa$`])`;

Computing $h\kappa'_{\mathcal{F}}$ for p'

  $h\bar{\kappa}_{\mathcal{F}} \leftarrow \frac{1}{2}(h\kappa_{\mathcal{F}} + h\kappa'_{\mathcal{F}})$;

Averaging into $h\bar{\kappa}_{\mathcal{F}}$

  `// Linearly blending neural and numerical estimations near zero`

  if $|h\kappa| \leqslant h\kappa^{up}_{\min}$ **then**

    $\lambda \leftarrow (h\kappa^{up}_{\min} - |h\kappa|)/(h\kappa^{up}_{\min} - h\kappa^{low}_{\min})$;

    $h\bar{\kappa}_{\mathcal{F}} = (1-\lambda)h\bar{\kappa}_{\mathcal{F}} + \lambda(-|h\kappa|)$;

Transitioning from $|h\kappa|$ to $|h\bar{\kappa}_{\mathcal{F}}|$ linearly in the range of $[h\kappa^{low}_{\min}, h\kappa^{up}_{\min}]$

  end

  $h\kappa^\star \leftarrow$`Sign(`$h\kappa$`)`$\cdot|h\bar{\kappa}_{\mathcal{F}}|$;

else

  $h\kappa^\star \leftarrow h\kappa$;

end

return $h\kappa^\star$;

$\mathbf{p} \leftarrow$ Preprocess$(p,\ h)$

$p.\phi_{ij} \leftarrow \frac{1}{h}p.\phi_{ij}$, for $i, j \in \{m, 0, p\}$;

$\mathbf{q} \leftarrow$ Vectorize(p);

$\mathcal{Q} \leftarrow$ GetStats();

$(\boldsymbol{\mu}, \boldsymbol{\sigma}) \leftarrow \mathcal{Q}$.StdScaler; // Z-scoring.

$\mathbf{z} \leftarrow (\mathbf{q} - \boldsymbol{\mu}) \oslash \boldsymbol{\sigma}$;

$\left(V^{(m_\iota)}, S^{(m_\iota)}\right) \leftarrow \mathcal{Q}$.PCA; // Dim. reduction.

$\mathbf{p} \leftarrow \left(S^{(m_\iota)}\right)^{-1/2}\left(V^{(m_\iota)}\right)^T\mathbf{z}$;

return p;

We have chosen

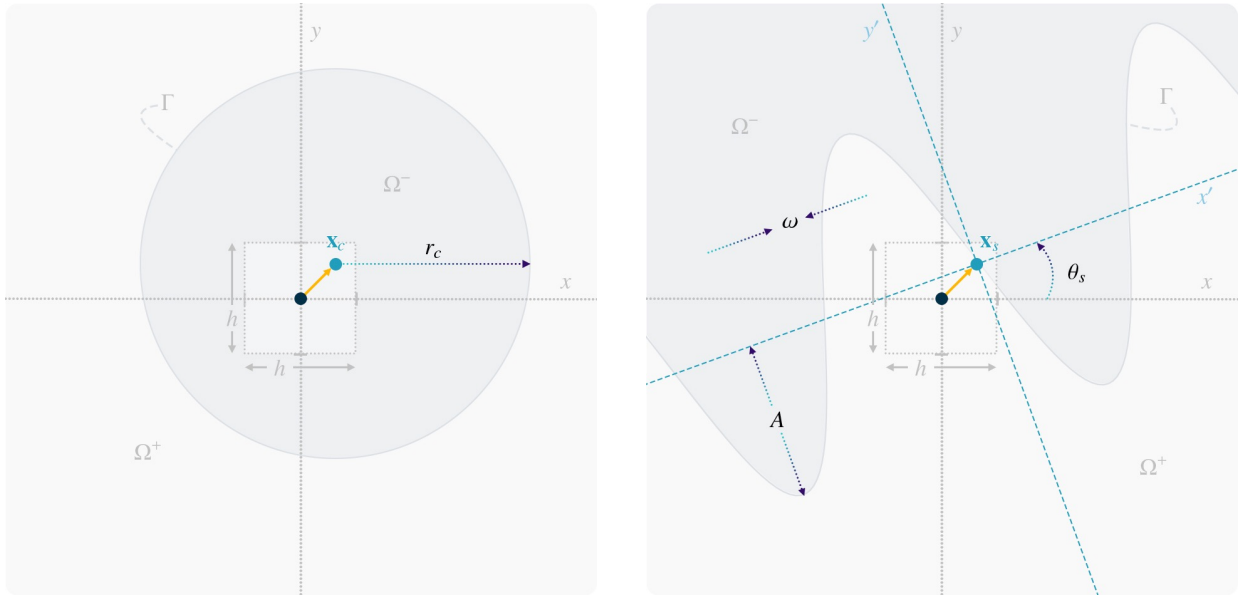
$$h\kappa^{low}_{\min} = 0.004 = h\kappa^{*}_{\min}$$

and

$$h\kappa^{up}_{\min} = 0.007$$

Novel **Error-Correcting Neural Network for 2D Curvature Computation**

- ***Data-set generation*** like [LaLC and Gibou 21b], on circular and sinusoidal interfaces, except for non-dimensional parametrization: **Scalable across grid resolutions**!
  - Built $\mathcal{D} = \mathcal{D}_c \cup \mathcal{D}_s$ with tuples with $|h\kappa^*|$ between $h\kappa^*_{\min} = \mathbf{0.004}$ and $h\kappa^*_{\max} = \frac{2}{3}$ (← equivalent to a circle with $r = 1.5h$).
  - Data-packet reorientation and reflection-based augmentation helped **constrain** the problem and **increase** $\mathcal{F}_\kappa(\cdot)$'s **capacity**.
  - Not considering exact-signed-distance samples anymore, only $\phi_c^{rls}(\mathbf{x})$ and $\phi_s^{rls}(\mathbf{x})$ redistanced with $\nu = 10$ iterations: **Cut down $|\mathcal{D}|$ at least in half**!
  - Improved probabilistic subsampling in both kinds of interfaces to give each $h\kappa^*$ "class" a fair probability of being accounted for during training.



Training interfaces and their tunable parameters in black and blue.

Assembling $\mathcal{D}_c$ with circular-interface samples

$\mathcal{D}_c \leftarrow$ **GenerateCircularDataSet**($\eta, h\kappa^*_{\min}, h\kappa^*_{\max}$, CPH, SPH2, ν, KeepEveryX)

$h \leftarrow 2^{-\eta}$;
$\kappa^*_{\min} \leftarrow h\kappa^*_{\min}/h$; $\kappa^*_{\max} \leftarrow h\kappa^*_{\max}/h$; $r_{\min} \leftarrow 1/\kappa^*_{\max}$; $r_{\max} \leftarrow 1/\kappa^*_{\min}$;
NC $\leftarrow \left\lceil \text{CPH} \cdot \left(\frac{r_{\max} - r_{\min}}{h} + 1 \right) \right\rceil$;

Find min/max r and κ^* and NC for current mesh size

CPH = 2

$\bar{r} = \frac{1}{2}(r_{\min} + r_{\max})$;
AvgSPR $\leftarrow \frac{1}{\text{KeepEveryX}} \left\lceil \text{SPH2} \cdot \frac{\pi}{h^2} \left(\bar{r}^2 - (\bar{r} - h)^2 \right) \right\rceil$;
SPR $\leftarrow$ Linspace(0.75 · AvgSPR, 1.25 · AvgSPR, NC);

Use mean $\bar{r}$ to prescribe the number of samples per r_c into SPR

SPH2 = 5
KeepEveryX = 4

// Data generation loop
$\mathcal{D}_c \leftarrow \varnothing$;
TgtK $\leftarrow$ Linspace($\kappa^*_{\max}, \kappa^*_{\min}$, NC);
for c $\leftarrow 0, 1, \ldots,$ NC $- 1$ **do**
  $\kappa^* \leftarrow$ TgtK[c]; $r_c \leftarrow 1/\kappa^*$;

Spacing out κ^* from $\kappa^*_{\max}$ to $\kappa^*_{\min}$ into TgtK. Use this to extract r_c

  // Collect samples for radius r_c into the set $\mathcal{S}$
  $\mathcal{S} \leftarrow \varnothing$;
  while $|\mathcal{S}| < 2 \cdot$ SPR[c] **do**
    $\mathbf{x}_c \sim \mathcal{U}(-h/2, +h/2)$;
    $\phi_c(\cdot) \leftarrow$ CircleLevelSet($\mathbf{x}_c$, r_c);
    let $\mathcal{B}_\Omega$ be the domain bounds ensuring that $\|\mathbf{x} - \mathbf{x}_c\|_\infty \leqslant r_c + 4h$ holds for any $\mathbf{x} \in \Omega$;
    $\mathcal{G} \leftarrow$ GenerateGrid($\phi_c(\cdot)$, η, $\mathcal{B}_\Omega$, $2h\sqrt{2}$);

Gather r_c samples until $|\mathcal{S}| \geq 2 \cdot$ SPR[c]. Use a random $\mathbf{x}_c$ with r_c to spawn $\phi_c(\mathbf{x})$ and discretize Ω.

    $\boldsymbol{\phi} \leftarrow$ Evaluate($\mathcal{G}$, $\phi_c(\cdot)$);
    $\boldsymbol{\phi} \leftarrow$ Reinitialize($\boldsymbol{\phi}$, ν);
    $\hat{N} \leftarrow$ ComputeNormals($\mathcal{G}$, $\boldsymbol{\phi}$);
    $\boldsymbol{\kappa} \leftarrow$ NumCurvature($\mathcal{G}$, $\boldsymbol{\phi}$, $\hat{N}$);

Evaluate $\phi_c(\mathbf{x})$ on $\mathcal{G}$, reinitialize it, and compute normal vectors ($\widehat{N}$) and curvature ($\boldsymbol{\kappa}$)

    $\mathcal{N} \leftarrow$ GetNodesNextToGamma($\mathcal{G}$, $\boldsymbol{\phi}$);
    foreach *node* $n \in \mathcal{N}$ **do**
      if $\mathcal{U}(0, 1) \geqslant 1/$KeepEveryX **then** skip node n;

Sample each n along Γ with a probability of 1/KeepEveryX

      $p \leftarrow$ CollectFeatures(n.stencil, $\boldsymbol{\phi}$, $\hat{N}$);
      $\mathbf{x}^\Gamma_n \leftarrow n.\mathbf{x} - \boldsymbol{\phi}[n]\hat{N}[n]$ $p.h\kappa \leftarrow h\cdot$Interpolate($\mathcal{G}$, $\boldsymbol{\kappa}$, $\mathbf{x}^\Gamma_n$);
      let $\boldsymbol{\xi}$ be the learning tuple $(p, h\kappa^*)$ with inputs p and expected output $h\kappa^* := h \cdot \kappa^*$;
      transform $\boldsymbol{\xi}$ so that $p.h\kappa^*$ is negative;
      rotate $\boldsymbol{\xi}$ so that the angle of $p.\hat{\mathbf{n}}_{00}$ lies between 0 and $\pi/2$;
      add $\boldsymbol{\xi}$ to $\mathcal{S}$;

Tuple for p

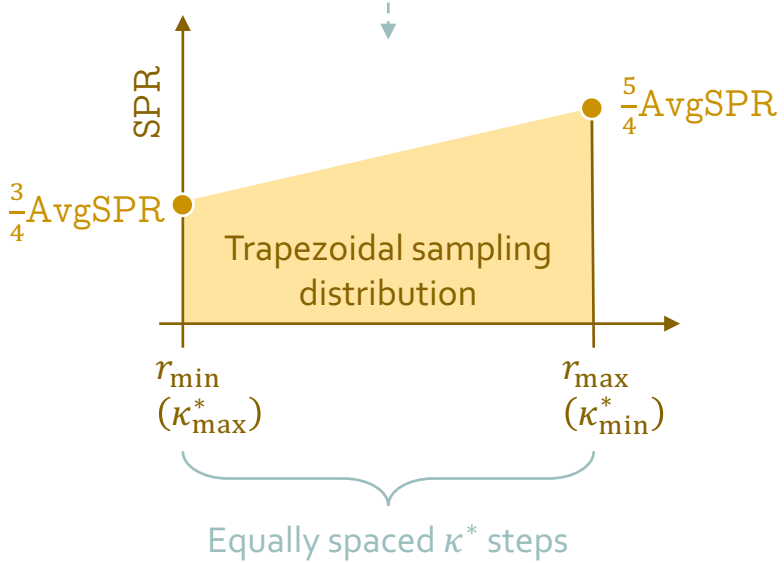
      let $\boldsymbol{\xi}'$ be the reflected tuple about the line $y = x + \beta$ going through $n.\mathbf{x}$;
      add $\boldsymbol{\xi}'$ to $\mathcal{S}$;

Tuple for p'

    end
  end
  $\mathcal{D}_c \leftarrow \mathcal{D}_c \cup$ RandomSamples($\mathcal{S}$, $2 \cdot$ SPR[c]);

Randomly subsampling if we went over $2 \cdot$ SPR[c]

end
return $\mathcal{D}_c$;



$\mathcal{D}_s \leftarrow$ **GenerateSinusoidalDataSet**(η, $h\kappa^*_{\min}$,$h\kappa^*_{\max}$, EaseOffMidMaxHKPr, NA, NT, ν)

```
6  D_s ← ∅;
7  foreach amplitude A ∈ Linspace(A_min, A_max, NA) do
8      ω_min ← sqrt(hκ^low_max/(h·A));   ω_max ← sqrt(hκ^up_max/(h·A));
9      ω_d ← (π/2)(ω_min^-1 − ω_max^-1);
10     NF ← ⌈ω_d/h⌉ + 1;
11     foreach frequency ω ∈ Linspace(ω_min, ω_max, NF) do
12         S ← ∅;                          // Collect samples for interface A sin(ωt) into S
13         foreach angle θ_s ∈ Linspace(−π/2, +π/2, NT), excluding right end point do
14             x_s ~ U(−h/2, +h/2);
15             φ_s(·) ← SineLevelSet(A, ω, x_s, θ_s);
16             let B_Ω be the domain bounds ensuring that ||x − x_s||_∞ ⩽ r_sam + 4h holds for any x ∈ Ω;
17             G ← GenerateGrid(φ_s(·), η, B_Ω, 4h√2);        // Discretizing Ω with an adaptive mesh
18             φ ← Evaluate(G, φ_s(·));
19             φ ← Reinitialize(φ, ν);
20             N̂ ← ComputeNormals(G, φ);
21             κ ← NumCurvature(G, φ, N̂);
22             N ← GetNodesNextToGamma(G, φ);
23             foreach node n ∈ N do
24                 compute the target hκ* at the nearest point, x_n^⊥, to n.x on the sinusoidal interface;
25                 if ||n.x − x_s||_2^2 > r_sam^2 or |hκ*| < hκ*_min then skip node n;
                   // Prob. sampling: Pr(|hκ*| ⩾ hκ̄*_max) = EaseOffMidMaxHKPr and Pr(|hκ*| = hκ*_min) = 0.01
26                 q ← min(1, (|hκ*| − hκ*_min)/(hκ̄*_max − hκ*_min));
27                 if U(0, 1) ⩽ EaseOff(0.01, EaseOffMidMaxHKPr, q) then
28                     p ← CollectFeatures(n.stencil, φ, N̂);
29                     x_n^Γ ← n.x − φ[n]N̂[n];   p.hκ ← h·Interpolate(G, κ, x_n^Γ);
30                     let ξ be the learning tuple (p, hκ*) with inputs p and expected output hκ*;
31                     transform ξ so that p.hκ* is negative;
32                     rotate ξ so that the angle of p.n̂_00 lies between 0 and π/2;
33                     add ξ to S;
34                     let ξ′ be the reflected tuple about the line y = x + β going through n.x;
35                     add ξ′ to S;
36                 end
37             end
38         end
39         D_s ← D_s ∪ S;
40     end
41 end
42 return D_s;
```

For each A, set the frequency $\omega_{\min}$ and $\omega_{\max}$ bounds to ensure $|h\kappa^*_{\max}| \in [\kappa^{low}_{\max}, \kappa^{up}_{\max}]$

Dist. between the first crests at $\omega_{\min}$ and $\omega_{\max}$ defines # of ω steps

For each (A, ω) pair, vary θ_s and pick a random $\mathbf{x}_s$

Evaluate $\phi_s(\mathbf{x})$ on $\mathcal{G}$, reinitialize it, and compute normal vectors ($\widehat{N}$) and curvature ($\boldsymbol{\kappa}$)

Only nodes inside samp. circ. or if $|h\kappa^*| \geq h\kappa^*_{\min}$

Sample each n by following an ease-off prob. distribution

Tuple for p

Tuple for p'

```
1  h ← 2^−η;
2  κ*_min ← hκ*_min/h;   κ*_max ← hκ*_max/h;   A_min ← 4/κ*_max;   A_max ← 1/κ*_min;
3  r_sam ← 2A_max;
4  hκ^low_max ← (1/2)hκ*_max;   hκ^up_max ← hκ*_max;
5  hκ̄*_max ← (1/2)(hκ^low_max + hκ^up_max);
```

Find min/max κ^* and A for h
Define sampling radius r_{sam}

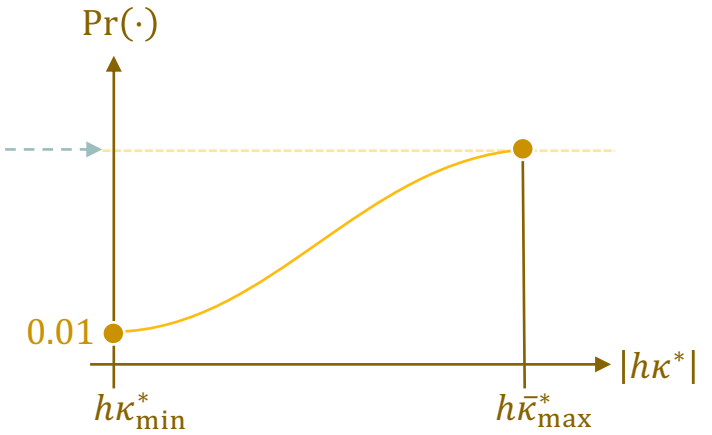
Set bounds $h\kappa^{low}_{\max} \leq \max_{\Omega}|h\kappa^*| \leq h\kappa^{up}_{\max}$ along crests and the mean value $h\bar{\kappa}^*_{\max}$

NA = 34

NT = 38

EaseOffMidMaxHKPr = 0.4

Assembling $\mathcal{D}_s$ with sinusoidal-interface samples

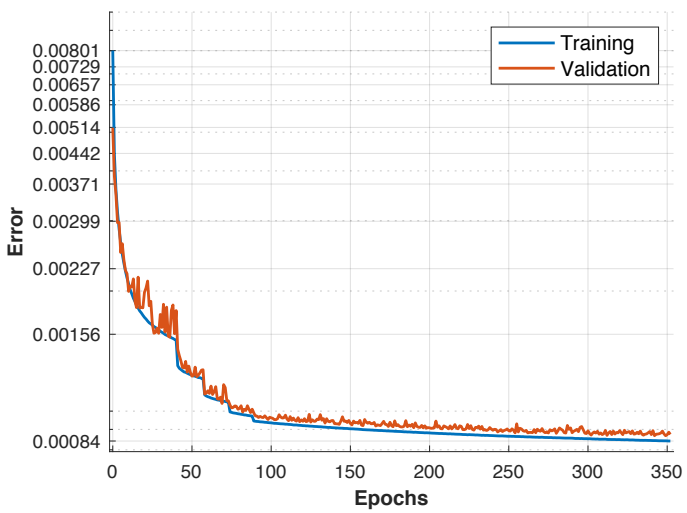


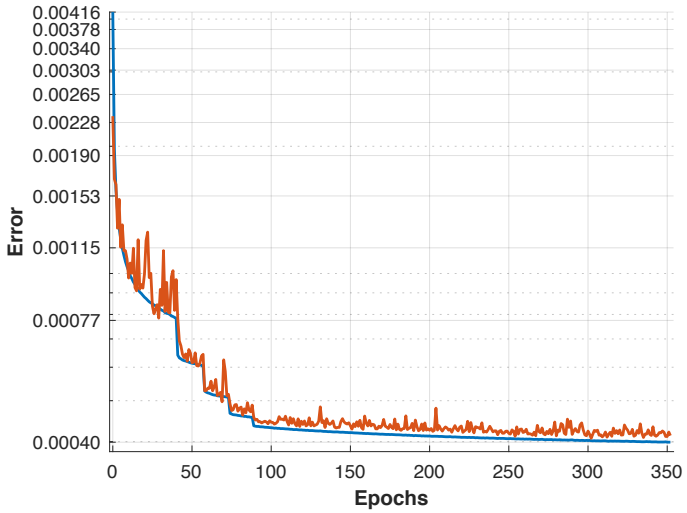
Postprocessing histogram-based subsampling steps

1. Distribute $\mathcal{D}_s$ into 100 bins in histogram $\mathcal{H}$, depending on $h\kappa^*$.
2. Randomly subsample overpopulated bins until $|b| \leq \frac{2}{3} m_{\mathcal{H}}$ holds for all $b \in \mathcal{H}$, where $m_{\mathcal{H}}$ is $\mathcal{H}$'s median.

Novel **Error-Correcting Neural Network for 2D Curvature Computation**

- ***Training technical details***:
  - Introduced Panda's $\mathrm{cut}()$ function to create $100\ h\kappa^*$ labels within $\mathcal{D}$. Then, we used these labels with SciKit-Learn's $\mathrm{StratifiedKFold.split}()$ with $K = 10$ to split $\mathcal{D}$ into training (70%), testing (10%), and validation (10%) subsets.
    - We discarded the remaining 10%.
    - We did not perform K-fold cross-validation for efficiency; only a couple of training instances with random weights.
  - Added $\mathrm{L2}$ kernel **regularization** ($\mathcal{O}(10^{-6})$) in all hidden layers.
  - Used Root MSE (RMSE), which **favored learning convergence** even more than MSE!
  - Halved the learning rate from $1.5{\times}10^{-4}$ down to 10^{-5} after seeing 15 epochs of no improvement.
  - Stopped optimization after 50 epochs of validation MAE stagnation.

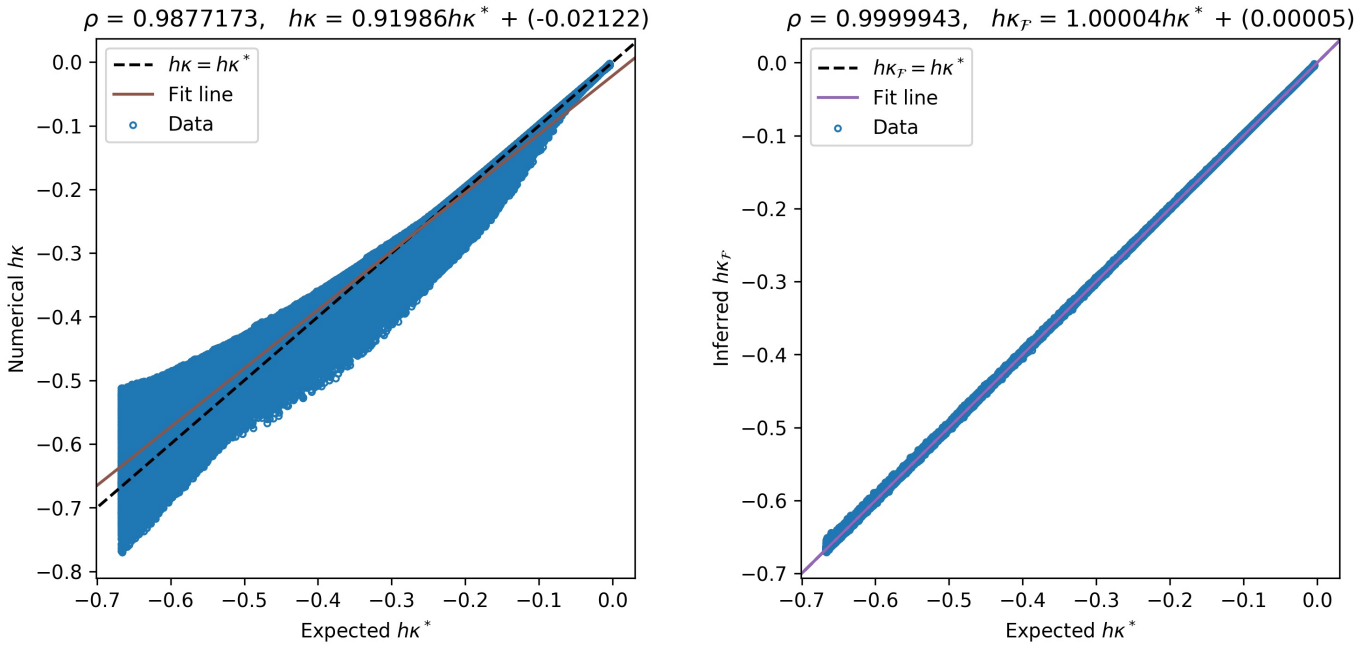




Learning curves for $\eta = 7$.
Left: RMSE.
Right: MAE.

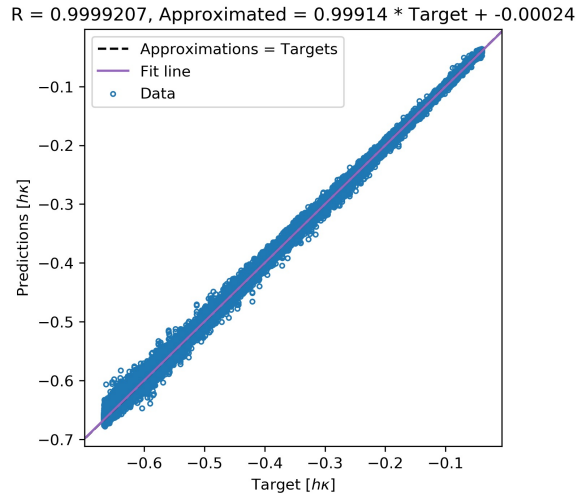
Novel **Error-Correcting Neural Network for 2D Curvature Computation**

- Consider $\mathcal{F}_\kappa(\cdot)$ for $\eta = 7$ (i.e., $h = 1/128$), where
  - $N_h^i = 130$ ReLU neurons in the first four hidden layers.
  - $m_\iota = 18$ linear input neurons (i.e., PCA with $m_\iota < 28$ components).
  - $\mathrm{L2} = 5{\times}10^{-6}$ kernel-regularization factor in all five hidden layers.



Fit quality on $\mathcal{D}$ for (left) numerical method and (right) $\mathcal{F}_\kappa(\cdot)$ trained for $\eta = 7$.

Compare



Fit quality on $\mathcal{D}^{rls}$ for prior $F_h(\cdot)$ trained for $\eta = 7$.

η	Method	MAE	MaxAE	RMSE
7	$\mathcal{F}_\kappa(\cdot)$	3.923205×10^{-4}	1.541521×10^{-2}	6.269966×10^{-4}
	Baseline	1.977165×10^{-2}	1.541942×10^{-1}	3.090897×10^{-2}

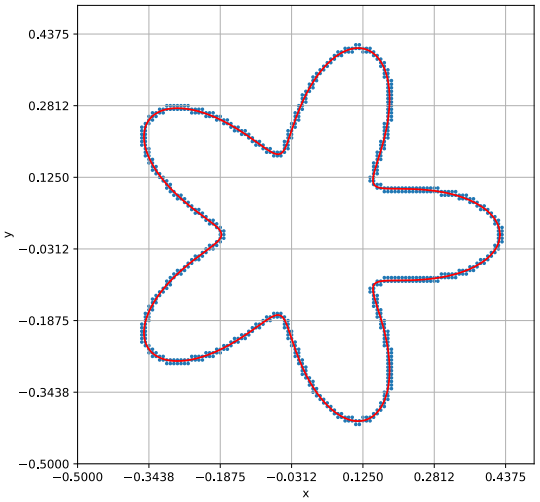
Learning $h\kappa$ and $h\kappa_{\mathcal{F}}$ error statistics on $\mathcal{D}$ for $\eta = 7$.

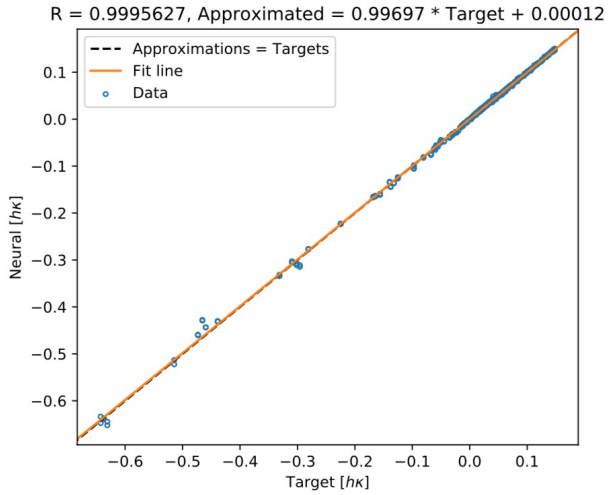
Novel **Error-Correcting Neural Network for 2D Curvature Computation**

Testing MLCurvature() for $h = 2^{-7}$ on $\phi_{rose}(\mathbf{x})$ with $p = 5$, $a = 0.12$, and $b = 0.305$.

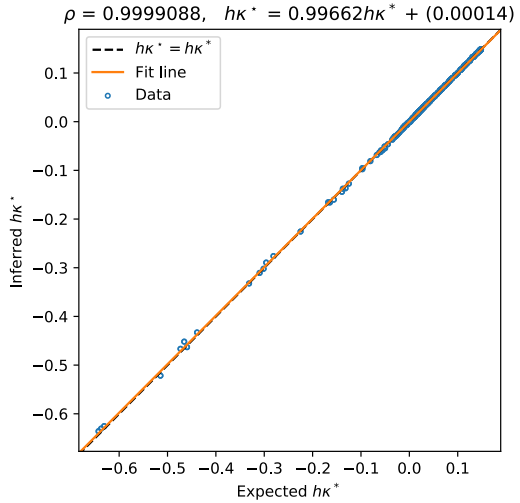
Method	MAE	Reduct. factor	MaxAE	Reduct. factor
MLCurvature()	7.37148×10^{-4}	–	1.36763×10^{-2}	–
Previous system	1.38981×10^{-3}	1.89	3.78224×10^{-2}	2.77
Baseline ($\nu = 10$)	3.46929×10^{-3}	4.71	1.34981×10^{-1}	9.87
Baseline ($\nu = 20$)	3.16221×10^{-3}	4.29	1.36403×10^{-1}	9.97

Comparing $h\kappa$ results for $h = 1/128$.

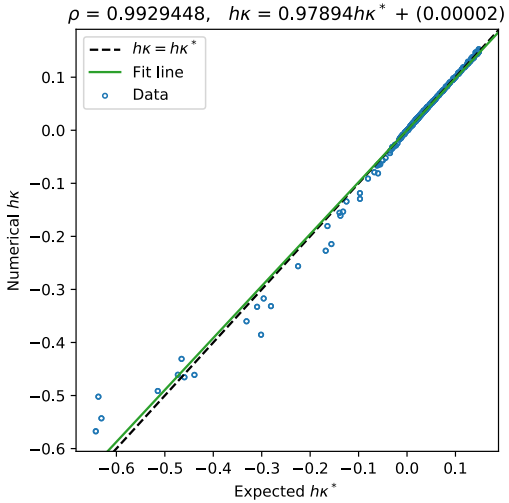




(a) Old hybrid approach ($\nu = 10$).



(b) MLCurvature() ($\nu = 10$).



(c) Numerical method ($\nu = 10$).

Fit quality for $h\kappa$ and $h\kappa^{\star}$ using the old hybrid approach, MLCurvature(), and the numerical baseline.

Novel **Error-Correcting Neural Network for 2D Curvature Computation**

Showing scalability to higher resolutions.

η	m_ι	N_h^i	L2 factor	$\lvert\mathcal{D}\rvert$	Parameters
6	20	130	5×10^{-6}	1,464,020	53,951
7	18	130	5×10^{-6}	1,473,359	53,691
8	18	120	5×10^{-6}	1,472,775	45,961
9	18	130	5×10^{-6}	1,464,407	53,691
10	18	130	1×10^{-5}	1,468,513	53,691
11	18	120	7×10^{-6}	1,462,741	45,961

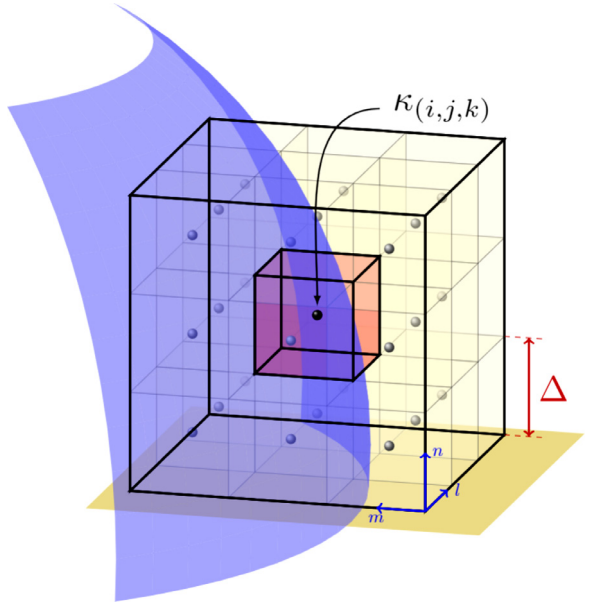
Architectural data for $6 \leq \eta \leq 11$.

η	Method	MAE	MaxAE	RMSE
6	$\mathcal{F}_\kappa(\cdot)$	3.660305×10^{-4}	1.397054×10^{-2}	5.713122×10^{-4}
	Baseline	1.975861×10^{-2}	1.542492×10^{-1}	3.095603×10^{-2}
7	$\mathcal{F}_\kappa(\cdot)$	3.923205×10^{-4}	1.541521×10^{-2}	6.269966×10^{-4}
	Baseline	1.977165×10^{-2}	1.541942×10^{-1}	3.090897×10^{-2}
8	$\mathcal{F}_\kappa(\cdot)$	3.991765×10^{-4}	1.369361×10^{-2}	6.332734×10^{-4}
	Baseline	1.979684×10^{-2}	1.549115×10^{-1}	3.094043×10^{-2}
9	$\mathcal{F}_\kappa(\cdot)$	3.952077×10^{-4}	1.295839×10^{-2}	6.336660×10^{-4}
	Baseline	1.983267×10^{-2}	1.524857×10^{-1}	3.097143×10^{-2}
10	$\mathcal{F}_\kappa(\cdot)$	4.427094×10^{-4}	1.342434×10^{-2}	7.062269×10^{-4}
	Baseline	1.982013×10^{-2}	1.549863×10^{-1}	3.095504×10^{-2}
11	$\mathcal{F}_\kappa(\cdot)$	4.293997×10^{-4}	1.456045×10^{-2}	6.879887×10^{-4}
	Baseline	1.983361×10^{-2}	1.551221×10^{-1}	3.098202×10^{-2}

Learning $h\kappa$ and $h\kappa_{\mathcal{F}}$ error statistics for $6 \leq \eta \leq 11$.

Networks and preprocessing objects in JSON format are available at https://github.com/UCSB-CASL/Curvature_ECNet_2D

Ongoing and Future Work



A 27-cell stencil to compute $\kappa_{(i,j,k)}$ in VOF
(Courtesy of [Patel *et al.* 19])

- Incorporating restrictions to enforce continuity or curvature smoothness along successive interface nodes [Beucler *et al.* 21]?
- Adding a well-resolution classifier [Buhendwa *et al.* 21][Ray and Hesthaven 18] [Morgan *et al.* 20] instead of hand-tuning $hk^*_{\min}$?
- Three-dimensional curvature computation:
  - Based on study by [Patel *et al.* 19] in VOF.
  - Their models were trained with well-balanced, randomized data sets of spherical patches and tested for ellipsoid, waves, and Gaussian shapes, as well as in a multiphase flow solver.
  - Experience with LSM in 2D shows that we require samples from more than just spherical interfaces.
- Per results in [Egan and Gibou 21], should we consider adding (scaled) white noise to $\phi(\mathbf{x})$ prior to reinitialization?

Thank You!

References

[Osher and Fedkiw 02] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002. New York, NY.

[Osher and Sethian 88] S. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[Gibou *et al.* 18] F. Gibou, R. Fedkiw, and S. Osher. A review of level-set methods and some recent applications. *J. Comput. Phys.*, 353:82–109, 2018.

[Sussman *et al.* 94] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.*, 114(1):146–159, 1994.

[Salih and Ghosh 13] A. Salih and S. Ghosh Moulic. A mass conservation scheme for level set method applied to multiphase incompressible flows. *Int. J. Comput. Meth. Eng. Sci. Mech.*, 14:271–289, 2013.

[LaLC and Gibou 21a] L. Á. Larios-Cárdenas and F. Gibou. A deep learning approach for the computation of curvature in the level-set method. *SIAM J. Sci. Comput.*, 43(3):A1754–A1779, 2021.

[LaLC and Gibou 21b] L. Á. Larios-Cárdenas and F. Gibou. A hybrid inference system for improved curvature estimation in the level-set method using machine learning. Submitted, https://arxiv.org/abs/2104.02951, 2021.

[LaLC and Gibou 21c] L. Á. Larios-Cárdenas and F. Gibou. Error-correcting neural networks for semi-Lagrangian advection in the level-set method. Submitted, https://arxiv.org/abs/2110.11611, 2021.

[LaLC and Gibou 22] L. Á. Larios-Cárdenas and F. Gibou. Error-correcting neural networks for two-dimensional curvature computation in the level-set method. Submitted, https://arxiv.org/abs/2201.12342, 2022.

[Popinet 18] S. Popinet. Numerical models of surface tension. *Annu. Rev. Fluid Mech.*, 50(1):49–75, 2018.

[Lervåg 14] K.Y. Lervåg. Calculation of interface curvature with the level-set method. https://arxiv.org/abs/1407.7340, 2014.

[Jiang and Peng 00] G.-S. Jiang and D. Peng. Weighted ENO schemes for Hamilton-Jacobi equations. *SIAM J. Sci. Comput.*, 21:2126–2143, 2000.

[Russo and Smereka 00] G. Russo and P. Smereka. A remark on computing distance functions. *J. Comput. Phys.*, 163(1): 51–67, 2000.

[Egan and Gibou 21] R. Egan and F. Gibou. DNS Interfacial Processes. *MURI: Predicting turbulent multi-phase flows with high fidelity*. Fortnightly Meetings. March 30, 2021.

References

[Patel *et al.* 19] H. V. Patel, A. Panda, J. A. M. Kuipers, and E. A. J. F. Peters. Computing interface curvature from volume fractions: A machine learning approach. *Comput. Fluids*, 193:104263, 2019.

[Ray and Hesthaven 18] D. Ray and J. S. Hesthaven. An artificial neural network as a troubled-cell indicator. *J. Comput. Phys.*, 367:166– 191, 2018.

[Morgan *et al.* 20] N. R. Morgan, S. Tokareva, X. Liu, and A. D. Morgan. A machine learning approach for detecting shocks with high-order hydrodynamic methods. *AIAA SciTech Forum*, 2020.

[LeCun *et al.* 12] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. *Efficient BackProp*, volume 7700 of *Lecture Notes in Comput. Sci.*, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[Dong *et al.* 14] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision* – ECCV 2014, pages 184–199, Cham, 2014. Springer International Publishing.

[Min and Gibou 07] C. Min and F. Gibou. A second order accurate level set method on non-graded adaptive Cartesian grids. *J. Comput. Phys.*, 225(1):300–321, 2007.

[Du Chéné *et al.* 08] A. du Chéné, C. Min, and F. Gibou. Second-order accurate computation of curvatures in a level set framework using novel high-order reinitialization schemes. *J. Sci. Comput.*, 35:114–131, 2008.

[Mirzadeh *et al.* 16] M. Mirzadeh, A. Guittet, C. Burstedde, and F. Gibou. Parallel level-set methods on adaptive tree-based grids. *J. Comput. Phys.*, 322:345–364, 2016.

[Pathak *et al.* 20] J. Pathak, M. Mustafa, and K. Kashinath. Using machine learning to augment coarse-grid computational fluid dynamics simulations. https://arxiv.org/abs/2010.00072, 2020.

[Qi *et al.* 19] Y. Qi, J. Lu, R. Scardovelli, S. Zaleski, and G. Tryggvason. Computing curvature for volume of fluid methods using machine learning. *J. Comput. Phys.*, 377:155–161, 2019.

[Strain 99] J. Strain. Semi-Lagrangian methods for level set equations. *J. Comput. Phys.*, 151:498–533, 1999.

[Beucler *et al.* 21] T. Beucler, M. Pritchard, S. Rasp, J. Ott, P. Baldi, and P. Gentine. Enforcing analytic constraints in neural networks emulating physical systems. *Phys. Rev. Lett.*, 126(9):098302, 2021.

[Buhendwa *et al.* 21] A. B. Buhendwa, D. A. Bezgin, and N. Adams. Consistent and symmetry preserving data-driven interface reconstruction for the level-set method. https://arxiv.org/abs/2104.11578, 2021.