# KINETIC-BASED REGULARIZATION: LEARNING SPATIAL DERIVATIVES AND PDE APPLICATIONS

**Abhisek Ganguly*** **& Santosh Ansumali**
Engineering Mechanics Unit
Jawaharlal Nehru Centre for Advanced Scientific Research
Jakkur, Bengaluru-560064, Karnataka, India
{abhisek,ansumali}@jncasr.ac.in

**Sauro Succi**
Center for Life Nano-Neuroscience at La Sapienza
Fondazione Istituto Italiano di Tecnologia
Viale Regina Elena 291, 00161 Roma, Italy
sauro.succi@iit.it

## ABSTRACT

Accurate estimation of spatial derivatives from discrete and noisy data is central to scientific machine learning and numerical solutions of PDEs. We extend kinetic-based regularization (KBR), a localized multidimensional kernel regression method with a single trainable parameter, to learn spatial derivatives with provable second-order accuracy in 1D. Two derivative-learning schemes are proposed: an explicit scheme based on the closed-form prediction expressions, and an implicit scheme that solves a perturbed linear system at the points of interest. The fully localized formulation enables efficient, noise-adaptive derivative estimation without requiring global system solving or heuristic smoothing. Both approaches exhibit quadratic convergence, matching second-order finite difference for clean data, along with a possible high-dimensional formulation. Preliminary results show that coupling KBR with conservative solvers enables stable shock capture in 1D hyperbolic PDEs, acting as a step towards solving PDEs on irregular point clouds in higher dimensions while preserving conservation laws.

## 1 INTRODUCTION

Over the years, regression has seen significant progress within the frame of machine learning. However, accurate derivative evaluation is a key area currently driving machine learning applications in computational physics (Wang & Zhou (2020), Abramavicius et al. (2025)). Automatic differentiation (Baydin et al. (2018)) is now well-employed in modern deep learning frameworks, enabling the computation of spatial and temporal derivatives in physics-informed models and also forming the basis of backpropagation (Rumelhart et al. (1986)). However, it also faces significant challenges (Gholami et al. (2019), Margossian (2019), Hückelheim et al. (2024)).

There has been a marked interest in deep learning approaches for solving PDEs, e.g. PINNs (Raissi et al. (2019)). Recent efforts recognize the importance of conservative variants (Jagtap et al. (2020)) that improve flux consistency. Though being parameter-heavy with expensive optimization, PINNs' functionality in higher dimensions cannot be ignored (Kumar & Ranjan (2026)). Related operator and derivative learning frameworks (Lu et al. (2021), Li et al. (2021), Ledesma et al. (2020)) further highlight the growing need for robust, physics-aware differentiation. Many such learning schemes, however, lack rigorous convergence and error analysis Wang et al. (2022), which hinders systematic performance assessment.

*Correspondence to abhisek@jncasr.ac.in

Recently, kinetic-based regularization (KBR, Ganguly et al. (2025)) was introduced to revive Radial Basis Function networks, one of the earliest kernel-based neural models (Broomhead & Lowe (1988)). The physics-inspired kernel regression scheme adopts a fully localized formulation that avoids solving large-scale global systems of equations. This enables learnable noise removal using a single trainable parameter, independent of the problem dimensionality.

In this paper, we extend KBR to learn spatial derivatives with fully interpretable accuracy, avoiding direct kernel differentiation approach used by methods such as Gaussian Processes (Rasmussen & Williams (2006)) that is known to cause instabilities (Gingold & Monaghan (1977), Bardenhagen & Sulsky (2001), Monaghan (2005)) in numerical solvers. We present explicit and implicit schemes of derivative extraction from learned fields, each with its own key strengths. We show that KBR is capable of stable integration into conservative 1D hyperbolic PDE solvers where traditional PINN-based learning approaches often saturate. This bridges kernel learning with traditional solvers, a potential step towards simulating PDEs on irregular high-dimensional point clouds while respecting conservation laws.

## 2 METHODOLOGY

The original KBR aims to fit an unknown function with a locally quadratic polynomial, but gives us no information about the fitting constants involved. The fit is ensured by a point-wise enforcement of lower order moments, arising from the disagreement between continuous and discrete statistics. The desired local quadratic fit using KBR for some $\boldsymbol{x}$ for $\boldsymbol{x} \subset \mathbb{R}^D$ is:

$$\phi(\boldsymbol{x}) = a + \boldsymbol{b} \cdot \boldsymbol{x} + \boldsymbol{C} : \boldsymbol{x}\boldsymbol{x}^\dagger, \tag{1}$$

However, the original scheme fails to impose strict second-order accuracy anywhere except the training points. We address this issue by introducing an improved correction that eliminates this error entirely from the previously unseen points as well (see § A.1 for details). The spatial derivatives of the fitted function can then be given by,

$$\frac{\partial \phi}{\partial x_\alpha} = b_\alpha \delta_{\alpha\gamma} + c_{\gamma\kappa} x_\gamma \delta_{\alpha\kappa} + c_{\gamma\kappa} x_\kappa \delta_{\alpha\gamma}, \qquad \frac{\partial^2 \phi}{\partial x_\alpha \partial x_\beta} = c_{\gamma\kappa} \delta_{\beta\gamma} \delta_{\alpha\kappa} + c_{\gamma\kappa} \delta_{\beta\kappa} \delta_{\alpha\gamma}. \tag{2}$$

where $\phi'(x)$ denotes the gradient (first derivative), and $\phi''(x)$ denotes the Laplacian (second derivative). In 1D, the equations simplify to $\frac{\partial \phi}{\partial x} = b + 2cx$ and $\frac{\partial^2 \phi}{\partial x^2} = 2c$. The information about the spatial derivatives are present in the fit constants. We present two ways to extract this information:

- The **explicit** way is to calculate each of these quantities sequentially as closed-form expressions at a test point $x$ (refer to § A.2 for details).
- Alternatively in the **implicit** way, we perturb the test point $x$ by a small $\varepsilon$, evaluate the predictions at $x_0 \pm \varepsilon$, and recover the required local constants implicitly by solving the resulting system of equations (see § A.3). This scheme can be expanded to higher dimensions by design.

Derivative extraction follows KBR training (with exact second-order correction), with no additional information requirement. We demonstrate the performance of two schemes in 1D using a number of tests below, with the experimentation details given in § A.4.

## 3 RESULTS

**Explicit scheme is more stable for unknown data:** We test the derivative learning schemes on an unknown test data of $5,000$ random points for two 1D functions, namely the Camel function with strong multiplicative coupling and the fully separable Rastrigin function. The general functions are given as follows:

$$f_{\text{camel}}(\boldsymbol{x}) = \frac{1}{2(k\sqrt{\pi})^D} \left( \exp\left( -\sum_{i=1}^{D} \frac{\left(x_i - \frac{1}{3}\right)^2}{k^2} \right) + \exp\left( -\sum_{i=1}^{D} \frac{\left(x_i - \frac{2}{3}\right)^2}{k^2} \right) \right) \tag{3}$$

$$f_{\text{ras}}(\boldsymbol{x}) = \sum_{i=1}^{D} \left[ x_i^2 - A\cos(2\pi x_i) + A \right] \tag{4}$$

where $\boldsymbol{x} = (x_1, x_2, ....., x_D) \in \mathbb{R}^D$, with $k = 0.2$ and $A = 10$. For the present study, $D = 1$. We refer to Fig. 1. It is observed that the explicit scheme shows a more stable behavior towards convergence, as compared to the implicit algorithm. For the Camel function, we observe a threshold sampling size beyond which the explicit scheme's performance sharply starts approaching second-order finite difference accuracy. This tendency is less apparent for the mathematically simpler Rastrigin function.

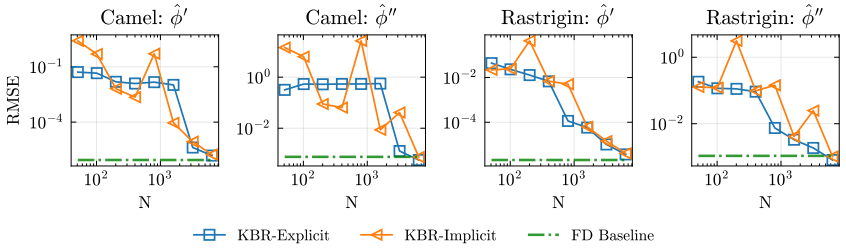


Figure 1: We show convergence of implicit and explicit derivative prediction schemes ($\hat{\phi}'$ and $\hat{\phi}''$) for two 1D functions with random sampling. RMSE is computed on a random test set of $5,000$ points, benchmarked against second-order non-uniform FD.

**Benchmarking on clean data:** We then move on to fields with known function values, eliminating the step of learning the function on unseen points. As a simple baseline validation study, we test our scheme against the so called Differential Neural Networks (Ledesma et al. (2020)), or DNN, from which the exact test conditions can be referred. Tab. 1 shows the present KBR scheme outperforming DNN significantly. The second-order accuracy is evident in the $x^2$ case, where explicit-scheme derivative errors reach machine precision.

Table 1: Comparison of non-normalized MSE (DNN vs KBR) on deployment/test data

| $\phi(x)$ | DNN | KBR-Implicit | | KBR-Explicit | |
|---|---|---|---|---|---|
| | $\phi'$ | $\phi'$ | $\phi''$ | $\phi'$ | $\phi''$ |
| $\sin(x)$ | $7.5 \times 10^{-6}$ | $6.60 \times 10^{-12}$ | $2.22 \times 10^{-4}$ | $3.12 \times 10^{-12}$ | $6.50 \times 10^{-5}$ |
| $x^2$ | $1.1 \times 10^{-3}$ | $8.14 \times 10^{-13}$ | $3.27 \times 10^{-4}$ | $3.24 \times 10^{-19}$ | $3.07 \times 10^{-14}$ |
| $\ln(x)$ | $3.6 \times 10^{-6}$ | $8.92 \times 10^{-12}$ | $2.52 \times 10^{-4}$ | $3.88 \times 10^{-12}$ | $1.34 \times 10^{-4}$ |

Convergence is tested on the 1D Camel function with increasing random sample sizes $N$, comparing derivative errors against non-uniform FD (Fornberg (1988); LeVeque (2007)). The top panel of Fig. 2 shows stable convergence for our schemes. While FD accuracy degrades on highly irregular grids (especially for $\phi''$, Crowder & Dalton (1971); De Hoog & Jackett (1985)), our method performs comparably. Both the theoretically second-order finite-difference (FD) scheme and the present scheme exhibit second-order convergence for $\phi'$ and first-order convergence for $\phi''$.

This scheme is generalizable in higher dimensions. A qualitative implementation result for the 2D Camel function is shown in § A.3.

**Implicit scheme outperforms explicit scheme for noisy data:** We refer to the bottom panel of Fig. 2. The implicit scheme is more robust with a much lesser error growth as the degree of corruption increases in the training data. For the smoothing cubic spline, a pre-determined window length proportional to the noise variance is used as a reference (see § A.4), whereas the KBR schemes aren't given any such pre-conditioning. It is apparent that the standard FD requires regularization for noisy data and isn't applicable here as is (Chartrand (2011)).

**Application of KBR in PDE Solutions:** The general conservative form of the hyperbolic PDEs considered in this work is given by

$$\frac{\partial \phi}{\partial t} + \frac{\partial F(\phi)}{\partial x} = 0, \tag{5}$$

where $\phi$ denotes the solution field and $F(\phi)$ represents the corresponding flux. We integrate KBR into conservative hyperbolic PDE solvers by replacing conventional flux evaluations with

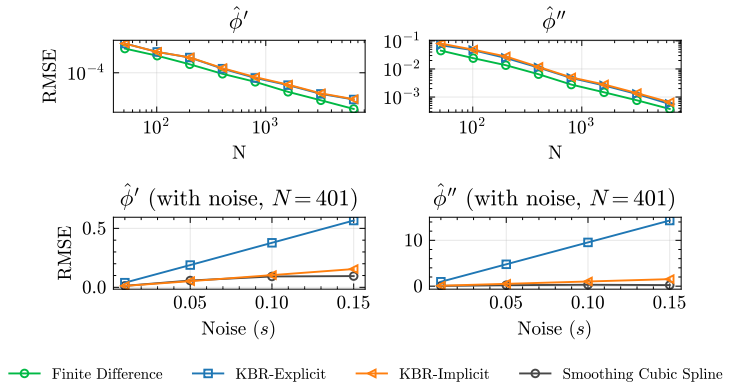


Figure 2: Convergence plot (top panel) comparing errors from KBR-predicted spatial derivatives with second-order FD for the 1D Camel function on a non-uniform grid. Bottom panel shows the errors in extracted derivative errors when the training data is corrupted with varying degrees of gaussian noise. Cubic spline-smoothened results are given as a reference.

KBR-based predictions, thereby promoting consistency with fundamental conservation laws beyond heuristic enforcement. The single trainable parameter $\theta$ may be retrained at every time step or periodically over multiple steps. Training is performed using grid-point data, while cell interfaces are treated as deployment locations for flux prediction owing to minimal training cost. The proposed approach is evaluated on two benchmark problems to assess the integration stability: the one-dimensional inviscid Burgers' equation with a Riemann-type initial condition on a uniform grid, and the one-dimensional Euler equations for the Sod shock tube problem (Sod (1978)) on a non-uniform grid. Implementation details are provided in § A.4.

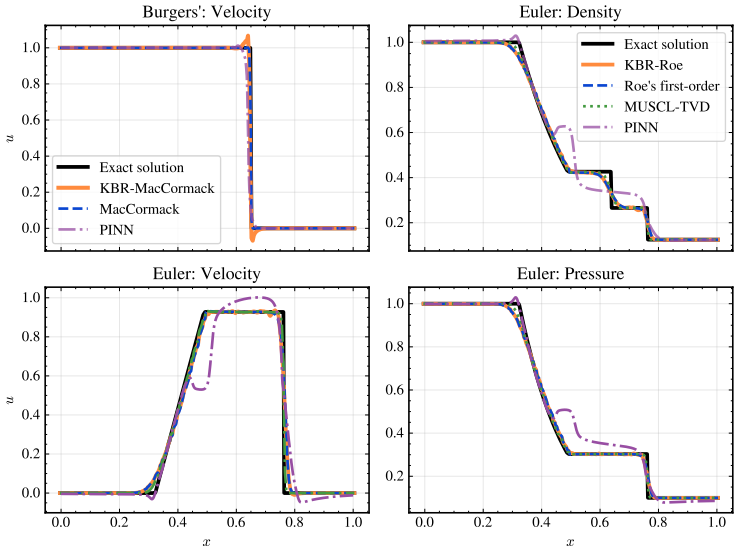


Figure 3: The figure shows the numerical solution at time $t = 0.3$, and $t = 0.15$ for 1D inviscid Burgers' and 1D Euler equations respectively, using strict FD and KBR-integrated schemes on a grid size of 251 points. Trained PINN predictions and exact solutions are given as a reference.

The KBR schemes stay stable across both PDEs (Fig. 3), though standard Gibbs oscillations (Wilbraham (1848)) appear for the Burgers' equation. This is because unlike the standard MacCormack, the KBR-integrated scheme uses a predicted second-order flux value at the cell faces. For the Euler equations, the KBR-integrated first-order Roe scheme is shown to achieve performance comparable to the standard Roe scheme in terms of shock resolution and error metrics (Table 2), with no error blow-ups over time. The results are also compared with those obtained using the higher-order flux-

Table 2: Shock-capturing metrics for density $\rho$ in the Sod shock tube problem on a non-uniform grid (KBR-Roe, Roe first-order, MUSCL-TVD). Region 1: shock $[0.62, 0.67]$, post-shock $[0.64, 0.72]$; Region 2: shock $[0.73, 0.77]$, post-shock $[0.755, 0.80]$.

| Region | Metrics | KBR-Roe | Roe | MUSCL-TVD |
|---|---|---|---|---|
| Shock 1 (Region 1) | L1 Error (Global) | $9.409 \times 10^{-3}$ | $1.014 \times 10^{-2}$ | $5.311 \times 10^{-3}$ |
| | L$\infty$ Error (Global) | $9.305 \times 10^{-2}$ | $9.021 \times 10^{-2}$ | $8.614 \times 10^{-2}$ |
| | Shock Thickness | $2.874 \times 10^{-2}$ | $3.221 \times 10^{-2}$ | $2.450 \times 10^{-2}$ |
| | Post-shock Osc. | $4.919 \times 10^{-2}$ | $5.162 \times 10^{-2}$ | $4.392 \times 10^{-2}$ |
| | Total Variation | $1.010 \times 10^{-1}$ | $9.646 \times 10^{-2}$ | $1.265 \times 10^{-1}$ |
| Shock 2 (Region 2) | L1 Error | $9.409 \times 10^{-3}$ | $1.014 \times 10^{-2}$ | $5.311 \times 10^{-3}$ |
| | L$\infty$ Error | $9.305 \times 10^{-2}$ | $9.021 \times 10^{-2}$ | $8.614 \times 10^{-2}$ |
| | Shock Thickness | $3.087 \times 10^{-2}$ | $1.638 \times 10^{-2}$ | $7.641 \times 10^{-3}$ |
| | Post-shock Osc. | $6.858 \times 10^{-2}$ | $7.641 \times 10^{-2}$ | $5.433 \times 10^{-2}$ |
| | Total Variation | $1.067 \times 10^{-1}$ | $7.656 \times 10^{-2}$ | $1.295 \times 10^{-1}$ |

limited MUSCL–TVD scheme (Laney (1998)), though a superiority is not expected given the order of MUSCL–TVD. Note that the variation in the observation window does not change relative trends significantly. The incorporation of KBR within TVD (Total Variation Diminishing) and WENO (Weighted Essentially Non-Oscillatory) schemes is currently under investigation.

In the examples considered, training of KBR is significantly faster than PINNs, using lesser computing resources. The objective of this demonstration is not to establish superiority over standard numerical schemes, but to illustrate the dynamic stability and competitive behavior of the proposed learning-incorporated approach. Integrating learning-based components with traditional numerical PDE solvers in this manner may provide an additional framework for robust spatial derivative evaluation, particularly for high-dimensional problems and unstructured grids.

## 4 Conclusion and Outlook

We have presented two spatial derivative learning schemes as an expansion of KBR, applicable in known, unknown and corrupt fields primarily in 1D and preliminary in 2D. Termed here as implicit and explicit, both the schemes achieve second-order FD accuracy for derivative learning while remaining reasonably robust to noise. Quantitative evaluation using the underlying theory and convergence rates provides a principled measure of performance, thereby enhancing the interpretability of the learning scheme. The presented KBR-integrated PDE simulations act as a step towards incorporating machine learning into PDE solvers in a manner that adheres to the basic conservation laws in a non-heuristic way. Future work targets improved implicit scheme stability and extension to high-dimensional PDE simulations in random point-clouds. The relevant scripts along with the result datasets are available at `https://github.com/abhishekganguly808/kbr-derivatives`.

### Acknowledgments

A. G. thanks Guruprasad S for all the meaningful discussions on the numerical solutions of hyperbolic PDEs.

## REFERENCES

Vytautas Abramavicius, Evan Philip, Kaonan Micadei, Charles Moussa, Mario Dagrada, Vincent E. Elfving, Panagiotis Barkoutsos, and Roland Guichard. Evaluation of derivatives using approximate generalized parameter shift rule, 2025.

Kober Bardenhagen and Sulsky. The material-point method for granular materials. *Computer Methods in Applied Mechanics and Engineering*, 187:529–541, 2001.

Atilim Günes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018.

D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2(3):321–355, 1988.

Rick Chartrand. Numerical differentiation of noisy, nonsmooth data. *International Scholarly Research Notices*, 2011, 2011.

H. J. Crowder and C. Dalton. Errors in the use of nonuniform mesh systems. *Journal of Computational Physics*, 7(1):32–45, 1971.

F. De Hoog and D. Jackett. On the rate of convergence of finite difference schemes on nonuniform grids. *ANZIAM Journal*, 26(3):247–256, 1985.

Bengt Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation*, 51(184):699–706, 1988.

Abhisek Ganguly, Alessandro Gabbana, Vybhav Rao, Sauro Succi, and Santosh Ansumali. A kinetic-based regularization method for data science applications. *Machine Learning: Science and Technology*, 6(3):035035, 2025.

Amir Gholami, Long Jin, Iman Kasaian, Kurt Keutzer, Jian Sun, Michael W. Mahoney, and George Biros. Anode: Unconditionally accurate memory-efficient gradients for neural odes. *Journal of Machine Learning Research*, 20(149):1–43, 2019.

Gingold and Monaghan. Smoothed particle hydrodynamics: theory and application. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.

Jan Hückelheim, Harshitha Menon, William Moses, Bruce Christianson, Paul Hovland, and Laurent Hascoët. A taxonomy of automatic differentiation pitfalls. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 14(5):e1555, 2024.

D. N. Jagtap, E. Kharazmi, and G. E. Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, 2020.

Prashant Kumar and Rajesh Ranjan. A robust data-free physics-informed neural network for compressible flows with shocks. *Computers & Fluids*, 308:106975, 2026.

Culbert B. Laney. *Computational Gasdynamics*. Cambridge University Press, Cambridge, UK, 1998.

Sergio Ledesma, Dora-Luz Almanza-Ojeda, Mario-Alberto Ibarra-Manzano, Eduardo Cabal Yepez, Juan Gabriel Avina-Cervantes, and Pascal Fallavollita. Differential neural networks (dnn). *IEEE Access*, 8:156530–156538, 2020.

Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. SIAM, 2007.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *ICLR*, 2021.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Karniadakis. Learning nonlinear operators via deeponet. *Nature Machine Intelligence*, 3:218–229, 2021.

Charles C. Margossian. A review of automatic differentiation and its efficient implementation. *arXiv preprint arXiv:1811.05031*, 2019.

Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68:1703–1759, 2005.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 2019.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. ISBN 9780262182539.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Gary A. Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1–31, 1978.

Hongqiao Wang and Xiang Zhou. Explicit estimation of derivatives from data and differential equations by gaussian process regression. *arXiv preprint arXiv:2004.05796*, 2020.

Sifan Wang, Yujun Teng, and Paris Perdikaris. When and why pinns fail to train. *Journal of Computational Physics*, 449:110768, 2022.

Henry Wilbraham. On a certain periodic function. *The Cambridge and Dublin Mathematical Journal*, 3:198–201, 1848.

# A Appendix

## A.1 Exact Second-Order Correction

The original KBR self-correction yields interpolated second-moment errors on test points, and the second-order accuracy on unseen points is no exact in the original work. We prove exact second-order accuracy via targeted error removal, and address this shortcoming.

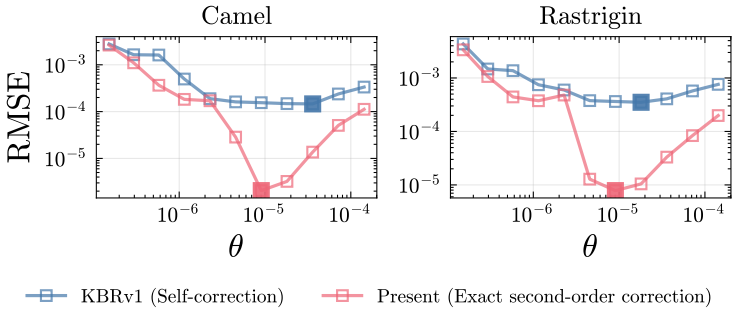


Figure 4: Comparison between the loss function landscapes for the case of self-correction and the proposed exact second-order correction, with respect to the single parameter $\theta$ and validation error in KBR. Training data consists of 501 randomly sampled points, with an 80:20 split between train and validation.

**Lemma A.1.1** (Exact Second-Order KBR). *Let $\{x_i, \phi(x_i)\}_{i=1}^N$ be a set of training points, For fully converged Lagrange multipliers $\tilde{x}$, define $\tilde{\xi}_i = x_i - \tilde{x}$. The corrected second-order prediction,*

$$\hat{\phi}^{(2)}(x) = \sum_i \phi(x_i) P(\|\tilde{\xi}_i\|, \theta) - \Theta(x) \sum_i \frac{\phi(x_i) - \phi^{(1)}(x)}{\Theta(x_i)} P(\|\tilde{\xi}_i\|, \theta), \tag{6}$$

*satisfies $\phi^{(2)}(x) = \phi(x)$ exactly for quadratic $\phi(x) = a + bx + cx^2$ at test points $x \notin \{x_i\}$.*

*Proof.* Define second-moment errors:
$\Theta(x) = \sum_i x_i^2 P(\|\tilde{\xi}_i\|, \theta) - x^2$ and $\Theta(x_k) = \sum_i x_i^2 P(\|\tilde{\xi}_i\|, \theta) - x_k^2$, for $k \in i$.

Converged first moments give exact first order prediction, $\hat{x}_i = x_i$. The original correction therefore expands as:

$$\begin{aligned}
\phi^{(2)}(x) &= \sum_i \big(2\phi(x_i) - \phi^{(1)}(x_i)\big) P(\|\tilde{\xi}_i\|, \theta) \\
&= a + bx + c \sum_i (2x_i^2 - \widehat{x^2}_i) P(\|\tilde{\xi}_i\|, \theta) \\
&= \phi(x) + c\,\big(\Theta(x) - \hat{\Theta}(x)\big)
\end{aligned} \tag{7}$$

where $\hat{\Theta}(x) = \sum_i \Theta(x_i) P(\|\tilde{\xi}_i\|, \theta)$ are the interpolated training errors from $\Theta(x_i)$, which fails to cancel out the thermal error at $x$. Also, $\widehat{x_i^2} = \sum_i x_i^2 P(\|\tilde{\xi}_i\|, \theta)$

For Eq. 6, the correction term simplifies:

$$\begin{aligned}
&\Theta(x) \sum_i \frac{\phi(x_i) - \phi^{(1)}(x)}{\Theta(x_i)} P(\|\tilde{\xi}_i\|, \theta) \\
&= c\,\Theta(x) \sum_i \frac{x_i^2 - (x_i^2 + \Theta(x_i))}{\Theta(x_i)} P(\|\tilde{\xi}_i\|, \theta) = c\,\Theta(x)
\end{aligned} \tag{8}$$

Thus, $\phi^{(2)}(x) = \sum_i \phi(x_i) P(\|\tilde{\xi}_i\|, \theta) - c\,\Theta(x) = \phi(x)$. □

This ensures exact second-order representation on unseen points locally. The immediate effect can be seen in Fig. 4, where the current corrections lead to a significantly lower validation error with respect to the trainable kernel parameter $\theta$.

**Algorithm 1** Exact second-order correction KBR

**Require:** Training data $\{x_i, \phi(x_i)\}_{i=1}^N$, test points $\{x\}$, kernel parameter $\theta$, max iterations $K$
**Ensure:** Second-order prediction $\hat{\phi}(x)$

**for** each test point $x$ **do**
  $\tilde{x} \leftarrow \text{LagrangeMultiplier}(x, \{x_i\}, \theta, K)$ ▷ Enforce moment constraint
  **for** $i = 1$ to $N$ **do**
    $P_i \leftarrow P(\|x_i - \tilde{x}\|, \theta)$
  **end for**
  $\phi^{(1)}(x) \leftarrow \sum_i \phi(x_i) P_i$ ▷ First-order prediction
  $\Theta(x) \leftarrow \sum_i (x_i^2 - x^2) P_i$ ▷ Second moment error
  **for** $i = 1$ to $N$ **do**
    $\Theta(x_i) \leftarrow \sum_j (x_j^2 - x_i^2) P_j$
  **end for**
  **for** $i = 1$ to $N$ **do**
    $c_i \leftarrow \dfrac{\phi(x_i) - \phi^{(1)}(x)}{\Theta(x_i)}$
  **end for**
  $\hat{\phi}(x) \leftarrow \phi^{(1)}(x) + \Theta(x) \sum_i c_i P_i$ ▷ Second-order correction
**end for**

### A.2 Explicit scheme for derivative calculation

**Laplacian Calculation:** We assume that the local field in the neighborhood of a test point $x_0$ is well represented by the quadratic polynomial $\phi(x) = a + bx + cx^2$, the exact second derivative is then $\phi''(x_0) = 2c$. Consider the uncorrected prediction at $x_0$,

$$\phi^{(0)}(x_0) = \sum_i \phi(x_i) P(\|\xi_i\|, \theta) = a + b\hat{x}_0 + c \sum_i x_i^2 P(\|\xi_i\|, \theta), \tag{9}$$

where $\hat{x}_0 = \sum_i x_i P(\|\xi_i\|, \theta)$. Rewriting,

$$\phi^{(0)}(x_0) = \phi(x_0) + b(\hat{x}_0 - x_0) + c\,\Theta^{(0)}(x_0), \tag{10}$$

After enforcing first-moment consistency via Lagrange multipliers, $\hat{x}_0 = x_0$, and the first-order corrected prediction becomes

$$\phi^{(1)}(x_0) = \sum_i \phi(x_i) P(\|\tilde{\xi}_i\|, \theta) = \phi(x_0) + c\,\Theta(x_0), \tag{11}$$

where $\Theta$ is defined in the proof of the previous section .

Subtracting the two predictions isolates the quadratic contribution,

$$\phi^{(0)}(x_0) - \phi^{(1)}(x_0) = c\,\Theta(x_0). \tag{12}$$

Since $\phi''(x_0) = 2c$, the Laplacian is obtained as

$$\phi''(x_0) = \frac{2}{\Theta(x_0)} \big[\phi^{(0)}(x_0) - \phi^{(1)}(x_0)\big]. \tag{13}$$

If the field value $\phi(x_0)$ is already known, it may be used directly in place of $\phi^{(1)}(x_0)$.

**Gradient Calculation:** Using the same quadratic representation, $\phi(x) = a + bx + cx^2$, the exact gradient at $x_0$ is

$$\phi'(x_0) = b + 2cx_0. \tag{14}$$

The uncorrected prediction can again be written as

$$\phi^{(0)}(x_0) = \phi(x_0) + b(\hat{x}_0 - x_0) + c\,\Theta^{(0)}(x_0), \tag{15}$$

while the first-moment corrected prediction satisfies

$$\phi^{(1)}(x_0) = \phi(x_0) + c\,\Theta(x_0). \tag{16}$$

Subtracting the two expressions yields

$$\phi^{(0)}(x_0) - \phi^{(1)}(x_0) = b(\hat{x}_0 - x_0) + c\big(\Theta^{(0)}(x_0) - \Theta(x_0)\big). \tag{17}$$

The gradient is recovered as

$$\phi'(x_0) = \frac{1}{\hat{x}_0 - x_0}\left(\phi^{(0)}(x_0) - \phi^{(1)}(x_0) - \frac{\phi''(x_0)}{2}\big[\Theta^{(0)}(x_0) - \Theta(x_0)\big]\right). \tag{18}$$

**Algorithm 2** Explicit scheme for KBR derivatives

**Require:** Training data $\{x_i, \phi(x_i)\}_{i=1}^N$, test points $\{x\}$, kernel parameter $\theta$, max iterations $K$
**Ensure:** Gradient $\phi'(x)$ and Laplacian $\phi''(x)$

**for** each test point $x$ **do**
  $\tilde{x} \leftarrow \text{LagrangeMultiplier}(x, \{x_i\}, \theta, K)$ ▷ Enforce first moment
  **for** $i = 1$ to $N$ **do**
    $P_i \leftarrow P(\|x_i - \tilde{x}\|, \theta)$
    $P_i^{(0)} \leftarrow P(\|x_i - x\|, \theta)$
  **end for**
  $\phi^{(0)}(x) \leftarrow \sum_i \phi(x_i) P_i^{(0)}$ ▷ Uncorrected prediction
  $\phi^{(1)}(x) \leftarrow \sum_i \phi(x_i) P_i$ ▷ Corrected prediction
  $\Theta^{(0)}(x) \leftarrow \sum_i (x_i^2 - x^2) P_i^{(0)}$
  $\Theta(x) \leftarrow \sum_i (x_i^2 - x^2) P_i$
  $\phi''(x) \leftarrow \dfrac{2\big(\phi^{(0)}(x) - \phi^{(1)}(x)\big)}{\Theta(x)}$ ▷ Laplacian
  $\hat{x} \leftarrow \sum_i x_i P_i^{(0)}$
  $\phi'(x) \leftarrow \dfrac{1}{\hat{x} - x}\left(\phi^{(0)}(x) - \phi^{(1)}(x) - \dfrac{\phi''(x)}{2}\big[\Theta^{(0)}(x) - \Theta(x)\big]\right)$ ▷ Gradient
**end for**

### A.3 Implicit scheme for derivative calculation

Consider the quadratic fit is given as

$$\phi(x) = a + bx + cx^2 \tag{19}$$

For a point $x_0$ of interest, three predictions can be made using slight perturbations in the Lagrange multipliers $\tilde{\xi}_i$, which goes into the kernel function $P$. It can be represented as:

$$\begin{aligned} \tilde{\xi}_i &= x - \tilde{x}_0, \\ \tilde{\xi}_i^- &= x - (\tilde{x}_0 - \varepsilon), \\ \tilde{\xi}_i^+ &= x - (\tilde{x}_0 + \varepsilon), \end{aligned} \tag{20}$$

where, $\varepsilon$ can be a small percentage of effective distance between the points, to preserve the local nature of derivative calculation. The problem can then be represented as:

$$\underbrace{\begin{bmatrix} \sum_i P(\tilde{\xi}_i) & \sum_i x_i P(\tilde{\xi}_i) & \sum_i x_i^2 P(\tilde{\xi}_i) \\ \sum_i P(\tilde{\xi}_i^-) & \sum_i x_i P(\tilde{\xi}_i^-) & \sum_i x_i^2 P(\tilde{\xi}_i^-) \\ \sum_i P(\tilde{\xi}_i^+) & \sum_i x_i P(\tilde{\xi}_i^+) & \sum_i x_i^2 P(\tilde{\xi}_i^+) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{X} = \underbrace{\begin{bmatrix} \hat{\phi}(x_0) \\ \hat{\phi}^+(x_0) \\ \hat{\phi}^-(x_0) \end{bmatrix}}_{B}$$

In the present study, $\varepsilon$ is chosen as $0.025\sqrt{\theta_o}$, where $\theta_o$ denotes the optimal kernel parameter obtained from the relation $\theta_o/d_{\text{typ}}^2 = k$. The dimensionless parameter $k$ is optimized instead of $\theta$. Here, $d_{\text{typ}}$ represents the effective local distance, evaluated as the average $k$-nearest-neighbor distance using five points. Empirically, a sweep of 10–15 iterations over $k \in [0.01, 100]$ is found to yield favorable local minima of the loss function.

**Algorithm 3** Implicit scheme for KBR derivatives

**Require:** Training data $\{x_i, \phi(x_i)\}_{i=1}^N$, test points $\{x\}$, kernel parameter $\theta$, max iterations $K$
**Ensure:** Gradient $\phi'(x)$ and Laplacian $\phi''(x)$

**for** each test point $x$ **do**
  $\tilde{x} \leftarrow \text{LagrangeMultiplier}(x, \{x_i\}, \theta, K)$ ▷ Moment consistency
  $\tilde{x}^- \leftarrow \tilde{x} - \varepsilon, \quad \tilde{x}^+ \leftarrow \tilde{x} + \varepsilon$ ▷ Perturbation
  **for** $i = 1$ to $N$ **do**
    $P_i \leftarrow P(\|x_i - \tilde{x}\|, \theta)$
    $P_i^- \leftarrow P(\|x_i - \tilde{x}^-\|, \theta)$
    $P_i^+ \leftarrow P(\|x_i - \tilde{x}^+\|, \theta)$
  **end for**
  Assemble matrix $A$:

$$A = \begin{bmatrix} \sum_i P_i & \sum_i x_i P_i & \sum_i x_i^2 P_i \\ \sum_i P_i^- & \sum_i x_i P_i^- & \sum_i x_i^2 P_i^- \\ \sum_i P_i^+ & \sum_i x_i P_i^+ & \sum_i x_i^2 P_i^+ \end{bmatrix}$$

  Assemble vector $B$:

$$B = \begin{bmatrix} \hat{\phi}(x) \\ \hat{\phi}^-(x) \\ \hat{\phi}^+(x) \end{bmatrix}$$

  Solve $AX = B$ for $X = [a\ b\ c]^T$ ▷ Local quadratic fit
  $\phi''(x) \leftarrow 2c$ ▷ Laplacian
  $\phi'(x) \leftarrow b + 2cx$ ▷ Gradient
**end for**

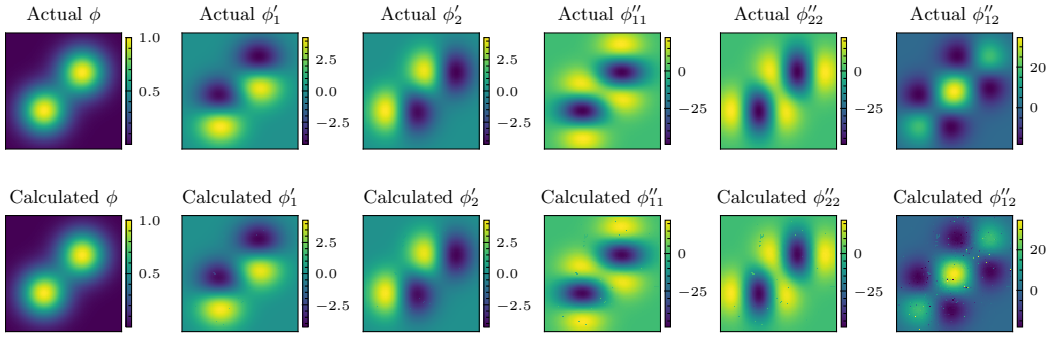


Figure 5: The actual and predicted fields, gradients, Laplacian and Hessian is shown for the 2D Camel function on a uniform $101 \times 101$ grid using KBR Implicit spatial derivative scheme.

**2D Formulation**: Let us take Eq. 1. At a point of interest $\boldsymbol{x} = (x_1, x_2)$, the gradients, Laplacian and Hessian can be represented as follows:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{bmatrix}, \tag{21}$$

$$\nabla\phi = \boldsymbol{b} + 2\boldsymbol{C}\boldsymbol{x}, \tag{22}$$

$$\nabla^2\phi = 2\boldsymbol{C}. \tag{23}$$

We can generate a response matrix using perturbations in the test point along principle directions, i.e., $(x_1 \pm \varepsilon, x_2 \pm \varepsilon)$, along with correction-invoked and uncorrected predictions at point $\boldsymbol{x}$. A qualitative result for the 2D Camel function (Eq. 3) is given in Fig. 5.

### A.4 Numerical experiment details

**Training splits and Error:** The train–validation split is set to 90:10 for derivative evaluations. The input field is normalized to $[-1, 1]$ using $\max|\phi(x)|$, and the spatial grid is normalized to $[0, 1]$ unless stated otherwise, in order to maintain a consistent reference frame. The RMSE is normalized accordingly and defined as

$$\mathrm{RMSE} = \frac{1}{\max|\phi(x)|}\left\langle \left(\hat{\phi}(x) - \phi(x)\right)^2 \right\rangle^{1/2}, \tag{24}$$

where $\hat{\phi}(x)$ and $\phi(x)$ denote the predicted and exact field values, respectively. Additional error metrics used for the analysis of 1D Euler equation are tabulated in Tab. 3.

Table 3: Metrics used for shock-capturing evaluation in the Sod shock tube problem.

| Metric | Description |
|---|---|
| L1 error | Mean absolute deviation from exact solution |
| L$\infty$ error | Maximum absolute deviation |
| Shock thickness | 10%–90% transition width of the discontinuity (measures smearing) |
| Post-shock oscillation | Maximum deviation in the post-shock region |
| Total variation | Sum of absolute differences in the shock window |

**Cubic smoothing spline:** The smoothing factor $w$ for the cubic spline is selected according to the heuristic $w \approx N\sigma^2$, with $\sigma = s/3$. This approximates the noise standard deviation induced by the multiplicative Gaussian perturbation with scale $s$, where $s = \%\,\text{noise}/100$. This keeps the corrupted function value at $\phi(x) \pm s\phi(x)$.

**KBR integration into PDE solvers:** For the PDE section of the results, the inviscid Burgers' equation and the Euler equation are solved in 1D. Tables 4 and 5 contain the details about the problem setups and the corresponding PINN hyperparameters, respectively.

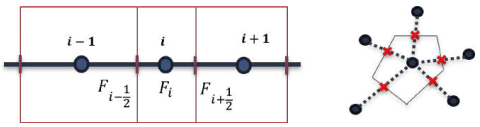


Figure 6: Schematic of (left) a one-dimensional non-uniform grid with interface flux locations, and (right) a 2D Voronoi diagram representation constructed from scattered points, representing an unstructured control volume discretization.

For the inviscid Burger's equation, we use the conservative MacCormack scheme (Laney (1998)).

$$\frac{\partial u}{\partial t} + \frac{\partial F(u)}{\partial x} = 0,$$

Where, $F(u) = \frac{1}{2}u^2$, is the flux. The MacCormack predictor-corrector scheme for a non-uniform grid (see Fig. 6) is given by,

- Predictor:

$$U_i^* = U_i^n - \frac{\Delta t}{x_{i+1} - x_i}\left(F_{i+1}^n - F_i^n\right), \tag{25}$$

- Corrector:

$$U_i^{n+1} = \frac{1}{2}\left[U_i^n + U_i^* - \frac{\Delta t}{x_i - x_{i-1}}\left(F_i^* - F_{i-1}^*\right)\right]. \tag{26}$$

The KBR implementation uses a finite volume-style strategy, using predicted flux value at cell interfaces. The equations then become:

- Predictor:

$$U_i^* = U_i^n - \frac{\Delta t}{x_{i+\frac{1}{2}} - x_i} \left( \hat{F}_{i+\frac{1}{2}}^n - F_i^n \right), \tag{27}$$

- Corrector:

$$U_i^{n+1} = \frac{1}{2} \left[ U_i^n + U_i^* - \frac{\Delta t}{x_i - x_{i-\frac{1}{2}}} \left( F_i^* - \hat{F}_{i-\frac{1}{2}}^* \right) \right], \tag{28}$$

Where, $\hat{F}_{i+\frac{1}{2}}$ and $\hat{F}^*_{i-\frac{1}{2}}$ are predicted fluxes at the interfaces. It can be easily simplified for the uniform grid here. In high dimensional problems with unstructured grids (see Fig. 6), integrating the KBR learning scheme in this manner would allow to keep the schemes strictly conservative in the domain.

The 1D Euler equation is solved using Roe's first order scheme (Laney (1998)). In a traditional sense, it uses an average central flux along with a dissipative component. At a point $x = x_{i-\frac{1}{2}}$, the average central flux is $\frac{1}{2}\left(F(x_{i-1}) + F(x_i)\right)$. The KBR-integrated scheme however does not perform this mean averaging and uses the predicted central flux $\hat{F}_{x-\frac{1}{2}}$ at the interface instead.

Table 4: Conservative form, initial conditions and boundary conditions for the numerical solution of the tested PDEs.

| **PDE** | **Conservative form** | **IC** ($t = 0$) | **BC** |
|---|---|---|---|
| Burgers' | $\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = 0$ | $u = \begin{cases} 1 & x < 0.5 \\ 0 & x \geq 0.5 \end{cases}$ | $u(0,t) = 1$<br>$u(1,t) = 0$ |
| Euler | $\frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix} \rho u \\ \rho u^2 + p \\ u(\rho E + p) \end{pmatrix} = \mathbf{0}$ | $\begin{pmatrix} \rho \\ u \\ p \end{pmatrix} = \begin{cases} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} & x < 0.5 \\ \begin{pmatrix} 0.125 \\ 0 \\ 0.1 \end{pmatrix} & x \geq 0.5 \end{cases}$ | outflow |

Table 5: PINN architecture and training hyperparameters used for both Euler equation (Sod shock tube test) and inviscid Burgers'equation with Riemann type initial conditions.

| **Component** | **Value / Description** | **Notes** |
|---|---|---|
| Input dimensions | 2 | $(x, t)$ |
| Output dimensions | 1 | $u$ |
| Hidden layers | 4 | |
| Neurons per hidden layer | 64 | |
| Activation function (hidden) | $\tanh$ | |
| Output activation | softplus + $10^{-6}$ | Ensures positivity |
| Kernel initializer | Glorot Uniform | |
| Optimizer | Adam | |
| Learning rate | $10^{-3}$ | |
| Total collocation points | 60,000 | |
| Initial condition points | 600 | |
| Boundary points per side | 300 | |
| Training epochs | 50,000 | |