∂^∞-Grid: A Neural Differential Equation Solver with Differentiable Feature Grids

NAVAMI KAIRANDA, Max Planck Institute for Informatics, Germany
SHANTHIKA NAIK, Indian Institute of Technology Jodhpur, India
MARC HABERMANN, Max Planck Institute for Informatics, Germany
AVINASH SHARMA, Indian Institute of Technology Jodhpur, India
CHRISTIAN THEOBALT, Max Planck Institute for Informatics, Germany
VLADISLAV GOLYANIK, Max Planck Institute for Informatics, Germany

We present ∂^∞-*Grid*, a differentiable grid-based representation for efficiently solving differential equations (DEs). Grid-based neural fields (*e.g.*, Instant-NGP, K-Planes) train fast via spatial locality but rely on linear interpolation, which lacks higher-order differentiability and cannot solve DEs. On the other hand, coordinate-based MLPs are infinitely differentiable but computationally expensive, requiring hours of training. ∂^∞-*Grid* combines the efficiency of feature grids with radial basis function (RBF) interpolation—smooth and C^∞—enabling accurate computation of gradients, Jacobians, and Laplacians for solving DEs. Our multi-resolution co-located grid captures high-frequency solutions with fast global gradient propagation. We validate ∂^∞-*Grid* on Poisson, Helmholtz, and Kirchhoff-Love thin-shell equations, achieving 5–20× speed-ups over coordinate-based MLPs while maintaining comparable accuracy. Project page: https://4dqv.mpi-inf.mpg.de/DInf-Grid/.

CCS Concepts: • **Computing methodologies** → **Modeling and simulation**; **Machine learning**.

1 INTRODUCTION

Finding representations that accurately and efficiently model fields is fundamental for solving physics governed by differential equations (DEs). Many such problems map spatial or spatio-temporal coordinates $\mathbf{x} \in \mathbb{R}^d$ to fields $\mathbf{u}(\mathbf{x})$ satisfying:

$$\mathcal{F}\Big(\mathbf{x}, \mathbf{u}, \nabla_{\mathbf{x}}\mathbf{u}, \nabla^2_{\mathbf{x}}\mathbf{u}, \ldots; g(\mathbf{x})\Big) = 0. \tag{1}$$

Current neural solvers predominantly use coordinate-based MLPs such as Siren [Sitzmann et al. 2020]. These are infinitely differentiable but computationally expensive—training times of hours—due to global weight updates per iteration. Grid-based representations [Fridovich-Keil et al. 2023; Müller et al. 2022] train faster by exploiting spatial locality, but rely on d-linear interpolation. This is only C^1 in the grid interior and C^0 at nodes; second-order derivatives vanish, making them unable to solve DEs.

We address both limitations with ∂^∞-*Grid*, which combines the efficiency of feature grids with radial basis function (RBF) interpolation—smooth and C^∞. Our core contributions are:

- A feature grid-based representation and formulation for recovering physical fields from DEs.
- Differentiable RBF interpolation enabling accurate higher-order derivative computation.
- Multi-resolution grids for fast global gradient propagation.

We validate ∂^∞-*Grid* on the Poisson equation for image reconstruction, the Helmholtz equation for wavefields, the Kirchhoff-Love boundary-value problem for cloth simulation, the Eikonal equation for signed distance functions, and heat and advection equations; see Fig. 1. Our method achieves 5–20× speed-ups over Siren while maintaining comparable accuracy.

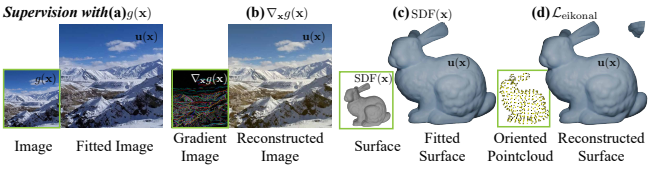


Fig. 1. Our proposed method, ∂^∞-*Grid*, is a feature grid-based representation capable of accurately modelling signals and their higher-order derivatives. It captures high-frequency details when fitting images and SDFs directly (a,c) and, unlike prior grid methods [Chen et al. 2023a; Müller et al. 2022] (Figs. IV and VII and Tab. 1), also succeeds with gradient and Eikonal supervision (b,d). Inputs are outlined in green.

2 METHOD

Our goal is to represent and solve for the unknown field $\mathbf{u}$ subject to $\mathcal{F}$ in Eq. (1); see Fig. 2. We parametrise $\mathbf{u}(\mathbf{x}) = \mathbf{d}(\mathbf{f}(\mathbf{x}; \mathbf{F}); \Theta)$, where $\mathbf{d}$ is a small MLP or linear decoder and the feature encoder $\mathbf{f}$ interpolates from a learnable grid $\mathbf{F}$. This exploits spatial locality: only the small neighbourhood of grid nodes adjacent to $\mathbf{x}$ participates in each gradient update, unlike coordinate-based MLPs where all weights are updated globally. We optimise $\mathbf{F}$ and Θ by minimising the PDE residual over sampled domain points, computing all required gradients, Jacobians, and Laplacians of $\mathbf{u}$ via autograd. To guarantee uniqueness, Dirichlet boundary conditions are imposed as hard constraints via a distance-based modulation:

$$\mathbf{u}(\mathbf{x}) = \mathbf{d}(\mathbf{f}(\mathbf{x}; \mathbf{F}); \Theta)\, \mathcal{B}(\mathbf{x}) + \mathbf{h}(\mathbf{x})\big(1 - \mathcal{B}(\mathbf{x})\big), \tag{2}$$

where $\mathcal{B}(\mathbf{x}) = 1 - e^{-\|\mathbf{x} - \mathbf{x}_{\partial\Omega}\|^2/\sigma}$ enforces $\mathbf{u}(\mathbf{x}) = \mathbf{h}(\mathbf{x})$ on $\partial\Omega$ [Kairanda et al. 2024; Lu et al. 2021].

Feature Grid Formulation. The domain Ω is discretised into a regular grid; each node stores a learnable feature vector. Given a query $\mathbf{x}$, the interpolated feature is:

$$\mathbf{f}(\mathbf{x}) = \sum_{i \in \mathcal{N}(\mathbf{x})} w(\mathbf{x}, \mathbf{x}_i)\, \mathbf{F}(\mathbf{x}_i), \tag{3}$$

where $\mathcal{N}(\mathbf{x})$ is the set of neighbouring nodes and w are interpolation weights. Only the small neighbourhood adjacent to $\mathbf{x}$ participates in each update, enabling efficient, localised training. We minimise the PDE residual over a sampled dataset $\mathcal{D} = \{(\mathbf{x}_i, g(\mathbf{x}_i))\}_i$:

$$\mathcal{L} = \sum_{\mathbf{x}_i \in \mathcal{D}} \mathcal{F}\Big(\mathbf{x}_i,\ \mathbf{u}_i,\ \nabla_{\mathbf{x}_i}\mathbf{u},\ \nabla^2_{\mathbf{x}_i}\mathbf{u},\ \ldots;\ g(\mathbf{x}_i)\Big), \tag{4}$$

with all derivatives computed via autograd through the RBF interpolation.

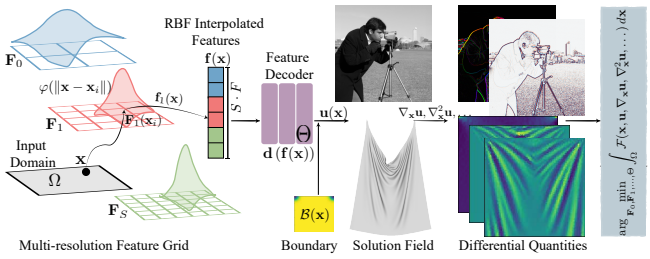


Fig. 2. Illustration of ∂^∞-*Grid*. Query coordinates $\mathbf{x}$ are encoded via RBF interpolation over multi-scale learnable feature grids $\{\mathbf{F}_s\}$, producing scale-wise features $\{\mathbf{f}_s(\mathbf{x})\}$ that are concatenated and passed through a decoder to produce the field $\mathbf{u}(\mathbf{x})$. The representation is optimised by minimising the governing differential equation $\mathcal{F}$ as a loss.

RBF Interpolation. d-Linear interpolation [Fridovich-Keil et al. 2023; Müller et al. 2022] is C^0 at grid nodes and C^1 in the interior; second-order derivatives vanish, blocking DE solving. We instead use RBF interpolation [Wright 2003] with Gaussian kernels $\varphi(r) = e^{-(\varepsilon r)^2}$, which are C^∞:

$$w(\mathbf{x}, \mathbf{x}_i) = \frac{\varphi(\|\mathbf{x} - \mathbf{x}_i\|)}{\sum_{j \in \mathcal{N}(\mathbf{x})} \varphi(\|\mathbf{x} - \mathbf{x}_j\|)}. \tag{5}$$

We truncate to a fixed ρ-ring neighbourhood ($\rho = 2$) with stratified sampling and precomputed indices, matching linear interpolation speed while maintaining smoothness and cross-cell overlap. We ablate the kernel parameters ε and ρ in App. C.

Multi-Resolution Feature Grid. A single grid updates only the small neighbourhood of each query point per step, requiring many iterations to propagate gradient information across the full domain. We use S co-located grids at resolutions $\{N_{\max}/2^s\}_{s=0}^{S-1}$, concatenating their interpolated features $\mathbf{f}(\mathbf{x}) = (\mathbf{f}_0(\mathbf{x}), \ldots, \mathbf{f}_{S-1}(\mathbf{x})) \in \mathbb{R}^{S \cdot F}$. Coarser grids enable fast global gradient flow; finer grids capture high-frequency detail. Together, the multi-scale features enable fast convergence with minimal additional parameter overhead. The effect of multi-resolution is ablated in App. C.

3 EXPERIMENTS

We demonstrate the versatility of ∂^∞-*Grid* across the Poisson, Helmholtz, and Kirchhoff-Love equations, as well as signal representation. Unlike prior grid-based methods (Instant-NGP [Müller et al. 2022], K-Planes [Fridovich-Keil et al. 2023], NeuRBF [Chen et al. 2023a]), our representation accurately solves for fields via higher-order derivative supervision. Additional experiments (Eikonal, heat, advection, and detailed comparisons) are in the appendices.

Poisson Equation. We reconstruct images from gradient or Laplacian supervision alone, *i.e.*,

$$\mathcal{L}_{\text{grad}} = \int_\Omega \|\nabla_{\mathbf{x}} \mathbf{u} - \nabla_{\mathbf{x}} g\| \, d\mathbf{x}, \quad \mathcal{L}_{\text{lapl}} = \int_\Omega \|\Delta \mathbf{u} - \Delta g\| \, d\mathbf{x}. \tag{6}$$

Our method achieves 20× faster convergence than Siren [Sitzmann et al. 2020] on 512×512 images; see Tab. 1. K-Planes [Fridovich-Keil et al. 2023] fails under $\mathcal{L}_{\text{lapl}}$: linear interpolation yields vanishing second derivatives.

Table 1. Image reconstruction: ∂^∞-*Grid* vs. Siren [Sitzmann et al. 2020] and K-Planes [Fridovich-Keil et al. 2023]. ∂^∞-*Grid* matches Siren in PSNR with substantially faster training; K-Planes fails under Laplacian supervision.

Model	K-Planes		Siren		Ours	
Supervision	Grad.	Lapl.	Grad.	Lapl.	Grad.	Lapl.
PSNR	17.96	-	32.12	11.82	32.24	12.19
Training time	9.5mins	-	10mins	1hr56mins	25s	15mins
Parameters	8.4m	-	329.99k	1.32m	702.94k	700.81k

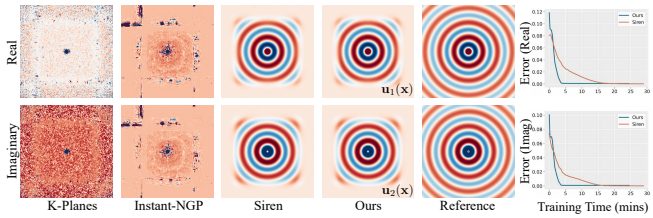


Fig. 3. Helmholtz Equation. ∂^∞-*Grid* matches the closed-form reference for a single point source, converging faster than Siren (see ℓ_1 error plot). K-Planes [Fridovich-Keil et al. 2023] and Instant-NGP [Müller et al. 2022] fail as their linear interpolation yields vanishing second-order derivatives.

Helmholtz Equation. We solve for complex-valued steady-state wavefields governed by $(\Delta + m\,\omega^2)\,\mathbf{u} = -g$ in a 2D domain with a single Gaussian point source, using perfectly matched layers for boundary absorption. Our method matches the closed-form Hankel reference solution and converges 4× faster than Siren; see Fig. 3. Grid-based baselines fail due to the lack of C^2 differentiability; see appendices for higher wavenumber results ($\omega = 30, 40$).

Cloth Simulation. We replace Siren with ∂^∞-*Grid* in NeuralClothSim [Kairanda et al. 2024], solving the Kirchhoff-Love thin-shell boundary-value problem. The model decomposes the deformation field $\mathbf{u} : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^3$ into membrane and bending strains; bending energy depends on second-order spatial derivatives of $\mathbf{u}$, making the total energy a fourth-order functional. Our representation recovers realistic folds and wrinkles (see Fig. XI in appendix); Instant-NGP and K-Planes fail, unable to model the required geometric strains.

Signal Representation. Beyond DE solving, ∂^∞-*Grid* also fits signals directly (images, SDFs) via L^1 supervision on the target field, matching grid-based baselines [Chen et al. 2023a; Müller et al. 2022]. It further supports indirect supervision—recovering images from gradient fields and SDFs from oriented point clouds via the Eikonal equation—where competing grid methods fail; see Fig. 1 and App. B.

4 CONCLUSION

We presented ∂^∞-*Grid*, a feature grid-based neural field representation with C^∞ RBF interpolation that enables accurate computation of higher-order derivatives for solving differential equations. ∂^∞-*Grid* achieves 5–20× speed-ups over coordinate-based MLPs while maintaining comparable accuracy across diverse PDE settings. One limitation is that RBF interpolation cost grows with input dimensionality; projection to lower-dimensional subspaces [Fridovich-Keil et al. 2023] is a promising direction to extend ∂^∞-*Grid* to higher-dimensional spatio-temporal domains.

REFERENCES

Jan-Hendrik Bastek and Dennis M Kochmann. 2023. Physics-Informed Neural Networks for shell structures. *European Journal of Mechanics-A/Solids* (2023).

John C. Butcher. 2016. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons.

Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. 2021. Physics-informed neural networks (PINNs) for fluid mechanics: a review. *Acta Mechanica Sinica* (2021).

Ang Cao and Justin Johnson. 2023. HexPlane: A Fast Representation for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. 2022. Efficient Geometry-Aware 3D Generative Adversarial Networks. In *Computer Vision and Pattern Recognition (CVPR)*.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. TensoRF: Tensorial Radiance Fields. In *European Conference on Computer Vision (ECCV)*.

Honglin Chen, Rundi Wu, Eitan Grinspun, Changxi Zheng, and Peter Yichen Chen. 2023b. Implicit Neural Spatial Representations for Time-dependent PDEs. In *International Conference on Machine Learning (ICML)*.

Zhongying Chen, Dongsheng Cheng, Wei Feng, and Tingting Wu. 2013. An optimal 9-point finite difference scheme for the Helmholtz equation with Pml. *International Journal of Numerical Analysis & Modeling* (2013).

Zhang Chen, Zhong Li, Liangchen Song, Lele Chen, Jingyi Yu, Junsong Yuan, and Yi Xu. 2023a. NeuRBF: A Neural Fields Representation with Adaptive Radial Basis Functions. In *International Conference on Computer Vision (ICCV)*.

Aditya Chetan, Guandao Yang, Zichen Wang, Steve Marschner, and Bharath Hariharan. 2025. Accurate differential operators for hybrid neural fields. In *Computer Vision and Pattern Recognition (CVPR)*.

Shin-Fang Chng, Sameera Ramasinghe, Jamie Sherrah, and Simon Lucey. 2022. Gaussian activated neural radiance fields for high fidelity reconstruction and pose estimation. In *European Conference on Computer Vision (ECCV)*.

David Clyde, Joseph Teran, and Rasmus Tamstorf. 2017. Modeling and data-driven parameter estimation for woven fabrics. In *Proc. ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)*.

Todd F Dupont and Yingjie Liu. 2003. Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. *J. Comput. Phys.* (2003).

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*.

Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. 2023. K-planes: Explicit radiance fields in space, time, and appearance. In *Computer Vision and Pattern Recognition (CVPR)*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings.

Shrisudhan Govindarajan, Zeno Sambugaro, Akhmedkhan Shabanov, Towaki Takikawa, Daniel Rebain, Weiwei Sun, Nicola Conci, Kwang Moo Yi, and Andrea Tagliasacchi. 2025. Lagrangian Hashing for Compressed Neural Field Representations. In *European Conference on Computer Vision (ECCV)*.

Antoine Guédon and Vincent Lepetit. 2024. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. *Computer Vision and Pattern Recognition (CVPR)* (2024).

Zhongkai Hao, Songming Liu, Yichi Zhang, Chengyang Ying, Yao Feng, Hang Su, and Jun Zhu. 2022. Physics-Informed Machine Learning: A Survey on Problems, Methods and Applications. *arXiv preprint arXiv:2211.08064* (2022).

Xinquan Huang and Tariq Alkhalifah. 2024. Efficient physics-informed neural networks using hash encoding. *J. Comput. Phys.* (2024).

Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. 2024. SC-GS: Sparse-Controlled Gaussian Splatting for Editable Dynamic Scenes. In *Computer Vision and Pattern Recognition (CVPR)*.

Navami Kairanda, Marc Habermann, Christian Theobalt, and Vladislav Golyanik. 2024. NeuralClothSim: Neural Deformation Fields Meet the Thin Shell Theory. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics* (2023).

Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. 2023. Neuralangelo: High-Fidelity Neural Surface Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. 2021. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing* (2021).

Nam Mai-Duy and Thanh Tran-Cong. 2003. Approximation of function and its derivatives using radial basis function networks. *Applied Mathematical Modelling* (2003).

Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. 2021. Acorn: adaptive coordinate networks for neural scene representation. *ACM Transactions on Graphics* (2021).

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Computer Vision and Pattern Recognition (CVPR)*.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics* (2022).

Tiago Novello, Vinícius da Silva, Guilherme Schardong, Luiz Schirmer, Hélio Lopes, and Luiz Velho. 2023. Neural Implicit Surface Evolution. In *International Conference on Computer Vision (ICCV)*.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Computer Vision and Pattern Recognition (CVPR)*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* (2011).

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* (2019).

Sameera Ramasinghe and Simon Lucey. 2022. Beyond periodicity: towards a unifying framework for activations in coordinate-MLPs. In *European Conference on Computer Vision (ECCV)*.

Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance Fields without Neural Networks. In *Computer Vision and Pattern Recognition (CVPR)*.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. 2020. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jonathan D. Smith, Kamyar Azizzadenesheli, and Zachary E. Ross. 2021. EikoNet: Solving the Eikonal Equation With Deep Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing* (2021).

Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes. In *Computer Vision and Pattern Recognition (CVPR)*.

Jiaxiang Tang. 2022. Torch-ngp: a PyTorch implementation of instant-ngp. https://github.com/ashawkey/torch-ngp.

Haoxiang Wang, Tao Yu, Tianwei Yang, Hui Qiao, and Qionghai Dai. 2024. Neural Physical Simulation with Multi-Resolution Hash Grid Encoding. In *AAAI Conference on Artificial Intelligence*.

Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. 2023. NeuS2: Fast Learning of Neural Implicit Surfaces for Multi-view Reconstruction. In *International Conference on Computer Vision (ICCV)*.

Grady Barrett Wright. 2003. *Radial basis function interpolation: numerical and analytical developments*. University of Colorado at Boulder.

Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. 2021. Geometry processing with neural fields. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. 2021. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems (NeurIPS)*.

APPENDICES

A RELATED WORK

Traditionally, differential equations are solved with numerical methods [Butcher 2016]. Classical solvers discretise governing equations but struggle with data incorporation, meshing, and high-dimensional costs, motivating neural representations as a promising alternative.

Neural Fields and DE Solvers. Neural fields are continuous and non-linear mapping functions parametrised by trainable neural network weights. They have been widely used in representing radiance fields [Mildenhall et al. 2020], signed distance and occupancy fields [Mescheder et al. 2019; Park et al. 2019; Yang et al. 2021], scene reconstruction [Guédon and Lepetit 2024; Li et al. 2023; Wang et al. 2023; Yariv et al. 2021], and beyond. Physics-Informed Neural Networks (PINNs) are neural fields that model solutions to differential equations, solved as optimisation problems supervised by physical laws [Hao et al. 2022; Raissi et al. 2019]. A wide range of problems, such as solving fluid dynamics [Cai et al. 2021], Eikonal equations [Smith et al. 2021], simulation [Bastek and Kochmann 2023], and many more, have been achieved with neural fields. Early works that used coordinate-based MLPs struggled to model high-frequency details in signals and derivatives. Siren [Sitzmann et al. 2020] introduces periodic activation functions for MLPs, facilitating gradient supervision with higher-order derivatives, enabling more complex applications [Chen et al. 2023b; Kairanda et al. 2024; Novello et al. 2023]. However, Siren and similar global MLP-based methods have shortcomings: they ignore the spatial structure of physical fields, require back-propagating through the full MLP for every coordinate sample, thus leading to slower training times compared to our proposed grid-based representation.

Feature Grids. In contrast to coordinate-based MLPs, discrete or hybrid representations learn compact localised features, reducing computation and training time while improving quality. Various discrete representations, such as grids [Martel et al. 2021; Sara Fridovich-Keil and Alex Yu et al. 2022], octrees [Takikawa et al. 2021] and feature planes [Cao and Johnson 2023; Chan et al. 2022; Chen et al. 2022; Fridovich-Keil et al. 2023], in combination with MLPs, have been introduced for faster learning. Müller et al. [2022] employ a multi-resolution structure with a hash encoding to provide a faster lookup, significantly speeding up computation. Unfortunately, the linear components (linear interpolation, ReLU) of these feature learning methods do not support higher-order derivative supervision required for solving DEs. While some methods [Huang and Alkhalifah 2024; Li et al. 2023; Wang et al. 2024] support higher-order derivative supervision using finite difference methods, they are known to be approximations and are susceptible to discretisation errors. [Chetan et al. 2025] recently introduced a post-hoc differential operator for pre-trained hybrid fields, such as Instant-NGP. Due to the reliance on a pre-trained representation, they fail to solve differential equations. On the other hand, our proposed method combines the advantages of localised grid feature learning with high-order differentiability to enable faster training and convergence.

Radial Basis Functions. RBF networks [Mai-Duy and Tran-Cong 2003] were among the first to employ RBFs as activation functions, with later works such as [Ramasinghe and Lucey 2022] and

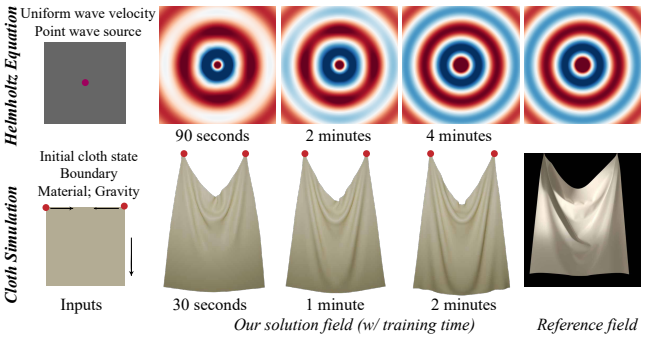


Fig. I. Our proposed method, ∂^∞-*Grid*, is a feature grid-based representation capable of accurately modelling signals and their higher-order derivatives. We demonstrate its effectiveness in solving complex differential equations, including the Helmholtz equation (top) and neural cloth simulation (bottom) [Kairanda et al. 2024] (reference adapted from [Clyde et al. 2017]), achieving both high speed and accuracy.

GARF [Chng et al. 2022] further demonstrating the utility of Gaussian activations. Recently, methods inspired by 3D Gaussian Splatting [Kerbl et al. 2023], including Lagrangian Hashing [Govindarajan et al. 2025] and SC-GS [Huang et al. 2024], leverage Gaussian interpolation for feature aggregation, though their focus is on radiance fields rather than solving DEs. The closest to ours is NeuRBF [Chen et al. 2023a], which introduced feature grids with RBF interpolation. However, its objectives and therefore the design differ fundamentally: NeuRBF proposes neural field representations that adapt to known target signals (images, SDFs, or Instant-NGP-distilled NeRFs), whereas ∂^∞-*Grid* solves PDEs from derivative-only supervision without observing the target field; see App. E for details. Their Gaussian interpolation remains discontinuous across grid cells due to the single-ring neighbourhood, and the method requires signal-dependent RBF initialisation with ground-truth signals (or proxy hybrids) in the loss formulation, leading to failure of NeuRBF for our problem setting.

B SIGNAL REPRESENTATION

Beyond solving DEs, ∂^∞-*Grid* can fit signals directly by using an L^1 loss on the target signal:

$$\mathcal{L}_{\text{signal}} = \int_\Omega \|\mathbf{u}(\mathbf{x}) - \mathbf{g}(\mathbf{x})\| \, d\mathbf{x}, \tag{7}$$

where $\mathbf{g}(\mathbf{x})$ is the known target (image or SDF). We show direct fitting of a 2D image and an SDF in Fig. 1-(a,c), where ∂^∞-*Grid* preserves high-frequency details on par with Instant-NGP [Müller et al. 2022] and NeuRBF [Chen et al. 2023a]. Our representation further supports indirect supervision: we recover images from gradient fields (Fig. 1-(b)) and SDFs from oriented point clouds via the Eikonal equation (Fig. 1-(d)). In these settings, competing grid-based methods fail (Figs. IV and VII and Tab. 1). Together, these results show that a single differentiable grid accommodates both direct signal fitting and PDE-constrained reconstruction without architectural changes.

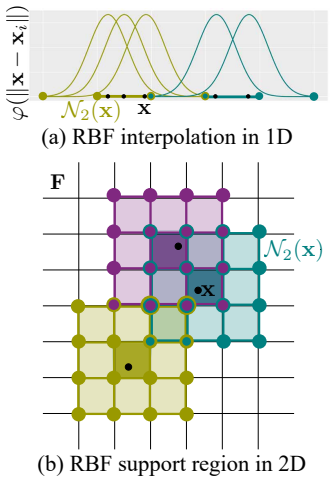


Fig. II. Illustration of effective neighbourhoods $\mathcal{N}_2(\mathbf{x})$ in $d = 1, 2$, determined by the shape parameter ε. An additional ring accounts for stratified sampling.

Table I. Ablation study for varying RBF shapes ε and neighbourhood sizes ρ for the high-resolution gradient-based image reconstruction in Fig. IV.

Shape ε	Ring ρ	PSNR ↑	SSIM ↑	Training ↓
0.6	4	28.74	0.87	15 min
1	4	28.74	0.87	17 min
1	3	28.73	0.87	12 min
1	2	28.82	0.86	12 min
1*	1	27.50	0.82	15 min
2*	1	17.72	0.74	6 min

C ABLATIONS

Neighbourhood Ring Selection. The RBF shape parameter ε and neighbourhood ring ρ balance efficiency and accuracy. We ablate these on 768×768 gradient image recovery (Fig. IV). Varying ε with adaptive ρ (so RBF weights are non-zero): smaller ε with larger rings gives smoother, more accurate results at higher cost; overly narrow kernels (*e.g.*, $\varepsilon = 2$, $\rho = 1$) introduce discontinuities resembling linear interpolation. Fixing $\varepsilon = 1$ and varying ρ: the 1-ring shows visible artefacts, while $\rho \geq 2$ yields stable training and smooth interpolation; see Tab. I.

Multi-Resolution Grid. Figure III demonstrates that a single-resolution grid restricts gradient flow to local regions, causing artefacts in SDF reconstruction. Multi-resolution grids enable propagation across the entire domain with minimal parameter overhead, and lead to significantly faster PSNR convergence on image recovery.

D REPRODUCIBILITY DETAILS

∂^∞-*Grid* Implementation. Our method is implemented in PyTorch, and all our experiments and the speed comparison for Siren are conducted on a single H100 GPU. Following [Müller et al. 2022], each feature grid is initialised with small random values drawn from a uniform distribution, *i.e.* $\mathbf{F}_s \sim \mathcal{U}(-10^{-4}, 10^{-4})$, whereas the decoder (whether linear or MLP) weights are initialised with Xavier [Glorot and Bengio 2010] and biases are set to zero. In the case of

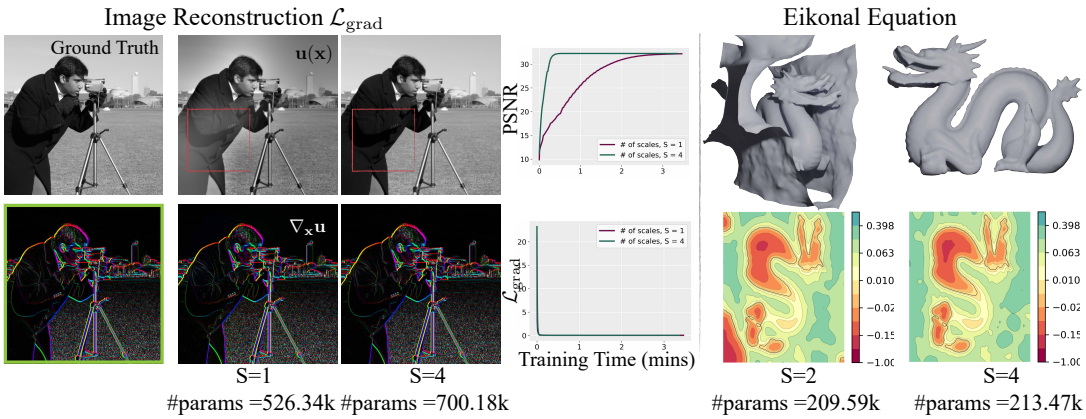


Fig. III. Effectiveness of the multi-resolution grids. While the single-resolution feature encoder converges at a similar rate to the multi-resolution grid in terms of the differential equation loss (as shown in the gradient loss plot), the *∂∞-Grid* multi-resolution architecture enables faster and more accurate field reconstruction (as illustrated in the PSNR plot, achieves higher PSNR in fewer iterations). The image reconstruction results are visualised at ≈30 seconds of training.

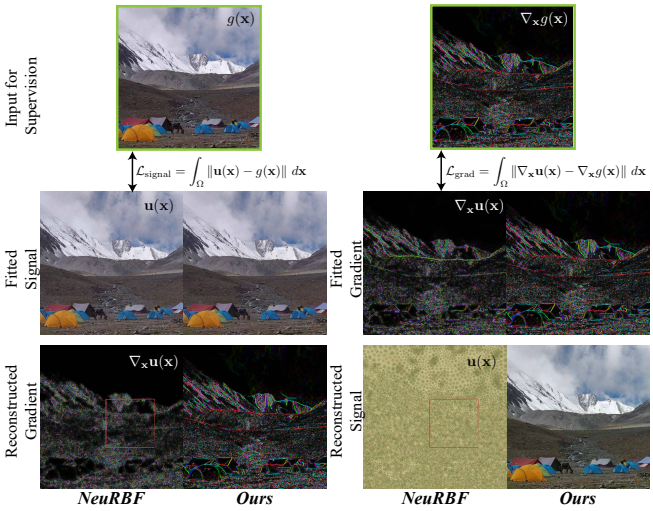


Fig. IV. Comparisons to NeuRBF [Chen et al. 2023a] for fitting 2D images with direct image supervision (left) and gradient image supervision (right). Note that while NeuRBF can fit signals well, it fails to compute accurate gradients (bottom left) and, consequently, cannot recover images with gradient supervision.

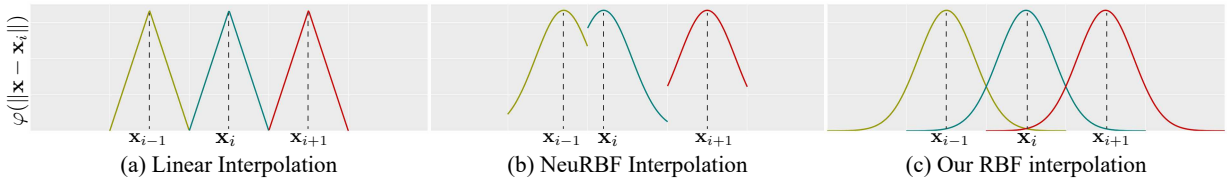
MLP decoder, we employ tanh as the activation. A learning rate of 5×10^{-3} is used with the Adam optimiser. We set $\varepsilon = 1$, which selects the neighbourhood $\mathcal{N}_3(\mathbf{x})$ (*i.e.*, $\rho = 3$) around each query point $\mathbf{x}$. Because a ring $\mathcal{N}_\rho(\mathbf{x})$ contains $(2\rho)^d$ neighbours per scale, the Gaussian RBF therefore uses 6^d neighbours at every scale/grid. Neighbourhood indices $\mathcal{N}_3(\mathbf{x}_i)$ are precomputed for all training and test samples $\{\mathbf{x}_i\}_i$. Additional hyperparameters, including the scale S, grid resolution $N_{\max}$, and feature dimension F, are experiment specific and detailed in the respective section for reproducibility.

K-Planes Comparison. We provide the settings used to reproduce our K-Planes results with the official implementation [Fridovich-Keil et al. 2023] on neural cloth simulation (Sec. 3) and Helmholtz equation (Sec. 3). We extend K-Planes using the same losses as in our method. The architectures are summarised as follows:

- NeuralClothSim: single-resolution 2D grid of size 64×64 with feature dimension 16, followed by an MLP with two 64-wide ReLU layers and a linear head; learning rate 10^{-2} with 0.1 decay every 2000 steps for 10000 epochs.
- Helmholtz: single-resolution 2D grid of size 128×128 with feature dimension 32, followed by an MLP with two 64-wide ReLU layers and a linear head; learning rate 2×10^{-5} for 50000 epochs. We employ single resolution as the multi-grid K-Planes struggles to converge otherwise.

Instant-NGP Comparison. We use the PyTorch implementation [Tang 2022] of Instant-NGP (that supports autograd) and apply the same losses as in our method. The architecture details are as follows:

- NeuralClothSim: eight-level hash grid (base resolution 4, final resolution 64) with feature dimension 2, followed by a linear head; learning rate 10^{-4} with 0.1 decay every 1000 steps for 10000 epochs.
- Helmholtz equation: eight-level hash grid (base resolution 4, final resolution 128) with feature dimension 2, followed by an MLP with two 64-wide ReLU layers and a linear head; learning rate 2×10^{-5} for 50000 epochs.
- SDF fitting: sixteen-level hash grid (base resolution 16, final resolution 2048) with feature dimension 2, followed by an MLP with two 64-wide ReLU layers and a linear head; learning rate 10^{-4} for 20 epochs.



(a) Linear Interpolation (b) NeuRBF Interpolation (c) Our RBF interpolation

Fig. V. Comparison of interpolation schemes. Linear interpolation in Instant-NGP/K-Planes [Fridovich-Keil et al. 2023; Müller et al. 2022] is only C^0 at grid nodes $\mathbf{x}_i$, whereas our RBF interpolation is C^∞ because centres are fixed on the grid with overlapping neighbourhoods. NeuRBF [Chen et al. 2023a] also uses RBFs but places adaptive centres off-grid and restricts each query to a single neighbourhood, leading to discontinuities.

E FURTHER COMPARISONS

E.1 Comparison with NeuRBF [Chen et al. 2023a]

NeuRBF and ∂^∞-*Grid* both use RBF-interpolated feature grids, yet they target very different problems. NeuRBF focuses on signal fitting: it reconstructs images and SDFs from ground-truth signals and tackles NeRFs via distillation, always coupled with a hybrid backbone such as Instant-NGP, K-Planes, or TensoRF. It does not attempt to solve PDEs directly from differential constraints, which is precisely our setting.

Consequently, NeuRBF assumes access to the full target signal (or a proxy provided by the hybrid model), whereas our losses (Eqs. (6) and (8)) only require derivative supervision. We recover solutions from gradients, Laplacians, or oriented point clouds without seeing the ground-truth field. For images and SDFs, NeuRBF augments its adaptive RBFs with Instant-NGP features for C^0 continuity; for NeRF, it first warms up K-Planes or TensoRF for 1–2k steps and then distils their predictions to initialise the RBFs. Such proxy supervision is infeasible in our PDE setting because Instant-NGP/K-Planes themselves fail when trained with purely differential signals.

NeuRBF chooses adaptive RBFs to position and scale kernels based on the known signal, while we adopt fixed-shape Gaussian RBFs to guarantee smooth, differentiable interpolation suitable for PDE losses. Even if one tried to adapt NeuRBF to PDEs, several critical problems remain that we summarise below:

- **Signal-dependent initialisation.** NeuRBF requires hand-crafted initialisation of RBF centres and shapes via weighted k-means. This biases the RBF distribution towards data points with higher weights (making their RBFs adaptive). Weights depend on the GT signal: $w_i = \|\nabla I(\mathbf{x}_i)\|$ for images, $w_i = 1/(|\mathrm{SDF}(\mathbf{x}_i)| + 10^{-9})$ for SDFs. For NeRF, NeuRBF first warms up K-Planes or TensoRF for 1–2k steps and distils their predictions to initialise the RBFs as $w_i = (1 - \exp(-\sigma(\mathbf{x}_i)\delta))\|\nabla \mathbf{f}_c(\mathbf{x}_i)\|$. Such proxy supervision is infeasible for PDEs, because Instant-NGP/K-Planes themselves fail in our settings, and no ground-truth field is available. By contrast, we place fixed-shape Gaussian RBFs on grid nodes without bespoke initialisation, which allows us to additionally precompute the effective neighbourhood; see Fig. V for RBF placement of ours vs NeuRBF.
- **Discontinuities in interpolation.** NeuRBF restricts interpolation to a single RBF ring, causing sharp discontinuities whenever the active neighbour set changes; they lean on Instant-NGP to enforce C^0 continuity at grid nodes (Fig. V), which is insufficient for PDE supervision. Expanding the neighbourhood is non-trivial because it would require KD-tree lookups at every query point as RBF centres change, and their signal-dependent initialisation scheme is provided only for the single-ring case. PDE losses such as the Eikonal constraint demand smooth gradients across cell boundaries, so we precompute overlapping $\mathcal{N}_3$ neighbourhoods on the grid for arbitrary query points, guaranteeing continuous interpolation and stable higher-order derivatives (Figs. II and V).

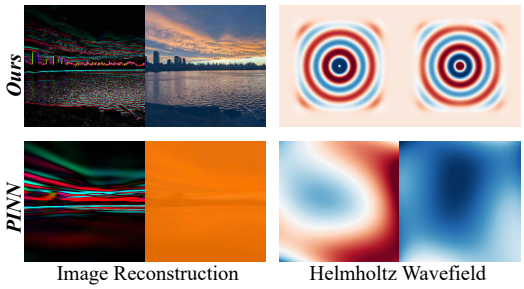


Fig. VI. Comparison to PINN. We compare ∂^∞-*Grid* to MLP with GELU activation function, which struggles with the high-frequency content.

Empirical comparison. To check if NeuRBF can handle PDE supervision, we modify the official NeuRBF implementation to (i) replace its image-fitting objective with our gradient-based Poisson loss (Eq. (6)) and (ii) optimise the Eikonal loss (Eq. (8)) for SDF reconstruction from oriented point clouds. We keep their initialisation scheme—weighting k-means by gradient magnitudes for images and by inverse SDF magnitudes for point clouds—yet note that no analogous scheme exists for higher-order PDE supervision (*e.g.*, Kirchhoff-Love problem for cloth simulation). Even under this favourable setup, NeuRBF failed to converge: Poisson reconstruction plateaued at PSNR/SSIM 10.85/0.17 versus 29.44/0.87 for ours (Fig. IV), and the Eikonal experiment did not recover a smooth surface (Fig. VII). These observations reinforce that NeuRBF's reliance on hybrid architecture (Instant-NGP), signal-dependent initialisation and discontinuous interpolation make it unsuitable for PDE solving, whereas our differentiable grid is explicitly engineered for derivative-only supervision.

E.2 Comparison with PINNs

We also compare against neural PDE solvers such as PINNs [Raissi et al. 2019]. While PINNs perform reasonably on smooth signals, they struggle with high-frequency content, failing to capture wrinkles in cloth simulations or oscillations in Helmholtz. For instance, in the image reconstruction experiment, a GELU-based PINN achieves

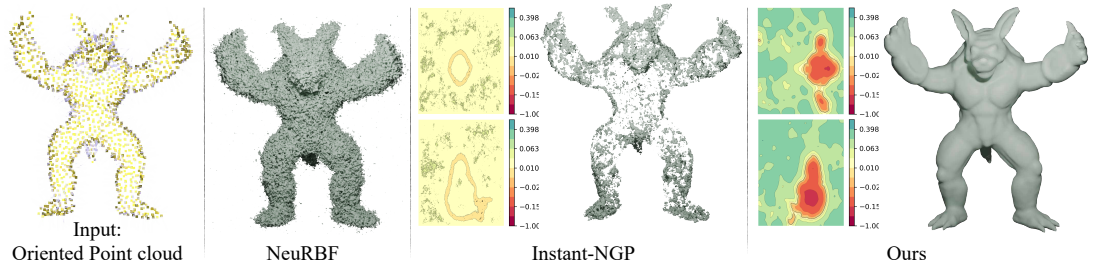


Fig. VII. Solutions to the Eikonal equation from oriented point clouds. For each method, we visualise 2D slices (left) and the mesh extracted from the SDF (right). Instant-NGP and NeuRBF can overfit when ground-truth SDF supervision ($\mathcal{L}_{\text{sdf}}$) is available, but they cannot reliably solve the Eikonal equation given partial point-cloud observations ($\mathcal{L}_{\text{eikonal}}$) because their gradients are not differentiable across grid boundaries (Fig. V). In contrast, our smooth formulation handles these gradients and solves the Eikonal problem comparably to baselines such as a sinusoidal MLP.

only PSNR/SSIM ≈19/0.65, compared to 29.44/0.87 with our method; results for image reconstruction from gradient field (Sec. 3) and Helmholtz equations (Sec. 3) are shown in Fig. VI. This limitation arises from their use of smooth activation functions (*e.g.*, Swish, GELU, tanh), which, though highly differentiable, are known to underfit high-frequency signals. We, therefore, adopt and primarily compare with Siren, which is better suited for such cases and widely used in physics-informed neural fields [Chen et al. 2023b; Kairanda et al. 2024].

F EIKONAL EQUATION

In addition to the experiments presented in the main paper, we demonstrate the learning of 3D shapes represented as signed distance functions (SDFs) by solving the Eikonal equation.

An SDF defines a mapping from 3D space to the distance of a point from a watertight surface, with the zero level set corresponding to the surface itself. Rather than assuming access to ground-truth SDF values, we explore fitting SDFs directly to oriented point clouds. This task corresponds to solving the Eikonal boundary value problem, a non-linear PDE, that enforces the constraint $\|\nabla_{\mathbf{x}}\mathbf{u}(\mathbf{x})\| = 1$ almost everywhere in the domain. Following the training procedure described in Sitzmann et al. [2020], we formulate the loss function as:

$$\begin{aligned}\mathcal{L}_{\text{eikonal}} = &\int_{\Omega} \big|\|\nabla_{\mathbf{x}}\mathbf{u}(\mathbf{x})\| - 1\big|\, d\mathbf{x} \\ &+ \int_{\Omega_0} \left(\|\mathbf{u}(\mathbf{x})\| + [1 - \langle \nabla_{\mathbf{x}}\mathbf{u}(\mathbf{x}), \mathbf{n}(\mathbf{x})\rangle]\right) d\mathbf{x} \\ &+ \int_{\Omega\setminus\Omega_0} \psi(\mathbf{u}(\mathbf{x}))\, d\mathbf{x},\end{aligned} \tag{8}$$

where Ω denotes the full domain, and Ω_0 refers to the zero level set of the SDF. The regularisation function is defined as $\psi(\cdot) = \exp(-\alpha \cdot |\mathbf{u}(\cdot)|)$, with $\alpha \gg 1$, which penalises off-surface points that produce SDF values close to zero. As illustrated in Figs. III and VII-(right), ∂^{∞}-*Grid* is able to accurately solve the Eikonal equation and recover high-quality SDFs from oriented point clouds.

Although the encoder lives on a spatio-temporal grid $[-1, 1]^3$, we support irregular geometries by sampling PDE residuals and boundary terms on the target (potentially unstructured) domain Ω_0. This works because $\mathbf{u}(\mathbf{x})$ in Eq. (2) is interpolated continuously, so arbitrary off-grid query sets (points and normals) can be supervised, enabling fits to surfaces (2D manifolds in 3D) from oriented point clouds. The trade-off is efficiency: empty grid points still store feature parameters, making our grid less parameter-efficient than coordinate-based MLPs.

F.1 Implementation Details

Dataset. In the SDF experiment, we learn a mapping $\mathbf{u}(\mathbf{x}) : [-1, 1]^3 \rightarrow \mathbb{R}$ using the loss function described above. The dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{n}(\mathbf{x}_i))\}_i$ comprises samples on the point cloud $\mathbf{x}$ and their ground-truth surface normals $\mathbf{n}(\mathbf{x})$.

Architecture. The feature encoder comprises $S = 4$ learnable grids, with the finest grid having a resolution of $N_{\max} = 56$, and each grid possessing a feature dimension of $F = 1$. Each query point is interpolated using $6^3 = 216$ features per scale. The feature decoder is a linear layer that maps the 3D input features to a 1D SDF.

Training. We employ stratified sampling over the domain, using 56 training samples along each input dimension. Additionally, an equal number of points are sampled on the surface, resulting in a total batch size of 2×56^2. A learning rate of 10^{-2} is used with the Adam optimiser. The SDF converges within 3000 iterations, requiring approximately six minutes of training time. For visualisation, the predicted SDF is evaluated on a uniformly sampled 112×112 grid covering the same domain, and meshes are extracted using the Marching Cubes algorithm.

F.2 Comparison with Grid-based Methods

Many grid-based methods [Li et al. 2023; Müller et al. 2022] represent signed distance functions (SDFs) using linear interpolation. In contrast to solving the Eikonal equation from oriented point clouds as in Eq. (8), these methods usually overfit to ground-truth SDFs computed from meshes, typically using a loss of the form:

$$\mathcal{L}_{\text{sdf}} = \int_{\Omega} \|\mathbf{u}(\mathbf{x}) - \text{SDF}(\mathbf{x})\|\, d\mathbf{x}, \tag{9}$$

where the loss penalises deviations from the ground-truth SDF. This approach is often used either directly [Chen et al. 2023a; Müller et al. 2022] or indirectly, for example, in multi-view surface reconstruction [Li et al. 2023; Wang et al. 2023] (sometimes with additional regulariser enforcing the Eikonal condition). While such methods

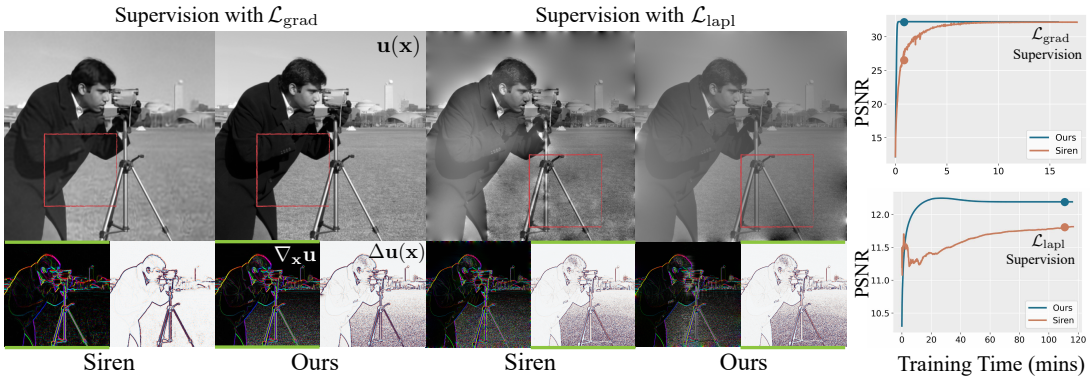


Fig. VIII. Image Reconstruction. Comparison of *∂^∞-Grid* with Siren [Sitzmann et al. 2020] for the reconstruction from gradient and Laplacian fields (Poisson equation). Results are visualised at training time marked by "•" and "•", and the supervision signal is highlighted with a green border. Our method achieves higher PSNR with faster convergence for high-resolution images, while Siren struggles, especially for Laplacian.

can produce accurate reconstructions when provided with high-quality ground-truth SDFs, they struggle when solving the Eikonal equation directly using $\mathcal{L}_{\text{eikonal}}$ (Eq. (8)). In particular, the Eikonal loss is not smoothly optimised across grid boundaries when using linear interpolation, which often results in incorrect SDF values (e.g., incorrect sign) on either side of the boundary. This issue is visualised in 2D slices for Instant-NGP in Fig. VII. In contrast, our method employs differentiable grid interpolation, enabling the recovery of consistent and smooth SDFs directly from oriented point clouds.

G POISSON EQUATION

G.1 Implementation Details

Dataset. We use the Camera image from sci-kit [Pedregosa et al. 2011]. Following Siren [Sitzmann et al. 2020], the ground-truth gradient image is computed using the Sobel filter and scaled by a constant factor of 10 for training. The ground-truth Laplacian image is computed using a Laplace filter and scaled by a factor of $10k$.

Architecture. The feature encoder comprises $S = 4$ learnable grids, with the finest grid of the resolution $N_{\text{max}} = 512$, and each grid of the feature dimension $F = 2$. We vary these hyperparameters for gradient and Laplacian supervision; see Tab. 1 for the number of trainable parameters. Each query point is interpolated using $6^2 = 36$ features per scale. The feature decoder is a linear layer.

Training. We train by evaluating on every pixel of the gradient or Laplacian image at each iteration. With regular fixed sampling over the domain, *i.e.*, 512 training samples along each input dimension, results in a total batch size of 512^2. A learning rate of 10^{-3} is used with the Adam optimiser. The convergence varies per experiment and is visualised in Fig. VIII and Tab. 1.

Discussion. We evaluate at 512×512 resolution—higher than the 256×256 used in the original Siren paper [Sitzmann et al. 2020]—to better illustrate the speed advantage of grid-based methods: localised gradient updates scale more favourably than full-network MLP backpropagation at high resolution. For colour images, gradients are supervised channel-wise; we visualise the red-channel field for clarity. Global intensity variations remain in the reconstructions due to the ill-posedness of the inverse problem: since no boundary conditions are explicitly specified, the absolute intensity level is underdetermined. Existing grid-based methods such as K-Planes [Fridovich-Keil et al. 2023] fail under $\mathcal{L}_{\text{lapl}}$ supervision: with d-linear interpolation, second derivatives vanish everywhere, so the Laplacian residual carries no gradient signal.

H HELMHOLTZ EQUATION

We describe the full details of the Helmholtz experiment, as provided in Sec. 3 of the main paper. The Helmholtz equation is a second-order partial differential equation commonly used to model steady-state wave behaviour and diffusion phenomena. It is expressed as:

$$\left(\Delta + m(\mathbf{x})\,\omega^2\right)\mathbf{u}(\mathbf{x}) = -g(\mathbf{x}), \tag{10}$$

where $g(\mathbf{x})$ denotes a known source function, $\mathbf{u}(\mathbf{x})$ is the unknown complex-valued wavefield, and $m(\mathbf{x}) = \frac{1}{c(\mathbf{x})^2}$ represents the squared slowness, defined in terms of the wave velocity $c(\mathbf{x})$. We solve for the wavefield $\mathbf{u}$ using the *∂^∞-Grid* representation, with results presented in Figs. 3 and I and error visualisation in Fig. IX. In the following, we outline the specific formulation of the Helmholtz equation variant used in our implementation. The model is trained using a weighted ℓ_1 residual loss:

$$\mathcal{L}_{\text{Helmholtz}} = \int_\Omega \lambda(\mathbf{x}) \left\| H(m)\,\mathbf{u}(\mathbf{x}) + g(\mathbf{x}) \right\|_1 d\mathbf{x}, \tag{11}$$

where $\lambda(\mathbf{x}) = k$ at source locations ($g(\mathbf{x}) \neq 0$) and $\lambda(\mathbf{x}) = 1$ elsewhere, balancing the inhomogeneous and homogeneous contributions. Unlike the image reconstruction setting, $\mathcal{L}_{\text{Helmholtz}}$ can be evaluated at arbitrary domain points, so we employ stratified sampling rather than dense pixel-grid supervision.

H.1 Perfectly Matched Layers Formulation

To obtain a unique solution to the Helmholtz equation within a bounded domain, we adopt a perfectly matched layer (PML) strategy, following the formulation introduced in [Chen et al. 2013; Sitzmann et al. 2020]. The PML serves to absorb outgoing waves at the domain

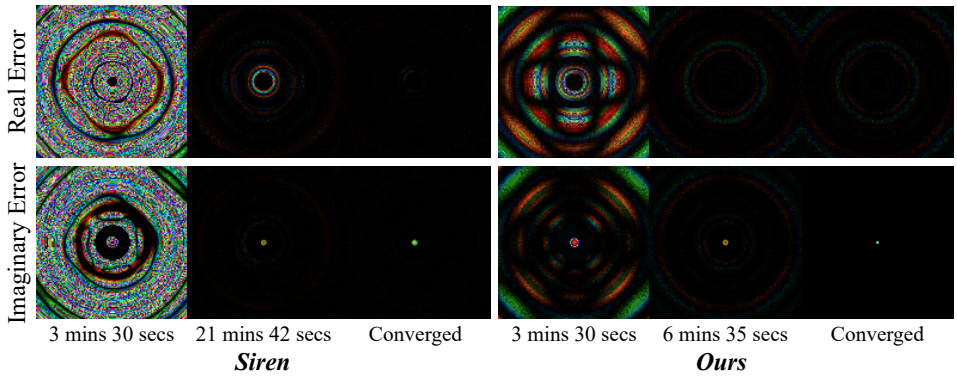


Fig. IX. ℓ_2-error for the solution to the Helmholtz equation. As presented in Fig. 3, while both ours and [Sitzmann et al. 2020] closely match the reference solution, our method trains significantly faster.

boundaries, thereby preventing artificial reflections. This results in the following variant of the Helmholtz equation:

$$\frac{\partial}{\partial x_1}\left(e_{x_1}e_{x_2}\frac{\partial \mathbf{u}(\mathbf{x})}{\partial x_1}\right)+\frac{\partial}{\partial x_2}\left(e_{x_1}e_{x_2}\frac{\partial \mathbf{u}(\mathbf{x})}{\partial x_2}\right)+e_{x_1}e_{x_2}k^2\mathbf{u}(\mathbf{x})=-g(\mathbf{x}), \tag{12}$$

where $\mathbf{x}=(x_1,x_2)\in\Omega$, the complex coefficients are given by $e_{x_i}=1-j\frac{\sigma_{x_i}}{\omega}$, and $k=\omega/c$. The damping term σ_{x_i} along each spatial axis is specified as:

$$\sigma_{x_i}=\begin{cases}a_0\omega\left(\frac{l_{x_i}}{L_{\text{PML}}}\right)^2 & \text{if } x_i\in\partial\Omega,\\ 0 & \text{otherwise,}\end{cases} \tag{13}$$

where a_0 determines the attenuation strength (set to $a_0=5$), and $c=1$ is the wave propagation speed. The term l_{x_i} represents the distance to the nearest PML boundary along the x_i-axis, while L_{PML} denotes the total PML width. The PML is applied exclusively near the domain boundary, defined as $\partial\Omega=\{\mathbf{x}\,|\,0.5<\|\mathbf{x}\|_\infty<1\}$, and the original Helmholtz equation remains unchanged elsewhere. We optimise Eq. (12) using Eq. (11), with the spatial weighting factor set to $\lambda(\mathbf{x})=k=\frac{\text{batch size}}{5\times10^3}$.

H.2 Implementation Details

Dataset. In the Helmholtz experiment, we learn a mapping $\mathbf{u}(\mathbf{x}):[-1,1]^2\to\mathbb{R}^2$ using the loss function described above. The dataset $\mathcal{D}=\{(\mathbf{x}_i,g(\mathbf{x}_i))\}_i$ comprises sampled spatial coordinates $\mathbf{x}_i$ and values of the Gaussian source function, defined as $g(\mathbf{x})=\mathcal{N}(\mathbf{x};[0,0],10^{-4})$.

Architecture. The feature encoder comprises $S=4$ learnable grids, with the finest grid having a resolution of $N_{\max}=128$, and each grid possessing a feature dimension of $F=2$. Each query point is interpolated using $6^2=36$ features per scale. The feature decoder is an MLP that takes the concatenated features as input, projects them to a hidden layer of width 64 with Swish activation, and produces a 2D output through a final linear layer. The total number of trainable parameters is approximately $45.19k$.

Training. We employ stratified sampling over the domain, with 256 training samples along each input dimension, resulting in a total batch size of 256^2. A learning rate of 1×10^{-3} is used with the Adam optimiser. The Helmholtz wavefields converge within 2500 iterations, requiring approximately 6 minutes of training time. For visualisation, the predicted wavefield is evaluated on a uniformly sampled 256×256 grid covering the same domain.

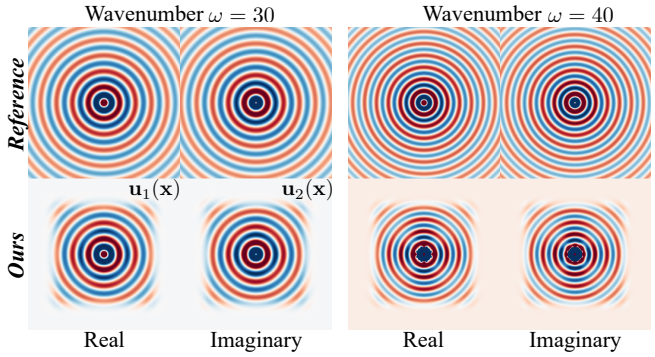


Fig. X. Results for Helmholtz equation with $\omega=30,40$ compared with closed-form solutions. *∂^∞-Grid* remains accurate overall, with minor errors near the source at $\omega=40$.

Table II. Helmholtz equation with higher wavenumbers. ℓ_1 error with the closed-form reference for the real and imaginary wavefield components. Parameter count: 45.19k for $\omega=20,30$; 176.71k for $\omega=40$.

ω	Real Error	Imag Error
20	0.0008	0.0007
30	0.0040	0.0040
40	0.0250	0.0240

H.3 Higher Wavenumber

To further evaluate our method under more challenging settings, we reproduce the Helmholtz equation experiments with higher wavenumbers, ω. While Fig. 3 presented results for $\omega=20$, here we extend the experiments to $\omega=30$ and $\omega=40$. Comparisons with the closed-form reference solutions are shown in Fig. X and Tab. II. For $\omega=30$, our method closely matches the analytical solution,

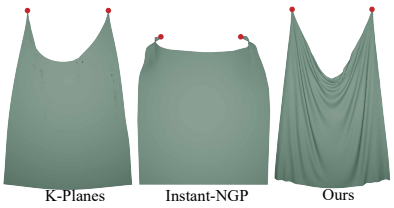


Fig. XI. Kirchhoff-Love Simulation [Kairanda et al. 2024]. *∂^∞-Grid* captures fine-grained cloth wrinkles by solving a highly nonlinear boundary value problem.

whereas for $\omega = 40$ we observe noticeable errors near the wave source but otherwise maintain good accuracy across the domain.

I CLOTH SIMULATION

We describe the further details of the cloth simulation experiment, as presented in Sec. 3. Given the rest state of the cloth and the material elastic parameters $\mathbf{\Phi}$, quasistatic cloth simulation involves finding the equilibrium deformation field $\mathbf{u} : \Omega \to \mathbb{R}^3$ of the midsurface under the influence of external forces $\mathbf{g}$ and boundary constraints on $\partial\Omega$. The stable equilibrium configuration is obtained by minimising the total potential energy functional subject to Dirichlet boundary constraints:

$$\begin{aligned} \mathbf{u}^* &= \arg\min_{\mathbf{u}} \int_\Omega \Psi[\boldsymbol{\varepsilon}(\mathbf{u}(\mathbf{x})), \boldsymbol{\kappa}(\mathbf{u}(\mathbf{x})); \mathbf{\Phi}] - \mathbf{g} \cdot \mathbf{u}(\mathbf{x})\, d\mathbf{x}, \\ \mathbf{u}(x_1, x_2) &= \mathbf{b}(x_1, x_2) \quad \text{on } \partial\Omega, \end{aligned} \tag{14}$$

where Ψ is the internal elastic energy density, and $\boldsymbol{\varepsilon}$, $\boldsymbol{\kappa}$ represent the membrane and bending strain components, respectively. We solve for the deformation field $\mathbf{u}$ using the *∂^∞-Grid* representation. Results are shown in Fig. XI.

As a specific example, we consider an analytically defined square piece of cloth of mass density ρ, held with two handles (Dirichlet boundary conditions) and subject to gravity $\mathbf{g} = [0, 0, -9.8\rho]$. Boundary conditions are imposed as hard constraints via the distance-based modulation of Eq. (2), and we assume a linear elastic material model for $\mathbf{\Phi}$. Since multiple equilibria are possible (no unique reference solution), we do not report quantitative accuracy metrics for this experiment.

I.1 Implementation Details

Dataset. In the simulation experiment, we learn a mapping $\mathbf{u}(\mathbf{x}) : [-1, 1]^2 \to \mathbb{R}^3$ using the loss function described above. The dataset $\mathcal{D}$ consists of sampled spatial coordinates $\mathbf{x}$ and their corresponding rest state positions, set as $[x_1, x_2, 0]$. The material properties are defined by the parameter set $\mathbf{\Phi}$, with values:$\rho = 0.144$, $h = 0.0012$, $E = 5000$, and $\nu = 0.25$. We impose Dirichlet boundary conditions as hard constraints (following NeuralClothSim [Kairanda et al. 2024]) by displacing the top-left and top-right vertices of the cloth towards the centre of the input domain by 0.4 units each. The external force is modelled as a constant gravitational acceleration $[0, -9.8, 0]$.

Architecture. The feature encoder consists of a single learnable grid ($S = 1$) with a feature dimension of $F = 4$. We use a grid resolution of $N = 32$ for the simulation in Fig. 1, and $N = 64$ for that in Fig. XI. Each query point is interpolated using $6^2 = 36$ features per scale. The feature decoder is a linear layer that maps the 3-dimensional input features to a 3D deformation field. The total number of trainable parameters is approximately $4.37k$ for the results in Fig. 1, and $16.91k$ for those in Fig. XI.

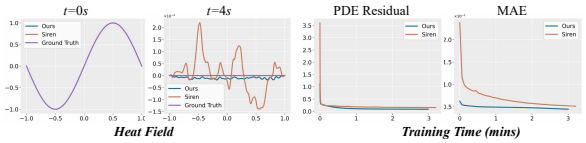


Fig. XII. Heat Equation. *∂^∞-Grid* solves the spatio-temporal heat equation: a sinusoidal wave diffuses over time (shown at $t = 4$s), matching the analytical solution. PDE residual and field error converge faster than Siren [Sitzmann et al. 2020].

Table III. 1D Advection comparison with INSR [Chen et al. 2023b] and Grid solver [Fedkiw et al. 2001].

Method	Error (MAE)	Training Time	Memory
INSR [Chen et al. 2023b]	0.0030	5.33h	3.53KB
Grid [Fedkiw et al. 2001]	0.0029	1.80s	27.35KB
Our *∂^∞-Grid* approach	0.0023	1.5mins	16.52KB

Training. Similar to the Helmholtz experiment, we employ stratified sampling over the domain, using 64 and 128 training samples along each input dimension for Figs. 1 and 6-(main), respectively. A learning rate of 1×10^{-2} is used with the Adam optimiser. The cloth simulation converges within 500 iterations, requiring at most 5 minutes of training time. For visualisation, the predicted deformation field is evaluated on a uniformly sampled 64×64 grid spanning the same domain.

J HEAT EQUATION

Here, we additionally demonstrate our result for the 1D heat equation. The heat equation is a diffusion PDE used to model temporal smoothing of signals:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \tag{15}$$

where $u(x, t)$ represents the temperature at position x and time t, and α is the thermal diffusivity constant. We consider a spatial domain $x \in [-1, 1]$ and a temporal domain $t \in [0, 4]$ with Dirichlet boundary conditions and an initial sinusoidal condition $u(x, 0) = \sin(\pi x)$ with $\alpha = 1$. It admits an analytical solution:

$$u(x, t) = e^{-\alpha \pi^2 t} \sin(\pi x), \tag{16}$$

which we use to quantitatively evaluate reconstruction fidelity. We train with the PDE loss with hard boundary constraints. Fig. XII summarises the experiment: the sinusoidal wave diffuses as expected and matches the analytical solution with a mean absolute error of 0.004 over uniformly sampled space-time points. Relative to the coordinate-based MLP baseline [Sitzmann et al. 2020], the PDE residual and the heat-field error reduce faster. Thus, *∂^∞-Grid* faithfully recovers the spatial and temporal evolution of the diffusion process.

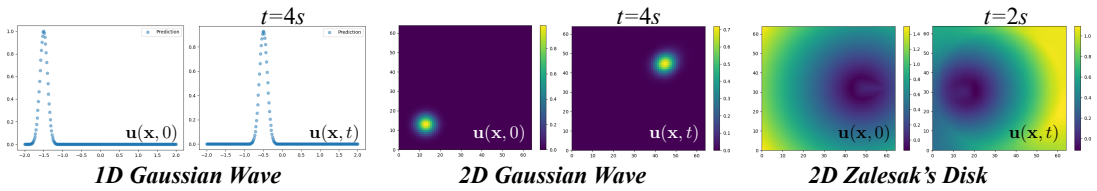


Fig. XIII. We solve spatio-temporal advection successfully for various 1D and 2D fields, including 1D and 2D Gaussian waves as well as 2D Zalesak's disk.

K ADVECTION EQUATION

We consider the spatio-temporal advection equation

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{a} \cdot \nabla_{\mathbf{x}} \mathbf{u} = 0, \tag{17}$$

which describes the transport of a field with advection velocity $\mathbf{a}$. We demonstrate our method on 1D and 2D cases, including Gaussian wave propagation and the Zalesak benchmark with a challenging discontinuous boundary.

K.1 1D Advection

We follow the INSR setup [Chen et al. 2023b], where a Gaussian-shaped wave, centred at $\mu = -1.5$ propagates rightwards in the spatial domain $[-2, 2]$, with advection velocity $a = 0.25$. In contrast to INSR, we impose boundary and initial conditions as hard constraints and treat time as a continuous input to our feature grid, rather than discretising it (they use midpoint integration). We present the results in Tab. III and Fig. XIII. Our results closely match both the analytical solution and INSR, with a mean absolute error (MAE) reported at $t = 4$ seconds. The INSR and the finite difference-based grid solver [Fedkiw et al. 2001] values are taken from Tab. 1 of [Chen et al. 2023b]. *∂^∞-Grid* suffers minimal dissipation; we observe an MAE of 2.3×10^{-3} at $t = 4$s, *i.e.*, less than 0.3% amplitude loss compared to the analytical solution, while still training 200× faster than INSR. Overall, our approach achieves lower error relative to the ground truth, reduced training time compared to INSR, and lower memory use compared to numerical solvers.

K.2 2D Advection

For the 2D Gaussian wave, we adopt the domain and initial conditions of [Chetan et al. 2025], simulating for 4 seconds. Relative to the closed-form solution, our method achieves mean advection errors of 0.0047 (after 3 minutes), 0.0032 (after 6 minutes), and 0.00246 (after 15 minutes, fully converged); see Fig. XIII-(centre) for qualitative results. Note that in [Chetan et al. 2025] advection example, they employ forward/explicit Euler (possible with a post hoc gradient operator) for timestepping, unlike INSR [Chen et al. 2023b] (which can support mid-point due to underlying Siren representation) and ours (which supports auto diff and mid-point similar to Siren/INSR).

K.3 Zalesak's Disk Problem

For the Zalesak benchmark, we follow the setup in [Dupont and Liu 2003], and solve for the signed distance function (SDF) of the slotted disk under advection. The results converge and preserve the overall shape, although some numerical diffusion is visible as blurring along the slot edges; see Fig. XIII-(right). After half a revolution, we obtain an ℓ_1-error of 0.065 compared to the closed-form rigid-body rotation solution with constant angular velocity.

L LIMITATIONS AND FUTURE WORK

While our method leverages differentiable interpolation to enable accurate representation of signals and their derivatives, it exhibits several limitations. First, the interpolated features are sensitive to the choice of the Gaussian RBF shape parameter, which in turn determines the effective neighbourhood. As a result, the representation can introduce some degree of smoothing, depending on this configuration. While we observed convergence across all experiments, further investigation is needed to establish theoretical guarantees for C^∞ under neighborhood truncation. A well-studied drawback of RBF methods, particularly near the domain boundaries, is the occurrence of boundary errors (Runge's phenomenon), which can negatively impact reconstruction accuracy in those regions. Our grid approach also suffers from the curse of dimensionality; for example, fitting SDFs in 3D is notably slower than in 2D. The efficiency gain of our method is primarily due to the use of structured Cartesian grids, which is lost when solving PDEs on manifolds. As a direction for future work, we suggest exploring planar projection strategies to improve scalability and computational efficiency in high-dimensional domains.

Moreover, our novel representation, although it improves speed over neural solvers and introduces differentiability into feature grids, it might not be as efficient as traditional numerical solvers, such as multi-grid methods. Since this requires further thorough investigation, in our evaluations, we provided initial experiments and comparisons to numerical methods help guide future work in this direction. In the future, a custom CUDA kernel implementation of *∂^∞-Grid* could further accelerate optimisation compared to our current PyTorch implementation.