PDEBENCH: AN EXTENSIVE BENCHMARK FOR SCIENTIFIC MACHINE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite some impressive progress in machine learning-based modeling of physical systems, there is still a lack of benchmarks for Scientific ML that are easy to use yet challenging and representative of a wide range of problems. We introduce PDEBENCH, a benchmark suite of time-dependent simulation tasks based on Partial Differential Equations (PDEs). PDEBENCH comprises both code and data to benchmark the performance of novel machine learning models against classical numerical simulations and ML baselines. Our proposed set of benchmark problems contribute the following features: (1) A much wider range of PDEs compared to existing benchmarks, ranging from relatively common examples to more realistic problems; (2) much larger ready-to-use datasets compared to prior work, comprising multiple simulation runs across a large number of initial and boundary conditions and PDE parameters; (3) more extensible source codes with user-friendly APIs for data generation and obtaining baselines of popular machine learning models (FNO, U-Net, PINN, Gradient-Based Inverse Method). PDEBENCH allows users to extend the benchmark freely for their own purposes using a standardized API and to compare the performance of new models to existing baseline methods. We also propose new evaluation metrics in order to provide a more holistic understanding of model performance in the context of Scientific ML.

1 MOTIVATION

In the emergent area of *Scientific Machine Learning* (SciML), recent progress has broadened the scope of traditional ML methods to include the time-evolution of physical systems. Within this field, rapid progress has been made in the use of neural networks to make predictions using functional observations over continuous domains (Chen et al., 2018; Rackauckas, 2019) or with challenging constraints and with physically-motivated conservation laws (Lu et al., 2021; Wang et al., 2020; Raissi et al., 2019). These neural networks provide an approach for solving PDEs complementing traditional numerical solvers. Moreover, neural models have the advantage of being continuously differentiable in their inputs, a useful property in several applications. While complex methods such as Bayesian optimisation Močkus (1975); Snoek et al. (2012); O'Hagan (1978) or reduced order modelling Guo & Hesthaven (2019) are in part an attempt to circumvent this lack of differentiability, gradients for neural networks are readily available and efficient.

For classical ML applications such as image classification, various popular benchmarks exist, and evaluations using these benchmarks provide a standardised means of testing the effectiveness and efficiency of ML models. As yet, a widely accessible and challenging benchmark with ready-to-use datasets to compare methods in SciML is missing. We aim to provide a benchmark that is more comprehensive with respect to the PDEs covered and which enables more diverse methods for evaluating the efficiency and accuracy of the ML method. The problems span a range of governing equations as well as different assumptions and conditions. Data may be generated by executing code through a common interface, or by downloading high-fidelity datasets of simulations. We also propose an API to ease the implementation and evaluation of new methods, provide recent competitive baseline methods such as FNOs and autoregressive models based on the U-Net, and a set of pre-computed performance metrics for these algorithms. We may thus compare their predictions against the "ground truth" provided by baseline simulators used to generate the data.

In this work, we propose a versatile benchmark suite for SciML (a) providing diverse datasets with distinct properties based on 11 well-known time-dependent and time-independent PDEs, (b) covering both "classical" forward learning problems and inverse settings, (c) all accessible via a uniform interface to read/store data across several applications, (d) extensible, (e) with results for popular state-of-the-art ML models (FNO, U-Net, PINN) for (f) a set of metrics that are better-suited for SciML, (g) with both data to download and code to generate more data, and (h) pre-trained models to compare against. The inverse problem scenarios comprise initial and boundary conditions and PDE parameters (e.g. viscosity). Each data set has a sufficiently large number of samples for training and testing for a variety of parameter values, with a resolution high enough to capture local dynamics. As an additional note, our goal is not to provide a complete benchmark that includes all possible combinations of inference tasks on all known experiments, but rather to ease the task for subsequent researchers to benchmark their favoured methods. Part of our goal here is to invite other researchers to fill in the gaps for themselves by leveraging our ready-to-run models.

2 PDEBench: A Benchmark for Scientific Machine Learning

In the following we describe the general learning problem addressed with the benchmark, the currently covered PDEs, existing implemented baselines (all developed using PyTorch (Paszke et al., 2019), and PINN specifically using DeepXDE (Lu et al., 2021)), and the ways in which the benchmark follows FAIR data principles Wilkinson et al. (2016).

2.1 Overview of Datasets and PDEs

The benchmark provides datasets generated for various PDEs ranging from 1 to 3 dimensional spatial domains. Each sample is generated with different parameters, initial conditions, and boundary conditions. Generalization to different parameters, varying initial conditions, and proper treatment of complex boundary conditions are still open challenges in SciML Karlbauer et al. (2022); Brandstetter et al. (2022); Belbute-Peres et al. (2021); Leiteritz et al. (2021). The parameters which are varied to provide several datasets include the advection speed in the advection equation, the forcing term in the Darcy flow, as well as the viscosity in the Burgers' and compressible Navier-Stokes equations all of which can lead to significantly different behaviors of the simulated systems. Additionally, besides the periodic boundary condition that is most commonly used in SciML studies, we also provide datasets generated with the Neumann boundary condition in the 2D diffusion-reaction and shallow-water equations, the Cauchy boundary condition in the diffusion-sorption equation, and the Dirichlet condition in the incompressible Navier-Stokes equation.

We designed this benchmark to represent a diverse set of challenges for emulation algorithms. In particular, we focus on hydromechanical field equations. Following this philosophy, we selected 6 basic and 3 advanced real-world problems. The basic PDEs are stylized, simple models: 1D advection/Burgers/Diffusion-Reaction/Diffusion-Sorption equations, 2D Diffusion-Reaction equation, and 2D DarcyFlow; the advanced and real-world PDEs incorporate features of real-world modeling tasks: Compressible and incompressible Navier-Stokes equations, and shallow-water equations. The PDEs exhibit a variety of behaviors of real-world significance which are known to challenge emulators, such as sharp shock formation dynamics, sensitive dependence on initial conditions, diverse boundary conditions, and spatial heterogeneity. Finding a surrogate model which can approximate these challenging dynamics with high fidelity, we argue is a necessary precondition to applying such models in the real world. While some of these have been used in prior work, a publicly available benchmark dataset is, to the best of our knowledge, not available.

2.2 Overview of Metrics

In SciML, standard evaluation metrics such as MSE and its variants fail to capture important features of physics-based problems. For example, physics-informed ML models that are intended to conserve physical quantities should be evaluated with appropriate metrics. Moreover, multiple evaluation metrics are also beneficial in comparing different aspects of ML models. Therefore, our proposed benchmark includes novel metrics which we believe provide a deeper understanding of the model's performance and are designed to evaluate the models both from the data-driven and physics perspectives. Such metrics include MSE, normalized MSE, maximum error, conservation

```python
from pdebench.models.fno.utils import FNODatasetSingle
filename = "data/2D_diff-react_NA_NA"
train_data = FNODatasetSingle(filename)
train_loader = torch.utils.data.DataLoader(train_data)
```

Listing 1: Using the PyTorch data loader.

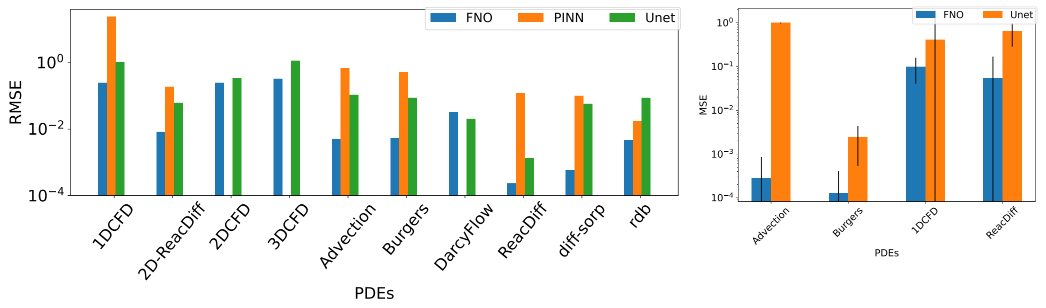


Figure 1: Comparisons of baseline models' performance for different problems for (a) the forward problem and (b) the inverse problem.

error, boundary condition error, and errors in the Fourier space. Further details can be found in Appendix B.

2.3 Data Format, Benchmark Access, Maintenance, and Extensibility

The benchmark consists of different data files, one for each equation, type of initial condition, and PDE parameter, using the HDF5 Group (2022) binary data format. Each such file contains multiple arrays where each array has the dimensions N, T, X, Y, Z, V with N the number of samples, T the number of time steps, X, Y, Z the spatial dimensions, and V the dimension of the field. Additional information on the data format is provided in the Supplementary Material.

PDEBench's datasets are stored and maintained using DaRUS, the University of Stuttgart's data repository based on the OpenSource Software DataVerse[1]. DaRUS follows the Findable, Accessible, Interoperable and Reusable (FAIR) data principles Wilkinson et al. (2016). All data uploaded to DaRUS gets a DOI as a persistent identifier, a license, and can be described with an extensive set of metadata, organized in metadata blocks. A dedicated team ensures that DaRUS is continuously maintained. To download the datasets, a file containing the download metadata and a script to download the data are provided, leveraging the `download_url` command of `torchvision`.

We also support a straightforward inclusion of the benchmark with a few lines of code. In Listing 1 we show an example leveraging pre-defined classes included in our benchmark code to load specific datasets as PyTorch (Paszke et al., 2019) `Dataset` classes. Subsequently, these can be used to construct common `DataLoader` instances for training custom ML models. We utilize the Hydra (Yadan, 2019) library simplifying the configuration of both surrogate model training as well as the generation of additional datasets. For the latter, we provide and expose various parameters of the underlying simulations for the end user to tweak. This provides a low barrier of entry for users to try out benchmarking with new experiments or baseline configurations.

3 A Selection of Experiments

In this section, we present a selection of experiments for the PDEBench datasets. An exhaustive discussion of all results is beyond the scope of this paper. An extensive set of additional results, tables, and plots can be found in the Appendix.

[1] `https://dataverse.org`

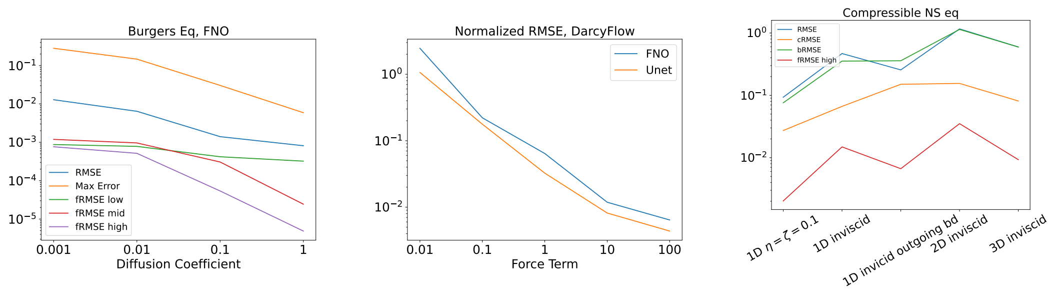


Figure 2: Detailed visualization of (a) Burgers', (b) DarcyFlow, and (c) Compressible NS eqs.

Baseline Setups We trained and tested the baseline emulator models, namely U-Net, FNO, and PINN with the datasets generated with the PDEs described in subsection 2.1. The data was split into training and test data: 90% was for training and 10% for test data. For FNO, we followed the original implementation, hyperparameters, and training protocols. We trained U-Net similar to FNO, but with the autoregressive methods with the pushforward trick with a slight modification to the original implementation (Brandstetter et al., 2022). The PINN baseline is implemented using the open-source DeepXDE (Lu et al., 2021) library. The training was performed on GeForce RTX 2080 GPUs for 1D/2D cases, and GeForce GTX 3090 for 3D cases.

Baseline Performance Figure 1 visualizes the RMSE performance of the surrogate models, averaged for each trained model over different PDE parameters. Among the baseline surrogate models, FNO provides the best prediction for most metrics. It learns the differential operators well, leading to low errors even for the conserved quantities and on the boundaries. Additionally, FNO has a consistent error of about 4×10^{-4} across the frequency spectrum for many problems highlighting its ability to learn in Fourier space. Our baseline results further indicate that the PINNs might deal better than expected with high-frequency features, despite prior observations (Wang et al., 2022). As an example for an inverse problem setup, we identify the initial condition to minimize the prediction error of the ML surrogate over 15 time steps horizon. In Figure 1b, we present the MSE of the prediction of estimated initial condition (with error bars) for 4 of the 11 datasets (1D). The results show that FNO outperforms U-Net also for the inverse problem. However, our benchmark also reveals several tasks which these methods cannot treat properly. First, Figure 2a shows that the FNO's error increases with decreasing diffusion coefficient where a strong discontinuity appears. This can be attributed to Gibb's phenomenon for FNO's limited maximum wave frequency in Fourier space, as shown by an increase of two orders of magnitude for high-frequency fRMSE. Second, Figure 2b shows that the normalized RMSE increases with decreasing force term, which is equivalent to decreasing the scale-value of the solution (in our case, force term 0.01 means $\mathrm{mean}(|u|) \approx 0.01$).Third, Figure 2c shows several metrics for the compressible Navier-Stokes equations. It shows the overall RMSE is very bad in comparison to the basic PDEs, such as the Burgers equation. Interestingly, the 3D inviscid case shows lower error than 2D inviscid case. We posit this is due to lower resolution resulting in smooth train/validation samples which FNO can learn very efficiently. This also indicates that high-resolution training samples should be used to create a surrogate model for real-world problems with a Reynolds number more than 10^6.

4 Conclusions

With PDEBench we contribute an extensive benchmark suite for comparing and evaluating methods on the realm of SciML. We provide both pre-computed datasets for easy access in a dataverse as well as the code to generate new data from configurable simulation runs. The focus is on time-dependent PDE problems ranging from simple 1D equations to challenging 3D coupled systems of equations featuring challenging boundary conditions. Furthermore, we present a wide variety of different evaluation metrics in order to better understand the strengths and weaknesses of the machine learning methods in a scientific computing context. We also provide an example application for utilizing SciML methods for inverse modeling tasks with our benchmark data. We believe this will be an important area in the future for machine learning models to produce competitive results both in accuracy as well as runtime when compared to numerical methods.

REFERENCES

Filipe de Avila Belbute-Peres, Yi-fan Chen, and Fei Sha. Hyperpinn: Learning parameterized differential equations with physics-informed hypernetworks. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/pdf?id=LxUuRDUhRjM`.

Deniz A. Bezgin, Aaron B. Buhendwa, and Nikolaus A. Adams. JAX-FLUIDS: A fully-differentiable high-order computational fluid dynamics solver for compressible two-phase flows, 2022. URL `https://arxiv.org/abs/2203.13760`.

Johannes Brandstetter, Daniel Worrall, and Max Welling. Message passing neural pde solvers. In *The Tenth International Conference on Learning Representations*, 2022. URL `https://openreview.net/pdf?id=vSix3HPYKSU`.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

Steven L. Brunton and Jose Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019. ISBN 978-1-108-42209-3. URL `https://databookuw.com`.

Kai Cao. *Inverse Problems for the Heat Equation Using Conjugate Gradient Methods*. PhD thesis, University of Leeds, 2018.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6572–6583. Curran Associates, Inc., 2018.

Richard P. Feynman. *Feynman lectures on physics - Volume 1*. 1963.

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021. URL `https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/3def184ad8f4755ff269862ea77393dd-Paper-round1.pdf`.

The HDF Group. An overview of the hdf5 technology suite and its applications, 2022. URL `https://portal.hdfgroup.org/display/HDF5/HDF5`.

Mengwu Guo and Jan S. Hesthaven. Data-driven reduced order modeling for time-dependent problems. *Computer Methods in Applied Mechanics and Engineering*, 345:75–99, 2019. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2018.10.029. URL `https://www.sciencedirect.com/science/article/pii/S0045782518305334`.

Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Max Rietmann, Jose del Aguila Ferrandis, Wonmin Byeon, Zhiwei Fang, and Sanjay Choudhry. NVIDIA SimNet$\{\hat{\text{TM}}\}$: An AI-accelerated multi-physics simulation framework. *arXiv:2012.07938 [physics]*, December 2020.

Philipp Holl and Vladlen Koltun. Phiflow: A differentiable PDE solving framework for deep learning via physical simulations. pp. 5, 2020.

Zizhou Huang, Teseo Schneider, Minchen Li, Chenfanfu Jiang, Denis Zorin, and Daniele Panozzo. A large-scale benchmark for the incompressible Navier-Stokes equations, 2021. URL `https://arxiv.org/abs/2112.05309`.

Tsuyoshi Inoue, Shu-ichiro Inutsuka, and Hiroshi Koyama. The Role of Ambipolar Diffusion in the Formation Process of Moderately Magnetized Diffuse Clouds. *The Astrophysical Journal*, 658 (2):L99–L102, April 2007. doi: 10.1086/514816.

Matthias Karlbauer, Timothy Praditia, Sebastian Otte, Sergey Oladyshkin, Wolfgang Nowak, and Martin V Butz. Composing partial differential equations with physics-aware neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Baltimore, USA, 16–23 Jul 2022.

K. Kashinath, M. Mustafa, A. Albert, J-L. Wu, C. Jiang, S. Esmaeilzadeh, K. Azizzadenesheli, R. Wang, A. Chattopadhyay, A. Singh, A. Manepalli, D. Chirila, R. Yu, R. Walters, B. White, H. Xiao, H. A. Tchelepi, P. Marcus, A. Anandkumar, P. Hassanzadeh, and null Prabhat. Physics-informed machine learning: Case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194): 20200093, April 2021. doi: 10.1098/rsta.2020.0093.

David I. Ketcheson, Kyle T. Mandli, Aron J. Ahmadia, Amal Alghamdi, Manuel Quezada de Luna, Matteo Parsani, Matthew G. Knepley, and Matthew Emmett. PyClaw: Accessible, Extensible, Scalable Tools for Wave Propagation Problems. *SIAM Journal on Scientific Computing*, 34(4): C210–C231, November 2012.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

G.A. Klaasen and W.C. Troy. Stationary wave solutions of a system of reaction-diffusion equations derived from the fitzhugh–nagumo equations. *SIAM Journal on Applied Mathematics*, 44(1): 96–110, 1984. doi: 10.1137/0144008.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning Maps Between Function Spaces. *arXiv:2108.08481 [cs, math]*, September 2021.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Alexander Lavin, Hector Zenil, Brooks Paige, David Krakauer, Justin Gottschlich, Tim Mattson, Anima Anandkumar, Sanjay Choudry, Kamil Rocki, Atılım Güneş Baydin, Carina Prunkl, Brooks Paige, Olexandr Isayev, Erik Peterson, Peter L. McMahon, Jakob Macke, Kyle Cranmer, Jiaxin Zhang, Haruko Wainwright, Adi Hanuka, Manuela Veloso, Samuel Assefa, Stephan Zheng, and Avi Pfeffer. Simulation Intelligence: Towards a New Generation of Scientific Methods. *arXiv:2112.03235 [cs]*, December 2021.

Raphael Leiteritz, Marcel Hurler, and Dirk Pflüger. Learning free-surface flow with physics-informed neural networks. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1668–1673, 2021. doi: 10.1109/ICMLA52953.2021.00266.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *International Conference on Learning Representations (ICLR)*, 2021.

G. Limousin, J.-P. Gaudet, L. Charlet, S. Szenknect, V. Barthès, and M. Krimissa. Sorption isotherms: A review on physical bases, modeling and measurement. *Applied Geochemistry*, 22(2):249–275, 2007. ISSN 0883-2927. doi: https://doi.org/10.1016/j.apgeochem.2006.09.010. URL https://www.sciencedirect.com/science/article/pii/S0883292706002629.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021. doi: 10.1137/19M1274067.

Dan MacKinlay, Dan Pagendam, Petra M Kuhnert, Tao Cui, David Robertson, and Sreekanth Janardhanan. Model Inversion for Spatio-temporal Processes using the Fourier Neural Operator. In *Neurips Workshop on Machine Learning for the Physical Sciences*, pp. 7, 2021.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1): 54–74, January 2020. doi: 10.1016/j.ijforecast.2019.04.014.

Sebastian K. Mitusch, Simon W. Funke, and Jørgen S. Dokken. Dolfin-Adjoint 2018.1: Automated adjoints for FEniCS and Firedrake. *Journal of Open Source Software*, 4(38):1292, June 2019. doi: 10.21105/joss.01292.

J. Močkus. On Bayesian Methods for Seeking the Extremum. In G. I. Marchuk (ed.), *Optimization Techniques IFIP Technical Conference: Novosibirsk, July 1–7, 1974*, Lecture Notes in Computer Science, pp. 400–404, Berlin, Heidelberg, 1975. Springer. ISBN 978-3-662-38527-2. doi: 10.1007/978-3-662-38527-2_55.

F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 1 edition, 2016. doi: 10.1007/978-3-319-16874-6.

Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

W. Nowak and A. Guthke. Entropy-based experimental design for optimal model discrimination in the geosciences. *Entropy*, 18(11), 2016. doi: 10.3390/e18110409.

A. O'Hagan. Curve Fitting and Optimal Design for Prediction. *Journal of the Royal Statistical Society: Series B (Methodological)*, 40(1):1–24, 1978. doi: 10.1111/j.2517-6161.1978.tb01643.x.

Randal S. Olson, William La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. PMLB: A large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, 10(1):36, December 2017. doi: 10.1186/s13040-017-0154-4.

Karl Otness, Arvi Gjoka, Joan Bruna, Daniele Panozzo, Benjamin Peherstorfer, Teseo Schneider, and Denis Zorin. An extensible benchmark suite for learning to simulate physical systems. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL https://openreview.net/forum?id=pY9MHwmrymR.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Christopher Rackauckas. The essential tools of scientific machine learning (Scientific ML). *The Winnower*, August 2019. doi: 10.15200/winn.156631.13064.

Maziar Raissi, P. 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*, 378:686–707, February 2019. doi: 10.1016/j.jcp.2018.10.045.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015.

Lars Ruthotto and Eldad Haber. Deep Neural Networks motivated by Partial Differential Equations. *arXiv:1804.04272 [cs, math, stat]*, April 2018.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 2951–2959. Curran Associates, Inc., 2012.

Kimberly Stachenfeld, Drummond B. Fielding, Dmitrii Kochkov, Miles Cranmer, Tobias Pfaff, Jonathan Godwin, Can Cui, Shirley Ho, Peter Battaglia, and Alvaro Sanchez-Gonzalez. Learned coarse models for efficient turbulence simulation, 2022. URL https://arxiv.org/abs/2112.15275.

James M. Stone and Michael L. Norman. ZEUS-2D: A Radiation Magnetohydrodynamics Code for Astrophysical Flows in Two Space Dimensions. I. The Hydrodynamic Algorithms and Tests. *The Astrophysical Journal Supplement*, 80:753, June 1992. doi: 10.1086/191680.

A. M. Stuart. Inverse problems: A Bayesian perspective. *Acta Numerica*, 19:451–559, 2010. doi: 10.1017/S0962492910000061.

Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, January 2005. ISBN 978-0-89871-792-1.

E. F. Toro, M. Spruce, and W. Speares. Restoration of the contact surface in the HLL-Riemann solver. *Shock Waves*, 4(1):25–34, July 1994. doi: 10.1007/BF01414629.

A. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, 237:37–72, 1952.

B. van Leer. Towards the Ultimate Conservative Difference Scheme. V. A Second-Order Sequel to Godunov's Method. *Journal of Computational Physics*, 32(1):101–136, July 1979. doi: 10.1016/0021-9991(79)90145-1.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards Physics-informed Deep Learning for Turbulent Flow Prediction. *arXiv:1911.08655 [physics, stat]*, June 2020.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2021.110768. URL `https://www.sciencedirect.com/science/article/pii/S002199912100663X`.

Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9, 2016.

Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL `https://github.com/facebookresearch/hydra`.

PDEBench: An Extensive Benchmark for Scientific Machine Learning. Supplementary Material

A Related Work

PDE benchmarking has particular challenges. Unlike many classic datasets, PDE datasets can be large on a gigabyte or terabyte scale and still contain only few data points. And unlike monolithic benchmark datasets such as ImageNet, the datasets for each PDE approximation task are specific to that task. Each set of governing equations or experiment design assumptions leads to a distinct dataset of PDE samples. Recent works in PDEs have attempted to produce standardised datasets covering well-known challenges (Otness et al., 2021; Huang et al., 2021; Stachenfeld et al., 2022). Huang et al. (2021) targets non-ML uses. Stachenfeld et al. (2022) is specialised for particular classes of equations. Of these, the excellent work of Otness et al. (2021) is most closely related, but with only four physical systems, it still lacks sufficient scale and diversity of data to challenge emerging ML algorithms. We expand the range of benchmarks in this domain by providing a larger, more diverse problem selection and scale than these previous attempts (11 PDEs with different parametrizations leading to 35 datasets). We additionally consider inverse problems for PDEs (Stuart, 2010; Tarantola, 2005), with the goal to identify unobserved latent parameters using ML. This has not been covered by benchmarks so far, despite its increasing importance in the community. Furthermore, most work in this scope considers classical statistical error measures such as the RMSE over the whole domain and at most PDE-motivated variants such as the RMSE of the gradient Otness et al. (2021). Measures based on properties of the underlying physical systems, as studied in this work, are lacking.

An overview and taxonomy of Scientific ML developments can be found in Lavin et al. (2021); Brunton & Kutz (2019). For developing our baselines, we focus on using neural network models to approximate the outputs of some *ground truth* PDE solver, given data generated by that solver, which itself aims to directly implement the numerical solution of a given partial differential equation. A range of methods aim to solve problems fitting this description, reviewed in Kashinath et al. (2021). Methods include Physics-informed neural networks (PINNs) (Raissi et al., 2019), Neural operators (NOs) (Li et al., 2021; Kovachki et al., 2021), treating ResNet as a PDE approximant (Ruthotto & Haber, 2018), custom architectures for specific problems such as TFNet for turbulent fluid flows (Wang et al., 2020), and generic image-to-image regression models such as the U-Net (Ronneberger et al., 2015). These approaches each have different assumptions, domains of applicability, and data processing requirements.

Benchmarks in machine learning are an ubiquitous feature of the field. In recent years, their design and implementation has become a research area of its own right. Easily accessible and widely used image classification benchmarks such as MNIST, CIFAR, and ImageNet are widely credited with accelerating progress in machine learning. Various domains in machine learning have widely influential datasets: In time series forecasting there are the Makridakis competitions (Makridakis et al., 2020), in reinforcement learning there is the OpenAI Gym (Brockman et al., 2016). Generic classification problems use, for example, the Penn Machine Learning Benchmark (Olson et al., 2017).

Closely related to the chosen Scientific ML baselines is the problem of directly differentiating through the numerical solver, which can itself be used in training an approximating model, or to directly solve some optimization or control problem of interest. Differentiable direct PDE solvers are increasingly available, e.g. Mitusch et al. (2019) and frequently built upon neural network technology stacks (Freeman et al., 2021; Bezgin et al., 2022; Holl & Koltun, 2020).

Recent efforts have attempted to unify Scientific ML surrogates for PDEs under a single interface. For example, NVIDIA's MODULUS/SimNet (Hennigh et al., 2020) implements a variety of methods in a single framework, although unfortunately under onerous intellectual property restrictions and an opaque contribution process. The DeepXDE project (Lu et al., 2021) is available under an open license and provides an impressive range of capabilities, but is largely restricted to PINN and DeepONet methods (Raissi et al., 2019).

B Detailed metrics description

The classic loss metrics we use are (1) root-mean-squared-error (RMSE), (2) normalized RMSE (nRMSE), (3) maximum error. These measure the emulating model's global performance but neglect local performance. Thus we include extra metrics to measure specific failure modes: (4) RMSE of the conserved value (cRMSE), (5) RMSE at boundaries (bRMSE), (6) RMSE in Fourier space (fRMSE) constrained to low, middle, and high-frequency regions.

The normalized RMSE is a variant of the RMSE to provide scale-independent information defined as:

$$\mathrm{nRMSE} \equiv \frac{||u_{\mathrm{pred}} - u_{\mathrm{true}}||_2}{||u_{\mathrm{true}}||_2}, \tag{1}$$

where $||u||_2$ is the L_2-norm of a (vector-valued) variable u, and $u_{\mathrm{true}}, u_{\mathrm{pred}}$ are true and predicted value, respectively. The maximum error measures the model's worst prediction, which quantifies both local performance and models' stability of their prediction. cRMSE is defined as $\mathrm{nRMSE} \equiv || \sum u_{\mathrm{pred}} - \sum u_{\mathrm{true}}||_2/N$, which measure the deviation of the prediction from some physically conserved value. bRMSE measures the error at the boundary, indicating if the model understand the boundary condition properly. Finally, fRMSE measures the error in low/middle/high-frequency ranges defined as

$$\frac{\sqrt{\sum_{k_{\min}}^{k_{\max}} |\mathcal{F}(u_{\mathrm{pred}}) - \mathcal{F}(u_{\mathrm{true}}))|^2}}{k_{\max} - k_{\min} + 1}, \tag{2}$$

where $\mathcal{F}$ is a discrete Fourier transformation, and $k_{\min}$, $k_{\max}$ are the minimum and maximum indices in Fourier coordinates. In our paper, we define the low/middle/high-frequency regions as Low: $k_{\min} = 0, k_{\max} = 4$, Middle: $k_{\min} = 5, k_{\max} = 12$, and High: $k_{\min} = 13, k_{\max} = \infty$. This allows a quantitative discussion of the model performance's dependence on the wavelength. In the multi-dimensional cases, we first integrate the angular coordinate direction of $|\mathcal{F}[u_{\mathrm{pred}} - u_{\mathrm{true}}](k)|^2$, and take the sum along the k-coordinate.

B.1 Inverse Problem Metrics

For the inverse problem setup, we selected various metrics. The major difference with respect to the forward metrics is that we have two main quantities to measure:

- the error of the *quantity* we want to estimate, in our case the initial condition u_0:

  $$\mathcal{L}(u_0, \hat{u}_0)$$

  where $\hat{u}_0$ is the estimated value;
- the error of the *prediction* based on the estimated initial condition $u(t, x|u_0)$,

  $$\mathcal{L}\left(u(t, x|u_0), u(t, x|\hat{u}_0)\right)$$

In general, we expect a larger error when we measure the error in the estimated quantity w.r.t. the predicted quantity. This is mainly due to the early decay of high frequencies of the PDE. We evaluated the error of the prediction at a specific instant in time $t = T$, that has been selected as $T = 15$ for all the tested datasets, expect $T = 5$ for the CFD dataset.

The metrics that we used for the inverse problem are: 1) MSE 2) the normalized ℓ_2 norm (L2), 3) the normalized ℓ_3 norm (L3); 4) the FFT MSE, the FFT L2, and 5) the FFT L3. For the frequency metrics we investigated the low frequency (between 0 and $1/4$ of the max frequency), the middle frequency (between $1/4$ and $3/4$) and high frequency (between $3/4$ and the maximum frequency) ranges. In Fig.9, the right figure shows the frequency power density, where we see that the largest error is found in the middle frequency range.

C Training Protocol and Hyperparameters

The model was trained for 500 epochs with the Adam optimizer (Kingma & Ba, 2014) as per the protocol of the original FNO. The initial learning rate was set as 10^{-3} and reduced by half after each

100 epochs. The datasets are split into 90% training and 10% validation and testing. For the PINNs, we use DeepXDE (Lu et al., 2021) implementation. The training was performed for 15,000 epochs with the Adam optimizer, with the learning rate set to 10^{-3}. As with the example problems from that library we use a fully-connected network of depth 6 with 40 neurons each. In contrast to the other surrogate models, the PINN baseline can be trained and tested only on a single sample, and is valid only for a specific initial and boundary condition. To get more reliable error bounds, we thus chose to train the PINN baseline for 10 different samples per dataset and average the resulting error metrics.

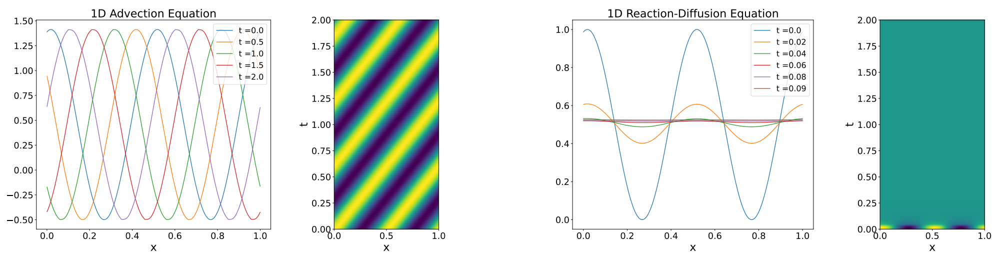
C.1 Inverse problem

For testing the power of surrogate models to solve inverse problems, we consider a simplified scenario where the machine learning model directly predicts a specific time in the future $t = T$. When training to predict a specific time in the future, we reduce the training time and avoid to consider the effect of training approaches (as discussed in the temporal analysis section **??**) in evaluating the surrogate models. We trained over $N_{\text{epoch}} = 20$ epochs and we selected as final time step $T = 15$ for all tested datasets, expect for the CFD dataset where we selected $T = 5$. We used similar parameters used in the forward training, while we selected 64 hidden values to be estimated for the initial condition and 100 samples to test and 0.2 as learning rate for the gradient method. The loss function for the gradient computation is the MSE.

D Detailed Problem Description

In this section, we provide more detailed descriptions of each PDE and its applications. Note that PDE is the basic mathematical tool to describe the evolution of the system in physics. Interested readers are referred to representative textbooks of physics, for example, (Feynman, 1963).

D.1 1D Advection Equation



1D Advection ($\beta = 0.4$) 1D Reaction-Diffusion ($\nu = 0.5, \rho = 1$)

Figure 3: Visualization of the time evolution of 1D Advection equation and Reaction-Diffusion equation.

The advection equation models pure advection behavior without non-linearity whose expression is given as:

$$\partial_t u(t,x) + \beta \partial_x u(t,x) = 0, \quad x \in (0,1), t \in (0,2], \tag{3}$$

$$u(0,x) = u_0(x),\ x \in (0,1), \tag{4}$$

where β is a constant advection speed. Note that the exact solution of the system is given as: $u(t,x) = u_0(x - \beta t)$.

In our dataset, we only considered the periodic boundary condition. As an initial condition, we use a super-position of sinusoidal waves as:

$$u_0(x) = \sum_{k_i=k_1,\dots,k_N} A_i \sin(k_i x + \phi_i), \tag{5}$$

where $k_i = 2\pi\{n_i\}/L_x$ are wave numbers whose $\{n_i\}$ are integer numbers selected randomly in $[1, n_{\max}]$, N is the integer determining how many waves to be added, L_x is the calculation domain size, A_i is a random float number uniformly chosen in $[0, 1]$, and ϕ_i is the randomly chosen phase in $(0, 2\pi)$. In 1D-advection case, we set $k_{\max} = 8$ and $N = 2$. After calculating Equation 5, we randomly operate the absolute value function with random signature and the window-function with 10% probability, respectively.

The numerical solution was calculated with the temporally and spatially 2nd-order upwind finite difference scheme.

D.2 1D Diffusion-Reaction Equation

Here, we consider a one-dimensional diffusion-reaction type PDE, that combines a diffusion process and a rapid evolution from a source term Krishnapriyan et al. (2021). The equation is expressed as:

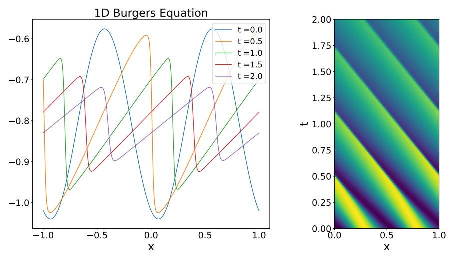
$$\partial_t u(t,x) - \nu \partial_{xx} u(t,x) - \rho u(1-u) = 0, \quad x \in (0,1), t \in (0,1], \tag{6}$$

$$u(0,x) = u_0(x), \quad x \in (0,1). \tag{7}$$

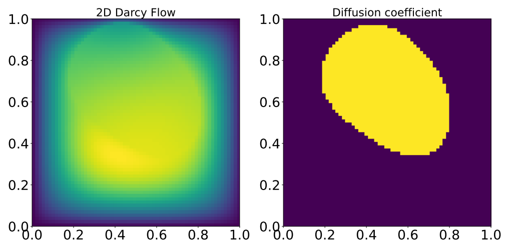
Note that the variable u develops at potentially exponential rate because of the force term which depends on u. measure the ability to capture very rapid dynamics.

Similar to the 1D advection equation case, we use the periodic boundary condition and Equation 5 as the initial condition. To avoid an ill-defined initial condition, we also applied the absolute value function and a normalization operation, dividing the initial condition by the maximum value. The numerical solution was calculated with the temporally and spatially 2nd-order central difference scheme. For the source term part, we use the piecewise-exact solution (PES) method (Inoue et al., 2007).

D.3 Burgers equation



1D Burgers ($\nu = 0.01$)



2D Darcy Flow ($\beta = 1.0$)

Figure 4: Visualization of the time evolution of 1D Burgers equation and 2D Darcy Flow.

The Burgers' equation is a PDE modeling the non-linear behavior and diffusion process in fluid dynamics as

$$\partial_t u(t,x) + \partial_x (u^2(t,x)/2) = \nu/\pi \partial_{xx} u(t,x), \quad x \in (0,1), t \in (0,2], \tag{8}$$

$$u(0,x) = u_0(x), \; x \in (0,1), \tag{9}$$

where ν is the diffusion coefficient, which is assumed constant in our dataset.

Note that setting $R \equiv \pi u L/\nu$ describes the system's evolution as the Reynolds number of the Navier-Stokes equation equation 13b; $R > 1$ means the strong non-linear case support forming shock phenomena, and $R < 1$ means the diffusive case.

Similar to the 1D advection equation case, we use the periodic boundary condition and Equation 5 as the initial condition. The numerical solution was calculated with the temporally and spatially 2nd-order upwind difference scheme for the advection term, and the central difference scheme for the diffusion term.

D.4 Darcy Flow

We experiment with the steady-state solution of 2D Darcy Flow over the unit square, whose viscosity term $a(x)$ is an input of the system. The solution of the steady-state is defined by the following equation

$$-\nabla(a(x)\nabla u(x)) = f(x), \quad x \in (0,1)^2, \tag{10}$$

$$u(x) = 0, \quad x \in \partial(0,1)^2. \tag{11}$$

In this paper, the force term f is set as a constant value β, changing the scale of the solution $u(x)$. Instead of directly solving Equation 10, we obtained the solution by solving a temporal evolution equation:

$$\partial_t u(x,t) - \nabla(a(x)\nabla u(x,t)) = f(x), \quad x \in (0,1)^2, \tag{12}$$

with random field initial condition, until reaching a steady state. The numerical calculation was performed the same as the case of the 1D Diffusion-Reaction equation.

D.5 Compressible Navier-Stokes equation

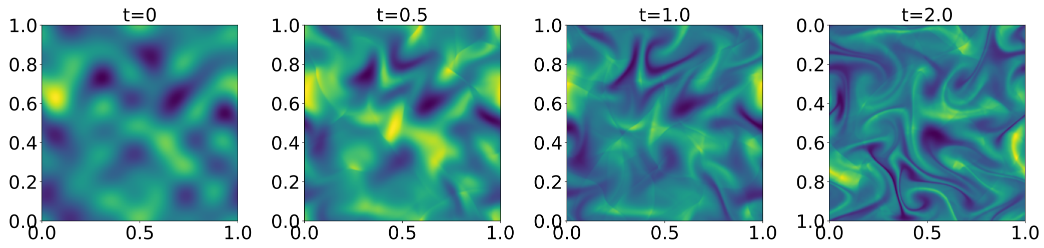


Figure 5: Visualization of the time evolution of the density in the case of 2D Compressible Navier-Stokes equations (inviscid, $M = 0.1$).

The compressible fluid dynamic equations describe a fluid flow,

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{v}) = 0, \tag{13a}$$

$$\rho(\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}) = -\nabla p + \eta \triangle \mathbf{v} + (\zeta + \eta/3)\nabla(\nabla \cdot \mathbf{v}), \tag{13b}$$

$$\partial_t \left[\epsilon + \frac{\rho v^2}{2}\right] + \nabla \cdot \left[\left(\epsilon + p + \frac{\rho v^2}{2}\right)\mathbf{v} - \mathbf{v} \cdot \sigma'\right] = 0, \tag{13c}$$

where ρ is the mass density, $\mathbf{v}$ is the velocity, p is the gas pressure, $\epsilon = p/(\Gamma - 1)$ is the internal energy, $\Gamma = 5/3$, σ' is the viscous stress tensor, and η, ζ are the shear and bulk viscosity, respectively.

PDEBench provides the following training datasets for the compressible Navier-Stokes equations:

N_d	initial field	boundary condition	(η, ζ, M)
1D	random field	periodic	$(10^{-8},\ 10^{-8},\ -)$
1D	random field	periodic	$(10^{-2},\ 10^{-2},\ -)$
1D	random field	periodic	$(10^{-1},\ 10^{-1},\ -)$
1D	random field	out-going	$(10^{-8},\ 10^{-8},\ -)$
1D	shock-tube	out-going	$(10^{-8},\ 10^{-8},\ -)$
2D	random field	periodic	$(10^{-8},\ 10^{-8},\ 0.1)$
2D	random field	periodic	$(10^{-2},\ 10^{-2},\ 0.1)$
2D	random field	periodic	$(10^{-1},\ 10^{-1},\ 0.1)$
2D	random field	periodic	$(10^{-8},\ 10^{-8},\ 1.0)$
2D	random field	periodic	$(10^{-2},\ 10^{-2},\ 1.0)$
2D	random field	periodic	$(10^{-1},\ 10^{-1},\ 1.0)$
2D	turbulence	periodic	$(10^{-8},\ 10^{-8},\ 0.1)$
2D	turbulence	periodic	$(10^{-8},\ 10^{-8},\ 1.0)$
3D	random field	periodic	$(10^{-8},\ 10^{-8},\ 1.0)$
3D	random field	periodic	$(10^{-2},\ 10^{-2},\ 1.0)$

where N_d is the number of spatial dimensions, $M = |v|/c_s$ is the Mach number, $c_s = \sqrt{\Gamma p/\rho}$ is the sound velocity. The outgoing boundary condition is copying the neighbor cell to the boundary region which allows waves and fluid to escape from the computational domain, and is popular for astrohydrodynamics simulations (Stone & Norman, 1992). The random field initial condition is applying Equation 5 which is extended to higher dimensions for the 2D and 3D cases. Note that density and pressure are prepared by adding a uniform background to the perturbation field Equation 5. The turbulence initial condition considers turbulent velocity with uniform mass density and pressure. The velocity is calculated similarly to Equation 5 as

$$\mathbf{v}(x, t = 0) = \sum_{i=1}^{n} \mathbf{A}_i \sin(k_i x + \phi_i), \tag{14}$$

where $n = 4$ and $A_i = \bar{v}/|k|^d$, and $d = 1, 2$ when considering 2D and 3D, respectively. $\bar{v}$ is determined by the initial Mach number as $\bar{v} = c_s M$. To reduce the compressibility effect, we subtracted the compressible field from Equation 14 by the Helmholtz-decomposition in the Fourier space.

The shock-tube initial field is composed as $Q(x, t = 0) = (Q_L, Q_R)$, where $Q = (\rho, \mathbf{v}, p)$ and Q_L, Q_R are randomly determined constant values. The location of the initial discontinuity is also randomly determined. This problem is called the "Riemann problem", and the initial discontinuity generates shocks and rarefaction depending on the values of Q_L, Q_R, which are very difficult to obtain without solving the PDEs. This scenario can be used for a rigorous test if ML models fully understand Equation 13a - Equation 13c. The numerical solution was calculated with the temporally and spatially 2nd-order HLLC scheme (Toro et al., 1994) with the MUSCL method (van Leer, 1979) for the inviscid part, and the central difference scheme for the viscous part.

D.6 Inhomogenous, incompressible Navier-Stokes

A popular simplification of the Navier-Stokes equation is the incompressible version, commonly used to model dynamics supposed to be far lower than the speed of propagation of waves in the medium,

$$\nabla \cdot \mathbf{v} = 0, \quad \rho(\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}) = -\nabla p + \eta \triangle \mathbf{v}. \tag{15}$$

These simplify the compressible Navier-Stokes equations Eq. equation 13b, by substituting the first term in Eq. equation 15 instead of the first term in equation 13, from which we can eliminate several elements in the second terms of Eq. equation 15. Additionally, we have introduced the assumption that the fluid is homogeneous (i.e. not a fluid comprising two or more substances of different density or viscosity).

We employ an augmented form of equation 15 which includes a vector field *forcing* term **u**,

$$\rho(\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}) = -\nabla p + \eta \triangle \mathbf{v} + \mathbf{u}. \tag{16}$$

Non-periodic conditions are included to challenge models which perform well upon periodic domains, such as the FNO (Li et al., 2021). The forcing term poses challenges based upon spatially heterogeneous dynamics. Firstly, this allows us to see if the prediction methods can successfully learn to predict in the presence of heterogeneity. Secondly, this permits us to use the spatially varying random field as a target for inverse inference.

Initial conditions $\mathbf{v}_0$ and inhomogeneous forcing parameters **u** are each drawn from isotropic Gaussian random fields with truncated power-law decay τ of the power spectral density and scale σ, where $\tau_{\mathbf{v}_0} = -3, \sigma_{\mathbf{v}_0} = 0.15, \tau_{\mathbf{u}} = -1, \sigma_{\mathbf{u}} = 0.4$. The variation in the resulting field is due to the alteration in the random seed. We set the domain to the unit square $\Omega = [0, 1]^2$, the viscosity to $\nu = 0.01$. Simulations are implemented using Phiflow Holl & Koltun (2020). Boundary conditions are Dirichlet, clamping field velocity to null at the perimeter.

D.7 2D Shallow-Water Equations

The shallow-water equations, derived from the general Navier-Stokes equations, present a suitable framework for modelling free-surface flow problems. In 2D, these come in the form of the following

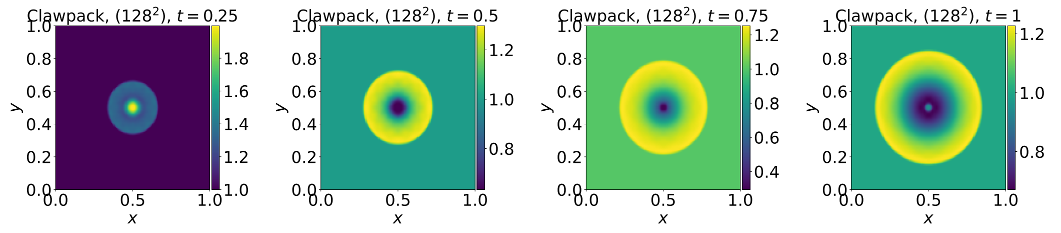


Figure 6: Visualization of the time evolution of the 2D shallow-water equations data.

system of hyperbolic PDEs,

$$\partial_t h + \partial_x hu + \partial_y hv = 0\,, \tag{17a}$$

$$\partial_t hu + \partial_x \left(u^2 h + \frac{1}{2} g_r h^2\right) + \partial_y uvh = -g_r h \partial_x b\,, \tag{17b}$$

$$\partial_t hv + \partial_y \left(v^2 h + \frac{1}{2} g_r h^2\right) + \partial_x uvh = -g_r h \partial_y b\,, \tag{17c}$$

with u, v being the velocities in horizontal and vertical direction, h describing the water depth and b describing a spatially varying bathymetry. hu, hv can be interpreted as the directional momentum components and g_r describes the gravitational acceleration.

The specific simulation we include in our benchmark for the shallow-water equations problem as introduced in D.7 is a 2D radial dam break scenario. On a square domain $\Omega = [-2.5, 2.5]^2$ we initialize the water height as a circular bump in the center of the domain

$$h(t=0,x,y) = \begin{cases} 2.0, & \text{for } r < \sqrt{x^2+y^2} \\ 1.0, & \text{for } r \geq \sqrt{x^2+y^2} \end{cases} \tag{18}$$

with the radius r randomly sampled from $\mathcal{U}(0.3, 0.7)$. For generating the datasets we simulate this problem using the PyClaw Ketcheson et al. (2012) Python package which offers a comprehensive finite volume solver. A time evolution visualization of the equation is shown in Figure 6.

D.8 Diffusion-Sorption Equation

The diffusion-sorption equation models a diffusion process which is retarded by a sorption process. The equation is written as

$$\partial_t u(t,x) = D/R(u) \partial_{xx} u(t,x), \quad x \in (0,1), t \in (0,500]. \tag{19}$$

where D is the effective diffusion coefficient, R is the retardation factor representing the sorption that hinders the diffusion process. Note that R is dependent on the variable u. This equation is applicable to real world scenarios, one of the most prominent being groundwater contaminant transport.

This equation is retarded by the retardation factor R which is dependent on u based on the Freundlich sorption isotherm Limousin et al. (2007):

$$R(u) = 1 + \frac{1-\phi}{\phi} \rho_s k n_f u^{n_f - 1}, \tag{20}$$

where $\phi = 0.29$ is the porosity of the porous medium, $\rho_s = 2\,880$ is the bulk density, $k = 3.5 \times 10^{-4}$ is the Freundlich's parameter, $n_f = 0.874$ is the Freundlich's exponent, and the effective diffusion coefficient $D = 5 \times 10^{-4}$. The initial condition is generated with a uniform distribution $u(0,x) \sim \mathcal{U}(0, 0.2)$ for $x \in (0,1)$. We provide datasets discretized into $N_x = 1024$ and $N_t = 501$, as well as the temporally downsampled version for the models training with $N_t = 101$. The spatial discretization is performed using the finite volume method Moukalled et al. (2016) and the time integration using the built-in fourth order Runge-Kutta method in the *scipy* package Virtanen et al. (2020).

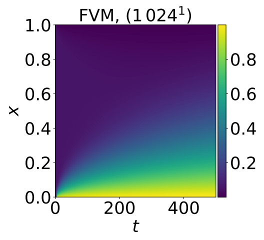


Figure 7: Visualization of the time evolution of the 1D diffusion-sorption equations data.

This particular example is interesting because of a few things. First, the diffusion coefficient becomes non-linear with dependency on u. And based on Equation 20, it is clear that there is a singularity when $u = 0$. Second, it is highly applicable to a real-world problem, namely the groundwater contaminant transport Nowak & Guthke (2016). To date, application of machine learning to real-world physics problems is still rare. Third, we employ boundary conditions that are not the usual zero or periodic conditions that can be easily padded in models with a convolutional structure. Here, we use $u(t, 0) = 1.0$ and $u(t, 1) = D\partial_x u(t, 1)$. The second boundary condition is particularly challenging since it uses a derivative instead of a constant value. For generating the datasets we simulate this problem using a standard finite volume solver. A time evolution visualization of the equation is shown in Figure 7.

D.9 2D Diffusion-Reaction Equation

In addition to the 1D diffusion-reaction equation, which involves only a single variable, we also consider extending the application to a 2D domain, with two non-linearly coupled variables, namely the activator $u = u(t, x, y)$ and the inhibitor $v = v(t, x, y)$. The equation is written as

$$\partial_t u = D_u \partial_{xx} u + D_u \partial_{yy} u + R_u, \quad \partial_t v = D_v \partial_{xx} v + D_v \partial_{yy} v + R_v \,, \tag{21}$$

where D_u and D_v are the diffusion coefficient for the activator and inhibitor, respectively, $R_u = R_u(u, v)$ and $R_v = R_v(u, v)$ are the activator and inhibitor reaction function, respectively. The domain of the simulation includes $x \in (-1, 1), y \in (-1, 1), t \in (0, 5]$. This equation is applicable most prominently for modeling biological pattern formation.

The reaction functions for the activator and inhibitor are defined by the Fitzhugh-Nagumo equation Klaasen & Troy (1984), written as:

$$R_u(u, v) = u - u^3 - k - v, \tag{22}$$

$$R_v(u, v) = u - v, \tag{23}$$

where $k = 5 \times 10^{-3}$, and the diffusion coefficients for the activator and inhibitor are $D_u = 1 \times 10^{-3}$ and $D_v = 5 \times 10^{-3}$, respectively. The initial condition is generated as standard normal random noise $u(0, x, y) \sim \mathcal{N}(0, 1.0)$ for $x \in (-1, 1)$ and $y \in (-1, 1)$. We provide datasets discretized into $N_x = 512$, $N_y = 512$ and $N_t = 501$, as well as the downsampled version for the models training with $N_x = 128$, $N_y = 128$, and $N_t = 101$. As in the 1D diffusion-sorption equation, the spatial discretization is performed using the finite volume method Moukalled et al. (2016), and the time integration is performed using the built-in fourth order Runge-Kutta method in the *scipy* package Virtanen et al. (2020).

We included the 2D diffusion-reaction equation as an example because it serves as a challenging benchmark problem. First, there are two variables of interest, namely the activator and inhibitor, which are non-linearly coupled. Second, it also has applicability in real-world problems, namely biological pattern formation Turing (1952). Third, we also employ a no-flow Neumann boundary condition, meaning that $D_u \partial_x u = 0$, $D_v \partial_x v = 0$, $D_u \partial_y u = 0$, and $D_v \partial_y v = 0$ for $x, y \in (-1, 1)^2$. For generating the datasets we simulate this problem using a standard finite volume solver. A time evolution visualization of the equation is shown in Figure 8.

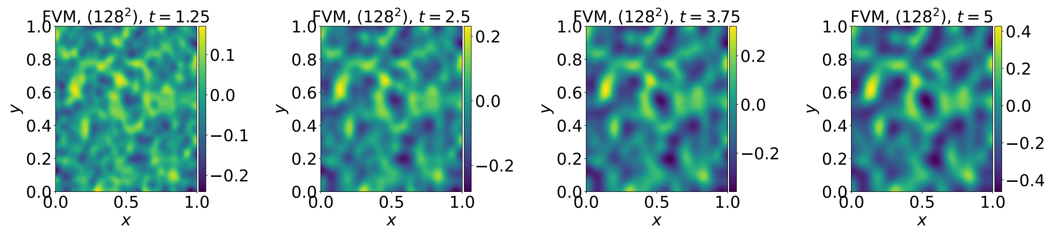


Figure 8: Visualization of the time evolution of the 2D diffusion-reaction equations data.

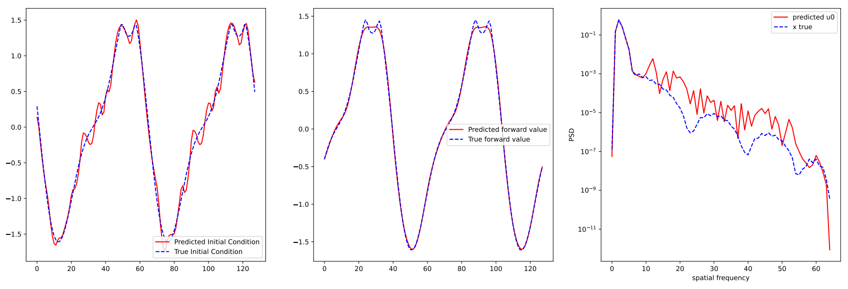


Figure 9: Inverse problem for the 1d advection equation with $\beta = 0.1$. The spectra density where most of the error is concentrated in the higher frequencies is depicted on the right.

D.10 Gradient-Based Inverse Method

The inverse problem aims at solving an inverse inference by minimising the prediction loss(Cao, 2018; Nocedal & Wright, 1999),

$$\begin{aligned}&\mathcal{L}(u(t=T,x|u_0), u(t=T,x|\hat{u}_0))\\ &\text{where} \quad \hat{u}_0 \sim p_\theta(u_0|u(t=T,x))\,.\end{aligned}$$

The generation process $p_\theta(u_0|u(t=T,x))$ is a deterministic function, whose parameters θ use a bilinear interpolation to recover the initial condition (MacKinlay et al., 2021).

Figure 9 shows the solution of the inverse problem for the 1d advection equation. On the left, we see the true and estimated initial condition, and on the right the power density in the frequency domain. As we can see, the error is concentrated in the mid-high frequencies. In the middle we have the true and predicted value at time $t = T$. The error is smaller then in the plot on the left.

Table 1, Table 2 and Table 3 show the error in the spatial and frequency domain of 4 datasets and using FNO and U-Net as surrogate models. In Fig.9, the left figure visualizes the true and the estimated initial condition, while the middle figure is the predicted and the true value. As shown in the figure on the right, the largest error is in the higher frequencies. This effect is also visible from the frequency metrics of Tab.2 and Tab.3. In the experiment we use the same initial and boundary conditions of the forward problem.

E Detailed Baseline Score

F Detailed Runtime Comparison

In this section we present the detailed comparison of computation time between the PDE solver used to generate the data and the baseline models used in this work, summarized in Table 14. The system listed in Table 13 was used to run all timing measurements regarding the Diffusion-sorption, 2D diffusion-reaction and Shallow-water equation scenarios. PyClaw (Ketcheson et al., 2012), a well-optimized finite-volume Fortran code, is used as PDE solver for the shallow-water equation data generation. Note that the experiment is only running on a single core due to its small size. Because the PINN model is not discretized, the inference time includes evaluating the trained model at the same discretization points of the reference simulation for the last 20 time steps of the data. E.g. the

		Forward model	
PDE	Metric	FNO	U-Net
Advection_beta4	MSE	$2.4 \times 10^{-3} \pm 3.4 \times 10^{-3}$	$1.0 \times 10^{+0} \pm 5.6 \times 10^{-2}$
	nL2	$3.9 \times 10^{-2} \pm 2.9 \times 10^{-2}$	$1.0 \times 10^{+0} \pm 2.8 \times 10^{-2}$
	nL3	$4.4 \times 10^{-2} \pm 3.3 \times 10^{-2}$	$1.0 \times 10^{+0} \pm 2.9 \times 10^{-2}$
	MSE'	$2.9 \times 10^{-4} \pm 5.8 \times 10^{-4}$	$9.9 \times 10^{-1} \pm 2.5 \times 10^{-2}$
	nL2'	$1.4 \times 10^{-2} \pm 1.1 \times 10^{-2}$	$1.0 \times 10^{+0} \pm 8.0 \times 10^{-3}$
	nL3'	$1.6 \times 10^{-2} \pm 1.3 \times 10^{-2}$	$1.0 \times 10^{+0} \pm 8.4 \times 10^{-3}$
Burgers_Nu1	MSE	$1.0 \times 10^{+0} \pm 2.2 \times 10^{-1}$	$1.3 \times 10^{+0} \pm 2.3 \times 10^{-1}$
	nL2	$1.0 \times 10^{+0} \pm 1.0 \times 10^{-1}$	$1.1 \times 10^{+0} \pm 1.0 \times 10^{-1}$
	nL3	$1.0 \times 10^{+0} \pm 1.0 \times 10^{-1}$	$1.1 \times 10^{+0} \pm 1.1 \times 10^{-1}$
	MSE'	$1.3 \times 10^{-4} \pm 2.8 \times 10^{-4}$	$2.5 \times 10^{-3} \pm 1.9 \times 10^{-3}$
	nL2'	$7.0 \times 10^{-1} \pm 4.6 \times 10^{-1}$	$1.6 \times 10^{+1} \pm 2.0 \times 10^{+1}$
	nL3'	$7.0 \times 10^{-1} \pm 4.4 \times 10^{-1}$	$1.7 \times 10^{+1} \pm 2.1 \times 10^{+1}$
CFD_Shock_Trans	MSE	$3.4 \times 10^{+0} \pm 5.3 \times 10^{-1}$	$1.1 \times 10^{+2} \pm 2.0 \times 10^{+1}$
	nL2	$1.8 \times 10^{+0} \pm 1.4 \times 10^{-1}$	$1.0 \times 10^{+1} \pm 1.1 \times 10^{+0}$
	nL3	$1.9 \times 10^{+0} \pm 2.7 \times 10^{-1}$	$1.1 \times 10^{+1} \pm 1.6 \times 10^{+0}$
	MSE'	$1.0 \times 10^{-1} \pm 5.9 \times 10^{-2}$	$4.2 \times 10^{-1} \pm 9.2 \times 10^{-1}$
	nL2'	$3.3 \times 10^{-1} \pm 8.5 \times 10^{-2}$	$5.8 \times 10^{-1} \pm 3.9 \times 10^{-1}$
	nL3'	$3.6 \times 10^{-1} \pm 9.6 \times 10^{-2}$	$6.0 \times 10^{-1} \pm 4.0 \times 10^{-1}$
ReacDiff_Nu1_Rho2	MSE	$1.7 \times 10^{+0} \pm 2.1 \times 10^{-1}$	$2.0 \times 10^{+0} \pm 3.8 \times 10^{-1}$
	nL2	$1.3 \times 10^{+0} \pm 8.4 \times 10^{-2}$	$1.4 \times 10^{+0} \pm 1.3 \times 10^{-1}$
	nL3	$1.3 \times 10^{+0} \pm 8.1 \times 10^{-2}$	$1.5 \times 10^{+0} \pm 1.3 \times 10^{-1}$
	MSE'	$5.4 \times 10^{-2} \pm 1.2 \times 10^{-1}$	$6.4 \times 10^{-1} \pm 3.5 \times 10^{-1}$
	nL2'	$1.2 \times 10^{-1} \pm 1.2 \times 10^{-1}$	$7.3 \times 10^{-1} \pm 5.1 \times 10^{-2}$
	nL3'	$1.2 \times 10^{-1} \pm 1.2 \times 10^{-1}$	$7.3 \times 10^{-1} \pm 5.0 \times 10^{-2}$

Table 1: Error of the inverse problem. The prime indicates the error of the predition, for example MSE' is the MSE at time $t = T$. The MSE for example in the first row is one order of magnitude lower. nL2 and nL3 are the normalized L2 and L3 norm error, nLp = $||\hat{\boldsymbol{y}} - \boldsymbol{y}||_p / ||\boldsymbol{y}||_p, p = 2, 3$.

		Forward model	
PDE	Metric	FNO	U-Net
Advection_beta4	fMSE	3.04×10^{-1}	$1.29\times10^{+2}$
	fMSE low	5.56×10^{-1}	$1.29\times10^{+2}$
	fMSE mid	5.26×10^{-2}	$1.29\times10^{+2}$
	fMSE high	3.03×10^{-1}	$1.29\times10^{+2}$
	fMSE’	3.67×10^{-2}	9.92×10^{-1}
	fMSE’ low	1.60×10^{-2}	9.92×10^{-1}
	fMSE’ mid	5.74×10^{-2}	9.92×10^{-1}
	fMSE’ high	3.68×10^{-2}	9.92×10^{-1}
	fL2	3.91×10^{-2}	$1.00\times10^{+0}$
	fL2 low	3.75×10^{-2}	$1.01\times10^{+0}$
	fL2 mid	$1.41\times10^{+1}$	$0.00\times10^{+0}$
	fL2 high	3.90×10^{-2}	$0.00\times10^{+0}$
Burgers_Nu1	fMSE	$1.29\times10^{+2}$	$1.59\times10^{+2}$
	fMSE low	$2.58\times10^{+2}$	$1.59\times10^{+2}$
	fMSE mid	1.19×10^{-1}	$1.59\times10^{+2}$
	fMSE high	$1.29\times10^{+2}$	$1.59\times10^{+2}$
	fMSE’	1.67×10^{-2}	2.46×10^{-3}
	fMSE’ low	3.36×10^{-2}	2.46×10^{-3}
	fMSE’ mid	9.26×10^{-7}	2.46×10^{-3}
	fMSE’ high	1.66×10^{-2}	2.46×10^{-3}
	fL2	9.98×10^{-1}	$1.11\times10^{+0}$
	fL2 low	9.98×10^{-1}	$1.11\times10^{+0}$
	fL2 mid	$3.50\times10^{+0}$	$0.00\times10^{+0}$
	fL2 high	9.98×10^{-1}	$0.00\times10^{+0}$
CFD_Shock_Trans	fMSE	$4.37\times10^{+2}$	$1.40\times10^{+4}$
	fMSE low	$4.37\times10^{+2}$	$1.40\times10^{+4}$
	fMSE mid	$4.37\times10^{+2}$	$1.40\times10^{+4}$
	fMSE high	$4.37\times10^{+2}$	$1.40\times10^{+4}$
	fMSE’	$1.28\times10^{+1}$	$2.19\times10^{+2}$
	fMSE’ low	$3.21\times10^{+1}$	$2.19\times10^{+2}$
	fMSE’ mid	$1.13\times10^{+0}$	$2.19\times10^{+2}$
	fMSE’ high	$8.98\times10^{+0}$	$2.19\times10^{+2}$
	fL2	$1.84\times10^{+0}$	$1.04\times10^{+1}$
	fL2 low	$1.51\times10^{+0}$	$9.95\times10^{+0}$
	fL2 mid	$0.00\times10^{+0}$	$0.00\times10^{+0}$
	fL2 high	$0.00\times10^{+0}$	$0.00\times10^{+0}$
ReacDiff_Nu1_Rho2	fMSE	$2.17\times10^{+2}$	$2.55\times10^{+2}$
	fMSE low	$6.10\times10^{+2}$	$2.55\times10^{+2}$
	fMSE mid	1.48×10^{-2}	$2.55\times10^{+2}$
	fMSE high	$1.28\times10^{+2}$	$2.55\times10^{+2}$
	fMSE’	$6.94\times10^{+0}$	6.35×10^{-1}
	fMSE’ low	$2.77\times10^{+1}$	6.35×10^{-1}
	fMSE’ mid	1.14×10^{-5}	6.35×10^{-1}
	fMSE’ high	1.29×10^{-4}	6.35×10^{-1}
	fL2	$1.30\times10^{+0}$	$1.41\times10^{+0}$
	fL2 low	$1.54\times10^{+0}$	$1.60\times10^{+0}$
	fL2 mid	$7.45\times10^{+0}$	$0.00\times10^{+0}$
	fL2 high	$1.00\times10^{+0}$	$0.00\times10^{+0}$

Table 2: Frequency error of the inverse problem. fMSE, fL2 and fL3 are the frequency version of the MSE, normalized L2 and L3 norm metrics. Low, mid and high is the range of frequencies. Prime is used for the error in the prediction, without the error of the initial condition estimation. Normalised metric are not well defined, when the original signal is zero.

		Forward model	
PDE	Metric	FNO	U-Net
Advection_beta4	fL2'	1.36×10^{-2}	$1.13\times10^{+1}$
	fL2' low	7.50×10^{-3}	$5.66\times10^{+0}$
	fL2' mid	$1.61\times10^{+0}$	$5.27\times10^{+1}$
	fL2' high	1.36×10^{-2}	$8.01\times10^{+0}$
	fL3	3.14×10^{-2}	$1.00\times10^{+0}$
	fL3 low	3.12×10^{-2}	$1.00\times10^{+0}$
	fL3 mid	$1.75\times10^{+1}$	$0.00\times10^{+0}$
	fL3 high	3.14×10^{-2}	$0.00\times10^{+0}$
	fL3'	9.51×10^{-3}	$5.04\times10^{+0}$
	fL3' low	5.62×10^{-3}	$3.18\times10^{+0}$
	fL3' mid	$1.47\times10^{+0}$	$2.77\times10^{+1}$
	fL3' high	9.51×10^{-3}	$4.00\times10^{+0}$
Burgers_Nu1	fL2'	7.00×10^{-1}	$1.82\times10^{+2}$
	fL2' low	7.99×10^{-1}	$6.28\times10^{+1}$
	fL2' mid	$1.03\times10^{+2}$	$5.84\times10^{+5}$
	fL2' high	5.39×10^{-1}	$1.31\times10^{+2}$
	fL3	9.97×10^{-1}	$1.04\times10^{+0}$
	fL3 low	9.97×10^{-1}	$1.04\times10^{+0}$
	fL3 mid	$3.58\times10^{+0}$	$0.00\times10^{+0}$
	fL3 high	9.97×10^{-1}	$0.00\times10^{+0}$
	fL3'	7.21×10^{-1}	$4.88\times10^{+1}$
	fL3' low	7.99×10^{-1}	$2.40\times10^{+1}$
	fL3' mid	$9.35\times10^{+1}$	$2.98\times10^{+5}$
	fL3' high	5.39×10^{-1}	$3.92\times10^{+1}$
CFD_Shock_Trans	fL2'	3.34×10^{-1}	$2.12\times10^{+0}$
	fL2' low	2.68×10^{-1}	$2.14\times10^{+0}$
	fL2' mid	$0.00\times10^{+0}$	$0.00\times10^{+0}$
	fL2' high	$0.00\times10^{+0}$	$0.00\times10^{+0}$
	fL3	$1.26\times10^{+0}$	$9.41\times10^{+0}$
	fL3 low	$1.11\times10^{+0}$	$9.36\times10^{+0}$
	fL3 mid	$0.00\times10^{+0}$	$0.00\times10^{+0}$
	fL3 high	$0.00\times10^{+0}$	$0.00\times10^{+0}$
	fL3'	2.16×10^{-1}	$2.19\times10^{+0}$
	fL3' low	1.96×10^{-1}	$2.20\times10^{+0}$
	fL3' mid	$0.00\times10^{+0}$	$0.00\times10^{+0}$
	fL3' high	$0.00\times10^{+0}$	$0.00\times10^{+0}$
ReacDiff_Nu1_Rho2	fL2'	1.23×10^{-1}	$1.18\times10^{+1}$
	fL2' low	1.23×10^{-1}	$5.83\times10^{+0}$
	fL2' mid	$1.89\times10^{+18}$	$1.90\times10^{+21}$
	fL2' high	$9.03\times10^{+18}$	$3.93\times10^{+21}$
	fL3	$1.27\times10^{+0}$	$1.29\times10^{+0}$
	fL3 low	$1.45\times10^{+0}$	$1.47\times10^{+0}$
	fL3 mid	$7.07\times10^{+0}$	$0.00\times10^{+0}$
	fL3 high	$1.00\times10^{+0}$	$0.00\times10^{+0}$
	fL3'	1.23×10^{-1}	$5.08\times10^{+0}$
	fL3' low	1.23×10^{-1}	$3.18\times10^{+0}$
	fL3' mid	$1.07\times10^{+18}$	$7.25\times10^{+20}$
	fL3' high	$6.54\times10^{+18}$	$1.14\times10^{+21}$

Table 3: Frequency error of the prediction of the inverse problem. fMSE, fL2 and fL3 are the frequency version sof the MSE, normalized L2 and L3 norm metrics. Low, mid and high is the range of the frequencies. Prime is used for the error in the prediction, without the error of the initial condition estimation. Normalised metric are not well defined, when the original signal is zero.

Table 4: Summary of the baseline models' performance for different evaluation metrics: RMSE, normalised RMSE (nRMSE), RMSE from conserved value (cRMSE), maximum error, RMSE at the boundaries (bRMSE), RMSE in Fourier space at low (fRMSE low), medium (fRMSE mid), and high frequency (fRMSE high) ranges applied to the diffusion-sorption, 2D diffusion-reaction, and shallow-water equations.

			Baseline model		
PDE	Parameter	Metric	U-Net	FNO	PINN
Diffusion-sorption	N/A	RMSE	5.8×10^{-2}	5.9×10^{-4}	9.9×10^{-2}
		nRMSE	1.5×10^{-1}	1.7×10^{-3}	2.2×10^{-1}
		max error	2.9×10^{-1}	7.8×10^{-3}	2.2×10^{-1}
		cRMSE	4.8×10^{-2}	1.9×10^{-4}	7.5×10^{-2}
		bRMSE	6.1×10^{-3}	2.0×10^{-3}	1.4×10^{-1}
		fRMSE low	1.9×10^{-2}	1.5×10^{-4}	3.5×10^{-2}
		fRMSE mid	4.7×10^{-3}	5.0×10^{-5}	5.2×10^{-3}
		fRMSE high	1.9×10^{-4}	7.1×10^{-6}	2.7×10^{-4}
2D diffusion-reaction	N/A	RMSE	6.1×10^{-2}	8.1×10^{-3}	1.9×10^{-1}
		nRMSE	8.4×10^{-1}	1.2×10^{-1}	$1.6 \times 10^{+0}$
		max error	1.9×10^{-1}	9.1×10^{-2}	5.0×10^{-1}
		cRMSE	3.9×10^{-2}	1.7×10^{-3}	1.3×10^{-1}
		bRMSE	7.8×10^{-2}	2.7×10^{-2}	2.2×10^{-1}
		fRMSE low	1.7×10^{-2}	8.2×10^{-4}	5.7×10^{-2}
		fRMSE mid	5.4×10^{-3}	7.7×10^{-4}	1.3×10^{-2}
		fRMSE high	6.8×10^{-4}	4.1×10^{-4}	1.5×10^{-3}
Shallow-water equation	N/A	RMSE	8.6×10^{-2}	4.5×10^{-3}	1.7×10^{-2}
		nRMSE	8.3×10^{-2}	4.4×10^{-3}	1.7×10^{-2}
		max error	4.4×10^{-1}	4.5×10^{-2}	1.3×10^{-3}
		cRMSE	1.3×10^{-2}	2.0×10^{-4}	1.7×10^{-2}
		bRMSE	4.2×10^{-3}	1.4×10^{-3}	1.5×10^{-1}
		fRMSE low	2.0×10^{-2}	2.6×10^{-4}	5.9×10^{-3}
		fRMSE mid	7.0×10^{-3}	3.1×10^{-4}	1.9×10^{-3}
		fRMSE high	8.6×10^{-4}	2.5×10^{-4}	6.0×10^{-4}

2D diffusion-reaction scenario is evaluated at $128^2 \times 20$ discrete points. Additionally, autoregression is not required and therefore, it leads to significantly faster computation time relative to FNO and U-Net.

As the case for 3D data, we also performed a similar experiment whose results are summarized in Table 16. The used system information is listed in Table 15. Because of the severe memory usage, the resolution was reduced to 64^3, though we provided a data with resolution 128^3 in our official dataset. Note that the training and inference time are shorter than the 2D cases in Table 14. This is because the number of time-step and sample numbers are less than the 2D cases to reduce dataset size.

G Resolution Sensitivity of Inference Time

Figure 10 plots the resolution dependence of the inference time of classical simulation and ML methods for 2D/3D compressible Navier-Stokes equations cases. To calculate the inference times, we used the same hardware resources to be a "fair" comparison as listed in Table 15.

The figure clearly shows that the ML inference time is nearly 3-order of magnitude smaller than that of the classical simulations. Concerning the resolution dependence, both of the ML models show a similar dependence to the inviscid classical simulation method. Importantly, the inference time of ML models is in general independent of the diffusion coefficient, such as viscosity. On the other hand, the classical simulation methods increase their computation time with diffusion

Table 5: Summary of the baseline models' performance for different evaluation metrics: RMSE, normalised RMSE (nRMSE), RMSE from conserved value (cRMSE), maximum error, RMSE at the boundaries (bRMSE), RMSE in Fourier space at low (fRMSE low), medium (fRMSE mid), and high frequency (fRMSE high) ranges applied to the advection equation with different parameter values.

			Baseline model		
PDE	Parameter	Metric	U-Net	FNO	PINN
Advection	$\beta = 0.1$	RMSE	3.8×10^{-2}	4.9×10^{-3}	7.8×10^{-1}
		nRMSE	6.0×10^{-2}	9.3×10^{-3}	9.1×10^{-1}
		max error	4.9×10^{-1}	1.4×10^{-1}	$1.5 \times 10^{+0}$
		cRMSE	1.5×10^{-2}	5.0×10^{-4}	5.5×10^{-3}
		bRMSE	6.4×10^{-2}	4.3×10^{-3}	6.8×10^{-1}
		fRMSE low	1.2×10^{-2}	4.1×10^{-4}	1.8×10^{-1}
		fRMSE mid	5.6×10^{-3}	4.4×10^{-4}	4.9×10^{-4}
		fRMSE high	8.6×10^{-4}	2.9×10^{-4}	6.1×10^{-6}
	$\beta = 0.4$	RMSE	3.6×10^{-1}	5.9×10^{-3}	9.2×10^{-1}
		nRMSE	6.7×10^{-1}	1.1×10^{-2}	$1.1 \times 10^{+0}$
		max error	$1.7 \times 10^{+0}$	2.0×10^{-1}	$1.7 \times 10^{+0}$
		cRMSE	2.6×10^{-1}	4.6×10^{-4}	1.9×10^{-3}
		bRMSE	3.7×10^{-1}	5.5×10^{-3}	7.7×10^{-1}
		fRMSE low	1.3×10^{-1}	4.4×10^{-4}	2.1×10^{-1}
		fRMSE mid	2.3×10^{-2}	4.7×10^{-4}	3.4×10^{-3}
		fRMSE high	2.3×10^{-3}	3.4×10^{-4}	9.8×10^{-6}
	$\beta = 1.0$	RMSE	1.2×10^{-2}	3.5×10^{-3}	4.0×10^{-1}
		nRMSE	2.0×10^{-2}	5.9×10^{-3}	4.7×10^{-1}
		max error	1.7×10^{-1}	8.5×10^{-2}	7.6×10^{-1}
		cRMSE	6.6×10^{-3}	1.8×10^{-4}	6.0×10^{-3}
		bRMSE	3.0×10^{-2}	2.6×10^{-3}	3.0×10^{-1}
		fRMSE low	3.8×10^{-3}	1.7×10^{-4}	9.7×10^{-2}
		fRMSE mid	1.5×10^{-3}	2.1×10^{-4}	1.2×10^{-3}
		fRMSE high	4.3×10^{-4}	2.2×10^{-4}	2.2×10^{-5}
	$\beta = 4.0$	RMSE	1.6×10^{-2}	5.8×10^{-3}	6.6×10^{-1}
		nRMSE	2.6×10^{-2}	1.0×10^{-2}	7.7×10^{-1}
		max error	1.4×10^{-1}	1.1×10^{-1}	$1.0 \times 10^{+0}$
		cRMSE	8.1×10^{-3}	3.9×10^{-4}	2.0×10^{-2}
		bRMSE	3.0×10^{-2}	5.1×10^{-3}	5.5×10^{-1}
		fRMSE low	4.6×10^{-3}	4.9×10^{-4}	1.5×10^{-1}
		fRMSE mid	1.8×10^{-3}	5.7×10^{-4}	3.4×10^{-4}
		fRMSE high	4.7×10^{-4}	2.9×10^{-4}	1.5×10^{-5}

coefficient because of the stability condition, known as Courant-Friedrich-Lewy (CFL) condition, $\Delta t \propto \Delta x^2/\eta$ in the case of the explicit method. Here $\Delta x, \Delta t$ are time-step size and mesh size, respectively, and η is the diffusion coefficient. This is much severer restriction than the inviscid case whose CFL condition is $\Delta t \propto \Delta x$. Hence, we can conclude that ML methods could even be suitable for solving for the problem with including strong-diffusive regime.

H Error Comparison with PDE Solver

To further assess the benefit of the trained baseline models, we generated the 2D diffusion-reaction data using a PDE solver with higher resolution (512×512), and downsampled them to lower resolution (128×128). These downsampled data were assumed as the ground truth (low discretization error) and then were used to train the baseline models. The trained baseline model predictions were

Table 6: Summary of the baseline models' performance for different evaluation metrics: RMSE, normalised RMSE (nRMSE), RMSE from conserved value (cRMSE), maximum error, RMSE at the boundaries (bRMSE), RMSE in Fourier space at low (fRMSE low), medium (fRMSE mid), and high frequency (fRMSE high) ranges applied to the Burgers' equation with different parameter values.

			Baseline model		
PDE	Parameter	Metric	U-Net	FNO	PINN
Burgers'	$\nu = 0.001$	RMSE	1.1×10^{-1}	1.3×10^{-2}	5.3×10^{-1}
		nRMSE	3.4×10^{-1}	4.2×10^{-2}	9.6×10^{-1}
		max error	5.7×10^{-1}	2.8×10^{-1}	8.2×10^{-1}
		cRMSE	5.9×10^{-2}	8.5×10^{-4}	5.1×10^{-1}
		bRMSE	1.0×10^{-1}	9.3×10^{-3}	5.2×10^{-1}
		fRMSE low	4.1×10^{-2}	8.7×10^{-4}	1.6×10^{-1}
		fRMSE mid	1.1×10^{-2}	1.2×10^{-3}	1.3×10^{-2}
		fRMSE high	1.5×10^{-3}	7.7×10^{-4}	4.7×10^{-4}
	$\nu = 0.01$	RMSE	9.7×10^{-2}	6.4×10^{-3}	5.3×10^{-1}
		nRMSE	3.0×10^{-1}	2.0×10^{-2}	9.5×10^{-1}
		max error	5.4×10^{-1}	1.5×10^{-1}	7.5×10^{-1}
		cRMSE	4.0×10^{-2}	7.2×10^{-4}	4.7×10^{-1}
		bRMSE	9.7×10^{-2}	7.6×10^{-3}	4.8×10^{-1}
		fRMSE low	3.5×10^{-2}	7.8×10^{-4}	1.8×10^{-1}
		fRMSE mid	1.0×10^{-2}	9.6×10^{-4}	2.3×10^{-2}
		fRMSE high	9.6×10^{-4}	5.2×10^{-4}	1.2×10^{-3}
	$\nu = 0.1$	RMSE	7.5×10^{-2}	1.4×10^{-3}	4.9×10^{-1}
		nRMSE	2.8×10^{-1}	4.5×10^{-3}	8.8×10^{-1}
		max error	4.6×10^{-1}	3.0×10^{-2}	6.6×10^{-1}
		cRMSE	3.0×10^{-2}	4.5×10^{-4}	4.7×10^{-1}
		bRMSE	1.0×10^{-1}	2.5×10^{-3}	3.4×10^{-1}
		fRMSE low	2.9×10^{-2}	4.2×10^{-4}	1.5×10^{-1}
		fRMSE mid	5.6×10^{-3}	3.1×10^{-4}	1.0×10^{-2}
		fRMSE high	8.4×10^{-4}	5.4×10^{-5}	5.1×10^{-4}
	$\nu = 1.0$	RMSE	6.0×10^{-2}	8.1×10^{-4}	5.4×10^{-1}
		nRMSE	3.6×10^{-1}	3.1×10^{-3}	9.9×10^{-1}
		max error	3.9×10^{-1}	5.9×10^{-3}	7.1×10^{-1}
		cRMSE	6.4×10^{-2}	2.4×10^{-4}	5.3×10^{-1}
		bRMSE	6.6×10^{-2}	8.6×10^{-4}	6.1×10^{-1}
		fRMSE low	2.5×10^{-2}	3.2×10^{-4}	1.5×10^{-1}
		fRMSE mid	3.0×10^{-3}	2.4×10^{-5}	4.9×10^{-3}
		fRMSE high	5.9×10^{-4}	4.9×10^{-6}	2.6×10^{-4}

compared against data that were generated using the same PDE solver but with coarser resolution (higher discretization error). The error comparison is summarized in Table 17. We observed that generating the data with lower resolution already accumulates high discretization error, relative to the baseline model prediction error. However, further sensitivity analysis with regards to different resolutions is required in future works to determine if the resolution is fine enough to be assumed as the ground truth.

I Visualization of Model Predictions

In this section, we present visualizations of the baseline model predictions, compared against the generated datasets for the diffusion-sorption equation (Figure 11), 2D diffusion-reaction equation (Figure 12, Figure 13, and Figure 14), the shallow-water equation (Figure 15, Figure 16, and Fig-

Table 7: Summary of the baseline models' performance for different evaluation metrics: RMSE, normalised RMSE (nRMSE), RMSE from conserved value (cRMSE), maximum error, RMSE at the boundaries (bRMSE), RMSE in Fourier space at low (fRMSE low), medium (fRMSE mid), and high frequency (fRMSE high) ranges applied to the Darcy flow equation with different parameter values.

			Baseline model	
PDE	Parameter	Metric	U-Net	FNO
DarcyFlow	$\beta = 0.01$	RMSE	4.0×10^{-3}	8.0×10^{-3}
		nRMSE	$1.1 \times 10^{+0}$	$2.5 \times 10^{+0}$
		max error	6.8×10^{-2}	1.5×10^{-1}
		cRMSE	5.8×10^{-3}	1.3×10^{-2}
		bRMSE	6.3×10^{-4}	4.7×10^{-3}
		fRMSE low	2.5×10^{-3}	5.2×10^{-3}
		fRMSE mid	1.3×10^{-4}	1.5×10^{-4}
		fRMSE high	2.1×10^{-5}	1.6×10^{-5}
	$\beta = 0.1$	RMSE	4.8×10^{-3}	6.2×10^{-3}
		nRMSE	1.8×10^{-1}	2.2×10^{-1}
		max error	7.0×10^{-2}	8.9×10^{-2}
		cRMSE	6.0×10^{-3}	7.7×10^{-3}
		bRMSE	8.6×10^{-4}	5.0×10^{-3}
		fRMSE low	2.6×10^{-3}	3.6×10^{-3}
		fRMSE mid	1.9×10^{-4}	2.6×10^{-4}
		fRMSE high	4.4×10^{-5}	4.5×10^{-5}
	$\beta = 1.0$	RMSE	6.4×10^{-3}	1.2×10^{-2}
		nRMSE	3.3×10^{-2}	6.4×10^{-2}
		max error	9.0×10^{-2}	1.1×10^{-1}
		cRMSE	6.0×10^{-3}	1.1×10^{-2}
		bRMSE	3.5×10^{-3}	5.5×10^{-3}
		fRMSE low	3.0×10^{-3}	5.2×10^{-3}
		fRMSE mid	3.4×10^{-4}	5.1×10^{-4}
		fRMSE high	1.3×10^{-4}	1.5×10^{-4}
	$\beta = 10.0$	RMSE	1.4×10^{-2}	2.1×10^{-2}
		nRMSE	8.2×10^{-3}	1.2×10^{-2}
		max error	2.4×10^{-1}	3.2×10^{-1}
		cRMSE	9.9×10^{-3}	1.5×10^{-2}
		bRMSE	9.4×10^{-3}	1.6×10^{-2}
		fRMSE low	5.8×10^{-3}	8.3×10^{-3}
		fRMSE mid	9.8×10^{-4}	1.3×10^{-3}
		fRMSE high	3.6×10^{-4}	5.7×10^{-4}
	$\beta = 100.0$	RMSE	7.3×10^{-2}	1.1×10^{-1}
		nRMSE	4.4×10^{-3}	6.4×10^{-3}
		max error	$1.7 \times 10^{+0}$	$2.1 \times 10^{+0}$
		cRMSE	5.1×10^{-2}	8.9×10^{-2}
		bRMSE	4.6×10^{-2}	7.9×10^{-2}
		fRMSE low	2.9×10^{-2}	4.6×10^{-2}
		fRMSE mid	5.3×10^{-3}	7.6×10^{-3}
		fRMSE high	2.5×10^{-3}	3.6×10^{-3}

ure 17), 1D Advection equation Figure 18, 1D Burgers equation Figure 19, 1D Reaction-Diffusion equation Figure 20, 1D compressible NS equations Figure 21, 2D Darcy flow Figure 22, and 2D compressible NS equations Figure 23.

Table 8: Summary of the baseline models' performance for different evaluation metrics: RMSE, normalised RMSE (nRMSE), RMSE from conserved value (cRMSE), maximum error, RMSE at the boundaries (bRMSE), RMSE in Fourier space at low (fRMSE low), medium (fRMSE mid), and high frequency (fRMSE high) ranges applied to the 1d diffusion-reaction equation with different parameter values.

			Baseline model		
PDE	Parameter	Metric	U-Net	FNO	PINN
ReacDiff	$\nu = 0.5, \rho = 1.0$	RMSE	3.1×10^{-3}	6.3×10^{-4}	4.5×10^{-2}
		nRMSE	6.0×10^{-3}	1.4×10^{-3}	8.0×10^{-2}
		max error	1.8×10^{-2}	8.7×10^{-3}	7.6×10^{-2}
		cRMSE	2.5×10^{-3}	1.3×10^{-3}	4.3×10^{-2}
		bRMSE	3.7×10^{-3}	6.7×10^{-4}	7.5×10^{-2}
		fRMSE low	1.1×10^{-3}	4.1×10^{-4}	1.4×10^{-2}
		fRMSE mid	1.8×10^{-4}	9.1×10^{-6}	2.4×10^{-4}
		fRMSE high	1.8×10^{-5}	1.7×10^{-6}	3.7×10^{-6}
	$\nu = 0.5, \rho = 10.0$	RMSE	6.2×10^{-8}	$0.0 \times 10^{+0}$	1.4×10^{-2}
		nRMSE	6.5×10^{-8}	$0.0 \times 10^{+0}$	1.4×10^{-2}
		max error	6.2×10^{-8}	$0.0 \times 10^{+0}$	2.6×10^{-2}
		cRMSE	6.2×10^{-8}	$0.0 \times 10^{+0}$	6.2×10^{-3}
		bRMSE	6.2×10^{-8}	$0.0 \times 10^{+0}$	2.3×10^{-2}
		fRMSE low	1.6×10^{-8}	$0.0 \times 10^{+0}$	4.3×10^{-3}
		fRMSE mid	$0.0 \times 10^{+0}$	$0.0 \times 10^{+0}$	2.5×10^{-4}
		fRMSE high	$0.0 \times 10^{+0}$	$0.0 \times 10^{+0}$	2.9×10^{-6}
	$\nu = 2.0, \rho = 1.0$	RMSE	2.3×10^{-3}	2.9×10^{-4}	3.9×10^{-1}
		nRMSE	4.5×10^{-3}	7.0×10^{-4}	7.3×10^{-1}
		max error	2.0×10^{-2}	4.2×10^{-3}	3.9×10^{-1}
		cRMSE	1.9×10^{-3}	6.4×10^{-4}	3.9×10^{-1}
		bRMSE	1.8×10^{-3}	4.1×10^{-4}	3.9×10^{-1}
		fRMSE low	7.7×10^{-4}	1.9×10^{-4}	9.7×10^{-2}
		fRMSE mid	1.7×10^{-4}	9.2×10^{-6}	6.2×10^{-5}
		fRMSE high	2.6×10^{-5}	1.8×10^{-6}	3.4×10^{-6}
	$\nu = 2.0, \rho = 10.0$	RMSE	3.1×10^{-8}	6.2×10^{-8}	3.2×10^{-2}
		nRMSE	3.2×10^{-8}	6.5×10^{-8}	3.3×10^{-2}
		max error	3.1×10^{-8}	6.2×10^{-8}	3.2×10^{-2}
		cRMSE	3.1×10^{-8}	6.2×10^{-8}	3.2×10^{-2}
		bRMSE	3.1×10^{-8}	6.2×10^{-8}	3.1×10^{-2}
		fRMSE low	7.8×10^{-9}	1.6×10^{-8}	8.0×10^{-3}
		fRMSE mid	$0.0 \times 10^{+0}$	$0.0 \times 10^{+0}$	6.4×10^{-6}
		fRMSE high	$0.0 \times 10^{+0}$	$0.0 \times 10^{+0}$	2.7×10^{-7}

J VISUALIZATION OF INITIAL CONDITIONS

In this section, we provide a collection of initial condition visualizations for each problem. Figure 24 shows different radius of the initial perturbation used as the initial condition for five different samples of the 2D shallow-water equation data. Figure 25 shows different random uniform initial condition used for five different samples of the 1D diffusion-sorption equation data. Figure 26 shows different random noise used as the initial condition for five different samples of the 2D diffusion-reaction equation data.

In Figure 27, we plotted the several samples of the initial condition for 1D Advection and Burgers equations. Figure 28 is also the similar plot of the initial condition for 1D Diffusion-Reaction equation. Note that in this case the value of the scalar function is limited between 0 to 1 because of

Table 9: Summary of the baseline models' performance for different evaluation metrics: RMSE, normalised RMSE (nRMSE), RMSE from conserved value (cRMSE), maximum error, RMSE at the boundaries (bRMSE), RMSE in Fourier space at low (fRMSE low), medium (fRMSE mid), and high frequency (fRMSE high) ranges applied to the 1d compressible Navier-Stokes equation with different parameter values.

			Baseline model	
PDE	Parameter	Metric	U-Net	FNO
1DCFD	$\eta = \zeta = 0.01$ Rand periodic	RMSE	9.9×10^{-1}	2.7×10^{-1}
		nRMSE	3.6×10^{-1}	9.5×10^{-2}
		max error	$7.8 \times 10^{+0}$	$4.1 \times 10^{+0}$
		cRMSE	3.6×10^{-1}	5.0×10^{-2}
		bRMSE	$1.0 \times 10^{+0}$	2.2×10^{-1}
		fRMSE low	3.6×10^{-1}	7.3×10^{-2}
		fRMSE mid	1.2×10^{-1}	5.5×10^{-2}
		fRMSE high	9.2×10^{-3}	3.7×10^{-3}
	$\eta = \zeta = 0.1$ Rand periodic	RMSE	6.6×10^{-1}	9.3×10^{-2}
		nRMSE	7.2×10^{-1}	6.8×10^{-2}
		max error	$5.3 \times 10^{+0}$	$1.5 \times 10^{+0}$
		cRMSE	3.5×10^{-1}	2.7×10^{-2}
		bRMSE	6.8×10^{-1}	7.6×10^{-2}
		fRMSE low	2.5×10^{-1}	2.8×10^{-2}
		fRMSE mid	5.7×10^{-2}	1.3×10^{-2}
		fRMSE high	7.7×10^{-3}	2.0×10^{-3}
	inviscid Rand periodic	RMSE	$1.7 \times 10^{+1}$	4.7×10^{-1}
		nRMSE	$1.1 \times 10^{+0}$	1.2×10^{-1}
		max error	$2.0 \times 10^{+1}$	$7.1 \times 10^{+0}$
		cRMSE	$1.7 \times 10^{+1}$	6.7×10^{-2}
		bRMSE	$1.6 \times 10^{+1}$	3.5×10^{-1}
		fRMSE low	5.3×10^{-1}	$4.5 \times 10^{+0}$
		fRMSE mid	1.9×10^{-1}	1.6×10^{-1}
		fRMSE high	2.1×10^{-2}	2.6×10^{-3}
	inviscid Rand Outgoing	RMSE	$1.6 \times 10^{+0}$	2.6×10^{-1}
		nRMSE	$1.1 \times 10^{+1}$	$6.7 \times 10^{+0}$
		max error	$1.2 \times 10^{+1}$	$4.3 \times 10^{+0}$
		cRMSE	$1.5 \times 10^{+0}$	1.5×10^{-1}
		bRMSE	$1.8 \times 10^{+0}$	3.6×10^{-1}
		fRMSE low	6.8×10^{-1}	9.0×10^{-2}
		fRMSE mid	1.2×10^{-1}	4.5×10^{-2}
		fRMSE high	1.6×10^{-2}	6.7×10^{-3}
	inviscid Shock Outgoing	RMSE	4.1×10^{-1}	1.6×10^{-1}
		nRMSE	1.7×10^{-1}	4.7×10^{-2}
		max error	$6.6 \times 10^{+0}$	$3.8 \times 10^{+0}$
		cRMSE	2.1×10^{-1}	5.3×10^{-2}
		bRMSE	5.6×10^{-1}	2.4×10^{-1}
		fRMSE low	1.4×10^{-1}	3.7×10^{-2}
		fRMSE mid	5.3×10^{-2}	2.6×10^{-2}
		fRMSE high	1.1×10^{-2}	6.7×10^{-3}

the form of the source term. Finally, we provided several samples of the 1D and 2D CFD cases in Figure 29 and Figure 30.

Table 10: Summary of the baseline models' performance for different evaluation metrics: RMSE, normalised RMSE (nRMSE), RMSE from conserved value (cRMSE), maximum error, RMSE at the boundaries (bRMSE), RMSE in Fourier space at low (fRMSE low), medium (fRMSE mid), and high frequency (fRMSE high) ranges applied to the 2d compressible Navier-Stokes equation with different parameter values (first part).

PDE	Parameter	Metric	Baseline model	
			U-Net	FNO
2DCFD	$M = 0.1$, inviscid Rand periodic	RMSE	4.0×10^{-1}	2.6×10^{-1}
		nRMSE	6.6×10^{-1}	2.8×10^{-1}
		max error	$5.1 \times 10^{+0}$	$4.2 \times 10^{+0}$
		cRMSE	1.5×10^{-1}	1.6×10^{-2}
		bRMSE	4.3×10^{-1}	2.6×10^{-1}
		fRMSE low	1.1×10^{-1}	4.5×10^{-2}
		fRMSE mid	4.8×10^{-2}	4.4×10^{-2}
		fRMSE high	1.7×10^{-2}	1.6×10^{-2}
	$M = 0.1, \eta = \zeta = 0.01$ Rand periodic	RMSE	9.1×10^{-2}	2.3×10^{-2}
		nRMSE	7.1×10^{-1}	1.7×10^{-1}
		max error	$1.1 \times 10^{+0}$	4.0×10^{-1}
		cRMSE	3.6×10^{-2}	5.3×10^{-3}
		bRMSE	1.1×10^{-1}	2.2×10^{-2}
		fRMSE low	2.7×10^{-2}	5.7×10^{-3}
		fRMSE mid	8.2×10^{-3}	2.7×10^{-3}
		fRMSE high	2.6×10^{-3}	6.3×10^{-4}
	$M = 0.1, \eta = \zeta = 0.1$ Rand periodic	RMSE	4.7×10^{-2}	4.9×10^{-3}
		nRMSE	$5.1 \times 10^{+0}$	3.6×10^{-1}
		max error	6.7×10^{-1}	8.7×10^{-2}
		cRMSE	3.2×10^{-2}	3.2×10^{-3}
		bRMSE	6.6×10^{-2}	4.3×10^{-3}
		fRMSE low	1.3×10^{-2}	1.4×10^{-3}
		fRMSE mid	4.2×10^{-3}	4.3×10^{-4}
		fRMSE high	2.2×10^{-3}	1.4×10^{-4}
	$M = 1.0$, inviscid Rand periodic	RMSE	$1.5 \times 10^{+0}$	$1.4 \times 10^{+0}$
		nRMSE	4.7×10^{-1}	3.5×10^{-1}
		max error	$1.6 \times 10^{+1}$	$1.6 \times 10^{+1}$
		cRMSE	4.8×10^{-1}	1.6×10^{-1}
		bRMSE	$1.5 \times 10^{+0}$	$1.3 \times 10^{+0}$
		fRMSE low	4.8×10^{-1}	4.0×10^{-1}
		fRMSE mid	1.2×10^{-1}	1.2×10^{-1}
		fRMSE high	3.9×10^{-2}	3.9×10^{-2}

Table 11: Summary of the baseline models' performance for different evaluation metrics: RMSE, normalised RMSE (nRMSE), RMSE from conserved value (cRMSE), maximum error, RMSE at the boundaries (bRMSE), RMSE in Fourier space at low (fRMSE low), medium (fRMSE mid), and high frequency (fRMSE high) ranges applied to the 2d compressible Navier-Stokes equation with different parameter values (second part).

PDE	Parameter	Metric	Baseline model	
			U-Net	FNO
2DCFD	$M = 1.0, \eta = \zeta =$ 0.01 Rand periodic	RMSE	3.4×10^{-1}	1.2×10^{-1}
		nRMSE	3.6×10^{-1}	9.6×10^{-2}
		max error	$3.7 \times 10^{+0}$	$1.7 \times 10^{+0}$
		cRMSE	1.1×10^{-1}	1.8×10^{-2}
		bRMSE	3.6×10^{-1}	1.3×10^{-1}
		fRMSE low	1.1×10^{-1}	3.3×10^{-2}
		fRMSE mid	2.7×10^{-2}	1.5×10^{-2}
		fRMSE high	6.2×10^{-3}	3.6×10^{-3}
	$M = 1.0, \eta = \zeta = 0.1$ Rand periodic	RMSE	1.1×10^{-1}	1.5×10^{-2}
		nRMSE	9.2×10^{-1}	9.8×10^{-2}
		max error	$1.3 \times 10^{+0}$	2.4×10^{-1}
		cRMSE	4.8×10^{-2}	4.8×10^{-3}
		bRMSE	1.5×10^{-1}	1.7×10^{-2}
		fRMSE low	3.0×10^{-2}	3.2×10^{-3}
		fRMSE mid	1.3×10^{-2}	1.5×10^{-3}
		fRMSE high	4.3×10^{-3}	8.9×10^{-4}
	$M = 0.1$, inviscid Turb periodic	RMSE	3.3×10^{-1}	2.8×10^{-1}
		nRMSE	1.9×10^{-1}	1.6×10^{-1}
		max error	$2.2 \times 10^{+0}$	$1.8 \times 10^{+0}$
		cRMSE	1.5×10^{-2}	1.2×10^{-2}
		bRMSE	3.6×10^{-1}	2.8×10^{-1}
		fRMSE low	6.5×10^{-2}	5.0×10^{-2}
		fRMSE mid	3.2×10^{-2}	3.1×10^{-2}
		fRMSE high	8.5×10^{-3}	6.5×10^{-3}
	$M = 1.0$, inviscid Turb periodic	RMSE	9.5×10^{-2}	9.2×10^{-2}
		nRMSE	1.4×10^{-1}	1.3×10^{-1}
		max error	8.2×10^{-1}	7.9×10^{-1}
		cRMSE	6.5×10^{-3}	4.3×10^{-3}
		bRMSE	1.1×10^{-1}	9.7×10^{-1}
		fRMSE low	1.3×10^{-2}	1.1×10^{-2}
		fRMSE mid	1.2×10^{-2}	1.2×10^{-2}
		fRMSE high	5.2×10^{-3}	5.2×10^{-3}

Table 12: Summary of the baseline models' performance for different evaluation metrics: RMSE, normalised RMSE (nRMSE), RMSE from conserved value (cRMSE), maximum error, RMSE at the boundaries (bRMSE), RMSE in Fourier space at low (fRMSE low), medium (fRMSE mid), and high frequency (fRMSE high) ranges applied to the 3d compressible Navier-Stokes equation with different parameter values.

			Baseline model	
PDE	Parameter	Metric	U-Net	FNO
3DCFD	$M = 1.0$ inviscid Rand periodic	RMSE	$2.2 \times 10^{+0}$	6.0×10^{-1}
		nRMSE	$1.0 \times 10^{+0}$	3.7×10^{-1}
		max error	$9.0 \times 10^{+0}$	$3.6 \times 10^{+0}$
		cRMSE	$2.3 \times 10^{+0}$	8.1×10^{-2}
		bRMSE	$2.1 \times 10^{+0}$	6.0×10^{-1}
		fRMSE low	7.3×10^{-1}	1.1×10^{-1}
		fRMSE mid	7.6×10^{-2}	4.4×10^{-2}
		fRMSE high	2.3×10^{-2}	9.3×10^{-3}
	$M = 1.0$ inviscid Turb periodic	RMSE	8.1×10^{-2}	8.2×10^{-2}
		nRMSE	2.3×10^{-1}	2.4×10^{-1}
		max error	5.0×10^{-1}	4.5×10^{-1}
		cRMSE	7.3×10^{-3}	2.8×10^{-3}
		bRMSE	9.9×10^{-2}	8.6×10^{-2}
		fRMSE low	1.1×10^{-2}	7.2×10^{-3}
		fRMSE mid	8.0×10^{-3}	9.4×10^{-3}
		fRMSE high	1.7×10^{-3}	4.5×10^{-3}

Table 13: System configuration 1

CPU	$2 \times$ AMD EPYC 7742
GPU	$1 \times$ NVIDIA Volta V100
Software	PyTorch@1.11, CUDA@11.3

Table 14: Comparison of computation time between the PDE solver used to generate a single data sample and single forward runs of FNO, U-Net, and PINN. Training time of the baseline models for one epoch are also presented in this table. The unit used for the time is seconds.

PDE	Resolution	Model	Training time ($\frac{\text{s}}{\text{epoch}}$)	Epochs	Inference time (s)
Diffusion-sorption	$1\,024^1$	PDE solver	–	–	59.83
		FNO	97.52	500	0.32
		U-Net	96.75	500	0.32
		PINN	0.011	15 000	0.0027
2D diffusion-reaction	128^2	PDE solver	–	–	2.21
		FNO	108.28	500	0.40
		U-Net	83.19	500	0.61
		PINN	0.022	100	0.0077
Shallow-water equation	128^2	PDE solver	–	–	0.62
		FNO	105.16	500	0.37
		U-Net	83.32	500	0.56
		PINN	0.041	15 000	0.00673

Table 15: System configuration 2

GPU	$1 \times$ NVIDIA GeForce RTX 3090
Software (ML methods)	PyTorch@1.11, CUDA@11.3
Software (simulations)	JAX@0.2.26, CUDA@11.3

Table 16: Comparison of computation time between the PDE solver used to generate a single data sample and single forward runs of FNO, U-Net. Training time of the baseline models for one epoch are also presented in this table. The unit used for the time is seconds.

PDE	Resolution	Model	Training time ($\frac{\mathrm{s}}{\mathrm{epoch}}$)	Epochs	Inference time (s)
3D CFD	64^3	PDE solver	–	–	60.07
		FNO	24.77	500	0.14
		U-Net	62.22	500	0.27

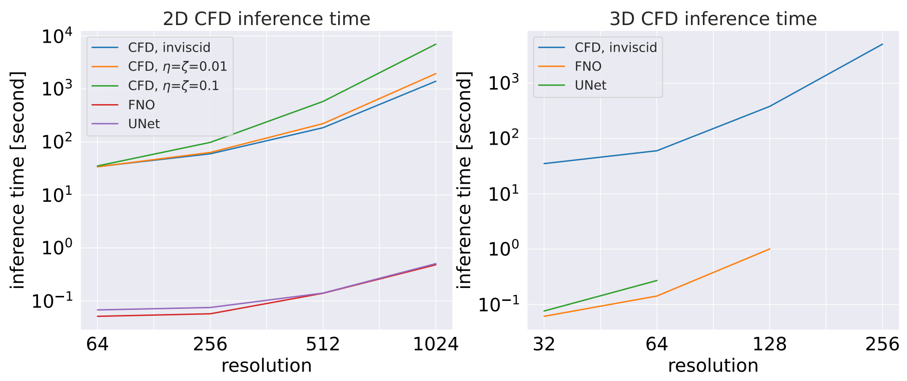


Figure 10: Plots inference time for 2D/3D CFD cases.

Table 17: Error comparison between of U-Net, FNO, and PINN prediction, as well as low-resolution PDE solver data, against the high-resolution PDE solver data (assumed as the ground truth) for the 2D diffusion-reaction scenario.

Error metric	U-Net	FNO	PINN	low-res PDE solver
RMSE	6.1×10^{-2}	8.1×10^{-3}	1.9×10^{-1}	1.8×10^{-1}
nRMSE	8.4×10^{-1}	1.2×10^{-1}	$1.6 \times 10^{+0}$	$2.8 \times 10^{+0}$
max error	1.9×10^{-1}	9.1×10^{-2}	5.0×10^{-1}	8.9×10^{-1}
cRMSE	3.9×10^{-2}	1.7×10^{-3}	1.3×10^{-1}	4.9×10^{-2}
bRMSE	7.8×10^{-2}	2.7×10^{-2}	2.2×10^{-1}	2.1×10^{-1}
fRMSE low	1.7×10^{-2}	8.2×10^{-4}	5.7×10^{-2}	4.9×10^{-2}
fRMSE mid	5.4×10^{-3}	7.7×10^{-4}	1.3×10^{-2}	2.2×10^{-2}
fRMSE high	6.8×10^{-4}	4.1×10^{-4}	1.5×10^{-3}	3.4×10^{-3}

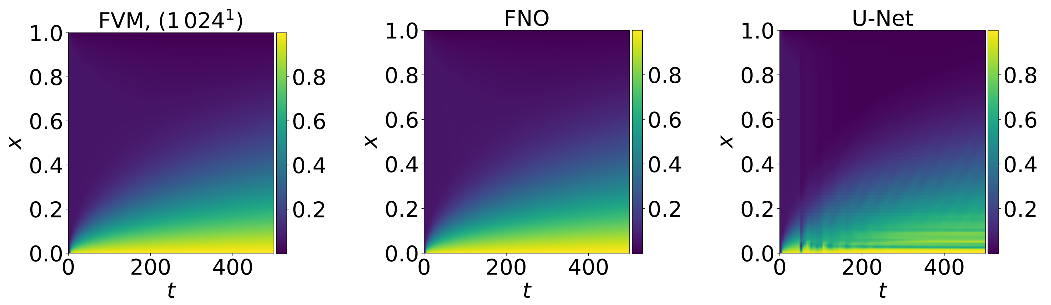


Figure 11: Visualization of the diffusion-sorption equation (a) data, (b) FNO prediction, and (c) U-Net prediction.

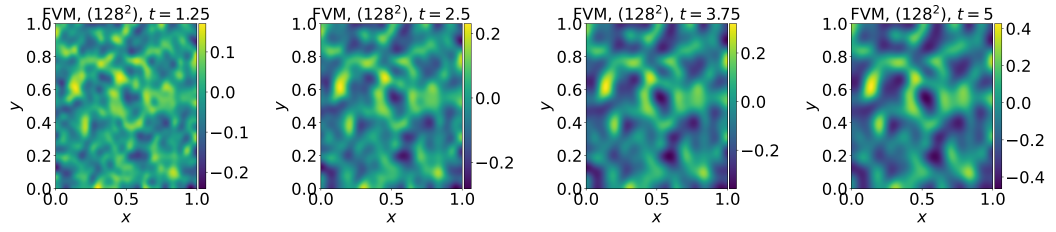


Figure 12: Visualization of the time evolution of the 2D diffusion-reaction equation data.

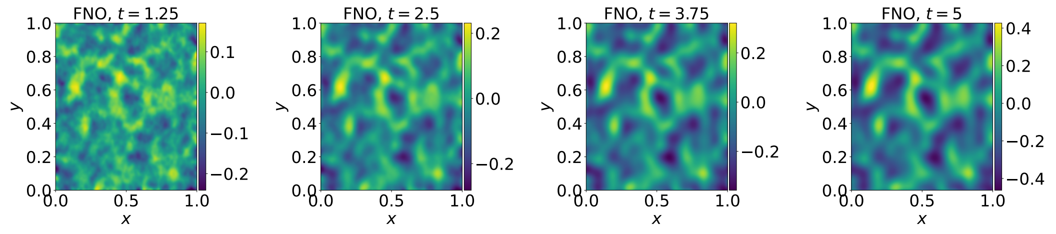


Figure 13: Visualization of the time evolution of the 2D diffusion-reaction equation predicted using FNO.

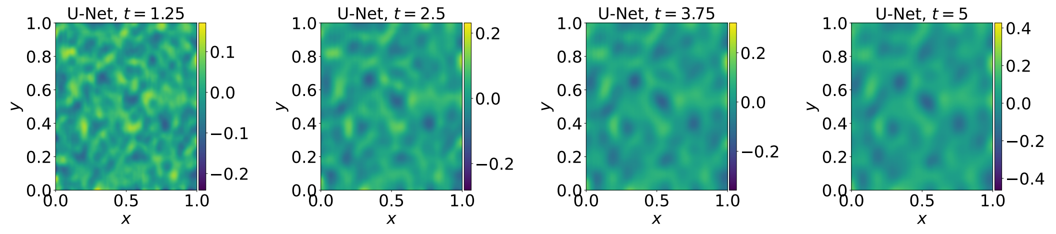


Figure 14: Visualization of the time evolution of the 2D diffusion-reaction equation predicted using U-Net.

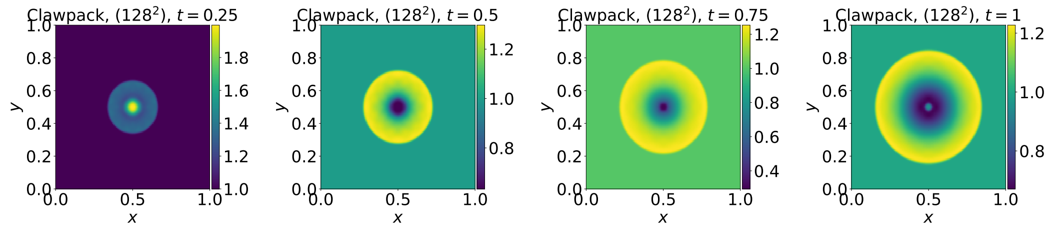


Figure 15: Visualization of the time evolution of the shallow water equation data.

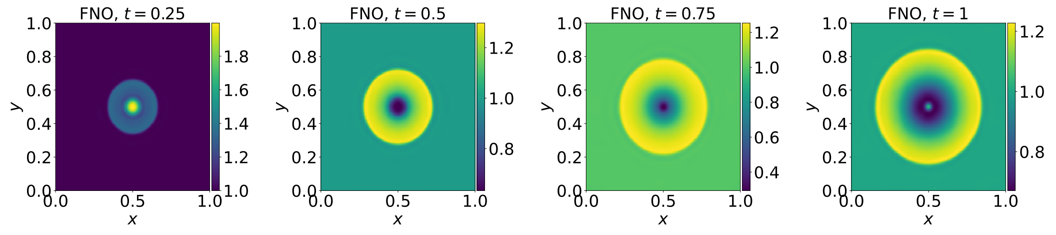


Figure 16: Visualization of the time evolution of the shallow water equation predicted using FNO.

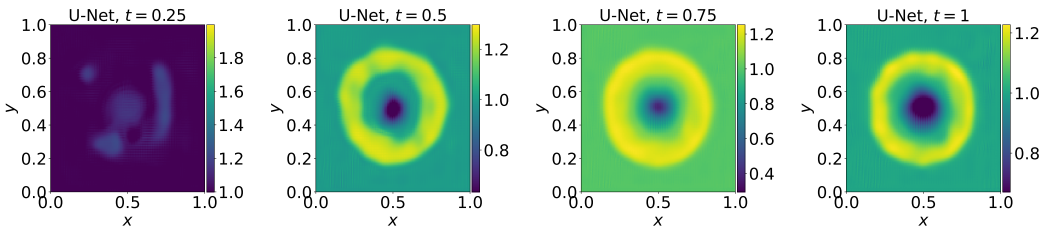


Figure 17: Visualization of the time evolution of the shallow water equation predicted using U-Net.

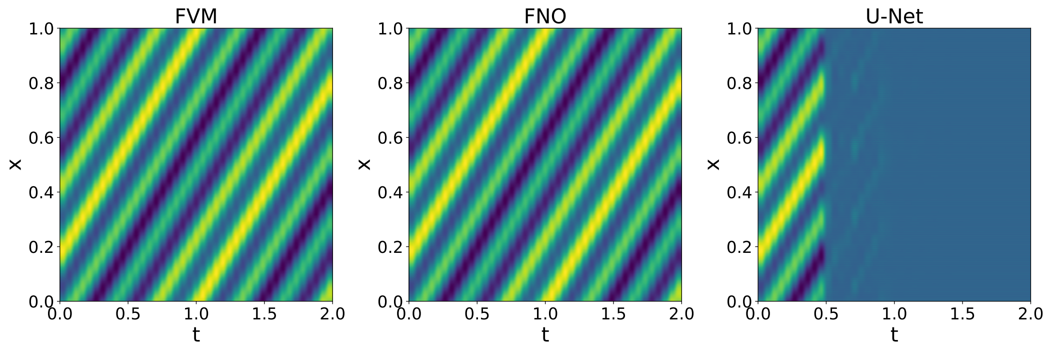


Figure 18: Plots of the predictions for 1D Advection equation.

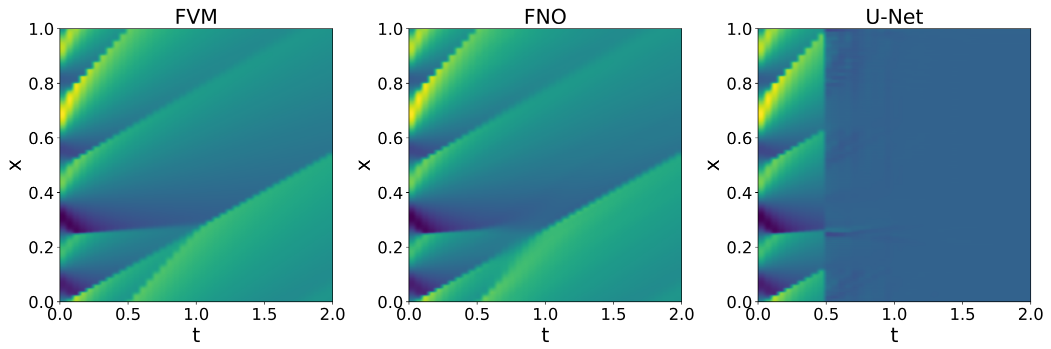


Figure 19: Plots of the predictions for 1D Burgers equation.

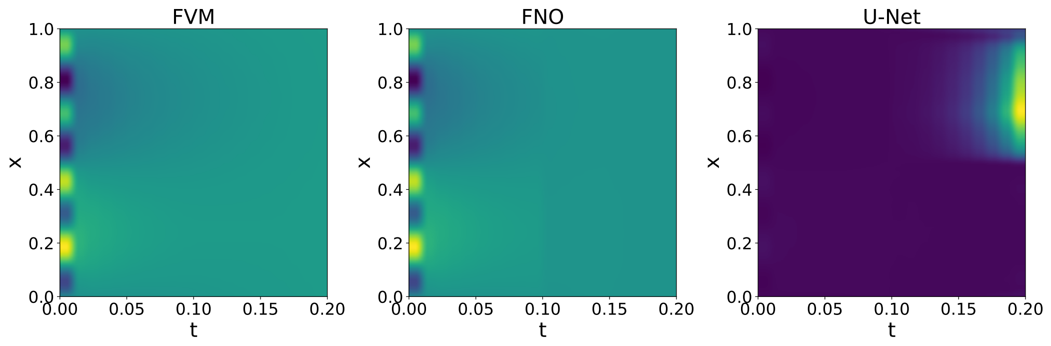


Figure 20: Plots of the predictions for 1D Reaction-Diffusion equation.

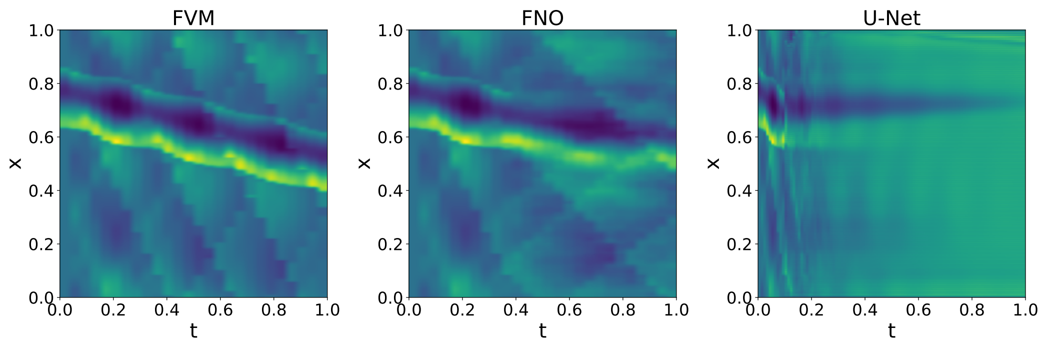


Figure 21: Plots of the predictions of density for 1D compressible NS equations.

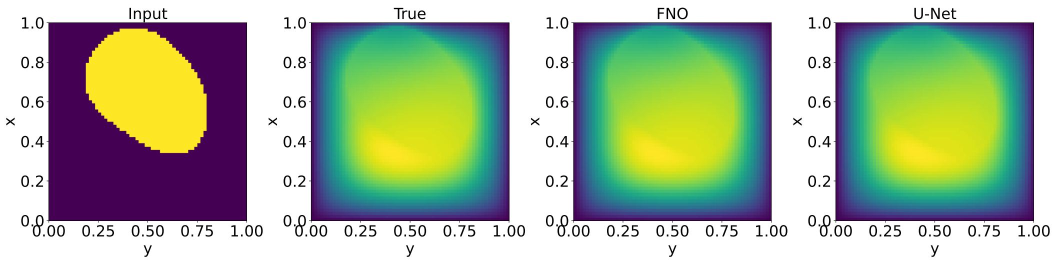


Figure 22: Plots of the predictions for 2D Darcy Flow.

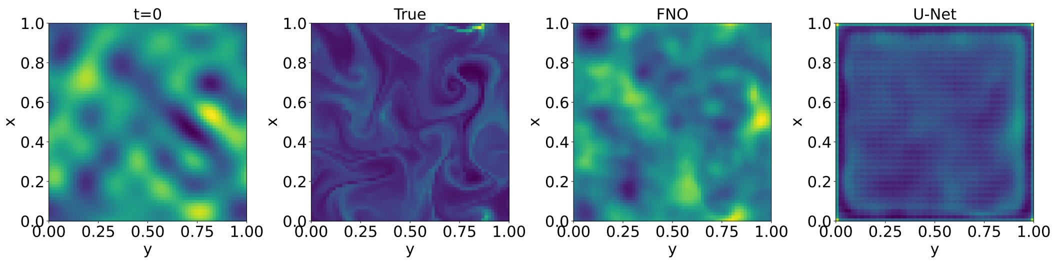


Figure 23: Plots of the predictions of the density for 2D compressible NS equations at the final time-step.

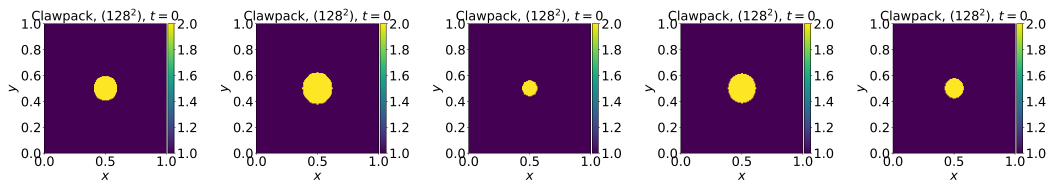


Figure 24: Visualization of the different radius of the initial perturbation used for the 2D shallow-water equations data.

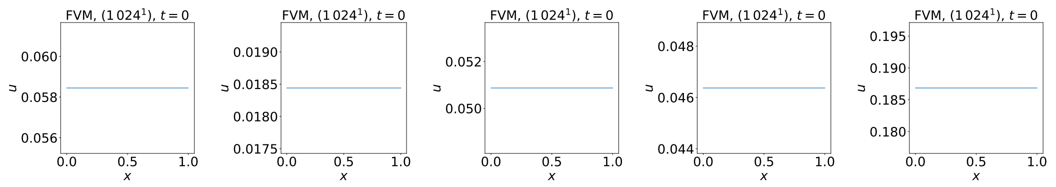


Figure 25: Visualization of the random uniform initial conditions used for the 1D diffusion-sorption equations data.

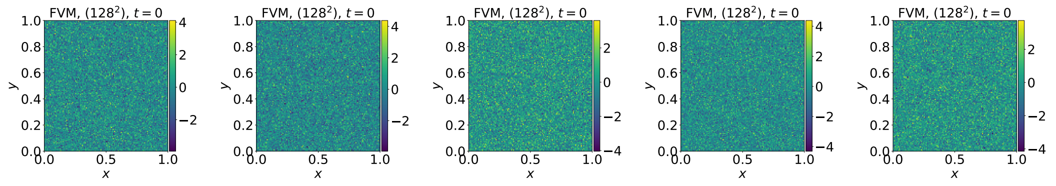


Figure 26: Visualization of the random initial conditions used for the 2D diffusion-reaction equations data.

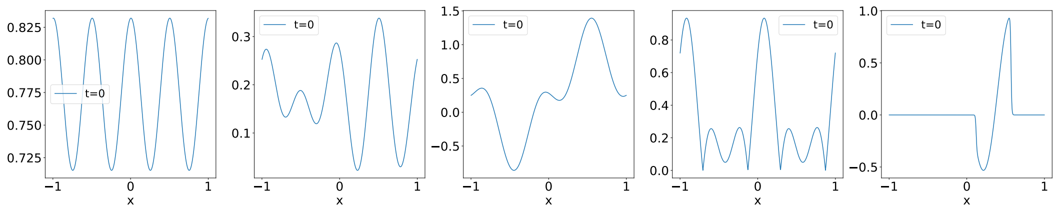


Figure 27: Visualization of the random initial conditions used for the 1D Advection/Burgers equations data.

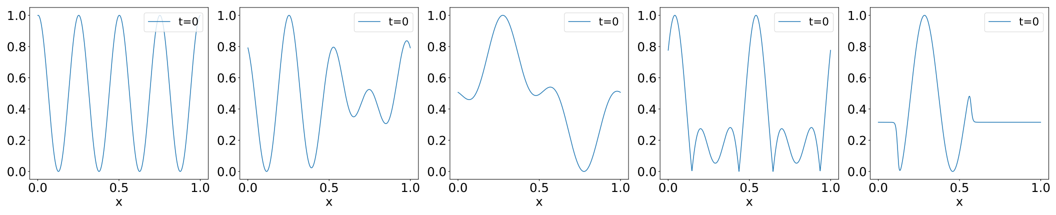


Figure 28: Visualization of the random initial conditions used for the 1D Reaction-Diffusion equations data.

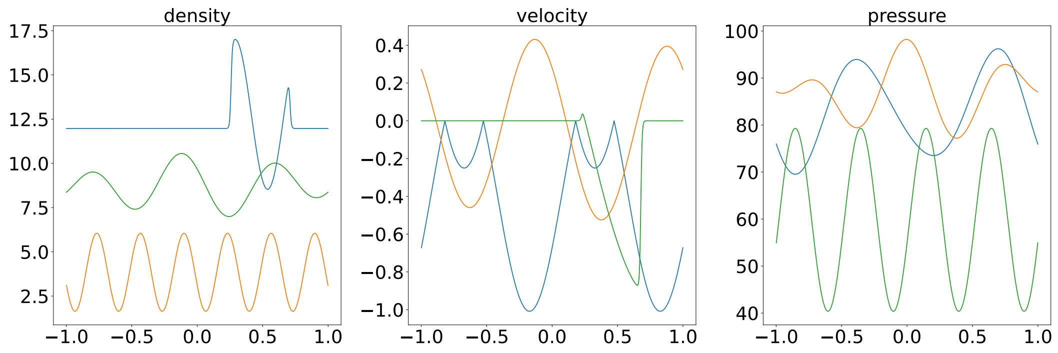


Figure 29: Visualization of the random initial conditions used for the 1D CFD data. The different colors mean the different samples.

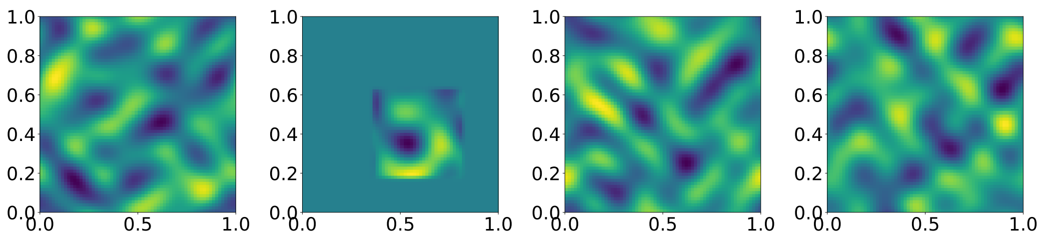


Figure 30: Visualization of the initial conditions of density used for the 2D CFD data.