Neural oscillators for generalizing parametric PDEs

Taniya Kapoor[1], Abhishek Chandra[2], Daniel M. Tartakovsky[3],
Hongrui Wang[1], Alfredo Núñez[1], Rolf Dollevoet[1]
[1] TU Delft, [2] TU Eindhoven, [3] Stanford University
t.kapoor@tudelft.nl

Abstract

Parametric partial differential equations (PDEs) are ubiquitous in various scientific and engineering fields, manifesting the behavior of systems under varying parameters. Predicting solutions over a parametric space is desirable but prohibitively *costly* and *challenging*. In addition, recent neural PDE solvers are usually *limited to interpolation* scenarios, where solutions are predicted for inputs within the support of the training set. This work proposes to utilize *neural oscillators* to extend predictions for parameters beyond the trained regime, effectively *extrapolating* the *parametric space*. The proposed methodology is validated on three parametric PDEs: linear advection, viscous burgers, and nonlinear heat. The results underscore the promising potential of neural oscillators in extrapolation scenarios for both linear and nonlinear parametric PDEs.

1 Introduction

Parametric partial differential equations (PDEs) are extensively used to model complex real-world problems in science and engineering. For instance, In aerospace engineering, aircraft designs are modeled by parametrizing airfoil shapes and operational conditions [1, 2]. In civil engineering and geosciences, parametric PDEs are often used to assess and optimize fundamental engineering structures like plates [3], beams [4] and characterize porous media flows [5], among other applications. Traditionally, these parametric PDEs are simulated through numerical methods. However, solving such problems for the *entire parametric space* $\mathcal{X}$ is *challenging* and *computationally costly*. Predicting solutions over the entire $\mathcal{X}$ is even more problematic for high-dimensional PDEs [6]. The challenge is not only limited to the numerical methods, but even the recent *neural PDE solvers* [7, 8] face challenges in solving the PDEs over $\mathcal{X}$ and finding the solution envelope $\mathcal{U}$.

Recently, *operator learning* frameworks [9, 10, 11, 12, 13] have been developed extensively focusing on learning a differential operator $\mathcal{L}$, which predicts $\mathcal{U}$ over the inputs $\mathcal{X}$. Although data-driven neural operators and neural PDE solvers, in general, often excel in interpolation tasks, making accurate predictions within the training domain, predictions *outside the training domain* are also of equal interest [14]. The extrapolated predictions are useful for situations where collecting data for every possible parameter value may not be feasible or practical, as presented in [15]. A deep operator network-based strategy is presented in [15] to balance bias and variance in the extrapolation region and incorporate governing PDEs or sparse data in the training to showcase its effectiveness on various parametric PDEs.

However, one could also formulate the problem of extrapolating the neural PDE solvers as a *sequential* deep learning problem [16, 17, 18]. Particularly, [16, 17] extrapolate the PDE solution over the *temporal* domain for operator networks and physics-informed architectures, respectively, using the recurrent neural network-based architectures. Similarly, the solution of the parametric PDEs $\mathcal{U}$ forms an envelope over the parametric space $\mathcal{X}$ and can be considered as a sequence varying on parameters. These parameters can also be considered as an additional dimension to the problem. Hence, the problem of extrapolating the parameter space could be tackled through similar strategies as presented

The Symbiosis of Deep Learning and Differential Equations workshop, NeurIPS 2023

in [16, 17, 19]. Consequently, we propose to employ recurrent neural network-based architecture to facilitate generalization in parametric space $\mathcal{X}$. Particularly, we employ *oscillator networks*, which are recurrent neural architectures that employ *ordinary differential equations* (ODEs) or *dynamic systems* to update the hidden states [20, 21, 22, 23, 24, 25]. By leveraging the *long-term memory* and the *universal approximation* properties of oscillators [26], we aim to investigate the potential of oscillator networks in mitigating the generalization challenge for parametric space.

The solution data corresponding to some parametric PDEs is required *at priori* to train the neural oscillators. This data can be acquired in several ways, including but not limited to physics-informed machine learning strategies, numerical techniques, and experiments or simulations. Subsequently, the oscillator network is trained and tested on *distinct* domains to infer the *viability* of the method. We demonstrate the efficacy of the proposed approach for three fundamental PDEs: linear advection, viscous Burgers and nonlinear heat. The rest of the paper is structured as follows: Section 2 presents the proposed methodology. Section 3 contains the three performed numerical experiments. Key findings resulting from this study are discussed and summarized in Section 4.

2 Method

We consider an abstract form of PDE, $\mathcal{L}[u(x,t;\lambda)] = 0$, where, $(x,t) \in D \times T$. Here, $D \subset \mathbb{R}$ and $T \subset \mathbb{R}$ represent the spatial and temporal domain, respectively. The parameter $\lambda \in \mathcal{X} := [\lambda_{\min}, \lambda_{\max}] \subset \mathcal{R}$. $\mathcal{L}$ is the differential operator and u is the quantity of interest. The *objective* is to predict the solutions of the parametric PDEs in the extrapolated parametric space. Precisely, training is performed in the parametric space $\mathcal{X}_1 := [\lambda_{\min}, \lambda^{'})$ and testing of the algorithm is performed in the extrapolation domain $\mathcal{X}_2 := [\lambda^{'}, \lambda_{\max}]$, where $\lambda_{\min} \leq \lambda^{'} \leq \lambda_{\max}$, and $\mathcal{X}_1, \mathcal{X}_2 \subset \mathcal{X}$. For instance, the dataset consisting of solutions of the PDE for n_λ number of parameters in $[\lambda_{\min}, \lambda^{'}]$ is collected through a suitable method, serving as the training set (input $(u(x,t,\lambda_i))$ - output $(u(x,t,\lambda_{i+1}))$ pairs) for the oscillator, where $\lambda_{\min} \leq \lambda_i \leq \lambda_{i+1} \leq \lambda_{\max}$. For the current study, the data is collected, and the predictions are made only at the *final time step* at n_x spatial locations.

We use the *long expressive memory* (LEM) [25] neural oscillator to update the hidden states. The neural oscillator processes the training data sequentially in $\mathcal{X}_1$ and predicts solutions outside the trained domain in $\mathcal{X}_2$. In LEM, like RNN, hidden states are updated using the current input and the previous hidden states. The *fundamental distinction* between vanilla or gated RNNs and neural oscillators lies in the hidden state update methodology. In neural oscillators, these updates are based on *systems of ODEs*, in contrast to algebraic equations used in typical RNNs. LEM updates the hidden states by solving the following ODEs

$$\begin{aligned} \mathbf{y}' &= \hat{\sigma}(\mathbf{W}_2\mathbf{y} + \mathbf{V}_2\mathbf{u} + \mathbf{b}_2) \odot [\sigma(\mathbf{W}_y\mathbf{z} + \mathbf{V}_y\mathbf{u} + \mathbf{b}_y) - \mathbf{y}] \\ \mathbf{z}' &= \hat{\sigma}(\mathbf{W}_1\mathbf{y} + \mathbf{V}_1\mathbf{u} + \mathbf{b}_1) \odot [\sigma(\mathbf{W}_z\mathbf{y} + \mathbf{V}_z\mathbf{u} + \mathbf{b}_z) - \mathbf{z}] \end{aligned} \tag{1}$$

The input to the oscillator is denoted by $\mathbf{u} \in \mathbb{R}^{n_x}$, which is the solution of a particular parametric PDE at final time. The hidden states within the framework of LEM are denoted as $\mathbf{y}$ and $\mathbf{z} \in \mathbb{R}^m$. Additionally, the weight matrices $\mathbf{W}_{1,2}$, $\mathbf{W}_{y,z} \in \mathbb{R}^{m\times m}$, and $\mathbf{V}_{1,2}$, $\mathbf{V}_{y,z} \in \mathbb{R}^{m\times n_x}$ are introduced, serving as the weight parameters. The bias vectors, denoted as $\mathbf{b}_{1,2}$ and $\mathbf{b}_{y,z} \in \mathbb{R}^m$. The activation function $\hat{\sigma}$ utilized is the sigmoid function, and the symbol $\odot$ signifies the element-wise product operation applied to vectors. A discretized representation of (1) could be written as,

$$\begin{aligned} \boldsymbol{\Delta t}_n &= \Delta t\hat{\sigma}(\mathbf{W}_1\mathbf{y}_{n-1} + \mathbf{V}_1\mathbf{u}_n + \mathbf{b}_1) \\ \overline{\boldsymbol{\Delta t}}_n &= \Delta t\hat{\sigma}(\mathbf{W}_2\mathbf{y}_{n-1} + \mathbf{V}_2\mathbf{u}_n + \mathbf{b}_2) \\ \mathbf{z}_n &= (1 - \boldsymbol{\Delta t}_n) \odot \mathbf{z}_{n-1} + \boldsymbol{\Delta t}_n \odot \sigma(\mathbf{W}_z\mathbf{y}_{n-1} + \mathbf{V}_z\mathbf{u}_n + \mathbf{b}_z) \\ \mathbf{y}_n &= (1 - \overline{\boldsymbol{\Delta t}}_n) \odot \mathbf{y}_{n-1} + \overline{\boldsymbol{\Delta t}}_n \odot \sigma(\mathbf{W}_y\mathbf{z}_n + \mathbf{V}_y\mathbf{u}_n + \mathbf{b}_y). \end{aligned} \tag{2}$$

The hidden states are augmented with a linear output state $\omega_n \in \mathbb{R}^{n_x}$ with $\omega_n = \mathcal{Q}\mathbf{y}_n$ and $\mathcal{Q} \in \mathbb{R}^{n_x\times m}$. Mean squared error loss function is utilized finally to train the LEM using ω_n and $u(x,t,\lambda_{i+1})$.

3 Numerical Experiments

Three numerical experiments on the linear advection equation, viscous Burgers equation, and the nonlinear heat equation are carried out to validate the presented method. For the associated challenge

and the rationale behind choosing the linear advection equation as the prototypical PDE to test the method, see [27, 28, 29]. The training dataset is the solution of the parametric PDE at the final time-varying spatial and parametric domain. For training and testing, we divide the entire parametric domain $\mathcal{X}$ into two regions, $\mathcal{X}_1 := [\lambda_{\min}, \lambda')$ and $\mathcal{X}_2 := (\lambda', \lambda_{\max}]$. For all the problems, we consider three cases, where $\lambda^{'} = p\lambda_{\max}$, dividing the training and testing set into $25:75, 50:50$, and $80:20$ ratio for $p = 0.25, 0.5$ and 0.8. The architectural setup employed for all three test cases is nearly identical, featuring a hidden size of 32, The Adam optimizer with a learning rate of 0.001, and 50000 epochs except for the first two cases of advection equation where 20000 epochs are performed. Fig. (1-3) show that the predictions are more accurate for $\lambda_{\max}$ when we utilize 80 percent of the parametric space for training LEM. Table 1 shows that when training with 25% and 50% parametric space datasets, the relative L^2 error at the final time is almost the same. The interpolation performance of the presented method is also tested for the advection equation and is presented in Appendix A.

3.1 Advection equation

The linear advection equation, along with the initial and boundary conditions, is given by,

$$\begin{aligned} u_t + \beta u_x = 0 \quad x \in [0, 2\pi], t \in [0, 1] \\ u(x, 0) = \sin(x), \quad u(0, t) = u(2\pi, t) \end{aligned} \tag{3}$$

where the parameter $\beta \in [0, 40]$. The analytic solution is $\sin(x - \beta t)$. The dataset is generated for $n_x = 5000$ and $|\mathcal{X}| = 1000$ different equidistant parameters.

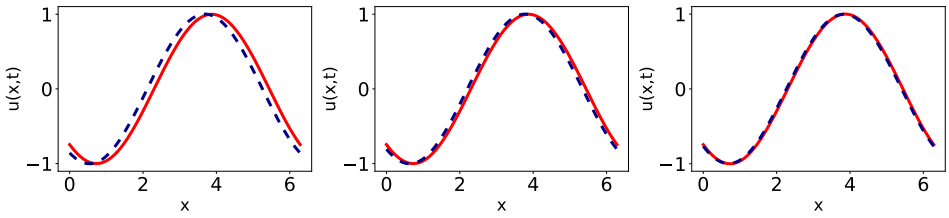

Figure 1: Snapshots for the extrapolation in the parametric space for the advection equation for $\beta = 40$; providing **Left**: 25% of data for the parametric space **Mid**: 50% of data for the parametric space **Right**: 80% of data for the parametric space. The solid red represents the reference solution, and the dashed blue represents the predictions. The order of the figures and the colors have the same meaning for all the following figures.

3.2 Burgers equation

The viscous Burgers equation, along with the initial and boundary conditions, is given by,

$$\begin{aligned} u_t + \frac{1}{2}(u^2)_x = \nu u_{xx} \quad x \in [-1, 1], t \in [0, 1] \\ u(x, 0) = -\sin(\pi x), \quad u(-1, t) = u(1, t) \end{aligned} \tag{4}$$

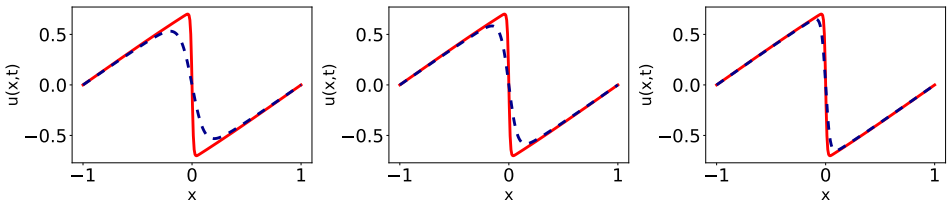

Figure 2: Snapshots for the extrapolation in the parametric space for the Burgers equation for $\nu = 0.05$

where parameter $\nu \in [0.005, 0.05]$. The reference solution is derived using Cole's transformation computed with Hermite integration [30, 31]. The dataset is generated for $n_x = 256$ and $|\mathcal{X}| = 1000$ different equidistant parameters.

Table 1: The extrapolation accuracy in terms of the relative errors in the L2-norm

Equation	Parameter	25%	50%	80%
Linear advection	$\beta = 32$	0.01383	0.00266	2.04e-5
	$\beta = 35$	0.02252	0.00222	0.04963
	$\beta = 40$	0.03472	0.00905	0.00128
Viscous Burgers	$\nu = 0.041$	0.02077	0.05566	9.91e-7
	$\nu = 0.044$	0.03288	0.07174	0.00208
	$\nu = 0.05$	0.01031	0.00602	0.00016
Nonlinear heat	$\alpha = 2.71$	0.36715	0.05332	6.79e-7
	$\alpha = 2.87$	0.51308	0.100799	0.00028
	$\alpha = \pi$	0.83146	0.22051	0.00035

3.3 Nonlinear heat equation

The nonlinear heat equation is given by,

$$u_t + \alpha u_{xx} + \tanh u = f \quad x \in [-1, 1], t \in [0, 1] \tag{5}$$

The analytic solution for this problem is $\sin(\pi x)e^{(-\alpha x^2)}e^{(-\alpha t^2)}$. The boundary, initial conditions and the source function are derived from the analytical solution. The parameter $\alpha \in [1, \pi]$. The dataset is generated for $n_x = 50$ and $|\mathcal{X}| = 1000$ different equidistant parameters.

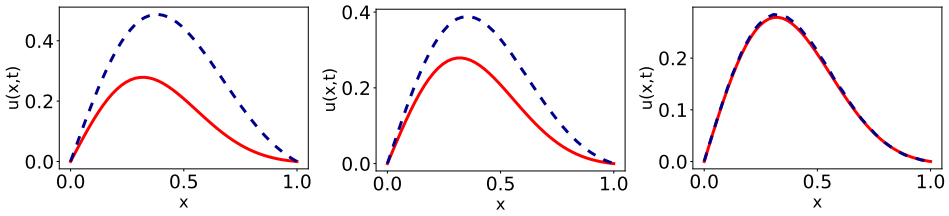

Figure 3: Snapshots for the extrapolation in the parametric space for the nonlinear heat equation for $\alpha = \pi$.

4 Conclusions

We introduced a neural oscillator-based approach for simulating parametric PDEs. In particular, we highlighted the efficacy of the proposed approach in predicting the solutions of parametric PDEs in an extrapolated parametric domain. Long expressive memory oscillator is used to train the network in an online stage using solutions of certain parametric PDEs. Subsequently, the trained network was applied to predict solutions beyond the range of trained parameters in an offline phase. Similar to generative methods in artificial intelligence and neural operators in scientific machine learning, our trained model could be used to generate solutions for parametric PDEs promptly. We demonstrated the effectiveness of our method on three benchmark PDEs: advection equation, viscous Burgers and nonlinear heat equation. It is observed that the predictive performance increased with the volume of the provided data for training the oscillator network. This study's predictions were limited to the final time within the domain. Future work will extend these predictions to encompass the entire space-time domain, which is conceptually straightforward but computationally intensive, aiming to predict solutions for parameters in a completely untrained space-time domain.

References

[1] Toni Lassila and Gianluigi Rozza. Parametric free-form shape design with PDE models and reduced basis method. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1583–1592, 2010.

[2] Kjetil O Lye, Siddhartha Mishra, Deep Ray, and Praveen Chandrashekar. Iterative surrogate model optimization (ISMO): An active learning algorithm for pde constrained optimization with deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 374:113575, 2021.

[3] Mohammad Vahab, Ehsan Haghighat, Maryam Khaleghi, and Nasser Khalili. A physics-informed neural network approach to solution and identification of biharmonic equations of elasticity. *Journal of Engineering Mechanics*, 148(2):04021154, 2022.

[4] Taniya Kapoor, Hongrui Wang, Alfredo Núñez, and Rolf Dollevoet. Physics-informed neural networks for solving forward and inverse problems in complex beam systems. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2023.

[5] Peng Wang, Daniel M Tartakovsky, KD Jarman Jr, and Alexandre M Tartakovsky. CDF solutions of Buckley–Leverett equation with uncertain parameters. *Multiscale Modeling & Simulation*, 11(1):118–133, 2013.

[6] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[7] Maziar Raissi, Paris Perdikaris, and George E 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, 2019.

[8] Arka Daw, Anuj Karpatne, William D Watkins, Jordan S Read, and Vipin Kumar. Physics-guided neural networks (PGNN): An application in lake temperature modeling. In *Knowledge Guided Machine Learning*, pages 353–372. Chapman and Hall/CRC, 2022.

[9] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

[10] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[11] Bogdan Raonić, Roberto Molinaro, Tobias Rohner, Siddhartha Mishra, and Emmanuel de Bezenac. Convolutional neural operators. *arXiv preprint arXiv:2302.01178*, 2023.

[12] Michael Prasthofer, Tim De Ryck, and Siddhartha Mishra. Variable-input deep operator networks. *arXiv preprint arXiv:2205.11404*, 2022.

[13] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science advances*, 7(40):eabi8605, 2021.

[14] Jungeun Kim, Kookjin Lee, Dongeun Lee, Sheo Yon Jhin, and Noseong Park. DPM: A novel training method for physics-informed neural networks in extrapolation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8146–8154, 2021.

[15] Min Zhu, Handi Zhang, Anran Jiao, George Em Karniadakis, and Lu Lu. Reliable extrapolation of deep neural operators informed by physics or sparse observations. *Computer Methods in Applied Mechanics and Engineering*, 412:116064, 2023.

[16] Katarzyna Michałowska, Somdatta Goswami, George Em Karniadakis, and Signe Riemer-Sørensen. Neural operator learning for long-time integration in dynamical systems with recurrent neural networks. *arXiv preprint arXiv:2303.02243*, 2023.

[17] Taniya Kapoor, Abhishek Chandra, Daniel M Tartakovsky, Hongrui Wang, Alfredo Nunez, and Rolf Dollevoet. Neural oscillators for generalization of physics-informed machine learning. *arXiv preprint arXiv:2308.08989*, 2023.

[18] Phillip Lippe, Bastiaan S Veeling, Paris Perdikaris, Richard E Turner, and Johannes Brandstetter. PDE-Refiner: Achieving accurate long rollouts with neural PDE solvers. *arXiv preprint arXiv:2308.05732*, 2023.

[19] Abhishek Chandra, Taniya Kapoor, Bram Daniels, Mitrofan Curti, Koen Tiels, Daniel M Tartakovsky, and Elena A Lomonova. Neural oscillators for magnetic hysteresis modeling. *arXiv preprint arXiv:2308.12002*, 2023.

[20] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[21] Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

[22] Bo Chang, Minmin Chen, Eldad Haber, and Ed H Chi. Antisymmetricrnn: A dynamical system view on recurrent neural networks. *arXiv preprint arXiv:1902.09689*, 2019.

[23] T Konstantin Rusch and Siddhartha Mishra. Coupled Oscillatory Recurrent Neural Network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies. *arXiv preprint arXiv:2010.00951*, 2020.

[24] T Konstantin Rusch and Siddhartha Mishra. Unicornn: A recurrent model for learning very long time dependencies. In *International Conference on Machine Learning*, pages 9168–9178. PMLR, 2021.

[25] T Konstantin Rusch, Siddhartha Mishra, N Benjamin Erichson, and Michael W Mahoney. Long expressive memory for sequence modeling. *arXiv preprint arXiv:2110.04744*, 2021.

[26] Samuel Lanthaler, T Konstantin Rusch, and Siddhartha Mishra. Neural oscillators are universal. *arXiv preprint arXiv:2305.08753*, 2023.

[27] 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*, 34:26548–26560, 2021.

[28] Tim De Ryck, Florent Bonnet, Siddhartha Mishra, and Emmanuel de Bézenac. An operator preconditioning perspective on training in physics-informed machine learning. *arXiv preprint arXiv:2310.05801*, 2023.

[29] Aleksandr Dekhovich, Marcel HF Sluiter, David MJ Tax, and Miguel A Bessa. ipinns: Incremental learning for physics-informed neural networks. *arXiv preprint arXiv:2304.04854*, 2023.

[30] Michael Penwarden, Shandian Zhe, Akil Narayan, and Robert M Kirby. A metalearning approach for physics-informed neural networks (PINNs): Application to parameterized PDEs. *Journal of Computational Physics*, 477:111912, 2023.

[31] Cea Basdevant, M Deville, P Haldenwang, JM Lacroix, J Ouazzani, R Peyret, Paolo Orlandi, and AT0612 Patera. Spectral and finite difference solutions of the burgers equation. *Computers & fluids*, 14(1):23–41, 1986.

A Appendix

The interpolation performance of the proposed method is presented for the linear advection equation in this appendix. Table 2 reports the error for the linear advection equation in the case of interpolating for $\beta = 4$, 8, and 10, where model is trained till $\beta = 10$, 32 corresponding to the 25%, and 80% cases respectively. Fig. 4 presents the comparisons of the obtained solution with the reference solution for these cases.

Table 2: The interpolation accuracy in terms of the relative errors in the L2-norm for the linear advection equation

Equation	Parameter	25%	80%
Linear advection	β = 4	0.00566	0.00011
	β = 8	0.00072	0.02903
	β = 10	0.00233	0.00010

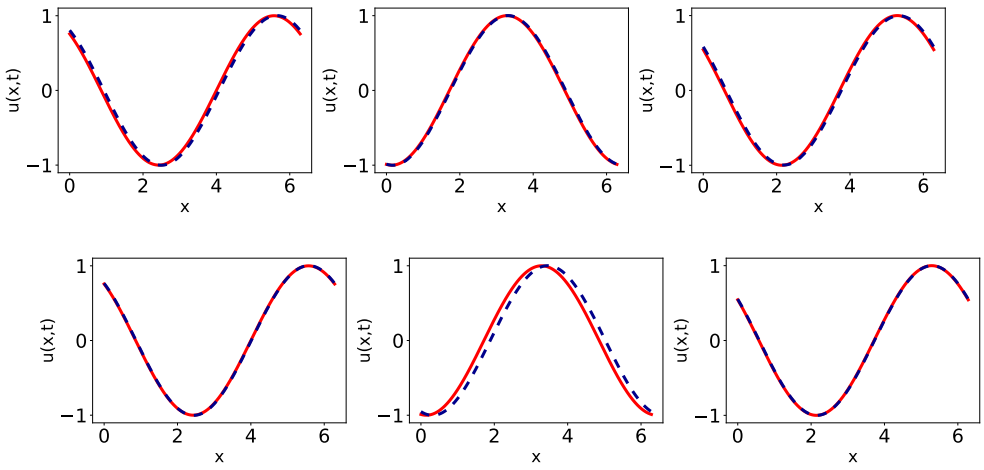

Figure 4: Snapshots for the interpolation in the parametric space for the advection equation for $\beta = 4, 8, 10$ (left to right); providing **Top**: 25% of data for the parametric space **Bottom**: 80% of data for the parametric space.