Physics Informed RNN-DCT Networks for Time-Dependent Partial Differential Equations

Benjamin Wu⋆ ⋆⋆1,2, Oliver Hennigh1, Jan Kautz1, Sanjay Choudhry1, and
Wonmin Byeon⋆1

1 NVIDIA, Santa Clara CA 95051, USA
{ohennigh,jkautz,schoudhry,wbyeon}@nvidia.com
2 National Astronomical Observatory of Japan, Mitaka Tokyo 181-8588
benwu.astro@gmail.com

Abstract. Physics-informed neural networks allow models to be trained by physical laws described by general nonlinear partial differential equations. However, traditional architectures of this approach struggle to solve more challenging time-dependent problems. In this work, we present a novel physics-informed framework for solving time-dependent partial differential equations. Using only the governing differential equations and problem initial and boundary conditions, we generate a latent representation of the problem's spatio-temporal dynamics. Our model utilizes discrete cosine transforms to encode spatial frequencies and re-current neural networks to process the time evolution. This efficiently and flexibly produces a compressed representation which is used for additional conditioning of physics-informed models. We show experimental results on the Taylor-Green vortex solution to the Navier-Stokes equations. Our proposed model achieves state-of-the-art performance on the Taylor-Green vortex relative to other physics-informed baseline models.

Keywords: Physics-informed neural networks · RNN · DCT · numerical simulation · PDEs · Taylor-Green vortex

1 Introduction

Numerical simulations have become an indispensable tool for modeling physical systems, which in turn drive advancements in engineering and scientific discovery. However, as the physical complexity or spatio-temporal resolution of a simulation increases, the computational resources and run times required to solve the governing partial differential equations (PDEs) often grow drastically.

ML-driven solvers. Recently, machine learning approaches have been applied to the domain of physical simulation to ameliorate these issues by approximating traditional solvers with faster, less resource-intensive ones. These methods generally fall into two main paradigms: data-driven supervision [2, 5,

---

⋆ Equal contribution.
⋆⋆ Work done during NVIDIA AI Research Residency

7, 12, 20] or physics-informed neural networks (PINNs) [1, 15, 16, 19]. *Data-driven approaches* excel in cases for which reliable training data is available and if the underlying physical equations are unknown. These generally utilize deep neural networks to parameterize the solution operator. *PINN-based solvers* parameterize the solution function directly as a neural network. This is typically done by passing a set of query points through a multilayer perceptron (MLP) and minimizing a loss function based on the governing PDEs, initial conditions (ICs) and boundary conditions (BCs). The simulation becomes constrained by physics alone and does not require any training data. However, the accuracy of traditional PINN-based approaches is limited to simpler problems in low dimensions and time-independent physics [10].

Learning time-dependent problems with PINNs. For time-dependent problems, traditional PINN-based models use a continuous time approach which treats the temporal and spatial dimensions in the same manner. To avoid increasingly poor performance as the simulation evolves in time, methods have been developed to split the domain into many short-time problems and solve each step using continuous-time PINNs [13, 14]. However, this results in additional model complexity and computational overhead. In principle, a well-constructed latent context grid allows the PINN to learn more easily while still relying on physics-constrained losses. In this work, we design a novel physics-informed MLP architecture by adding a new latent context generation process to effectively learn spatial-temporal physics problems.

Efficient learning in time and space. Typical feedforward neural networks lack notions of temporal relationships. Recurrent neural networks (RNNs) form graphs directed along a temporal sequence, allowing learning of time-dependent dynamics. Long Short-Term Memory (LSTM) [9] and Gated Recurrent Units (GRUs) [3] provide a gating mechanism to solve the problem of vanishing gradients and have become popular choices for RNNs, exhibiting high performance and efficiency. For spatial features, Xu et al. [18] demonstrate that compressing large images in the frequency domain using digital signal processing improved model accuracy while greatly reducing input size. This allows for high model efficiency while maintaining compatibility with standard spatial CNN architectures.

Contributions. Although PINN solvers provide a well-principled, machine learning approach that could enhance the capabilities of numerical simulations, their current constraints to problems with simple geometries and short times severely limits their real-world impact. We address these shortcomings by introducing novel design choices that improve the simulation accuracy and efficiency of PINN solvers on more challenging problems, particularly in the regime of long time evolution where current PINNs severely struggle.

Our key contributions are as follows: **(1)** We propose a new approach for latent context generation that requires no additional data and enables PINNs to learn complex time-dependent physics problems. **(2)** To the best of our knowledge, our work is the first to directly address space-time-dependent physics using PINNs. This is achieved by utilizing convolutional GRUs for learning the spatio-temporal dynamics of simulations. **(3)** We separate the spatial and frequency domains,
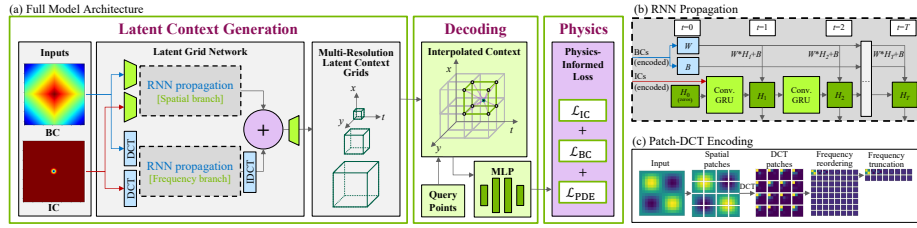
Fig. 1. (a) Full model architecture. (b) RNN propagation. (c) Patch-DCT encoding.

adding flexibility for the network to learn more diverse physical problems. **(4)** We test the new model against other architectures on benchmark transient simulation problems and demonstrate quantitative improvements in both accuracy and speed.

2   Methods

In this paper, we propose a new model that enables PINN-based neural solvers to learn temporal dynamics in both the spatial and frequency domains. Using no additional data, our architecture can generate a latent context grid that efficiently represents more challenging spatio-temporal physical problems.

Our full architecture is shown in Fig. 1 (a). It consists of three primary parts, which are explained in more detail below: (1) latent context generation, (2) decoding, and (3) physics-informing. The latent context generation stage takes as input the problem ICs and BCs and outputs spatio-temporal latent context grids. For the decoding stage, spatio-temporal query points along with additional vectors interpolated from the latent context grid are used as input. For each set of points, the MLP predicts corresponding output values, on which the physics-constrained losses are applied. Upon minimization of these losses, the MLP approximates the function governed by the underlying PDEs.

Latent grid network. The primary novelty of our method is the latent grid network that can generate context grids which efficiently represent the entire spatio-temporal domain of a physical problem without requiring additional data.

This network requires two inputs for the problem-specific constraints: ICs and BCs. The ICs are defined as $u_0 = u(x_{1,..,N}, t = 0)$ for each PDE solution function u over N spatial dimensions. The BCs are defined based on the geometry of the problem for each spatial dimension. An additional spatial weighting by signed distance functions (SDFs) can also be applied to avoid discontinuities at, e.g., physical boundaries, but would not be necessary for, e.g., periodic BCs. Each tensor undergoes an encoding step in either the frequency or spatial domain.

The *frequency branch* transforms the spatial inputs to frequencies via the discrete cosine transform (DCT), motivated by [18]. Fig. 1 (c) illustrates our patch-wise DCT encoding step. First, the ICs and BCs are separately split into spatial patches of size $p \times p$. DCTs are performed on each patch to yield the corresponding frequency coefficient array. The tensor is then reshaped such that the same coefficient across all patches forms each channel, and the channels are

reordered by increasing coefficient (i.e., decreasing energy). After the reordering, the channels are truncated by $n\%$, so the lowest $n\%$ of frequency coefficients (largest energies) are kept. This outputs highly compressed representations for the ICs and BCs, which are used as inputs for an RNN propagation branch.

The *spatial branch* follows a traditional ResNet [6] architecture, in which the ICs and BCs each pass through separate convolutional encoders comprising sets of convolutional blocks with residual connections. The inputs are downsampled with strided convolutions before entering the spatial RNN propagation branch.

RNN propagation. After compression, the representations enter the RNN propagation stage (Fig. 1 (b)), in which the BCs are split into an additive (B^{bc}) and multiplicative (W^{bc}) component and combined with an IC-informed state matrix (H_t). The final output at each timestep is computed as $S_t = W^{\mathrm{bc}}H_t + B^{\mathrm{bc}}$. This method offers flexibility and efficiency in learning the dynamics of compressed simulations [7]. To predict the simulation state at each successive timestep, the previous hidden state H_{t-1} is passed through a convolutional GRU (ConvGRU) along with the previous output S_{t-1}; for timestep 0, the initial state H_0 set to zero and ICs are used as inputs. This occurs in a recurrent manner until the final time T. Thus, for each timestep, the RNN propagation stage outputs S_t which is then sent to a decoding step corresponding to the original frequency or spatial encoding. $S_0 = u_0$, $H_0 = \mathbf{0}$, $H_t = \mathrm{ConvGRU}(S_{t-1}, H_{t-1})$, $S_t = W^{\mathrm{bc}}H_t + B^{\mathrm{bc}}$, $t \in \{1, \ldots, T\}$. The RNN propagation stage is duplicated across both frequency and spatial branches.

Latent grid generation. After RNN propagation, the outputs are combined to form the latent grid. In the frequency branch, the output state at each timestep from the RNN is converted back into the spatial domain: 1) reshaping the frequencies from coefficients to patches 2) performing IDCTs, and 3) merging the patches to reconstruct the spatial domain. The output of the frequency branch is denoted as O_t^f. The representation in the spatial domain O_t^s is then added with learnable weights W_t^o. Thus, the final output is computed as: $O_t = W_t^o O_t^s + O_t^f$. These combined outputs O_t for each timestep are used to form the spatio-temporal latent context grids. Finally, the multiple resolutions of grids are generated by upsampling the outputs O_t using transpose convolutional blocks.

Decoding step. The multi-resolution latent context grids generated from the previous step are then used to query points input to the MLP. This decoding step follows the same principles as [4]. Given a random query point $\mathbf{x} := (x, y, t)$, k neighboring vertices of $\mathbf{x}$ at each dimension are selected. Using these neighboring vertices, the final values of the context vector are then Gaussian-interpolated. This process is repeated for each of the multi-resolution grids allowing the PINN framework to learn spatio-temporal quantities at multiple resolutions.

Physics-informed loss. The MLP outputs predictions that are then subject to the loss function determined by the ICs, BCs, and the PDEs. The losses are backpropagated through the entire combined decoding and latent grid network and minimized via stochastic gradient descent. This end-to-end training allows our two-branch convGRU model to learn accurate time-evolution of the spatial and frequency domains in complex physical problems.

3 Experiments

We compare our model (**RNN-SpDCT**) against several other neural network solver architectures using the 2D Taylor-Green vortex problems. This problem is commonly used to test and validate spatial and temporal accuracy of both traditional and ML-based fluid solvers. We compare against PINN-based models and use the ICs, BCs, and PDE constraints for all comparing models. We used a single Tesla V100 16G or 32G for all experiments.

Taylor-Green vortex. The Taylor-Green vortex describes a decaying vortex flow which follows a special case of the Navier-Stokes equations [17]. The incompressible Navier-Stokes equations in 2D are $\partial_x u + \partial_y v = 0$, $\partial_t u + u\partial_x u + v\partial_y u = -\partial_x \rho/\rho + \nu(\partial_{xx} u + \partial_{yy} u)$, $\partial_t v + u\partial_x v + v\partial_y v = -\partial_y \rho/\rho + \nu(\partial_{xx} v + \partial_{yy} v)$, where u and v are the x- and y-velocities, respectively, $\nu \in \mathbb{R}_+$ is the kinematic viscosity, and ρ is the density.

The exact closed form solution for the Taylor-Green vortex over the domain $x \times y \times t \in [0, 2\pi] \times [0, 2\pi] \times [0, T]$ is $u = \cos x \sin y F(t)$, $v = -\sin x \cos y F(t)$, $p = \frac{-\rho}{4}(\cos 2x + \cos 2y)F^2(t)$, where $F(t) = e^{-2\nu t}$ and p is the pressure.

Dataset. We used 2π seconds for both training and testing. During training, the input size is set to 32×32. The first timestep is used as an initial condition, and x and y are used as boundary conditions. For testing, the size of x and y are set to 128×128, and 10 time steps are uniformly sampled between 0 and 2π.

Experimental setup. The number of interpolation points k used in the decoding step is 3, and the truncation ratio n is fixed to 25%. All models are trained with an Adam optimizer [11]. We found that patch-wise models need lower initial learning rate $1e-4$ and the others with higher initial learning rate $4e-4$ with 0.95 decay rate and different decay steps: 8000 steps for patch-based models, 2000 steps for other models.

Network architecture. The patch size p in the patch-based DCT models is set to 8. The number of (encoding) residual blocks are 2 for the spatial branch and 1 for the frequency branch. In the spatial branch, 4 additional residual blocks are used with stride 2 for downsampling . There is no downsampling layer for the frequency branch. All convolutional layers have a filter size of 3×3, and there are two RNN propagation layers. The number of upsampling layers is searched between 1 and 4, and the reported numbers are with 4 layers.

Baseline models. We compare our proposed model against several PINN-based approaches: MLP-PINN, RNN-S, RNN-pDCT, and RNN-SfDCT. All comparing models contain the RNN-propagation and decoding steps except for MLP-PINN and all use physics informed loss explained in section 2. All use the same hyper-parameters as our model except for learning rate and decay steps. **MLP-PINN**: a traditional MLP-based PINN solver used as a default model from `SimNet` [8]. **RNN-S**: a PINN solver with a latent grid network consisting of a single spatial branch (ResNet). **RNN-pDCT**: a PINN solver with a latent grid network consisting of a single frequency branch (DCT). **RNN-SfDCT**:a PINN solver with a latent grid network consisting of both spatial and frequency branches. The frequency branch in this model applies DCT/IDCT to the full input, foregoing the patching, coefficient channel reordering, and truncation steps.

Table 1. Quantitative comparisons. The averaged mean squared error (MSE) over 10 uniformly sampled time steps for 2π sec. is reported. ν is the kinematic viscosity of the fluid. F and S indicate frequency and spatial branches. FullDCT applies DCT to the entire input. The numbers are in the magnitude of 10^{-2}.

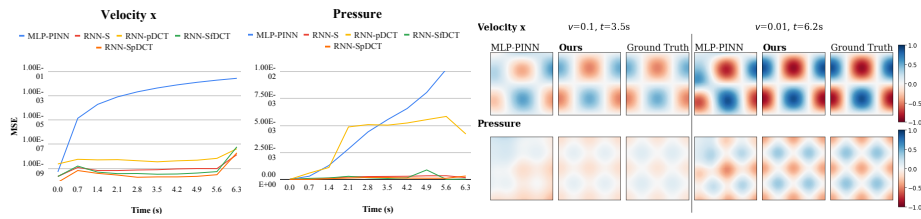| Model Name	Branch	DCT type	Taylor-Green Vortex					
			$\nu = 1.0$		$\nu = 0.1$		$\nu = 0.01$	
			velocity	pressure	velocity	pressure	velocity	pressure
MLP-PINN	-	-	0.033	5.910	1.769	0.782	0.824	0.522
RNN-S	S	-	6.683e-8	0.075	2.975e-7	0.138	2.527e-7	0.020
RNN-pDCT	F	patch	1.979e-6	0.172	5.957e-7	1.383	8.804e-7	0.508
RNN-SfDCT	S+F	full	9.171e-8	1.177	2.961e-7	0.301	7.015e-6	0.018
RNN-SpDCT	S+F	patch	1.408e-7	**0.044**	3.107e-7	**0.101**	1.328e-6	**0.012**

Fig. 2. Top: MSE comparisons over time. The MSE of the five models with respect to ground truth are shown over time for the Taylor-Green vortex problem. **Bottom: Visualization of the predictions** with the viscosity $\nu = 0.1$ at around 3.5 sec. (left) and $\nu = 0.01$ at around 6 sec. (right).

Results. Tab. 1 summarizes the performance of our model compared to the other PINN baselines. RNN-SpDCT achieves the best performance for all values of vorticity. All RNN models achieve extremely accurate velocities compared to MLP-PINN. Fig. 2 (top) shows the performance comparisons over time. Overall, for both x-velocity and pressure, RNN-based model produces much lower error throughout long time evolution compared to the baseline MLP-PINN, and RNN-SpDCT achieves the best overall performance. Fig. 2 (bottom) visualizes the predictions and compares with the analytical solution. The model produces much more accurate predictions for longer time steps (up to 2π sec.) compared to MLP-based PINNs.

4   Conclusion

We presented a novel extension to the PINN framework designed especially for time-dependent PDEs. Our model utilizes RNNs and DCTs to generate a multi-resolution latent context grid to condition the traditional MLP PINN architecture. We demonstrated that our model can accurately predict the solution functions in Taylor-Green vortex simulations (especially for pressures) and achieve state-of-the-art results. Future directions include experiments on more complex problems, higher dimensions, and longer time evolution.

Bibliography

[1] Bar, L., Sochen, N.: Unsupervised deep learning algorithm for pde-based forward and inverse problems. arXiv:1904.05417 (2019)

[2] Bhatnagar, S., Afshar, Y., et al.: Prediction of aerodynamic flow fields using convolutional neural networks. Computational Mechanics **64**(2), 525–545 (2019)

[3] Cho, K., et al.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv:1406.1078 (2014)

[4] Esmaeilzadeh, S., et al.: Meshfreeflownet: a physics-constrained deep continuous space-time super-resolution framework. In: SC20, pp. 1–15, IEEE (2020)

[5] Guo, X., Li, W., Iorio, F.: Convolutional neural networks for steady flow approximation. KDD (2016)

[6] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016)

[7] Hennigh, O.: Lat-net: compressing lattice boltzmann flow simulations using deep neural networks. arXiv:1705.09036 (2017)

[8] Hennigh, O., et al.: Nvidia simnet™: An ai-accelerated multi-physics simulation framework. In: ICCS, pp. 447–461, Springer (2021)

[9] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[10] Kashefi, A., Mukerji, T.: Physics-informed pointnet: A deep learning solver for steady-state incompressible flows and thermal fields on multiple sets of irregular geometries. arXiv:2202.05476 (2022)

[11] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)

[12] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., et al.: Fourier neural operator for parametric partial differential equations (2020)

[13] Mattey, R., Ghosh, S.: A physics informed neural network for time-dependent nonlinear and higher order partial differential equations. arXiv:2106.07606 (2021)

[14] Meng, X., Li, Z., Zhang, D., Karniadakis, G.E.: Ppinn: Parareal physics-informed neural network for time-dependent pdes. Computer Methods in Applied Mechanics and Engineering **370**, 113250 (2020)

[15] Raissi, M., Perdikaris, P., Karniadakis, G.E.: 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)

[16] Smith, J.D., Azizzadenesheli, K., Ross, Z.E.: Eikonet: Solving the eikonal equation with deep neural networks. TGRS (2020)

[17] Taylor, G., Green, A.: Mechanism of the production of small eddies from large ones. Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences (1937)

[18] Xu, K., Qin, M., Sun, F., Wang, Y., Chen, Y.K., Ren, F.: Learning in the frequency domain. In: CVPR, pp. 1740–1749 (2020)

[19] Yu, B., et al.: The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. arXiv:1710.00211 (2017)

[20] Zhu, Y., Zabaras, N.: Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. Journal of Computational Physics **366**, 415–447 (2018)