TOWARDS FAST SIMULATION OF ENVIRONMENTAL FLUID MECHANICS WITH MULTI-SCALE GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

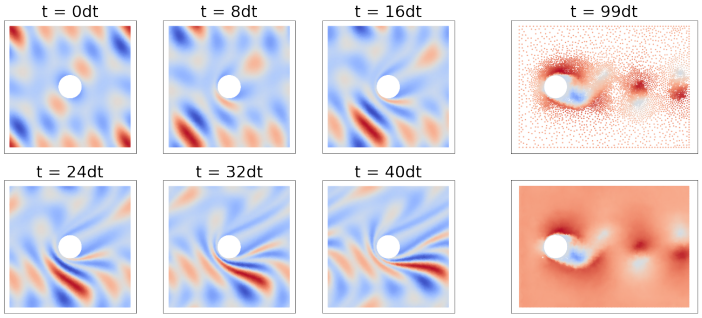
ABSTRACT

Numerical simulators are essential tools in the study of natural fluid-systems, but their performance often limits application in practice. Recent machine-learning approaches have demonstrated their ability to accelerate spatio-temporal predictions, although, with only moderate accuracy in comparison. Here we introduce MultiScaleGNN, a novel multi-scale graph neural network model for learning to infer unsteady continuum mechanics in problems encompassing a range of length scales and complex boundary geometries. We demonstrate this method on advection problems and incompressible fluid dynamics, both fundamental phenomena in oceanic and atmospheric processes. Our results show good extrapolation to new domain geometries and parameters for long-term temporal simulations. Simulations obtained with MultiScaleGNN are between two and four orders of magnitude faster than those on which it was trained.

1 INTRODUCTION

Forecasting the spatio-temporal mechanics of continuous systems is a common requirement in many areas of science and engineering, including environmental fluid dynamics (Rubin, 2001). Physical models for the transport and dispersion processes in natural fluid flows often consist on one or more partial differential equations (PDEs), whose complexity may preclude their analytic solution (Kim & Boysan, 1999). Numerical methods are well-established for approximating the solution of PDEs with high accuracy, but they are computationally expensive (Karniadakis & Sherwin, 2013). Deep learning techniques have recently been shown to accelerate physical simulations (Guo et al., 2016). Most of the recent work on deep learning to infer continuum physics has focused on the use of convolutional neural networks (CNNs). In part, the success of CNNs for these problems lies in their translation invariance and locality Goodfellow et al. (2016), which represent strong and desirable inductive biases for continuum-mechanics models. However, CNNs constrain input and output fields to be defined on rectangular domains represented by regular grids, which is not suitable for more complex domains. As with traditional numerical techniques, it is desirable to be able to vary the resolution in space, devoting more effort where the physics are challenging to resolve, and less effort elsewhere. An alternative approach to applying deep learning to geometrically and topologically complex domains are graph neural networks (GNNs), which can also be designed to satisfy spatial invariance and locality (Battaglia et al., 2018; Wu et al., 2020).

In this paper we describe a novel approach to applying GNNs for accurately forecasting the evolution of physical systems in complex and irregular domains. We propose MultiScaleGNN, a multi-scale GNN model, to forecast the spatio-temporal evolution of continuous systems discretised as unstructured sets of nodes. Each scale processes the information at different resolutions, enabling the network to more accurately and efficiently capture complex physical systems. We apply MultiScaleGNN to simulate advection and incompressible fluid dynamics, fundamental processes in oceanic and atmospheric research. Importantly, MultiScaleGNN is independent of the spatial discretisation and the mean absolute error (MAE) decreases linearly as the distance between nodes is reduced, which allows for the application of adaptive re-meshing (see Figure 1b) MultiScaleGNN simulations are between two and four orders of magnitude faster than the numerical solver used for generating the ground truth datasets, becoming a potential surrogate model for fast predictions.



(a) Advection (b) Incompressible flow

Figure 1: MultiScaleGNN forecasts continuum dynamics. We used it to simulate (a) advection [video] and, (b) the incompressible flow around a circular cylinder [video].

2 Related work

During the last five years, most deep neural networks (DNNs) architectures used for predicting fluid dynamics have included convolutional layers (Guo et al., 2016; Tompson et al., 2017; Lee & You, 2019; Kim et al., 2019; Wiewel et al., 2019; Fotiadis et al., 2020). These CNN-based solvers are between one and four orders of magnitude faster than numerical solvers (Guo et al., 2016; Lino et al., 2020), and some of them have shown good extrapolation to unseen domain geometries and initial conditions (Thuerey et al., 2018; Lino et al., 2020). Recently, GNNs have been used to simulate the motion of discrete systems of solid particles (Battaglia et al., 2016; Chang et al., 2017) and deformable solids and fluids discretised into Lagrangian (or *free*) particles (Li et al., 2019a; Mrowca et al., 2018b; Sanchez-Gonzalez et al., 2020). Further research in this area introduced more general message-passing (MP) layers (Sanchez-Gonzalez et al., 2018; Li et al., 2019b; Mrowca et al., 2018a), high-order time-integration (Sanchez-Gonzalez et al., 2019) and hierarchical models (Li et al., 2019a; Mrowca et al., 2018b). To the best of our knowledge Alet, et al. (2019) Alet et al. (2019) were the first to explore the use of GNNs to infer Eulerian mechanics by solving the Poisson PDE. However, their domains remained simple, used coarse spatial discretisations and did not explore the generalisation of their model. More closely related to our work, Pfaff et al. (2021) proposed a mesh-based GNN to simulate continuum mechanics, although they did not consider the use of MP at multiple scales of resolution. Li et al. (2020b) and Liu et al. (2021) used multi-resolution GNNs to infer steady solutions, but their pooling remained simple and did not explore extrapolation to unseen domain geometries or physical parameters. Li et al. (2020a) also considered multi-scale neural PDE modelling, with the mayor drawback that the spatial discretisations are constrained to uniform grids.

3 Model

3.1 Model definition

For a PDE $\frac{\partial \mathbf{u}}{\partial t} = \mathcal{F}(\mathbf{u})$ on a spatial domain $\mathcal{D} \subset \mathbb{R}^2$, MultiScaleGNN infers the temporal evolution of $\mathbf{u}(t, \mathbf{x})$ at a finite set of nodes V^1, with coordinates $\mathbf{x}_i^1 \in \mathcal{D}$. [1] Given an input field $\mathbf{u}(t_0, \mathbf{x}_{V^1})$, at time $t = t_0$ and at the V^1 nodes, a single evaluation of MultiScaleGNN returns $\mathbf{u}(t_0 + dt, \mathbf{x}_{V^1})$, where dt is a fixed time-step size. Each time-step is performed by applying MP layers in L graphs and between them, as illustrated in Figure 2. The high-resolution graph G^1 consists of the set of nodes V^1 and a set of directed edges E^1 connecting these nodes. In a complete graph there exist $|V^1|(|V^1| - 1)$ edges, however this would result in a high computational cost to MP. Instead, MultiScaleGNN connects each node in V^1 (receiver node) to its k-closest nodes (k sender nodes) using a k-nearest neighbours (k-NN) algorithm. This guarantees that in MP layers each node is receiving information from exactly k nodes, in contrast to Pfaff et al. (2021) and Sanchez-Gonzalez et al. (2018). This uniformity can ease the learning and improve the accuracy of the predictions. The attributes assigned to each node $\mathbf{v}_i^1$ are the concatenation of $\mathbf{u}(t_0, \mathbf{x}_i)$, $\mathbf{p}_i$ and Ω_i, where $\mathbf{p}_i$ is a set

[1] Model repository available on *revealed on publication.*

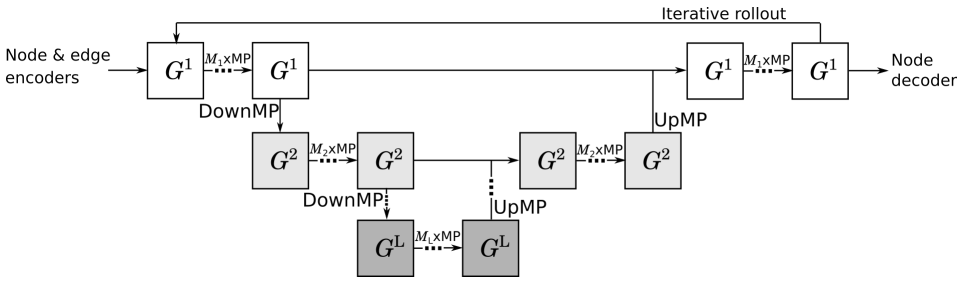


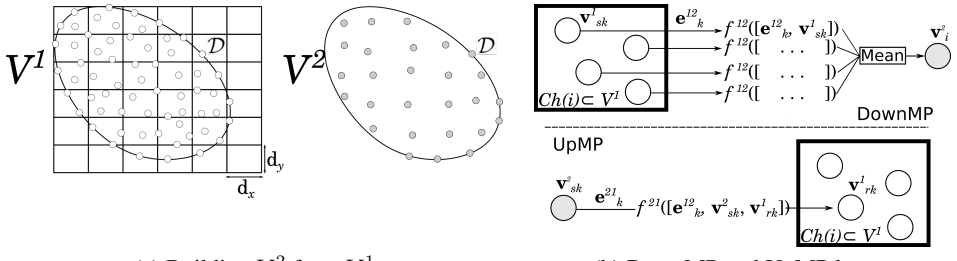
Figure 2: MultiScaleGNN architecture. G^1 is the high-resolution graph, G^2 and G^L are low-resolution graphs.

of physical parameters at $\mathbf{x}_i$ (such as the Reynolds number in fluid dynamics) and $\Omega_i = 1$ for nodes located on Dirichlet boundaries and zero elsewhere. Each edge attribute $\mathbf{e}_k^1$ is assigned the relative position between sender node s_k and receiver node r_k.

Node attributes and edge attributes are encoded through two independent multi-layer perceptrons (MLPs). A MP layer applied to G^1 propagates the nodal and edge information only locally between adjacent nodes. Nevertheless, most continuous physical-systems require this propagation at larger scales or even globally. To handle this, MultiScaleGNN processes the information at L scales, creating a graph for each level and propagating information between them in each pass. The lower-resolution graphs ($G^2, G^3, \ldots, G^L$; with $|V_1| > |V_2| > \cdots > |V_L|$) possess fewer nodes and edges, and hence, a single MP layer can propagate features over longer distances more efficiently. As depicted in Figure 2, the information is first diffused and processed in the high-resolution graph G^1 through M_1 MP layers. It is then passed to G^2 through a downward MP (DownMP) layer. In G^2 the attributes are again processed through M_2 MP layers and a DownMP layer to G^3. This process is repeated $L-1$ times. The lowest resolution attributes (stored in G^L) are then passed back to the scale immediately above through an upward message passing (UpMP) layer. Attributes are again successively passed through M_l MP layers at scale l and an UpMP layer from scale l to scale $l-1$ until the information is ultimately processed in G^1. Finally, a MLP decodes the nodal information to return the predicted field at time t_0+dt at the V^1 nodes. To apply MP in the L graphs, MultiScaleGNN uses the MP layer introduced by Sanchez-Gonzalez et al. (2018) and Battaglia et al. (2018), with the mean as the aggregation function.

3.2 MULTI-SCALE GRAPHS

Each graph $G^l = (V^l, E^l)$ with $l = 2, 3, \ldots, L$; is obtained from graph G^{l-1} by first dividing $\mathcal{D}$ into a regular grid with cell size $d_x^l \times d_y^l$ (Figure 3a exemplifies this for $l = 2$). For each cell, provided that there is at least one node from V^{l-1} in it, a node is added to V^l. The nodes from V^{l-1} in a given cell i and the node from V^l on the same cell are denoted as child nodes, $Ch(i) \subset V^{l-1}$, and parent node, $i \in V^l$, respectively. The coordinates of each parent node is the mean position of its children nodes. Each edge $k \in E^l$ connects sender node $s_k \in V^l$ to receiver node $r_k \in V^l$, provided that there exists at least one edge from $Ch(s_k)$ to $Ch(r_k)$ in E^{l-1}. Edge k is assigned the mean edge attribute of the set of edges going from $Ch(s_k)$ to $Ch(r_k)$.



(a) Building V^2 from V^1 (b) DownMP and UpMP layers

Figure 3: (a) V^2 is obtained from V^1 by partitioning $\mathcal{D}$ using a grid with cell size $d_x^2 \times d_y^2$. (b) DownMP and UpMP diagrams.

Downward message-passing (DownMP). To perform MP from G^{l-1} to G^l (see Figure 3b), a set of directed edges, $E^{l-1,l}$, is created. Each edge $k \in E^{l-1,l}$ connects node $s_k \in V^{l-1}$ to its parent node $r_k \in V^l$, with edge attributes assigned as the relative position between child and parent nodes. A DownMP layer applies a common edge-update function, $f^{l-1,l}$, to edge k and node s_k. It then assigns to each node attribute $\mathbf{v}_i^l$ the mean updated attribute of all the edges in $E^{l-1,l}$ arriving to node i, i.e.,

$$\mathbf{v}_i^l = \frac{1}{|Ch(i)|} \sum_{k:r_k=i} f^{l-1,l}([\mathbf{e}_k^{l-1,l}, \mathbf{v}_{s_k}^{l-1}]), \quad \forall i \in 1, \ldots, |V^l|. \tag{1}$$

Upward message-passing (UpMP). To pass and process the node attributes from G^{l+1} to G^l, MultiScaleGNN defines a set of directed edges, $E^{l+1,l}$. These edges are the same as in $E^{l,l+1}$, but with opposite direction. An UpMP layer applies a common edge-update function, $f^{l+1,l}$, to each edge $k \in E^{l+1,l}$ and both its sender (at scale $l+1$) and receiver (at scale l) nodes, directly updating the node attributes in G^l, i.e.,

$$\mathbf{v}_{r_k}^l = f^{l+1,l}([\mathbf{e}_k^{l+1,l}, \mathbf{v}_{s_k}^{l+1}, \mathbf{v}_{r_k}^l]), \quad \forall k \in E^{l+1,l}. \tag{2}$$

UpMP layers leave the edge attributes of G^l unaltered. To model functions $f^{l-1,l}$ and $f^{l+1,l}$ we use MLPs.

4 TRAINING DATASETS

Datasets `AdvBox` and `AdvInBox`, both used simultaneously for training, contain simulations of a scalar field advected under a uniform velocity field on a square domain ($[0,1] \times [0,1]$) and a rectangular domain ($[0,1] \times [0,0.5]$) respectively. `AdvBox` domains have periodic conditions on all four boundaries, whereas `AdvInBox` domains have upper and lower periodic boundaries, a prescribed Dirichlet condition on the left boundary, and a zero-Neumann condition on the right boundary. The initial states at t_0 are derived from two-dimensional truncated Fourier series with random coefficients and a random number of terms. For advection models, $\mathbf{u}(t, \mathbf{x}_i) \in \mathbb{R}$ is the advected field and $\mathbf{p}_i \in \mathbb{R}^2$ are the two components of the velocity field at $\mathbf{x}_i$.

Dataset `NS` contains simulations of the periodic vortex shedding around a circular cylinder at Reynolds numbers between 500 and 1000. The upper and lower boundaries are periodic, and the vertical distance between cylinders is randomly sampled between 4 and 6. Each domain is discretised into approximately 7000 nodes. For flow models, $\mathbf{u}(t, \mathbf{x}_i) \in \mathbb{R}^3$ contains the velocity and pressure fields and $\mathbf{p}_i \in \mathbb{R}$ is the Reynolds number. Further details of the training and testing datasets are included in Appendix A. [2]

5 RESULTS AND DISCUSSION

We first consider the MultiScaleGNN model trained to infer advection. Despite MultiScaleGNN being trained on square and rectangular domains and uniform velocity fields, it generalises to complex domains and non-uniform velocity fields. As an example of a closed domain, we consider the Taylor-Couette flow in Figure 4, where after 49 time-steps a MultiScaleGNN with $L = 1$ maintains high accuracy in transporting both the lower and the higher frequencies generated due to the shear flow. We also evaluated the predictions of MultiScaleGNN on open domains containing obstacles of different shapes (circles, squares, ellipses and closed spline curves). Figure 1 shows the predictions obtained with a MultiScaleGNN with $L = 2$ for a field advected around a circular cylinder and upper-lower and left-right periodic boundaries. MultiScaleGNN was also trained to simulate unsteady incompressible fluid dynamics in a range of Reynolds numbers between 500 and 1000. It shows very good interpolation to unseen Reynolds numbers within this range. For instance, Figure 1b shows the horizontal velocity field predicted by MultiScaleGNN with $L = 4$ after 99 time-steps for $Re = 700$. The MAE increases for Reynolds numbers lower than 400 and higher than 1200, but the predictions remain visually realistic in the range 300 to 1500, indicating that the network has learnt some fundamental aspects of the fluid dynamics. [3]

[2] Datasets available on *revealed on publication*.

[3] Animations with the ground truth and best-model predictions can be found *here*.

We evaluate the accuracy of MultiScaleGNN with $L = 1, 2, 3$ and 4; the architectural details of each model are included in Appendix B. Tables 1 and 2 collect the MAE for the last time-point and the mean of all the time-points on the testing datasets. Incompressible fluids have a global behaviour, and the addition of coarser layers helps the network to learn this characteristic and achieve significantly lower errors. Hence, for the N-S testing dataset there is a clear benefit from increasing the number of scales. In contrast, for the advection datasets, the lowest MAE values are obtained for $L = 1$ or 2, since in advection the information is propagated only locally. As a comparison to Pfaff et al. (2021), a GNN with 16 sequential MP-layers (GN-Blocks) results in a MAE of 5.852×10^{-2} on our `NSMidRe` dataset; whereas MultiScaleGNN with the same number and type of MP layers, but distributed among 3 scales, results in a lower MAE of 3.081×10^{-2}. A comparison of our coarsening/pooling algorithm to Liu et al. (2021) is included in Appendix C.

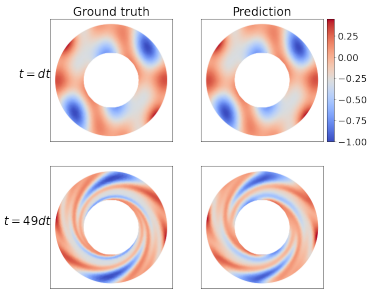


Figure 4: Advection of a scalar field in a Taylor-Couette flow [`video`].

Table 1: MAE $\times 10^{-2}$ on the advection testing datasets for MultiScaleGNN models with $L = 1, 2, 3, 4$

Datasets	$L=1$		$L=2$		$L=3$		$L=4$	
	Step 49	All	Step 49	All	Step 49	All	Step 49	All
`AdvTaylor`	**6.940**	**3.195**	7.914	3.676	8.037	3.739	8.790	4.050
`AdvCircle`	4.057	2.112	**3.690**	**1.817**	3.911	1.916	3.827	1.995
`AdvCircleAng`	**3.870**	2.030	3.962	**1.890**	4.300	2.074	4.428	2.249
`AdvSquare`	4.250	2.178	**4.175**	**1.991**	4.420	2.107	4.279	2.141
`AdvEllipseH`	4.462	2.241	**4.328**	**2.017**	4.567	2.136	4.449	2.179
`AdvEllipseV`	4.265	2.236	**4.132**	**2.014**	4.297	2.100	4.207	2.22
`AdvSplines`	4.484	2.293	**4.426**	**2.088**	4.636	2.199	4.58	2.300
`AdvInCir`	**11.7**	**7.101**	25.369	18.548	27.981	22.626	27.379	22.034

Table 2: MAE $\times 10^{-2}$ on the N-S testing datasets for MultiScaleGNN models with $L = 1, 2, 3, 4$

Datasets	$L=1$		$L=2$		$L=3$		$L=4$	
	Step 99	All	Step 99	All	Step 99	All	Step 99	All
`NSMidRe`	9.765	6.108	4.759	3.663	3.851	3.081	**3.456**	**2.825**
`NSLowRe`	9.43	7.327	12.346	8.211	11.707	8.203	**7.338**	**5.532**
`NSHighRe`	9.487	10.598	7.879	9.002	6.98	7.096	**5.826**	**5.871**

6 Conclusion

MultiScaleGNN is a novel multi-scale GNN model for inferring mechanics on continuous systems discretised into unstructured sets of nodes. Unstructured discretisations allow complex domains to be accurately represented and the node count to be adjusted over space. Multiple coarser levels allow high and low-resolution mechanics to be efficiently captured. In global and local problems, such as incompressible fluid dynamics, the coarser graphs are particularly advantageous, since they enable global characteristics to be learnt. MultiScaleGNN interpolates to unseen spatial discretisations of the physical domains, allowing it to adopt efficient discretisations and to dynamically and locally modify them to further improve the accuracy. MultiScaleGNN also generalises to advection on complex domains and velocity fields and it interpolates and extrapolates to unseen Reynolds numbers in fluid dynamics. Inference is between two and four orders of magnitude faster than with the high-order solver used for generating the training datasets. This work is a significant advancement in the design of flexible, accurate and efficient neural simulators for fluid dynamics and in general for continuum mechanics.

REFERENCES

Ferran Alet, Adarsh Keshav Jeewajee, Maria Bauza Villalonga, Alberto Rodriguez, Tomas Lozano-Perez, and Leslie Kaelbling. Graph element networks: adaptive, structured computation and memory. In *International Conference on Machine Learning*, pp. 212–222. PMLR, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Peter W Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *38th Neural Information Processing Systems, NeurIPS 2016*, 2016.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Filipe de Avila Belbute-Peres, Thomas Economon, and Zico Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *International Conference on Machine Learning*, pp. 2402–2411. PMLR, 2020.

Chris D Cantwell, David Moxey, Andrew Comerford, Alessandro Bolis, Gabriele Rocco, Gianmarco Mengaldo, Daniele De Grazia, Sergey Yakovlev, J-E Lombard, Dirk Ekelschot, et al. Nektar++: An open-source spectral/hp element framework. *Computer physics communications*, 192:205–219, 2015.

Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

Stathi Fotiadis, Eduardo Pignatelli, Mario Lino, Chris Cantwell, Amos Storkey, and Anil A. Bharath. Comparing recurrent and convolutional neural networks for predicting wave propagation. In *ICLR Workshop on Deep Neural Models and Differential Equations*, 2020.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. MIT press Cambridge, 2016.

Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 481–490, 2016.

George Karniadakis and Spencer Sherwin. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, 2013.

Byungsoo Kim, Vinicius C Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. In *Computer Graphics Forum*. Wiley Online Library, 2019.

Sung-Eun Kim and Ferit Boysan. Application of cfd to environmental flows. *Journal of Wind Engineering and Industrial Aerodynamics*, 81(1-3):145–158, 1999.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *arXiv preprint arXiv:1706.02515*, 2017.

Sangseung Lee and Donghyun You. Data-driven prediction of unsteady flow over a circular cylinder using deep learning. *Journal of Fluid Mechanics*, 879:217–254, 2019.

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *7th International Conference on Learning Representations, ICLR 2019*, 2019a.

Yunzhu Li, Jiajun Wu, Jun Yan Zhu, Joshua B. Tenenbaum, Antonio Torralba, and Russ Tedrake. Propagation networks for model-based control under partial observation. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May:1205–1211, 2019b. ISSN 10504729. doi: 10.1109/ICRA.2019.8793509.

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*, 2020a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020b.

Mario Lino, Chris Cantwell, Stathi Fotiadis, Eduardo Pignatelli, and Anil Anthony Bharath. Simulating surface wave dynamics with convolutional networks. In *NeurIPS Workshop on Interpretable Inductive Biases and Physically Structured Learning*, 2020.

Wenzhuo Liu, Mouadh Yagoubi, and Marc Schoenauer. Multi-resolution graph neural networks for pde approximation. In *International Conference on Artificial Neural Networks*, pp. 151–163. Springer, 2021.

Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B. Tenenbaum, and Daniel L.K. Yamins. Flexible neural representation for physics prediction. *Advances in Neural Information Processing Systems*, 2018-Decem:8799–8810, 2018a. ISSN 10495258.

Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li F Fei-Fei, Josh Tenenbaum, and Daniel L Yamins. Flexible neural representation for physics prediction. In *Advances in neural information processing systems*, pp. 8799–8810, 2018b.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. In *(accepted) 38th International Conference on Machine Learning, ICML 2021*, 2021.

Hillel Rubin. *Environmental fluid mechanics*. CRC Press, 2001.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. *35th International Conference on Machine Learning, ICML 2018*, 10:7097–7117, 2018.

Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian graph networks with ode integrators. *arXiv preprint arXiv:1909.12790*, 2019.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020.

Nils Thuerey, Konstantin Weissenow, Lukas Prantl, and Xiangyu Hu. Deep Learning Methods for Reynolds-Averaged Navier-Stokes Simulations of Airfoil Flows. *AIAA Journal*, 58(1):15–26, oct 2018. ISSN 0001-1452. doi: 10.2514/1.j058291. URL `http://arxiv.org/abs/1810.08217`.

Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning*, pp. 3424–3433. PMLR, 2017.

Steffen Wiewel, Moritz Becher, and Nils Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer Graphics Forum*. Wiley Online Library, 2019.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

A DATASETS DETAILS

A.1 ADVECTION DATASETS

We solved the two-dimensional advection equation using Nektar++, an spectral/hp element solver (Karniadakis & Sherwin, 2013; Cantwell et al., 2015). As initial condition, φ_0, we take an scalar field derived from a two-dimensional Fourier series with $M \times N$ random coefficients, specifically

$$\varphi_0 = \sum_{m=0}^{M} \sum_{n=0}^{N} c_{m,n} \varphi_{m,n} \cdot \exp\Big(-2(x-x_c)^2 - 2(y-y_c)^2 \Big), \tag{3}$$

$$\text{with} \quad \varphi_{m,n} = \Re\Big\{ \exp\Big(i2\pi(mx+ny) \Big) \Big\}. \tag{4}$$

Coefficients $c_{m,n}$ are sampled from a uniform distribution between 0 and 1, and integers M and N are randomly selected between 3 and 8. In equation (4), x_c and y_c are the coordinates of the centre of the domain. The initial field φ_0 is scaled to have a maximum equal to 1 and a minimum equal to -1. We created training and testing datasets containing advection simulations with 50 time-points each, equispaced by a time-step size $dt = 0.03$. A summary of these datasets can be found in Table 3.

Training datasets. We generated two training datasets: `AdvBox` with 1500 simulations and `AdvInBox` with 3000 simulations. In these datasets we impose a uniform velocity fields with random values for u and v, but constrained to $u^2 + v^2 \leq 1$. In dataset `AdvBox` the domain is a square ($\mathbf{x} \in [0,1] \times [0,1]$) with periodicity in x and y. In dataset `AdvInBox` the domain is a rectangle ($\mathbf{x} \in [0,1] \times [0,0.5]$) with periodicity in y, a Dirichlet condition on the left boundary and a homogeneous Neumann condition on the right boundary – as an additional constraint, $u \geq 0$. During training, a new set of nodes V^1 is selected at the beginning of every iteration. The node count was varied smoothly across the different regions of the domains. The sets of node were created with Gmsh, a finite-element mesher. The *element size* parameter was set to 0.012 in the corners and the centre of the training domains, and set to $\sqrt{10}$ or $1/\sqrt{10}$ times that value at one random control point on each boundary. The mean number of nodes in $|V_1|$ for `AdvBox` and `AdvInBox` predictions are 9802 and 5009 respectively.

Testing datasets. We generated eight testing datasets, each of them containing 200 simulations. These datasets consider advection on more complex open and closed domains with non-uniform velocity fields. The domains employed are represented in Figure 5, and the testing datasets are listed in Table 3. The velocity fields were obtained from the steady incompressible Navier-Stokes (N-S) equations with $Re = 1$. In dataset `AdvTaylor` the inner and outer walls spin at a velocity randomly sampled between -1 and 1. In datasets `AdvCircle`, `AdvSquare`, `AdvEllipseH`, `AdvEllipseV` and `AdvSplines` there is periodicity along x and y, and a horizontal flow rate between 0.2 and 0.75 is imposed. The obstacles inside the domains on the `AdvSplines` dataset are made of closed spline curves defined from six random points. Dataset `AdvCircleAng` is similar to `AdvCircle`, but the flow rate forms an angle between -45 deg and 45 deg with the x axis. The domain in dataset `AdvInCir` has periodicity along y, a Dirichlet condition on the left boundary (with $0.2 \leq u^2 + v^2 \leq 0.75$ and $-45\,\text{deg} \leq \arctan(v/u) \leq 45\,\text{deg}$), and a homogeneous Neumann condition on the right boundary. The set of nodes V^1 were generated using Gmsh with an element size equal to 0.005 on the walls of the obstacles and 0.01 on the remaining boudaries.

A.2 INCOMPRESSIBLE FLUID DYNAMICS DATASETS

We solved the two-dimensional incompressible Navier-Stokes equation using the high-order solver Nektar++. We consider the flow around an infinite vertical array of circular cylinders, with diameter $D = 1$, equispaced a distance H randomly sampled between $4D$ and $6D$. The width of the domain is $7D$ and the cylinders axis is at $1.5D$ from the left boundary. The left boundary is an inlet with $u = 1$, $v = 0$ and $\partial p/\partial x = 0$; the right boundary is an outlet with $\partial u/\partial x = 0$, $\partial v/\partial x = 0$ and $p = 0$; and, the cylinder walls have a no-slip condition. In our simulations we select Re values that yield solutions in the laminar vortex-shedding regime, and we only include the periodic stage to our datasets. The sets of nodes V^1 employed for each simulation were created using Gmsh placing more

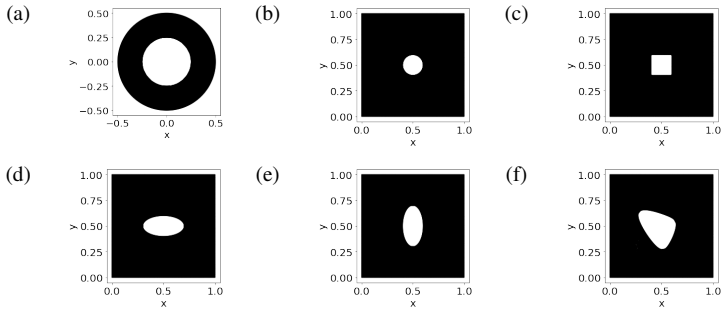


Figure 5: Physical domains (black areas) on our testing datasets.

Table 3: Advection training and testing datasets

Dataset	Flow type	Domain	#Nodes	Train/Test
`AdvBox`	Open, periodic in x and y	$[0,1] \times [0,1]$	9601-10003	Training
`AdvInBox`	Open, periodic in y	$[0,1] \times [0,0.5]$	4894-5135	Training
`AdvTaylor`	Closed, Taylor-Couette flow	Figure 5a	7207	Testing
`AdvCircle`	Open, periodic in x and y	Figure 5b	19862	Testing
`AdvCircleAng`	Open, periodic in x and y	Figure 5b	19862	Testing
`AdvSquare`	Open, periodic in x and y	Figure 5c	19956	Testing
`AdvEllipseH`	Open, periodic in x and y	Figure 5d	20210	Testing
`AdvEllipseV`	Open, periodic in x and y	Figure 5e	20221	Testing
`AdvSplines`	Open, periodic in x and y	Figure 5f	19316-20389	Testing
`AdvIncir`	Open, periodic in y	Figure 5b	19862	Testing

nodes around the cylinders walls. The mean number of nodes in these sets is 7143. Each simulation contains 100 time-points equispaced by a time-step size $dt = 0.1$. The training and testing datasets are listed in Table 4.

B MODEL DETAILS

Hyper-parameters choice. The number of edges going to each node was set to $k = 6$, and the number of layers in each MLP (encoder, decoder and edge and node update functions) was three, with 128 neurons per hidden layer. All MLPs (except the decoder) use SELU activation functions (Klambauer et al., 2017), and, batch normalisation (Ba et al., 2016). The grid sizes used for generating the coarser graphs are collected in Table 5. The number of MP layers we used at each scale are listed in the Table 6.

Training details. We trained MultiScaleGNN models on a internal cluster using 4 CPUs, 86GB of memory, and a RTX6000 GPU with 24GB. We fed 8 graphs per batch. First, each training iteration predicted a single time-point, and, every time the training loss decreased below a threshold (0.01 for advection and 0.005 for fluid dynamics) we increased the number of iterative time-steps by one, up

Table 4: Incompressible flow training and testing datasets

Dataset	Re	#Simulations	Train/Test
`NS`	500-1000	1000	Training
`NSMidRe`	500-1000	250	Testing
`NSLowRe`	100-500	250	Testing
`NSMidRe`	1000-1500	250	Testing

Table 5: Cell sizes for coarsening to levels 2, 3 and 4

Cell size	Advection	Fluid dynamics
d_x^2, d_y^2	0.02	0.15
d_x^3, d_y^3	0.04	0.30
d_x^4, d_y^4	0.08	0.60

Table 6: Number of MP layers at each scale for $L = 1, 2, 3$ and 4

L	Advection	Fluid dynamics
$L = 1$	$M_1 = 4$	$M_1 = 8$
$L = 2$	$M_1 = 2, M_2 = 4$	$M_1 = 4, M_2 = 4$
$L = 3$	$M_1 = 2, M_2 = 2, M_3 = 4$	$M_1 = 4, M_2 = 2, M_3 = 4$
$L = 4$	$M_1 = 2, M_2 = 2, M_3 = 2, M_4 = 4$	$M_1 = 4, M_2 = 2, M_3 = 2, M_4 = 4$

to a limit of 10. We used the loss function given by

$$\begin{aligned}\mathcal{L} = \mathrm{MSE}\Big(\hat{\mathbf{u}}(t, \mathbf{x}^1_{V^1}), \mathbf{u}(t, \mathbf{x}^1_{V^1})\Big) \\ + \lambda_d \,\mathrm{MAE}\Big(\hat{\mathbf{u}}(t, \mathbf{x}^1_{V^1} \in \partial_D \mathcal{D}), \mathbf{u}(t, \mathbf{x}^1_{V^1} \in \partial_D \mathcal{D})\Big) \\ + \frac{\lambda_e}{|E^1|} \sum_{\forall k \in E^1} \mathrm{MSE}\left(\frac{\hat{\mathbf{u}}(t, \mathbf{x}^1_{r_k}) - \hat{\mathbf{u}}(t, \mathbf{x}^1_{s_k})}{||\mathbf{e}_k||_2}, \frac{\mathbf{u}(t, \mathbf{x}^1_{r_k}) - \mathbf{u}(t, \mathbf{x}^1_{s_k})}{||\mathbf{e}_k||_2}\right).\end{aligned} \tag{5}$$

with $\lambda_d = 0.25$, and, $\lambda_e = 0.5$ for advection and $\lambda_e = 0$ for fluid dynamics. The initial time-point was randomly selected for each prediction, and, we added to the initial field noise following a uniform distribution between -0.01 and 0.01. After each time-step, the models' weights were updated using the Adam optimiser with its standard parameters Kingma & Ba (2015). The learning rate was set to 10^{-4} and multiplied by 0.5 when the training loss did not decrease after six consecutive epochs, also, we applied gradient clipping to keep the Frobenius norm of the weights gradients below or equal to one.

C COARSENING COMPARISON

An important question in the design of MultiScaleGNN was how to coarsen V_l to V_{l+1}. In the field of numerical simulations several coarsening algorithms have been developed to guarantee the stability and accuracy of the simulations. Liu et al. (2021) and Belbute-Peres et al. (2020) used coarsening techniques from numerical simulations and then interpolated the nodes attributes into the coarsened set of nodes. We found that our *cell-grid* coarsening and learnt MP from the high to the low-resolution graph (see section 3.2) performs significantly better than such coarsening-interpolation approach. Table 7 shows the MAE on the incompressible flow testing datasets for MultiScaleGNN models using both pooling strategies and a random coarsening, which randomly drops nodes to match the same $|V_l|/|V_{l+1}|$ ratio. In contrast to coarsening algorithms adopted from numerical simulations, cell-grid coarsening does not preserve the spatial node density from the original discretisation, but it results into a pseudo-structured discretisation. This discretisation may help to spread the node information more efficiently on all directions and regions and to ease the learning of the parameters of the MP layers at low resolution scales. Also, instead of a vanilla interpolation, in the DownMP the node information is passed to the low-resolution graph using a learnt MP layer, and the edge information is also passed to the edges of the low-resolution graph.

Table 7: MAE $\times 10^{-2}$ on the N-S testing datasets for $L = 1, 2, 3, 4$ with three different coarsening algorithms

Datasets	L	Cell grid		Guillard's		Random	
		Step 99	All	Step 99	All	Step 99	All
NSMidRe	2	**4.759**	**3.663**	6.756	4.393	7.191	4.741
	3	**3.851**	**3.081**	5.735	3.952	6.431	4.606
	4	**3.456**	**2.825**	4.494	3.330	6.337	4.428
NSLowRe	2	12.346	8.211	**9.053**	**6.378**	10.736	7.255
	3	11.707	8.203	9.099	6.946	**8.977**	**6.672**
	4	**7.338**	**5.532**	14.358	9.164	10.335	7.521
NSHighRe	2	**7.879**	9.002	10.776	7.608	11.024	**7.842**
	3	**6.980**	**7.096**	10.331	7.556	9.631	7.28
	4	**5.826**	**5.871**	11.222	7.837	9.572	7.202