IMPROVING REGIONAL WEATHER FORECASTS WITH NEURAL INTERPOLATION

James Jackaman
Norwegian University of Science and Technology
james.jackaman@ntnu.no

Oliver J. Sutton
King's College London and Synoptix Ltd
oliver.sutton@kcl.ac.uk

ABSTRACT

In this paper we design a neural interpolation operator to improve the boundary data for regional weather models, which is a challenging problem as we are required to map multi-scale dynamics between grid resolutions. In particular, we expose a methodology for approaching the problem through the study of a simplified model, with a view to generalise the results in this work to the dynamical core of regional weather models. Our approach will exploit a combination of techniques from image super-resolution with convolutional neural networks *(CNNs)* and residual networks, in addition to building the flow of atmospheric dynamics into the neural network.

1 Motivation

In recent years, machine learning has revolutionised many areas of scientific computing. One area of recent significant advancement is weather prediction. Due to its multi-scale nature and the chaotic nature of the underlying dynamics, weather forecasting has long employed statistical tools. Indeed, to improve the accuracy of forecasts models are enhanced through the incorporation of observations (known as data assimilation) [25, 28, 13]. Additionally, unresolvable phenomena (such as cloud coverage and precipitation) are often approximated by statistical models [1, 31].

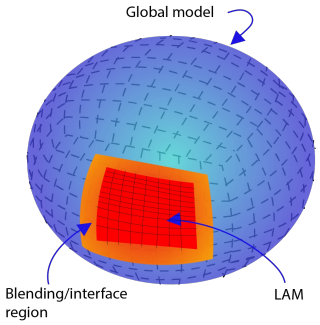
Machine learning weather forecasting models have recently been developed through graph neural networks by Google DeepMind [20], and have been matured to the extent they can compete with operational forecasts over 10 days [21]. One (current) limitation of such models is they do not take careful consideration of physical laws, although significant progress is being made in this direction, [4, 10, 18, 7, e.g.], and the field is rapidly developing. Our philosophy behind this work is to marry the recent progress in machine learning with traditional approaches taken by meteorological centres.

Due to the chaotic and multi-scale nature of atmospheric dynamics, to accurately simulate weather locally one must resolve the problem to a high resolution. While the method of simulation has changed over time and varies between meteorological institutes [32, 24, c.f.], existing models have invariably been derived from numerical discretisations of PDEs. In practice, these models reduce to solving a (non)linear system of equations, which for a high resolution model becomes prohibitively (computationally) expensive. A key tool in improving this are parallelisable iterative linear solvers [34], which have recently been implemented for Navier-Stokes [17]. To avoid prohibitive computational costs, one approach meteorologists take is to restrict their model to a region of high interest and run this much smaller model at a high resolution [2, 3]. Unfortunately, this approach has a major limitation. Information must both enter and leave the region in a physically meaningful (and qualitatively accurate) manor. This has long been the bane of regional weather models, and is a prime area for enhancement by neural networks. Here, we shall refer to regional weather models as limited area models *(LAMs)*.

Typically, fine scale regional weather models are driven by a coarse global model which informs the boundary data. A large amount of nonphysical behaviour appears near the boundaries of the LAM due to differences in physics at different scales. While not typical in meteorological models, it is possible to decrease this effect by grading the mesh to slowly increase resolution on the boundary. Unfortunately, this approach remains unable to resolve some fine-grain phenomena (such as eddies and vortices) moving onto the LAM, as they do not appear on the coarse global model. One approach often taken at meteorological centres can be described as follows: Around the area of interest, one may introduce an interface (or blending) region, as shown in Figure 1. This region varies in size, but typically requires an area similar to that of the LAM. Initially, the solution in the interface region and LAM is interpolated directly from the coarse global model. The model is then run over time, and at every time step, the solution on the global model and LAM are averaged in the blending region. After a certain

amount of time, the dynamics in the LAM stabilise and appear physically meaningful. This is known as spinning up the model.

Figure 1: A visualisation of the coarse global model, LAM (regional model) and interface region.



In this work, *we aim to improve the spinning up of the model by replacing the averaging between the global model and LAM with a neural interpolation operator*. This interpolation aims not only to accurately carry information from the coarse grid to the fine grid, but also to infer unresolvable features of the fine grid from high resolution training data. Our approach will utilise techniques from image super-resolution [12, 33], residual neural networks [15, 5], and will also incorporate the flow dynamics over time. To clarify the exposition (as well as highlighting potential pitfalls) we consider a significantly simplified model with an aim to generalise what we learn in our simplified setting to more complex fluid dynamics models. In fact, for the approach we outline the main difficulty in generalising lies in data representation and not in simulating the appropriate physics. This will be the subject of forthcoming future work.

2 Methodology

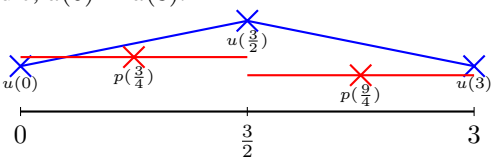
Atmospheric dynamical cores are typically driven by the Euler equations in an ideal gas. Here, we consider the simplified model of the shallow water equation *(SWE)*, as this is often the first case study considered when developing new dynamical cores, [30, 19, e.g.]. The SWE describes a shallow layer of fluid in hydrostatic balance bounded from below by bottom topography and above by a free surface. In two dimensional space, the equation is described by two variables, namely velocity in the (x, y)-direction and pressure. Similarly to [30], to further simplify we make the assumption that our solution is constant in y, reducing the equations as follows: Let $x \in S^1$ be periodic and $t \in [0, T]$ for some fixed end time T, then the SWE is

$$u_t - fv + p_x = 0, \quad v_t + fu = 0, \\ p_t + gu_x = 0, \tag{1}$$

where $u = u(t, x), v = v(t, x)$ are velocities in the x and y direction respectively, and $p = p(t, x)$ is the pressure. Further, f and g are positive constants where f represents the Coriolis parameter and g an initial reference pressure value. We note that, even though our model assumes functions are constant in y, the velocity in the y-direction is not constant due to the Coriolis parameter. While vortices and eddies are not captured by this simplified model, mapping solutions between scales still represents a significant challenge. The SWE supports three key types of waves, gravity waves, inertial waves and travelling waves. High frequency travelling waves are particularly difficult to model in the discrete setting. Beyond a certain frequency these are known to behave nonphysically and (on sufficiently coarse meshes) alias to lower frequency waves, see [23, e.g.]. In §3 we shall consider an amalgamation of these types of waves to maximise the expressivity of our neural interpolant.

To maximise compatibility between models (and in line with the UK Met Office's next generational goals [2]), we utilise the same discretisations for the global model and LAM. In particular, a compatible finite element discretisation [8, 9] reduced to 1D space. We outline this discretisation in more detail in Appendix A, including the temporal discretisation. After partitioning our periodic spatial domain, our discretisation is given by *piecewise polynomials* within each element of the domain. In particular, each component of our velocity space is described by a *continuous piecewise linear function*, and the pressure by a *discontinuous piecewise constant function*. Our velocity space can be uniquely characterised by the values of the solution at nodal points, and conversely the pressure by the value of the solution at the midpoint of each element, as visualised in Figure 2. The points at which we evaluate the solution are known as degrees of freedom *(DOF)*. Our velocity and pressure may be represented as vectors comprised of the DOF in our approximation, and these vectors form the data structures we need to process with our neural network. Before proceeding, we make two important remarks: Firstly, that the velocity space and pressure space fundamentally different and should be treated as such. Secondly, that the DOF are equidistributed, which is required to act on these vectors with a CNN.

Figure 2: A coarse visualisation of the 1D mesh partitioned into three nodes with two elements. We represent a function in the velocity space in blue, marking the DOF with crosses, and a function in the pressure space in red, again marking the DOF with crosses. Note that, as the domain is periodic, $u(0) = u(3)$.



For clarity of exposition, here we envision our entire domain as the blending region of the LAM, that is to say we consider both a coarse "global" model over the periodic interval $[0, 3)$ and the "regional" model over the same in-

terval. In this case study, our models will simply differ in resolution, with the LAM having *four times* the resolution of the coarse global model. We fix our temporal resolution of both models to be the same, although note that this is not fundamental to our methodology.

Our goal is to take the DOF of a solution simulated on the coarse scale and interpolate this to the DOF of a solution on the fine scale. Our approach is to initialise both the coarse and fine models with highly oscillatory data, and as such, the dynamics will evolve at different speeds on different meshes. We aim to learn not only how to interpolate the initial conditions onto a finer mesh, but also the difference in how the dynamics evolve on the different meshes with a neural interpolator. Here, our methodology takes inspiration from image super-resolution [12]: We begin by interpolating our coarse data onto the fine mesh data structure, which is an exact operation if our spatial meshes are nested. After interpolating onto the fine mesh, we label this data as $\boldsymbol{x}_c$, which will be the input data for our neural network. Here $\boldsymbol{x}_c \in \mathbb{R}^{N\times D}$ where N is our number of time steps and D the total dimension of the spatial problem on the fine mesh. This neural network will be trained against the same initial data which has evolved natively on the fine mesh, denoted $\boldsymbol{x}_f \in \mathbb{R}^{N\times D}$. The core of our model is a CNN, inspired largely by their relatively small kernel size and ability to capture and incorporate derivative information, [27, 11, 6], and is structured similarly to a U-Net [22, 29]. In practice, here we will stack time into the channel dimension and construct U-Net-like CNNs for each component in the velocity and pressure independently. This choice is made due to the differences in data structure between velocity and pressure. Following PyTorch syntax, this leads to a neural network of the type

```
def UNet(x):
  skip = x
  x = ReLU(Conv1d(N, s1, kernel_size=3)(x))
  x = ReLU(Conv1d(s1, s2, kernel_size=5)(x))
  x = ReLU(Conv1d(s2, s2, kernel_size=7)(x))
  x = ReLU(Conv1d(s2, s1, kernel_size=5)(x))
  x = Conv1d(s1, N, kernel_size=3)(x) + skip
  return x
```

where `x` may be $\boldsymbol{u}_c$, $\boldsymbol{v}_c$ or $\boldsymbol{p}_c$ with $\boldsymbol{x}_c =: [\boldsymbol{u}_c, \boldsymbol{v}_c, \boldsymbol{p}_c]^T$. To simplify notation we will later write `UNet` acting on $\boldsymbol{x}_c$, however, its action on each component of the solution in this work is independent. We note the skip connection here is crucial. For example, if we aim to learn the identity operator, this is a surprisingly difficult task. However, through the inclusion of a skip connection we may capture the identity mapping through choosing all weights to be zero. For our problem, the inclusion of a skip connection means the neural network only needs to learn the error between the desired output and the identity operator.

As we understand how the dynamics evolve on the fine scale over time, it is possible to improve upon our network. In particular, we aim to give our network the ability to learn the difference between dynamics at different scales. Fortunately, our problem is linear, so it is possible to find the flow map of the fine dynamics over time, which is a matrix operator $A \in \mathbb{R}^{D\times D}$ mapping the solution between time levels following the fine dynamics. With this operator in mind, we modify our network by

```
def LearnFlow(x):
  for i in range(N-1):
    x[i+1,:] = A x[i,:]
  x = UNet2(x)
  return x
```

where $\mathtt{x} = \boldsymbol{x}_c$ and `UNet2` does not share weights with the previously introduced `UNet` but for simplicity has the same structure. Astute readers may question the validity of utilising the operator A. We believe this is a reasonable object to include in our neural network as in the full problem we will only utilise it in the blending region, and it will have a complexity on the order of magnitude of the LAM. For more complex dynamics we may employ cheaper approximations inspired by iterative (non)linear solvers. With these functions in mind, we define our neural interpolant as

$$\mathrm{NN}(x_c) := \mathrm{UNet}(x_c) + \mathrm{LearnFlow}(\mathrm{UNet}(\mathrm{x_c})) \quad (2)$$

Before discussing the implementation of (2), we briefly remark upon some useful properties of the SWE. The SWE (1) is a *conservative* PDE. A prime example of this is the conservation of energy over time. In the continuous setting, this conservation can be expressed as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{S^1} \left(g\left[u^2+v^2\right]+p^2\right)\,\mathrm{d}x = 0. \quad (3)$$

This is a powerful result, as (3) immediately provides stability of the solution over time in the spatial L_2 norm. In finite element analysis, when mapping solutions between mesh resolutions, it is typical to utilise a projection operator which is stable in an appropriate norm. To be stable in terms of (3) one may choose the L_2 norm. Ideally, this property is something we would like to mimic in our neural interpolant (2). Unfortunately, the value of the discrete energy is dependent on the mesh resolution. That is to say, the discrete energies on the coarse and fine grid will not be the same, and we cannot know the correct energy on the fine grid without knowing the fine solution. This prohibits us from being able to exactly enforce conservation of energy in our neural interpolator, which in turn would guarantee stability of the operator. We can, however, improve stability through weakly enforcing that the *coarse* energy is preserved. That is to say, we can compute the initial energy of the input $\boldsymbol{x}_c$ and penalise the energy of the output to be approximately the same. This is physically meaningful (assuming the energies do not vary significantly between grids) and adds a form of L_2 regularisation to the optimisation problem.

3 Implementation

We will now discuss the implementation of our methodology. Our implementation can be found at [16], and depends on PyTorch 2.1.2 [26]. Experiments are run on an Apple

M2 Pro with 16GB of RAM. The training data was generated with Firedrake [14]. All experiments in this section are driven by the finite element method described in Appendix A, where we fix our coarse mesh to have an element size of 0.04 compared to a fine mesh element size of 0.01. Throughout, we fix our time step to be 0.01. Further, we choose the parameters of (1) to be $f = 0.1$ and $g = 1$. Our training data is generated by the initial conditions

$$u(0,x) = 0, \quad v(0,x) = 0,$$
$$p(0,x) = \alpha \left(e^{(-\beta x - x_0)^2} + \frac{1}{10} \sin\left(2\pi k (x - x_0)\right) \right), \tag{4}$$

where $\alpha \approx \mathcal{U}(\frac{1}{2}, 2)$, $\beta \approx \mathcal{N}(100, 6)$, $x_0 \approx \mathcal{U}(1, 2)$ and $k \approx \mathcal{U}^{\text{int}}(4, 10)$ with $\mathcal{U}, \mathcal{U}^{\text{int}}, \mathcal{N}$ uniform, uniform integer and normal distributions respectively. Such waves are an amalgamation of interio-gravity and travelling waves. We generate 1000 runs and utilise $N = 10$ time steps for training. We use 70% of our data for training and the remainder for verification.

We build the neural interpolator (2) by fixing the UNet sizes to be $s_1 = 2^3 \cdot N$ and $s_2 = 2^6 \cdot N$ in all instances. To train our network, we use a batch sizes of 16 and optimise against the square of the L_2 error in the finite element space. We note that if we instead considered the mean squared error, our errors would be smaller in magnitude, but here we choose the L_2 error as the mean squared error decreases as the mesh resolution is increased, and is not a trustworthy measure of error for a finite element function. We use the Adam optimiser with a learning rate of 10^{-3} and train over 300 epochs. Every 30 steps, the learning rate decays by a factor of 10.

As discussed in §2, we shall also study the weak enforcement of energy, as this gives a weak notion of stability to our network in addition to regularising the network. Our approach is given as follows: For an input $\boldsymbol{x}_c$, we extract the value of the energy at the initial time and penalise the difference between the initial energy and the energy of $\text{NN}(\boldsymbol{x}_c)$ in absolute value. This penalty is averaged over time and over the batch, and is multiplied by some fixed penalty parameter σ in the loss function.

We run simulations with no regularisation ($\sigma = 0$), and varying levels of regularisation, and in Table 1 report the loss after 300 epochs.

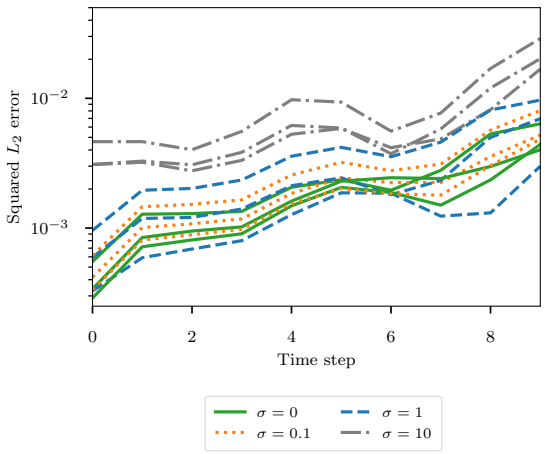
Table 1: Training loss for various values of regularisation

σ	0	0.1	1	10
Loss	$6 \cdot 10^{-6}$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-4}$	$4 \cdot 10^{-3}$

We observe that, with no regularisation, our method can be trained to the order of 10^{-6}. Unfortunately, we are not currently able to train our network to a higher accuracy than this.

After training we apply our validation data in Figure 3. In this figure, all information we display the average of a batch of 16 runs. Further, for transparency, we display three random batches of runs for each trained model. We notice that, as we increase the regularisation, the error increases. In fact, as confirmed by the training loss, once $\sigma = 10$ our model becomes significantly less accurate. For small values of σ, the error remains competitive.

Figure 3: The square of the L_2 error for the neural interpolator applied to the validation dataset with various regularisation parameters σ.



We note that, on average, the energy is conserved similarly well for various σ, however, the standard deviation significantly increases as σ increases. We further note that these errors (compared against errors in finite element analysis) are by no means small, and further study is required to fully exploit the expressivity of the network.

4 Conclusion

In this paper, we introduced the problem of LAM and outlined how neural networks can be used to improve regional weather forecasts. We considered a simplified set up, and designed a neural interpolation operator to map dynamics between differing grid scales. We found that, while these operators can be learnt, they have large errors relative to the finite element dynamics. These errors are most likely due to suboptimalities in our optimisation routine. Training the network to a higher accuracy is crucial before these methods can be effectively utilised for meteorological applications. In addition, our techniques need to be generalised to be applied in $2D$ space and for higher order finite element methods, which requires graph neural networks and is ongoing work.

Acknowledgements

This work was supported by the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 101108679 (JJ) and by the UKRI Turing AI Acceleration Fellowship EP/V025295/2 (OJS)

References

[1] L. M. Berliner. Physical-statistical modeling in geophysics. *Journal of Geophysical Research: Atmospheres*, 108(D24), Sept. 2003.

[2] M. Bush, T. Allen, C. Bain, I. Boutle, J. Edwards, A. Finnenkoetter, C. Franklin, K. Hanley, H. Lean, A. Lock, J. Manners, M. Mittermaier, C. Morcrette, R. North, J. Petch, C. Short, S. Vosper, D. Walters, S. Webster, M. Weeks, J. Wilkinson, N. Wood, and M. Zerroukat. The first met office unified model–jules regional atmosphere and land configuration, ral1. *Geoscientific Model Development*, 13(4):1999–2029, Apr. 2020.

[3] M. Bush, I. Boutle, J. Edwards, A. Finnenkoetter, C. Franklin, K. Hanley, A. Jayakumar, H. Lewis, A. Lock, M. Mittermaier, S. Mohandas, R. North, A. Porson, B. Roux, S. Webster, and M. Weeks. The second met office unified model–jules regional atmosphere and land configuration, ral2. *Geoscientific Model Development*, 16(6):1713–1734, Mar. 2023.

[4] E. Cardoso-Bihlo and A. Bihlo. Exactly conservative physics-informed neural networks and deep operator networks for dynamical systems. *Neural Networks*, 181:106826, Jan. 2025.

[5] E. Celledoni, M. J. Ehrhardt, C. Etmann, R. I. Mclachlan, B. Owren, C.-B. Schonlieb, and F. Sherry. Structure-preserving deep learning. *European Journal of Applied Mathematics*, 32(5):888–936, May 2021.

[6] E. Celledoni, J. Jackaman, D. Murari, and B. Owren. Predictions based on pixel data: Insights from pdes and finite differences. *arXiv*, May 2024.

[7] E. Celledoni, D. Murari, B. Owren, C.-B. Schönlieb, and F. Sherry. Dynamical systems–based neural networks. *SIAM Journal on Scientific Computing*, 45(6):A3071–A3094, Dec. 2023.

[8] C. Cotter and J. Thuburn. A finite element exterior calculus framework for the rotating shallow-water equations. *Journal of Computational Physics*, 257:1506–1526, Jan. 2014.

[9] C. J. Cotter. Compatible finite element methods for geophysical fluid dynamics. *Acta Numerica*, 32:291–393, May 2023.

[10] M. David and F. Méhats. Symplectic learning for hamiltonian neural networks. *Journal of Computational Physics*, 494:112495, Dec. 2023.

[11] B. Dong, Q. Jiang, and Z. Shen. Image restoration: Wavelet frame shrinkage, nonlinear evolution pdes, and beyond. *Multiscale Modeling and Simulation*, 15(1):606–660, Jan. 2017.

[12] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, Feb. 2016.

[13] M. Fisher, J. Nocedal, Y. Trémolet, and S. J. Wright. Data assimilation in weather forecasting: a case study in pde-constrained optimization. *Optimization and Engineering*, 10(3):409–426, July 2008.

[14] D. A. Ham, P. H. J. Kelly, L. Mitchell, C. J. Cotter, R. C. Kirby, K. Sagiyama, N. Bouziani, S. Vorderwuelbecke, T. J. Gregory, J. Betteridge, D. R. Shapero, R. W. Nixon-Hill, C. J. Ward, P. E. Farrell, P. D. Brubeck, I. Marsden, T. H. Gibson, M. Homolya, T. Sun, A. T. T. McRae, F. Luporini, A. Gregory, M. Lange, S. W. Funke, F. Rathgeber, G.-T. Bercea, and G. R. Markall. *Firedrake User Manual*. Imperial College London and University of Oxford and Baylor University and University of Washington, first edition edition, 5 2023.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[16] J. Jackaman. Implementation of "Improving regional weather forecasts with neural interpolation". https://doi.org/10.5281/zenodo.14888898, 2025.

[17] J. Jackaman and S. MacLachlan. Space-time waveform relaxation multigrid for Navier-Stokes. arXiv, 2024.

[18] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028, June 2020.

[19] J. Kent, T. Melvin, and G. A. Wimmer. A mixed finite-element discretisation of the shallow-water equations. *Geoscientific Model Development*, 16(4):1265–1276, Feb. 2023.

[20] R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, O. Vinyals, J. Stott, A. Pritzel, S. Mohamed, and P. Battaglia. Graphcast: Learning skillful medium-range global weather forecasting. Dec. 2022.

[21] R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, O. Vinyals, J. Stott, A. Pritzel, S. Mohamed, and P. Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, Dec. 2023.

[22] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[23] T. Melvin. Dispersion analysis of the P_n–P_{n-1}^{DG} mixed finite element pair for atmospheric modelling.

Journal of Computational Physics, 355:342–365, Feb. 2018.

[24] T. Melvin, B. Shipway, N. Wood, T. Benacchio, T. Bendall, I. Boutle, A. Brown, C. Johnson, J. Kent, S. Pring, C. Smith, M. Zerroukat, C. Cotter, and J. Thuburn. A mixed finite-element, finite-volume, semi-implicit discretisation for atmospheric dynamics: Spherical geometry. *Quarterly Journal of the Royal Meteorological Society*, 150(764):4252–4269, July 2024.

[25] I. M. Navon. *Data Assimilation for Numerical Weather Prediction: A Review*, pages 21–65. Springer Berlin Heidelberg.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[27] W. K. Pratt. *Digital Image Processing: PIKS Scientific Inside*. Wiley, June 2006.

[28] F. Rabier. Overview of global data assimilation developments in numerical weather-prediction centres. *Quarterly Journal of the Royal Meteorological Society*, 131(613):3215–3233, Oct. 2005.

[29] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni. U-net and its variants for medical image segmentation: A review of theory and applications. *IEEE Access*, 9:82031–82057, 2021.

[30] A. Staniforth, T. Melvin, and C. Cotter. Analysis of a mixed finite-element pair proposed for an atmospheric dynamical core. *Quarterly Journal of the Royal Meteorological Society*, 139(674):1239–1254, Jan. 2013.

[31] J. Straka. *Cloud and Precipitation Microphysics: Principles and Parameterizations*. Cambridge University Press, 2009.

[32] P. A. Ullrich, C. Jablonowski, J. Kent, P. H. Lauritzen, R. Nair, K. A. Reed, C. M. Zarzycki, D. M. Hall, D. Dazlich, R. Heikes, C. Konor, D. Randall, T. Dubos, Y. Meurdesoif, X. Chen, L. Harris, C. Kühnlein, V. Lee, A. Qaddouri, C. Girard, M. Giorgetta, D. Reinert, J. Klemp, S.-H. Park, W. Skamarock, H. Miura, T. Ohno, R. Yoshida, R. Walko, A. Reinecke, and K. Viner. Dcmip2016: a review of non-hydrostatic dynamical core design and intercomparison of participating models. *Geoscientific Model Development*, 10(12):4477–4509, Dec. 2017.

[33] Z. Wang, J. Chen, and S. C. H. Hoi. Deep learning for image super-resolution: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3365–3387, Oct. 2021.

[34] P. Wesseling and C. Oosterlee. Geometric multigrid with applications to computational fluid dynamics. *Journal of Computational and Applied Mathematics*, 128(1–2):311–334, Mar. 2001.

A Discrete dynamical core

In this appendix, we shall discuss how the SWE (1) is discretised to generate our coarse and fine scale data. Spatially, we discretise with finite elements and temporally with finite differences. We subdivide the periodic interval $[0, 3)$ into partitions such that $0 := x_0 < x_1 < ... < x_M =: 3$ defining an element such as $I_m = (x_{m-1}, x_m)$. We define the continuous finite element space $\mathbb{V}_1$, to be the space of functions which are linear over each I_m and are globally continuous. We further define the discontinuous finite element space $\mathbb{V}_0$ as the space of functions which are constant over each I_m. With these spaces in mind, we spatially discretise (1) by seeking some $U, V \in \mathbb{V}_1$ and $P \in \mathbb{V}_0$ such that

$$\begin{aligned} \langle U_t - fV, \phi\rangle - \langle P, \phi_x\rangle &= 0 \qquad \forall \phi \in \mathbb{V}_1 \\ \langle V_t + fU, \psi\rangle &= 0 \qquad \forall \psi \in \mathbb{V}_1 \\ \langle P_t + gU_x, \chi\rangle &= 0 \qquad \forall \chi \in \mathbb{V}_0, \end{aligned} \tag{5}$$

where $\langle \cdot, \cdot \rangle$ denotes the spatial L_2 inner product over the periodic spatial domain. We note that this spatial discretisation is energy conserving, i.e.,

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[g\left\langle U, U\right\rangle + g\left\langle V, V\right\rangle + \left\langle P, P\right\rangle\right] = 0. \tag{6}$$

Temporally, we discretise with the method of lines. Let $U^0, V^0 \in \mathbb{V}_1$ and $P^0 \in \mathbb{V}_0$ be given (by the interpolant of the initial data into the finite element space), then we seek $U^1, V^1 \in \mathbb{V}_1$ and $P^1 \in \mathbb{V}_0$ such that

$$\begin{aligned} \left\langle \frac{U^1 - U^0}{\tau} - fV^{\frac{1}{2}}, \phi \right\rangle - \left\langle P^{\frac{1}{2}}, \phi_x \right\rangle &= 0 \qquad \forall \phi \in \mathbb{V}_1 \\ \left\langle \frac{V^1 - V^0}{\tau} + fU^{\frac{1}{2}}, \psi \right\rangle &= 0 \qquad \forall \psi \in \mathbb{V}_1 \\ \left\langle \frac{P^1 - P^0}{\tau} + gU_x^{\frac{1}{2}}, \chi \right\rangle &= 0 \qquad \forall \chi \in \mathbb{V}_0, \end{aligned} \tag{7}$$

where $U^{\frac{1}{2}} := \frac{1}{2}\left(U^1 + U^0\right)$ and $V^{\frac{1}{2}}, P^{\frac{1}{2}}$ are defined similarly. This fully discrete scheme has been designed such that energy is still preserved, i.e., that

$$\begin{aligned} g\left\langle U^1, U^1\right\rangle + g\left\langle V^1, V^1\right\rangle + \left\langle P^1, P^1\right\rangle = \\ g\left\langle U^0, U^0\right\rangle + g\left\langle V^0, V^0\right\rangle + \left\langle P^0, P^0\right\rangle, \end{aligned} \tag{8}$$

and guarantees numerical stability over long time by preserving a modified L_2 norm of the solution.