Supporting Information for "PseudospectralNet: Towards hybrid atmospheric models for climate simulations"

Maximilian Gelbrecht[1,2] *, Milan Klöwer[3,4], Niklas Boers[1,2,5]

[1]Earth System Modelling, School of Engineering & Design, Technical University Munich, Munich, Germany

[2]Potsdam Institute for Climate Impact Research, Potsdam, Germany

[3]Massachusetts Institute of Technology, Cambridge, MA, USA

[4]University of Oxford, Oxford, UK

[5]Department of Mathematics and Global Systems Institute, University of Exeter, Exeter, UK

Contents of this file

1. Model description of the QG3 model
2. Description of the Julia implementation QG3.jl
3. Description of the Julia implementation of PseudoSpectralNet
4. Description of the training procedure and hyperparameters used
5. Description of all baselines used in the study
6. Additional analysis of the kinetic energy in the main experiments

---

*maximilian.gelbrecht@tum.de

1. Quasigeostrophic Model

In this project we use the quasigeostrophic model of (Marshall & Molteni, 1993), both as part of the physics-based components of PseudospectralNet, and to simulate training and test data in 2. The model is formulated as a prognostic equation for quasigeostrophic potential vorticity $\mathbf{q}$ in the spherical harmonics space at the three pressure levels 200 hPa, 500 hPa and 800 hPa to discretizze the vertical. We use a triangular truncation at wavenumber 42 throughout this paper, and a corresponding 128x64 Gaussian grid in grid space. Its governing equation and its components with variables and parameters listed in Table S1 follow as

$$\frac{\partial q}{\partial t} = -J(\psi, q) - D(\psi, q) + F \tag{1}$$

with

$$J(\psi, q) = \frac{\partial \psi}{\partial \mu}\frac{\partial q}{\partial \lambda} - \frac{\partial \psi}{\partial \lambda}\frac{\partial q}{\partial \mu} \qquad \text{(advection)}$$

$$D(\psi, q) = TR(\psi) + H(q') + EK(\psi) \qquad \text{(dissipation)}$$

$$TR(\psi) = \begin{pmatrix} -\tau_R^{-1} R_1^{-2} (\psi_1 - \psi_2) \\ \tau_R^{-1} R_1^{-2} (\psi_1 - \psi_2) - \tau_R^{-1} R_2^{-2} (\psi_2 - \psi_3) \\ \tau_R^{-1} R_2^{-2} (\psi_2 - \psi_3) \end{pmatrix} \qquad \text{(temperature relexation)}$$

$$H(q') = \tau_H^{-1} a^8 \Delta^4 (l_{max}(l_{max} - 1))^{-4} q' \qquad \text{(horizontal diffusion)}$$

$$EK(\psi) = \begin{pmatrix} 0 \\ 0 \\ \nabla k \nabla \psi_3 + k \Delta \psi_3 \end{pmatrix} \qquad \text{(Ekman dissipation)}$$

$$C(\lambda, \phi) = \tau_E^{-1} (1 + \alpha_1 LS(\lambda, \phi) + \alpha_2 (1 - \exp(-h(\lambda, \phi)/1000m))) \qquad \text{(drag)}$$

$$f(\phi) = \begin{pmatrix} 2\Omega \sin \phi \\ 2\Omega \sin \phi \\ 2\Omega \sin \phi (1 + h/H_0) \end{pmatrix} \qquad \text{(Coriolis factor)}$$

and the transform between quasigeostrophic potential vorticity q and streamfunction ψ defined as

$$q_1 = \Delta\psi_1 - (\psi_1 - \psi_2)/R_1^2 + f_1 \tag{2}$$

$$q_2 = \Delta\psi_2 + (\psi_1 - \psi_2)/R_1^2 - (\psi_2 - \psi_3)/R_2^2 + f_2 \tag{3}$$

$$q_3 = \Delta\psi_3 + (\psi_2 - \psi_3)/R_2^2 + f_3 \tag{4}$$

or short

$$q = T_{\psi q}\psi \tag{5}$$

and the time-dependent part of the potential vorticity as

$$q' = q - F. \tag{6}$$

The forcing F used here is computed according to (Roads, 1987; Corti et al., 1997), so that for a climatological ensemble the average potential vorticity tendencies vanish. With $\psi =< \psi >_t + \psi' = \psi_c + \psi'$, these are computed as

$$S = J(\psi_c, q_c) + < J(\psi, q') >_t + D(\psi_c) \tag{7}$$

with streamfunction data computed from ERA5 reanalysis (Hersbach et al., 2020). We restricted this data to the winter months, in accordance with (Marshall & Molteni, 1993). With such a constant forcing the model therefore also doesn't exhibit seasonality.

In order to simplify the models computations, during the QG3 computations we use a unit system so that $a = 1$ and $2\Omega = 1$. Units for all other variables follow accordingly from this.

2. QG3.jl

The QG3 model used throughout this project was implemented in the Julia language (Bezanson et al., 2017) to be differentiable by reverse-mode automatic differentiation such as Zygote.jl (Innes, 2019) and to run on GPUs and CPUs in the package QG3.jl (Gelbrecht, 2024). The model, and most importantly the transforms and derivatives used by it, were implemented to be computed pseudospectrally using a Gaussian grid. The Gaussian grid uses Gaussian latitudes, meaning latitudes on the zeros of the associated Legendre polynomials, and equiangle-spaced longitudes. As the neural networks generally need real-valued inputs, we also use real spherical harmonics with degree $l \in [0, l_{max}]$ and order $m \in [-l; l]$ defined as

$$Y_{l,m}(\mu,\lambda) = (-1)^m \sqrt{\frac{2l+1}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}} P_l^{|m|}(\mu) \begin{cases} \sqrt{2}\sin(|m|\lambda) & \text{if } m < 0 \\ 1 & \text{if } m = 0 \\ \sqrt{2}\cos(m\lambda) & \text{if } m > 0 \end{cases} \tag{8}$$

with associated Legendre polynomials $P_l^m(x)$. The transforms are defined as

$$f_{l,m} = \frac{1}{4\pi}\int_{-1}^{1}\int_{0}^{2\pi} f(\mu,\lambda) Y_{l,m}(\theta,\lambda) d\mu d\lambda = \mathcal{S}\{f(\mu,\lambda)\} \tag{9}$$

$$f(\mu,\lambda) = \sum_{l=0}^{l_{max}} \sum_{m=-l}^{l} f_{l,m} Y_{l,m}(\mu,\lambda) = \mathcal{S}^{-1}\{f_{l,m}\}. \tag{10}$$

In order to achieve differentiability and GPU compatibility the model was mostly written in matrix form, and avoiding inplace mutation of the variables. For example, transforms and meridional derivatives are solved by a pre-planned real-to-real FFT and Legendre-Gauss quadrature via a matrix multiplication with pre-computed associated Legendre Polynomials (or their derivatives) as follows

$$f_{l,m} = (\mathrm{FFT}_\lambda \cdot f(\mu, \lambda)) \times (wP)_{lm,\mu_i} \tag{11}$$

$$f(\mu, \lambda) = \mathrm{iFFT}_\lambda \cdot (f_{l,m} \times P_{lm,\mu_i}) \tag{12}$$

$$(\partial_\mu f_{l,m})(\mu, \lambda) = \mathrm{iFFT}_\lambda \cdot (f_{l,m} \times dP_{lm,\mu_i}), \tag{13}$$

with FFT_λ and iFFT_λ as fast Fourier transform plans along the longitude, $(wP)_{lm,\mu_i}$ as pre-computed quadrature weights and Legendre Polynomials, P_{lm,μ_i} as pre-computed Legendre Polynomials and dP_{lm,μ_i} as pre-computed derivatives of the Legendre Polynomials. The multiplication $\times$ is along the latitude dimension in case of the forward transform to $f_{l,m}$ (Eq. 11) and along the l-dimension in case of the inverse transform to $f(\mu, \lambda)$ (Eq. 12). All nonlinearieties in the models are computed with the pseudospectral approach by element-wise multiplication in grid space.

The remaining derivative operators follow accordingly to the known relations of real spherical harmonic functions as

$$\Delta f_{l,m} = -l(l+1)/a^2 f_{l,m} \tag{14}$$

$$\partial_\lambda f_{l,m} = -m f_{l(-m)} \tag{15}$$

with gridded versions of those derivatives computed by applying the previously defined transform afterwards.

The correctness of the gradients, computed with Zygote.jl by reverse-mode automatic differentiation of the transforms and all model components, is verified by cross checking them with a finite difference forward differentiation. The library QG3.jl itself does not include any differential equation solvers. Instead, it defines all necessary functions to

define the right-hand side Eq. 1, which can be used with differential equations solvers from DifferentialEquations.jl (Rackauckas & Nie, 2017).

3. NeuralQG3.jl

The Julia package NeuralQG3.jl (Gelbrecht, 2025) builts upon QG3.jl to define the PseudospectralNet proposed in this paper and to run all experiments presented in this paper. It uses Lux.jl (Pal, 2023) to built the model and its ANN components, and otherwise many parts of the Flux.jl ecosystem (Innes, 2018).

The gradients of the one-step-ahead error ($N_{max} = 1$ in Eq. 3 in the main text) are computed directly via AD (discritize-then-optimize). In order to do so, we implemented a single step 4th order Runge-Kutta solver to reduce any possible overhead, increase performance, and make it easily GPU compatible and usable with batches of trajectories and input data. Gradients of longer trajectories ($N_{max} > 1$) are then computed via adjoint sensitivity analysis with SciMLSensitivity.jl (Rackauckas et al., 2021; Ma et al., 2021), which is significantly slower than our own implementation for the one-step-ahead error.

3.1. Knowledge Layers

The process-based layers $\mathcal{K}_{l,m}$ and $\mathcal{K}(x)$ incorporate the phyiscal model components of the QG3 Model. $\mathcal{K}_{l,m}$ computes those with output in the spherical harmonics space, and $\mathcal{K}$ those with output in the grid space. Table 2 in the main text gives an overview of all the processes considered. The rationale behind the selection of those processes was to give the hybrid model access to all individual parts and components of the QG3. The components computed online can be computationally costly and are only used when

absolutely needed for the differential operators. The pre-computed static components on the other hand consume hardly any additional computational power, which is why we add additional fields here much more liberally. The layers take the potential vorticity and stream function, concatenated along the dimension of the horizontal level, as an input, and concatenate the output of all process-based components along this dimension as well to form the output of the layer. This dimension that the outputs are concatenated along also serves as the channel dimension for the subsequent neural networks. So, for example, the $\mathcal{K}_{l,m}$ layer for the QG3 experiment follows as

$$\mathbf{K}_{l,m}(q,\psi) = \begin{pmatrix} q \\ \psi \\ \Delta q \\ \Delta\psi \\ \partial_\lambda q \\ \partial_\lambda \psi \\ l \\ m \end{pmatrix}. \tag{16}$$

All parameters of all process-based components can be either kept fixed or made trainable as well. This therefore also allows to re-calibrate those components during training. However, in this paper we keep all parameters fixed with the values presented in Table S1.

3.2. ANN Details

Next to the two process-based layers $\mathcal{K}_{l,m}$ and $\mathcal{K}(x)$, PseudospectralNet consists of three different ANNs, as also shown e.g in Fig. 1 in the main text. The networks NN_{SH} and NN_{grid} are both defined point-wise, meaning they fit a function $\mathbb{R}^{k_{in}} \to \mathbb{R}^{k_{out}}$ with k_{in} denoting the number of input channels, which correspond to the number of output channels of $\mathcal{K}_{l,m}$ and $\mathcal{K}(x)$, respectively. Both NN_{SH} and NN_{grid} are multi-layer perceptrons, i.e. fully connected feed forward networks, with their hyperparameters listed in Table S2. NN_{SH} has an additional element-wise linear transformation layer as its last layer with

different, learnable parameters for each input (and therefore also for each spectral coefficient). The UNet largely follows the original UNet design of (Ronneberger et al., 2015). Our implementation features four levels with upscaling between them through transpose convolution and downscaling through strided convolution. Three convolutional layers were used for each convolutional block per level. The UNet performs 2D convolutions on the grid space with the vertical levels incorporated into the channel dimension as reported in the previous section 3.1. Appropriate periodic padding along the longitude dimension is applied. Exact hyperparameters are given in Table S2 as well. Each of the ANNs is preempted by a Z-normalization that normalizes each channel separately. This is needed as some of the process-based components have very different scales.

3.3. Hyperparameter Search

The hyperparameters of PseudoSpectralNet were tuned by running a hyperparameter search with randomly selected parameters. The hyperparameters that we tuned were the number of channels in the UNet, the kernel size in the UNet ($(2, 2)$ and $(3, 3)$ considered), the number of layers N_{layers} and nodes per layer in the MLPs N_{Nodes}, the activation function (Swish and ReLu were considered). 50 different randomly drawn configurations were tested and the best is reported here. Overall results only differed within a small margin for most of these hyperparameters. We do however note that PSNs with ReLU activaction function were not stable for very long integrations, whereas those with Swish were (as reported in the main text).

4. Baselines

Several different baselines are used to compare the PseudospectralNet results against. As a basic baseline we use the climatology of the training data q_{data}

$$q_{n+1} =< q_{data} >_t . \tag{Clim}$$

Then, we have as process-based baselines the QG3 advection only, therefore,

$$\dot{q} = -J(\psi, q) \tag{J}$$

and the full QG3 Model

$$\dot{q} = -J(\psi, q) - D(\psi, q) + S. \tag{QG3}$$

Then, we have two purely data-driven baselines. Both are built around a UNet as used by PseudospectralNet. While Grid UNet and PS UNet share the hyperparameters and exact architecture with the UNet used in PseudospectralNet (see Table S2, they are trained independently. The baseline *Grid UNet* is set up in grid space as a RNN, so that

$$q_{n+1}(\mathbf{x}) = \mathrm{UNet}(q_n(\mathbf{x})). \tag{Grid UNet}$$

Grid UNet is trained by minimizing the one step ahead mean square error. The baseline *PS UNet* is set up as a PseudospectralNet without its process-based components, so that

$$\dot{q} = NN_{SH}(\mathcal{S}\{NN_{grid}(\mathrm{UNet}(q(\mathbf{x}, t),)\})). \tag{PS UNet}$$

PS UNet is trained by minimizing the same error as for the PseudospectralNet (Eq. 3 in the main text).

5. Angular Power Spectra

To investigate the stability of the trained models further, we compute the angular power of PseudoSpectralNet and the PS UNet baseline over a long roll out. We compute the

angular energy spectra averaged over a 4-year-window of a trajectory, then evolve the state for four years and repeat this procedure to compute the spectra of 10 windows over a time period of 80 y total. We compute the spectra S of the vorticity q_i at the i-th level at wavenumber l as

$$S_i(l) = \frac{1}{2l+1} \sum_{m=-l}^{+l} |q_{i,l,m}|^2. \tag{17}$$

Fig. S1 shows the power spectra analysis of the QG3 experiment. The spectra of a 60 year long trajectory was computed in 3-year-windows. The results for PseudoSpectralNet in the top row show that spectra remains stable as the individual lines are indistinguishable. In the bottom row for PS UNet baseline on the other hand we see a trend in the spectra that will ultimately also result in numerical instability.

In Fig. S2 we analyse the power spectra of the SpeedyWeather.jl primitive equation experiment. Here, the spectra of the PseudoSpectralNet show a small offset to higher tails after the initial window and remain relatively stable further in the integration but do a exhibit a small trend afterwards. Compared to that the spectra PS UNet are never stable and the model also becomes quickly numerically unstable and diverges so that the integration had to stop.

6. Average Kinetic Energy

The QG3 model doesn't directly exhibit invariants as a forced-dissipative system. However, we can still compute and track the average kinetic energy of the system. While the energy does fluctuate considerably, it has to stay bounded over the course of long simulations. The kinetic energy of the i-th layer

$$E^i_{kin}(t) = \frac{1}{2} \int \int \left(u^2(\lambda, \mu, i, t) + v^2(i, \lambda, \mu, i, t) \right) \mathrm{d}\lambda \mathrm{d}\mu \tag{18}$$

is computed in the spectral domain as

$$E^i_{kin}(t) = \frac{1}{2} \left(u^2_{0,0}(i, t) + v^2_{0,0}(i, t) \right) \tag{19}$$

with the meridional and zonal wind

$$u = -\frac{\partial \psi}{\partial \phi} \tag{20}$$

$$v = \frac{1}{a \cos \phi} \frac{\partial \psi}{\partial \lambda}. \tag{21}$$

Figure S3 shows the kinetic energy of the QG3 experiment shown in 2. The PseudospectralNet slightly overestimates the energy of the ground truth, but stays consistent for the whole time span of the simulation. The PS UNet baseline, while not shown in the plot, slightly underestimates the energy of the ground truth for several years of integration, until blowing up very rapidly within a few time steps.

Figure S4 shows the kinetic energy of the SpeedyWeather experiment. The PSN slowly builds up kinetic energy in every time step. While this process is comparably slow for the PSN, over thousands of time steps, the PS UNet baseline blows up rapidly after 400 discrete integration time steps (about two weeks). In future work, considering explicit energy constraint, e.g. via an added stabilization (White et al., 2024), therefore seems like an natural next step to improve the performance of PSN.

Table S1. List of variables and parameters used in the QG3 model

Symbol	Name	Value / data source
λ	longitude	
ϕ	latitude	
θ	colatitude	
μ	sine of latitude	$\sin(\phi) = \cos(\theta)$
l_{max}	maximum wave number	42
R_1	Rossby deformation radius	700 km
R_2	Rossby deformation radius	450 km
h	orography	ERA5 data used
LS	land-sea mask	ERA5 data used, $LS \in [0, 1]$
H_0	orography scale constant	9 km
F	forcing	computed with ERA5 data according to (Roads, 1987; Corti et al., 1997)
τ_H	horizontal damping time scale	2 days
τ_E	Ekman time scale	3 days
a	Earth's radius	6371 km = 1 (model units)
Ω	Earths' angular velocity	$2\pi/24h = 1/2$(model units)

Table S2. Hyperparameters of ANNs used in the experiments

NN_{Grid}	NN_{SH}	UNet	Optimisation
$N_{\text{Nodes}} = 50$	$N_{\text{Nodes}} = 50$	Channels: $[24, 48, 96, 96]$	AdamW
$N_{\text{Layers}} = 3$	$N_{\text{Layers}} = 3$	Kernel: $(3, 3)$	$\eta_0 = 10^{-3}$
Swish activation	Swish activation	Swish activation	Decayed sinusoidal η
		Upsampling: transpose conv.	scheduling
		Downsampling: strided conv.	$\gamma = 10^{-6}$
		Output channels: 6	max. trajectory
		Conv layers per block: 3	length: $N_{max} = 6$

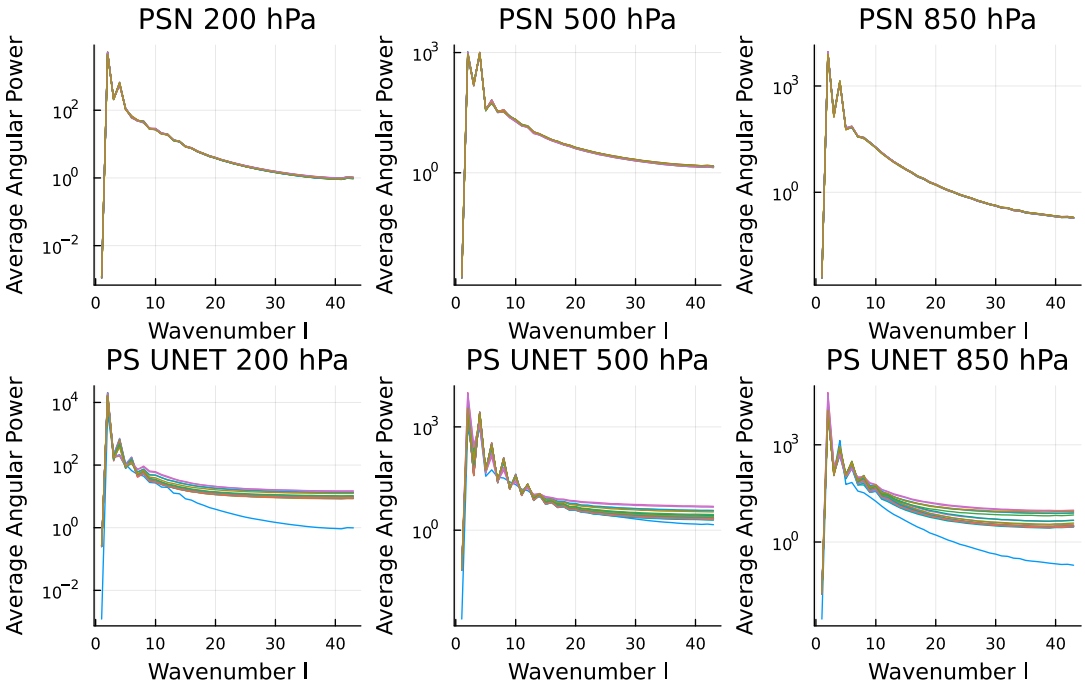

Figure S1. Angular Power Spectra of the application on the QG3 data. The top row shows the spectra of the three layers of PseudoSpectralNet, the bottom row the spectra of the PS UNet baseline

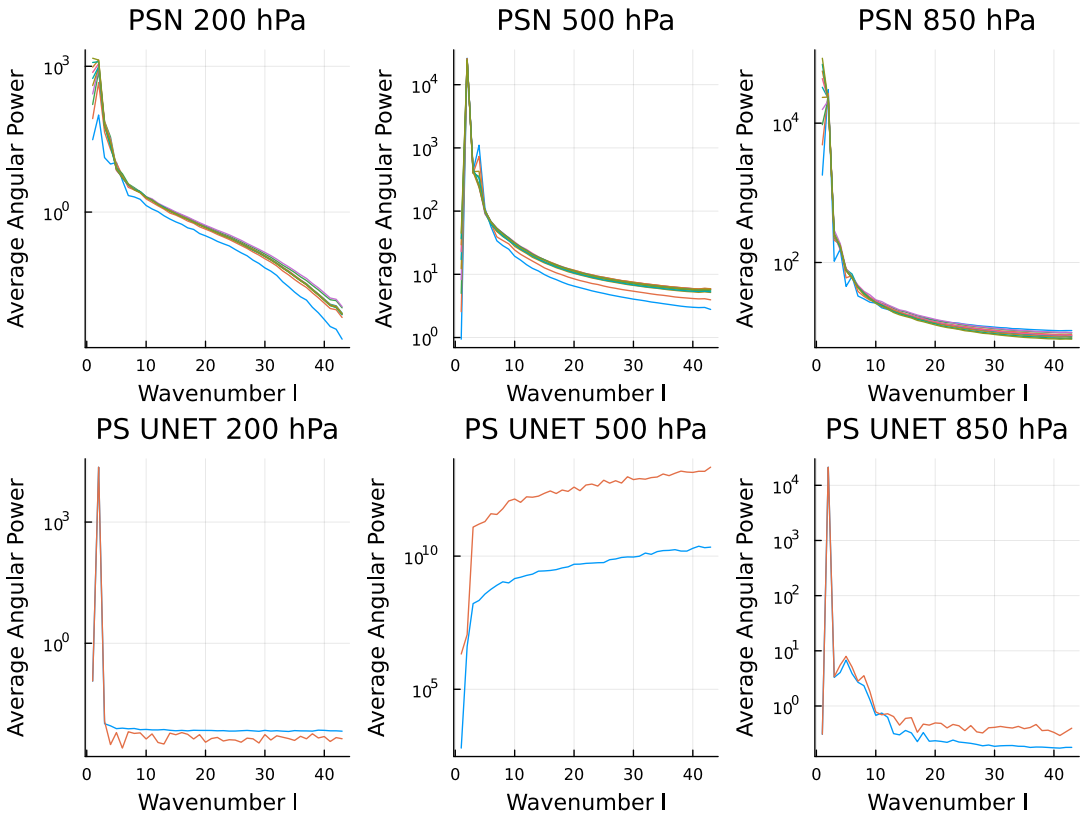

Figure S2. Angular Power Spectra of the application on the SpeedyWeater.jl primitive equation data. The top row shows the spectra of the three layers of PseudoSpectralNet, the bottom row the spectra of the PS UNet baseline

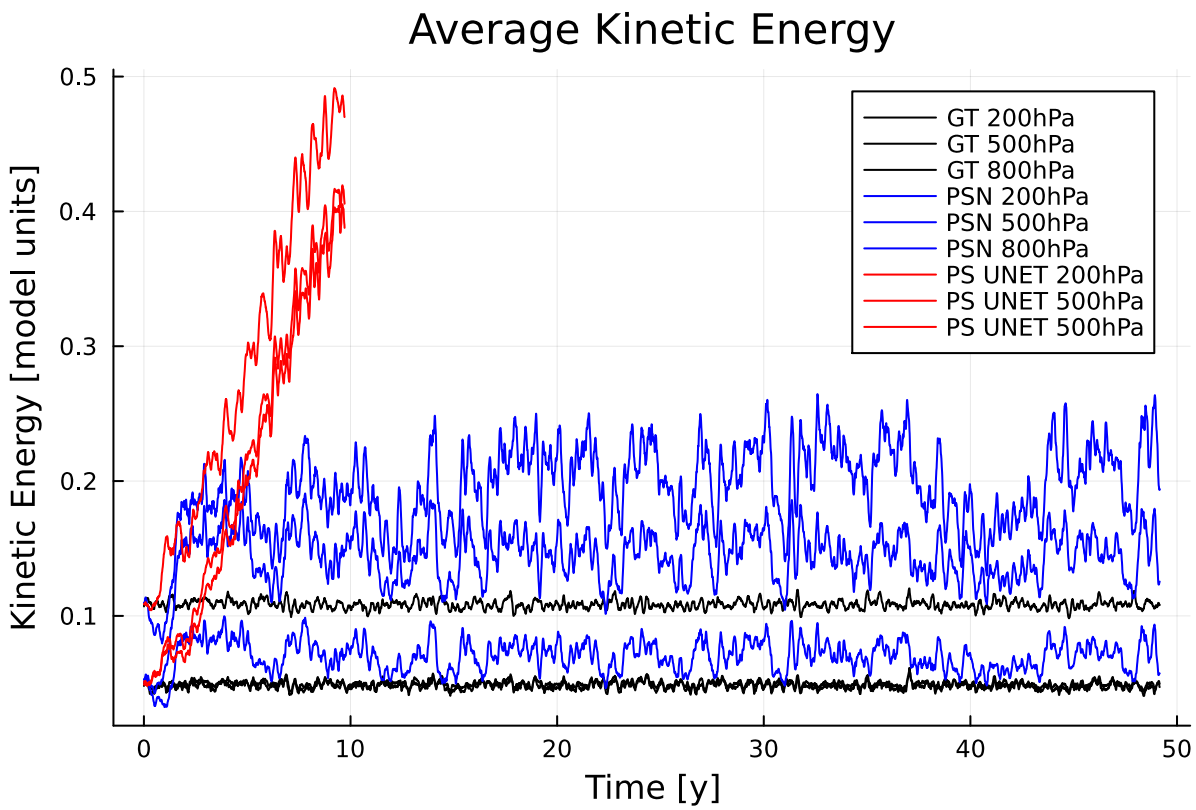

Figure S3. Kinetic Energy in the QG3 experiment for the ground truth (GT) and PseudoSpectralNet (PSN). **A** Temporal evolution of the kinetic energy in each layer over a 50 year integration. A 6-day running mean is applied to the energy data in the temporal plot. **B-D** Distribution of the kinetic energy in each layer over a 50 year integration (without the running mean applied).

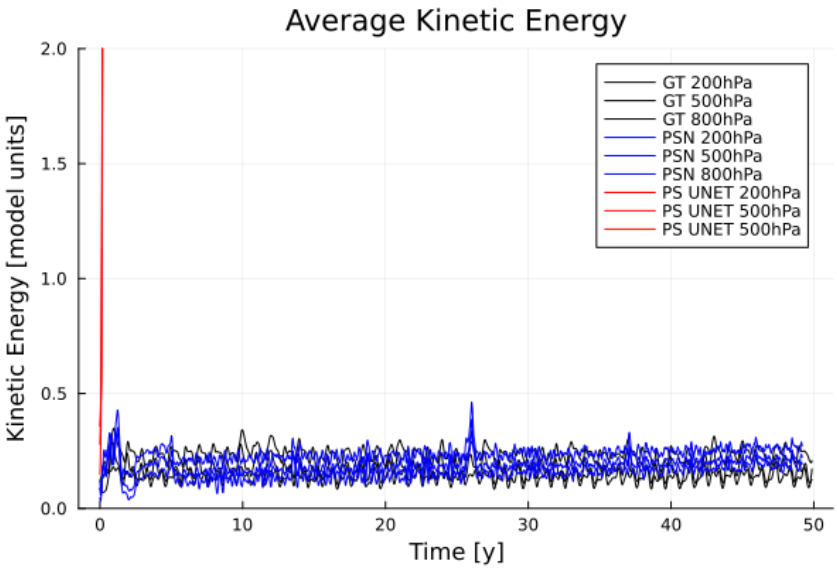

Figure S4. Average Kinetic Energy in the SpeedyWeather experiment for the ground truth (GT) PseudoSpectralNet (PSN) and the PS UNet baseline. The ground truth energy is non-zero, but stays close to 0.07 model units for the whole integration.

References

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65-98. Retrieved from https://doi.org/10.1137/141000671 doi: 10.1137/141000671

Corti, S., Giannini, A., Tibaldi, S., & Molteni, F. (1997). Patterns of low-frequency variability in a three-level quasi-geostrophic model. *Climate Dynamics*, *13*(12), 883–904. Retrieved from https://doi.org/10.1007/s003820050203 doi: 10.1007/s003820050203

Gelbrecht, M. (2024, December). *Qg3.jl.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.14547916 doi: 10.5281/zenodo.14547916

Gelbrecht, M. (2025, January). *maximilian-gelbrecht/neuralqg3.jl: Initial paper release.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.14711058 doi: 10.5281/zenodo.14711058

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., ... Thépaut, J.-N. (2020). The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, *146*(730), 1999-2049. Retrieved from https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.3803 doi: https://doi.org/10.1002/qj.3803

Innes, M. (2018). Flux: Elegant machine learning with julia. *Journal of Open Source Software*, *3*(25), 602. Retrieved from https://doi.org/10.21105/joss.00602 doi: 10.21105/joss.00602

Innes, M. (2019). *Don't unroll adjoint: Differentiating ssa-form programs.*

Ma, Y., Dixit, V., Innes, M. J., Guo, X., & Rackauckas, C. (2021). A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions. In *2021 ieee high performance extreme computing conference (hpec)* (p. 1-9). doi:

10.1109/HPEC49654.2021.9622796

Marshall, J., & Molteni, F. (1993). Toward a dynamical understanding of planetary-scale flow regimes. *Journal of Atmospheric Sciences*, *50*(12), 1792 - 1818. Retrieved from https://journals.ametsoc.org/view/journals/atsc/50/12/1520-0469_1993_050_1792_taduop_2_0_co_2.xml doi: 10.1175/1520-0469(1993)050⟨1792:TADUOP⟩2.0.CO;2

Pal, A. (2023, April). *Lux: Explicit Parameterization of Deep Neural Networks in Julia.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.7808904 (If you use this software, please cite it as below.) doi: 10.5281/zenodo.7808904

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., ... Edelman, A. (2021). *Universal differential equations for scientific machine learning.*

Rackauckas, C., & Nie, Q. (2017). DifferentialEquations.jl–a performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software*, *5*(1).

Roads, J. O. (1987). Predictability in the extended range. *Journal of Atmospheric Sciences*, *44*(23), 3495 - 3527. Retrieved from https://journals.ametsoc.org/view/journals/atsc/44/23/1520-0469_1987_044_3495_piter_2_0_co_2.xml doi: 10.1175/1520-0469(1987)044⟨3495:PITER⟩2.0.CO;2

Ronneberger, O., P.Fischer, & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention (miccai)* (Vol. 9351, pp. 234–241). Springer. Retrieved from http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a ((available on arXiv:1505.04597 [cs.CV]))

White, A., Kilbertus, N., Gelbrecht, M., & Boers, N. (2024). *Stabilized neural differential equations for learning dynamics with explicit constraints.*