A RELATED CONCEPTS OF DIFFERENTIAL FORMS

Differential forms aim to provide a unified approach to define integrands. In this section, we briefly review a few related mathematical concepts in the paper, which can be found in Richter-Powell et al. (2022); Do Carmo (1998); Weintraub (2014).

Definition A.1. *Let k be a non-negative integer and $\mu(\mathbf{x}) = f(x_1, \cdots, x_p)$ be a smooth function on $\mathbb{R}^p$, a monomial $k-$form on $\mathbb{R}^p$ is an expression $\mu\, dx_{i_1} \wedge \cdots \wedge dx_{i_k} = \mu dx_I$, where $I := \{i_1, \cdots, i_k$. A $k-$form is a sum of monomial $k-$forms, which will be denoted as*

$$\mu = \sum_{1 \le i_1, i_2, \cdots, i_k \le p} \mu_{(i_1, \cdots, i_k)} dx_{i_1} \wedge \cdots \wedge dx_{i_k} := \sum_I \mu_I dx_I.$$

Definition A.2. *The wedge product, $\wedge$, is an associative operation on differential forms. When ω is a $k-$form and η is a $l-$form, $\omega \wedge \eta$ is a $(k+l)-$form. Moreover, $\wedge$ satisfies the following properties:*

- *(Distribution Law) $(\omega_1 + \omega_2) \wedge \eta = \omega_1 \wedge \eta + \omega_1 \wedge \eta$, $\omega \wedge (\eta_1 + \eta_2) = \omega \wedge \eta_1 + \omega \wedge \eta_2$.*
- *(Associative Law) $(f\omega) \wedge \eta = f(\omega \wedge \eta)$.*
- *(Skew Symmetry) $\eta \wedge \omega = -\omega \wedge \eta$.*

Here ω, ω_1, ω_2 are $k-$forms, η, η_1, η_2 are $l-$forms, and f is a function.

Definition A.3. *The exterior derivative (or differential) operator d maps a $k-$form to a $(k+1)-$form. In particular, given a $k-$form $\mu = \sum_{1 \le i_1 < i_2 < \cdots < i_k \le p} \mu_{(i_1, \cdots, i_k)} dx_{i_1} \wedge \cdots \wedge dx_{i_k}$, one has*

$$\begin{aligned} d\mu &= \sum_{1 \le i_1, i_2, \cdots, i_k \le p} d\mu_{(i_1, \cdots, i_k)} dx_{i_1} \wedge \cdots \wedge dx_{i_k} \\ &= \sum_{1 \le i_1, i_2, \cdots, i_k \le p} \sum_{1 \le l \le p} \frac{\partial \mu_{(i_1, \cdots, i_k)}}{\partial dx_l} dx_l \wedge dx_{i_1} \wedge \cdots \wedge dx_{i_k}. \end{aligned}$$

Moreover, the following properties hold for the exterior derivative operator:

- *For each function f, $ddf = 0$, and $df \wedge df = 0$.*
- *For each $k-$form ω, $dd\omega = 0$.*
- *For each function f and $k-$form ω, $d(f\omega) = df \wedge \omega + f d\omega$.*

Definition A.4. *The Hodge $\star$-operator on $\mathbb{R}^p$ is a function that takes a $k-$form to a $(p-k)-$form defined as follows:*

- *Let $I = \{i_1, \cdots, i_k\}$ be an ordered subset of $\{1, \cdots, p\}$ and $I^c = \{j_1, \cdots, j_{p-k}\}$ be its complement, ordered so that $dx_{i_1} \wedge \cdots dx_{i_k} \wedge dx_{j_1} \wedge \cdots dx_{j_{p-k}} = dx_1 \wedge \cdots \wedge dx_p$, then*

$$\star dx_I = \star(dx_{i_1} \wedge \cdots dx_{i_k}) := dx_{j_1} \wedge \cdots dx_{j_{p-k}} = dx_{I^c}.$$

- *Given any $k-$form $\mu = \sum_{1 \le i_1, i_2, \cdots, i_k \le p} \mu_{(i_1, \cdots, i_k)} dx_{i_1} \wedge \cdots \wedge dx_{i_k}$, then*

$$\star\mu := \sum_{1 \le i_1, i_2, \cdots, i_k \le p} \mu_{(i_1, \cdots, i_k)} \star (dx_{i_1} \wedge \cdots \wedge dx_{i_k}).$$

Then, the following property holds for any $k-$form μ:

$$\star \star \mu = (-1)^{k(p-k)} \mu.$$

Given a vector-valued function on $\mathbb{R}^p$ $\mathbf{u}(\mathbf{x}) = [u_1(\mathbf{x}), \cdots, u_p(\mathbf{x})]$, its divergence $\text{div}(\mathbf{u}) = \sum_{i=1}^p \frac{\partial u_i}{\partial x_i}$ can be defined via exterior derivative:

$$\begin{aligned} d \star \sum_{i=1}^p u_i dx_i =& d \sum_{i=1}^p u_i \star dx_i = d \sum_{i=1}^p (-1)^{i-1} u_i dx_{\{1,\cdots,n\}\setminus i} \\ =& \sum_{i=1}^p (-1)^{i-1} du_i \wedge dx_{\{1,\cdots,n\}\setminus i} = \sum_{i=1}^p \frac{\partial u_i}{\partial x_i} dx_{\{1,\cdots,n\}}. \end{aligned}$$

Therefore, one can denote the divergence of $\mathbf{u}$ as $d \star \mathbf{u}$. On the other hand, combining the fundamental properties of $\star$ and d yields $\text{div}(\star d\mu) = d \star \star d\mu = (-1)^{k(p-k)} dd\mu = 0$ for any $k-$form μ. Therefore, when $\mathbf{u} = \star d\mu$, where μ is a $(p-2)-$form, it is guaranteed to be divergence free. Since any $(p-2)-$form μ can be represented as

$$\mu = \sum_{1 \le i_1, \cdots, i_{p-2} \le p} \mu_{(i_1,\cdots,i_{p-2})} dx_{i_1} \wedge \cdots \wedge dx_{i_{p-2}} = \sum_{1 \le i,j \le p} \mu_{(i,j)} \star (dx_i \wedge dx_j),$$

where $\mu_{(i,j)} := (-1)^{\sigma(\{i,j\},\{i_1,\cdots,i_{p-2}\})} \mu_{(i_1,\cdots,i_{p-2})}$ when $\{i,j\} \cup \{i_1,\cdots,i_{p-2}\} = \{1,\cdots,p\}$. And note that the skew symmetry property of $\wedge$ yields $\mu_{(i,j)} := \mu_{ij} = -\mu_{ji}$. We further expand the representation of $\star d\mu$ as:

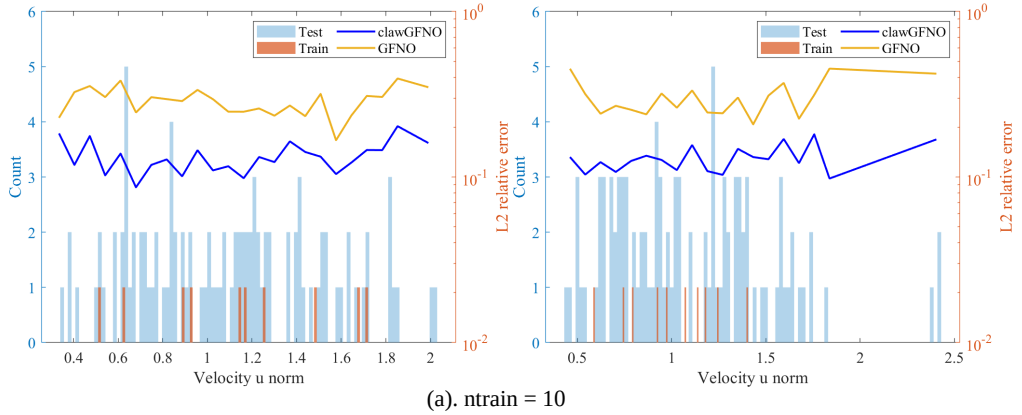
$$\begin{aligned} \star d\mu =& \star \sum_{1 \le i,j \le p} d\mu_{(i,j)} \star (dx_i \wedge dx_j) \\ =& \star \left(\sum_{1 \le i,j \le p} \frac{\partial \mu_{ij}}{\partial x_i} dx_i \wedge \star (dx_i \wedge dx_j) - \frac{\partial \mu_{ji}}{\partial x_j} dx_j \wedge \star (dx_i \wedge dx_j) \right) \\ =& \star \left(\sum_{1 \le i,j \le p} \frac{\partial \mu_{ij}}{\partial x_i} dx_i \wedge \star (dx_i \wedge dx_j) + \frac{\partial \mu_{ij}}{\partial x_i} dx_i \wedge \star (dx_i \wedge dx_j) \right) \\ =& 2 \star \sum_{1 \le i,j \le p} \frac{\partial \mu_{ij}}{\partial x_i} dx_i \wedge \star (dx_i \wedge dx_j) = 2 \star \sum_{1 \le i,j \le p} \frac{\partial \mu_{ij}}{\partial x_i} \star dx_j \\ =& 2 \sum_{1 \le i,j \le p} \frac{\partial \mu_{ij}}{\partial x_i} \star \star dx_j = (-1)^{p-1} 2 \sum_j \left(\sum_i \frac{\partial \mu_{ij}}{\partial x_i} \right) dx_j. \end{aligned}$$

That means, taking any $\mu := [\mu_{ij}(\mathbf{x})]$ satisfies $\mu_{ij} = -\mu_{ji}$ (a p by p skew-symmetric matrix-valued function), we have $\text{div}(\star d\mu) = 0$, where $\star d\mu = [\text{div}(\mu_1), \cdots, \text{div}(\mu_p)]^T$ and μ_i stands for the $i-$th row of μ.

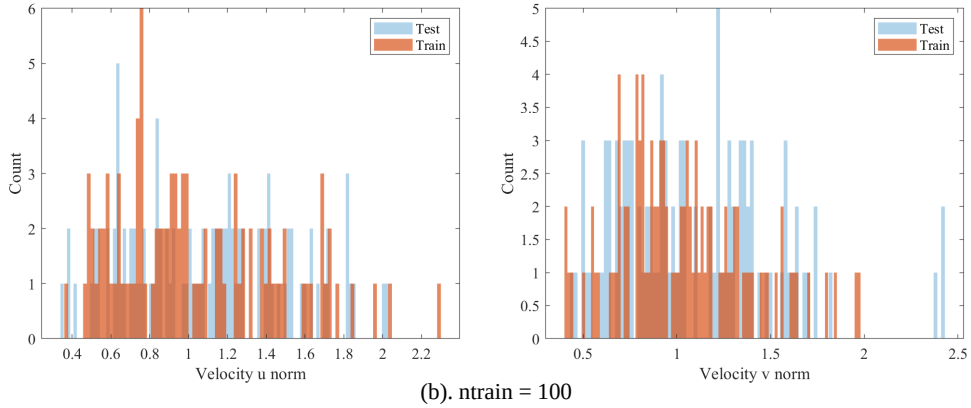
B Data generation and training strategies

B.1 Example 1 – incompressible Navier-Stokes equation

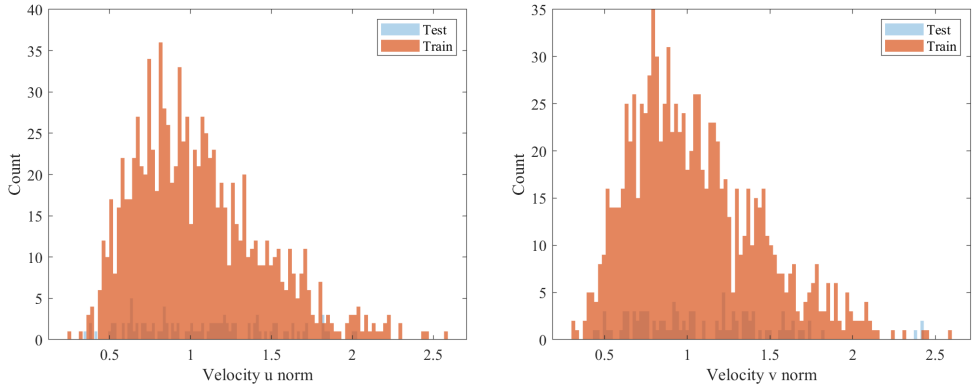
For the two case study datasets in the incompressible Navier-Stokes example, we generate a total of 1,200 samples using the pseudo-spectral Crank-Nicolson solver available in Li et al. (2020c). We then split the generated dataset into 1,000, 100 and 100 for training, validation and testing, respectively. A histogram demonstration of the dataset distributions in small, medium and large data regimes is illustrated in Figure 3, where the test set of the small dataset of ntrain=10 samples exhibits a wider data distribution compared to its training dataset, which is aimed to test the data efficiency and out-of-distribution performances of the learned models. The fluid viscosity employed in the physics solver is $\nu = 10^{-4}$, and the timestep size is $\Delta t = 10^{-4}$ s. The solutions are obtained on a 256×256 spatial grid and the total duration of the simulation is 24 s. The obtained solutions are then downsampled to a $64 \times 64 \times 30$ grid, with the 3rd dimension being the temporal dimension. Note that, in downsampling the spatial dimensions, we employ a 2×2 mean pooling. This strategy is suggested in Helwig et al. (2023) to mitigate the spurious numerical errors not existed in the original data.



(a). ntrain = 10



(b). ntrain = 100



(c). ntrain = 1000

Figure 3: The data distribution of velocity in L2 norm, in the incompressible Navier-Stokes dataset. Left: the x component of velocity. Right: the y component of the velocity. Cases (a)-(c) represent the histogram of sample distributions in the small, medium and large data regimes, respectively, with blue representing the test samples and orange for the training samples. The per-sample relative L2 error on the test set is also plotted in (a), comparing clawGFNO (in navy) with GFNL (in yellow). This result demonstrates the improved the accuracy of clawNO, comparing to its counterpart, in small data regime.

To provide more details on the model performance, we plot the per-time-step prediction errors in terms of L2 relative error on the test dataset in Figure 4 using the best models trained with ntrain=1000 samples. Perhaps unsurprisingly, the prediction error increases as the prediction time step grows, due to the accumulation of error. All models have a similar growth rate, while clawNOs,

together with FNO, significantly outperform other baselines in accuracy. We also list a number of performance metrics in Table 5, including the total number of model parameters, the per-epoch runtime, the inference time, as well as the peak GPU usage. To quantitatively evaluate the divergence of the predicted solutions, we compute the averaged L2-norm of the divergence on the test dataset for all the models trained with ntrain=1000 samples (cf. the last row of Table 5). Because the additional layer in clawNOs are with pre-calculated weights, it barely adds any extra burden into GPU memory compared with its NO base model. Similar observation also applies to the runtime in large data regime. The inference time has a minor increase, due to the fact that the inferred solution will go through the an additional layer which avoidably increases the computational cost. When comparing the divergence of predicted solutions, we can see that both clawNOs predictions have much smaller divergence compared with all baselines. However, we point out that the solution divergence from clawNOs is not exactly zero, due to the numerical errors as discussed in Theorem 3.1.

Remark: The complexity of our clawNOs is very simular to their NO counterparts. Taking clawFNOs for example, the trainable part of the clawNO model consists of two fixed-size MLPs for the lifting layer and the projection layer, and L numbers of Fourier layers between them. Denote d_u as the input function dimension, H as the latent dimension after lifting, M as the total number of grids, m as the number of Fourier modes on each dimension after truncation (which is often taken as half of the number of grids in each dimension, $M^{1/p}$, in practice), and p as the problem dimension. During the lifting layer a vector valued input function taking values in $\mathbb{R}^{d_u}$ is linearly and locally mapped to the first layer feature function $\mathbf{h}(\cdot, 0)$ taking values in $\mathbb{R}^H$, and hence the number of trainable parameters is $Hd_u + H$. Then, each iterative Fourier layers involves the integral with a trainable kernel weight matrix in the Fourier domain, which is of size $2H^2m^p = 2^{1-p}H^2M$, and a local linear transformation which involves $H^2 + H$ numbers of trainable parameters. Then, the last iterative layer feature function, $\mathbf{h}(\cdot, L)$, is projected to the skew symmetric matrix-valued function μ with a two-layer MLP. Since the skew symmetric matrix-valued function μ is of degree of freedom $p(p-1)/2$ at each point $\mathbf{x}$, the projection layer maps a size H input vector to a size $p(p-1)/2$ vector. Assume that the hidden layer of this MLP is of d_Q neurons, the total number of trainable parameters in projection layer will be $Hd_Q + d_Q + d_Qp(p-1)/2 + p(p-1)/2$. Finally, μ will go through the pre-calculated differentiation layer, with $p^2(p-1)M^2/2 = p^2(p-1)m^{2p}/2$ numbers of non-trainable parameters in the FNO case. From the above calculation, we can see that clawFNO involves $(d_Q+H(d_u+d_Q+L+1)+LH^2)+\frac{(d_Q+1)}{2}p(p-1)+2LH^2m^p$ numbers of trainable parameters, while the vanilla FNO involves $(d_Q+H(d_u+d_Q+L+1)+LH^2)+(d_Q+1)p+2LH^2m^p$ numbers of trainable parameters. Therefore, the number of trainable parameters in clawFNO and FNO only differs in the second part of their projection layer, where clawFNO has $\frac{(d_Q+1)}{2}p(p-1)$ numbers of parameters and FNO has $(d_Q+1)p$. When $p > 3$, clawFNO will have a larger number of trainable parameters. However, we want to point out that since the number of parameter in the iterative layer ($2LH^2m^p$) grows exponentially with dimension p, it dominates the cost, and the differences between clawFNO and FNO are negligible. This is consistent with what we observed in Table 5: the number of trainable parameters and the GPU cost of clawNOs and their counterparts are almost the same. During the inference, the non-trainable parameters in clawNO will play a role and we therefore observe an increase in the inference time.

Table 5: Performance comparison of selected models in incompressible Navier-Stokes case 1, in terms of the total number of parameters, the per-epoch runtime, inference time, peak GPU usage, and the L2-norm divergence of prediction. The runtime is evaluated on a single NVIDIA V100 GPU. Note that the case of ntrain=10 requiring more time to run compared to ntrain=100 is due to the reduced batch size of 2, as opposed to the batch size of 20 in ntrain=100.

Case	ntrain	clawGFNO	clawFNO	GFNO	FNO	UNet	LSM	UNO	KNO
nparam (M)		0.85	0.93	0.85	0.93	0.92	1.21	1.07	0.89
runtime (s)	10	7.10	4.76	6.14	3.80	4.85	9.86	4.24	4.31
	100	4.89	2.42	4.39	2.03	2.39	4.98	2.69	2.35
	1000	41.75	19.56	40.06	17.38	20.80	37.86	18.78	18.20
inf. time (s)		0.077	0.072	0.062	0.058	0.075	0.183	0.066	0.045
GPU (GB)	1000	0.68	0.34	0.68	0.34	0.12	13.20	2.14	0.16
L2(div)	1000	1.2e-3	3.8e-4	6.6e-2	4.7e-2	3.5e-1	5.4e-1	5.8e-1	1.8e-1

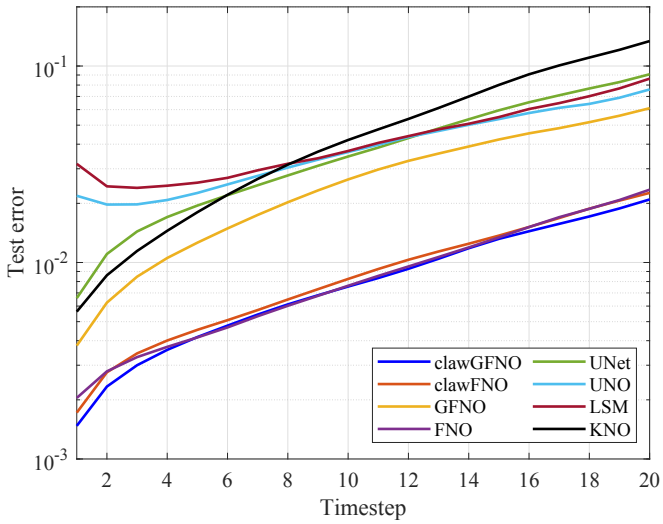


Figure 4: The per-time-step prediction error on the test dataset of the incompressible Navier-Stokes case 1, with ntrain=1000 samples.

B.2 Example 2 – radial dam break

In generating the radial dam break dataset, we closely follow the numerical procedure in Takamoto et al. (2022), where we slightly modified the code to output the velocity fields in addition to the water height. A total of 1,000 samples are generated and are subsequently split into n_{train}, 100, 100 and training, validation, and testing, respectively, with n_{train} the size of the training dataset depending on the adopted data regime. We run the numerical simulation on a $128 \times 128 \times 100$ grid and downsample the obtained solution to $32 \times 32 \times 25$ for training, where the first two dimensions are the spatial dimensions and the last is the temporal dimension. Analogous to the incompressible NS dataset, we perform 2×2 mean pooling in downsampling the spatial dimensions to maintain symmetry in data.

B.3 Example 3 – atmospheric modeling

As SpeedyWeather.jl uses spherical harmonics to solve the shallow water equations, we set the initial conditions η_0 for the aforementioned random waves through random coefficients of the spherical harmonics. The spherical harmonics are denoted as $Y_{\ell,m}$ with degree $\ell \geq 0$ and order m with $-\ell \leq m \leq \ell$. Using a standard complex normal distribution $\mathcal{CN}(0,1) = \mathcal{N}(0,\frac{1}{2}) + i\mathcal{N}(0,\frac{1}{2})$, the random coefficients $\eta_{\ell,m}$ are drawn for degrees $10 \leq \ell < 20$ from $\mathcal{CN}(0,1)$, but $\eta_{\ell,0} \sim \mathcal{N}(0,1)$ for the zonal modes $m = 0$, and $\eta_{\ell,m} = 0$ otherwise. The wave lengths are $2\pi R\ell^{-1}$, about 2000 to 4000 km. The initial u_0, v_0, η_0 on a grid can be obtained through the spherical harmonic transform as

$$\begin{aligned} u_0 &= v_0 = 0, \\ \eta_0 &= A \sum_{\ell=0}^{\ell_{max}} \sum_{m=-\ell}^{\ell} \eta_{\ell,m} Y_{\ell,m}. \end{aligned} \tag{14}$$

The amplitude A is chosen so that $\max(|\eta_0|) = 2000$ m. The resolution of the simulation is determined by the largest resolved degree ℓ_{max}, we use $\ell_{max} = 63$. In numerical weather prediction this spectral truncation is widely denoted as T63. We combine this spectral resolution with a regular longitude-latitude grid of 192x95 grid points ($\Delta\lambda = \Delta\theta = 1.875°$, about 200 km at the Equator, no grid points on the poles), also called a full Clenshaw-Curtis grid because of the underlying quadrature rule in the Legendre transform (Hotta & Ujiie, 2018). Non-linear terms are calculated on the grid, while the linear terms are calculated in spectral space of the spherical harmonics, and the model transforms between both spaces on every time step. This is a widely adopted method in global numerical weather prediction models.

For numerical stability, an implicitly calculated horizontal diffusion term of the form $-\nu\nabla^8\zeta, -\nu\nabla^8\mathcal{D}$, is added to the vorticity and the divergence equation, respectively. The power-4 Laplacian is chosen to be very scale-selective such that energy is only removed at the highest wave numbers, keeping the simulated flow otherwise largely unaffected. In practice, we use a non-dimensional Laplace operator $\tilde{\nabla}^2 = R^2\nabla^2$, such that the diffusion coefficient becomes an inverse time scale of $\tau = 2.4$ hours. The shallow water equations, equation 12, do not have a forcing or drag term such that the horizontal diffusion is the only term through which the system loses energy over time. The shallow water equations are otherwise energy conserving.

SpeedyWeather.jl employs a RAW-filtered (Williams, 2011) Leapfrog-based time integration with a time step of $\Delta t = 15$ min at T63 resolution. At this time step, the CFL number $C = c_{ph}\Delta t(\Delta x)^{-1}$ with equatorial $\Delta x = 2\pi R\frac{\Delta\lambda}{360°}$ is typically between $C = 1$ and $C = 1.4$, given wave speeds $c_{ph} = \sqrt{gh}$ between 280 and 320 ms^{-1}. Thanks to a centred semi-implicit integration of the linear terms (Hoskins & Simmons, 1975), the simulation remains stable without aggressively dampening the gravity waves with larger time steps or with a backwards implicit scheme. The continuity equation with the centred semi-implicit leapfrog integration reads as (the RAW-filter is neglected)

$$\frac{\eta_{i+1} - \eta_{i-1}}{2\Delta t} = -\tfrac{1}{2}\nabla\cdot(\mathbf{u}_{i+1}h_{i+1}) - \tfrac{1}{2}\nabla\cdot(\mathbf{u}_{i-1}h_{i-1}) \tag{15}$$

with previous time step $i-1$, and next time step $i+1$. The RAW-filter then acts as a weakly dampening Laplacian in time, coupling the tendencies at $i-1, i$ and $i+1$ to prevent a computational mode from growing.

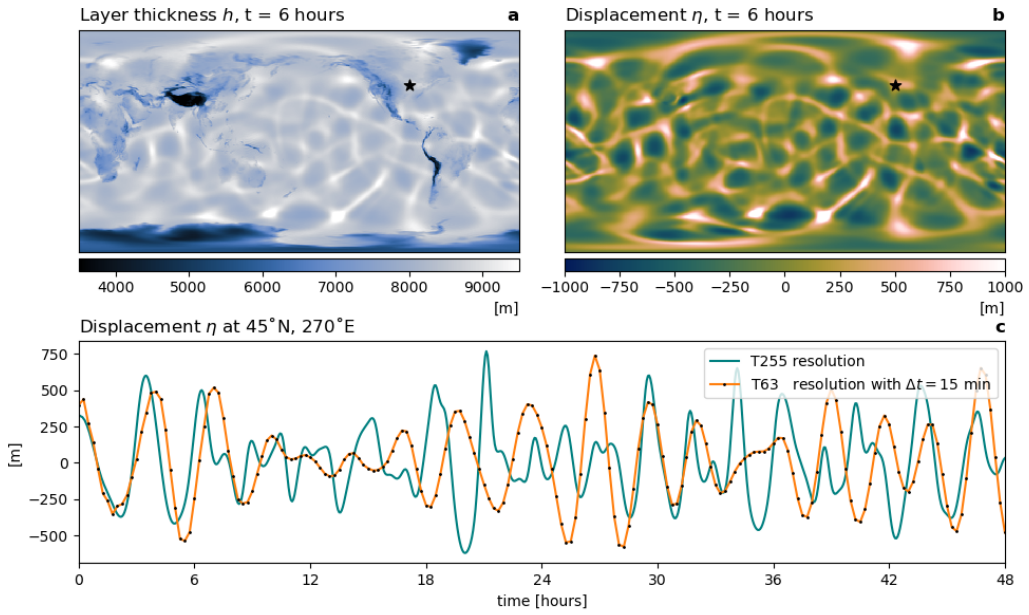


Figure 5: Atmospheric gravity waves as simulated by SpeedyWeather.jl. **a**, Layer thickness h and **b** displacement η after $t = 6$ hours at a resolution of T255 (about 50 km). The layer thickness h includes, in contrast to η, clearly the signal of the underlying orography of Earth. **c** Time series of η over the USA (45°N, 90°W, marked with a black star in **a**,**b**). Black circle markers denote the time step $\Delta t = 15$ min used for model integration and training data. Both simulations, T63 and T255, started from the same initial conditions, illustrating the limited predictability of the shallow water equations.

While the initial conditions contain only waves of wave lengths 2000 to 4000 km shorter waves are created during the simulation due to non-linear wave-wave interactions and interactions with the Earth's orography (Fig. 5). The time scale of these gravity waves is on the order of hours (Fig. 5c), which is why we use a training data sampling time step of $\Delta t = 15$ min. Much longer time steps as used by Gupta & Brandstetter (2022) would therefore fail to capture the wave dynamics present in the shallow water simulations. It is possible to use initial conditions that are closer to geostrophy, such that the presence of gravity waves from geostrophically-unbalanced initial conditions is

reduced. In such a setup, the predictability horizon is given by the chaotic vorticity advection and turbulence that evolves over longer time scales, which would justify a longer data sampling time step. However, the Gupta & Brandstetter (2022) setup includes a strong gravity wave in the initial conditions that propagates meridionally, while also including some slowly evolving vortices. Our setup therefore represents a physically clearer defined problem, one that focuses on the non-linear gravity wave propagation in the shallow water system. Following equation 14, our setup can be easily recreated in other models for further studies. In this context, we generate a total of 1,200 samples and split them into 1,000/100/100 for training/validation/testing, respectively.

B.4 Example 4 – constitutive modeling of material deformation

We generate the material deformation dataset as follows. Firstly, we generate a solution A from a steady-state 2D Darcy flow with the diffusion coefficient specified as a random Gaussian field, following the numerical procedure in Li et al. (2020c). The ground truth $\mathbf{u}$ is generated as

$$\mathbf{u} = (\frac{\partial A}{\partial y}, -\frac{\partial A}{\partial x}),$$

which is guaranteed divergence-free, and the body load $\mathbf{f}$ is then computed following equation 13. The derivatives are numerically computed using FFT on a regular mesh. A total of 300 samples are generated and are subsequently split into n_{train}, 100, 100 for training, validation, and testing, respectively, with n_{train} the size of the training dataset depending on the adopted data regime. We run the numerical simulation on a 80×80 uniform mesh and downsample the solution to a grid of 246 spatial points through interpolation on a circular domain of radius 0.4 and centers at the origin point.

B.5 Training strategies

We run three replicates for all the experiments and report the mean and standard deviation of the L2 relative error for comparison metrics. For all Fourier-domain models, we closely follow the model setup in Helwig et al. (2023), employing four Fourier layers and keeping only the 12 lowest Fourier modes in all the 2D models and 8 spatial and 6 temporal lowest modes in all the 3D models. An exception is in the atmospheric modeling problem, where we truncate the spatial Fourier modes to 22 for correct physical realizability. For fair comparison, we adopt Cartesian encoding in all the models.

Similar to the model size in Helwig et al. (2023), we set the latent dimension in FNO and clawFNO to 20, whereas we counterbalance the additional dimensions introduced due to equivariance in GFNO and clawGFNO by reducing the latent dimension to 10 in all the 2D models and 11 in all the 3D models. For UNet, in order to arrive at a similar number of model parameters, we increase the first-layer dimension to 11 in the incompressible NS problem, to 15 in the radial dam break problem, and to 96 in weather modeling.

As suggested in Tran et al. (2022) and Helwig et al. (2023), we turn to the teacher forcing strategy to facilitate the learning process. We set the batch size to 20 for all 2D models and 10 for all 3D models, with the exception in small and medium data regimes, where we set batch size to 2 and 1 when the training datasets are of size 10 and 2, respectively. We employ cosine annealing learning rate scheduler that decays the initial learning rate to 0. All the 2D models are trained for a total of 100 epochs whereas all the 3D models are trained for 500 epochs with an early stop if the validation loss stops improving for consecutive 100 epochs. 2D models are trained with less number of epochs as one training sample in 3D corresponds to $(T - T_{in})$ training samples in 2D. We directly take the baseline models in GFNO (Helwig et al., 2023), and further tune the hyperparameters (i.e., the learning rate and weight decay in the Adam optimizer) in clawNOs. All the experiments are carried out on a single NVIDIA A6000 40GB GPU.

For both of the two graph-based models (i.e., INO and GNO), we closely follow the model setup in Liu et al. (2023) and Li et al. (2020c). In order to be consistent across clawINO, INO and GNO, the latent width is set to 64 and the kernel width is set to 1,024 in all models. We employ a total of 4 integral layers in all models, whereas the shallow-to-deep technique is equipped for initialization in INO-based models. The batch size is set to 2 for all the irregular-mesh models, with the exception in the first case in the constitutive modeling example where a batch size equal to 1 is employed.

We adopt the cosine annealing learning rate scheduler that decays the initial learning rate to 0. All the models are trained for 500 epochs with an early stop if the validation loss stops improving for consecutive 200 epochs.

C ROLLOUT VISUALIZATIONS

We illustrate the rollout of randomly selected trajectories in the test dataset using clawNO predictions, along with the comparisons against ground truth data and the corresponding absolute errors. The rollouts of incompressible NS, radial dam break, and atmospheric modeling are showcased in Figure 6, Figures 8 and 9, Figures 11 and 2, respectively. We also showcase the material deformation prediction in 6 different test samples in Figure 15. To provide a visual comparison across models, we plot the final-step prediction of all the models in the incompressible NS example in Figure 7, the radial dam break example in Figure 10, and the atmospheric modeling example in Figure 14.

D DETAILED DERIVATIONS

D.1 PROOF OF THEOREM 3.1

Since the Fourier spectral differentiation error estimate is a direct result of Trefethen (2000, Page 34), we provide the detailed derivation of equation 7 in this section. According to the basic properties of Fourier transform, if f is a differential and periodic function on $[0, L]$ with its Fourier representation:

$$f(\mathbf{x}) = \sum_{\xi=-N/2}^{N/2-1} \hat{f}(\xi) e^{i2\pi\xi/L},$$

its derivative can be given as

$$f'(\mathbf{x}) = \sum_{\xi=-N/2}^{N/2-1} \frac{i2\pi\xi}{L} \hat{f}(\xi) e^{i2\pi\xi/L} \approx \mathcal{F}^{-1}\left[\frac{i2\pi\xi}{L}\mathcal{F}[f](\xi)\right].$$

As such, equation 7 can be obtained by applying the above property to approximate every derivative term $\dfrac{\partial \mu_{jk}^{(N^{(1)},\cdots,N^{(p)})}}{\partial x_k}$.

D.2 PROOF OF THEOREM 3.2

It suffices to show that $\left|\dfrac{\partial \psi}{\partial x_k}(\mathbf{x}_i) - \sum_{\mathbf{x}_l \in \chi \cap B_\delta(\mathbf{x}_i)} (\psi(\mathbf{x}_l) - \psi(\mathbf{x}_i))\omega_{i,l}^{(k)}\right| \leq C\Delta x^{m+1}$ for any $\psi \in C^{m+1}(\Omega)$. Denote $I[\psi](\mathbf{x}_i) := \dfrac{\partial \psi}{\partial x_k}(\mathbf{x}_i)$, $I_{\Delta x}[\psi](\mathbf{x}_i) := \sum_{\mathbf{x}_l \in \chi \cap B_\delta(\mathbf{x}_i)} (\psi(\mathbf{x}_l) - \psi(\mathbf{x}_i))\omega_{i,l}^{(k)}$ and let ϕ_m denote the $m-$th order truncated Taylor series of ψ about $\mathbf{x}_i$ with associated remainder r_m, such that

$$\psi(\mathbf{y}) = \phi_m(\mathbf{y}) + r_m(\mathbf{y}) = \sum_{|\alpha| \leq m} \frac{D^\alpha \psi(\mathbf{x}_i)}{\alpha!}(\mathbf{y} - \mathbf{x}_i)^\alpha + \sum_{|\beta|=m+1} R_\beta(\mathbf{y})(\mathbf{y} - \mathbf{x}_i)^\beta,$$

where $R_\beta(\mathbf{y}) := \dfrac{|\beta|}{\beta!}\int_0^1 (1-\tau)^{|\beta|-1} D^\beta u(\mathbf{y} + \tau(\mathbf{y} - \mathbf{x}_i))d\tau$ and therefore $|R_\beta(\mathbf{y})| \leq \dfrac{1}{\beta!}\max_{|\alpha|=m+1}\max_{\mathbf{y}\in B_\delta(\mathbf{x}_i)} |D^\alpha \psi(\mathbf{y})| \leq C||\psi||_{C^{m+1}}$. We then have

$$\begin{aligned}|\psi - \phi_m|(\mathbf{y}) = &|r_m|(\mathbf{y}) = \left|\sum_{|\beta|=m+1} R_\beta(\mathbf{y})(\mathbf{y} - \mathbf{x}_i)^\beta\right| \\ \leq &|\mathbf{y} - \mathbf{x}_i|^{m+1} \sum_{|\beta|=m+1} |R_\beta(\mathbf{y})| \leq C||\psi||_{C^{m+1}}|\mathbf{y} - \mathbf{x}_i|^{m+1}.\end{aligned}$$

To bound the approximation error, we apply the triangle inequality and the reproducing condition of polynomial ϕ_m:

$$\begin{aligned}|I[\psi](\mathbf{x}_i) - I_{\Delta x}[\psi](\mathbf{x}_i)| \leq & |I[\psi](\mathbf{x}_i) - I[\phi_m](\mathbf{x}_i)| + |I[\phi_m](\mathbf{x}_i) - I_{\Delta x}[\psi](\mathbf{x}_i)| \\ = & |I[\psi](\mathbf{x}_i) - I[\phi_m](\mathbf{x}_i)| + |I_{\Delta x}[\phi_m](\mathbf{x}_i) - I_{\Delta x}[\psi](\mathbf{x}_i)|.\end{aligned}$$

Here, the first term vanishes since ϕ_m is the truncated Taylor series of ψ and $m \geq 1$, and for the second term we have

$$\begin{aligned}|I_{\Delta x}[\phi_m](\mathbf{x}_i) - I_{\Delta x}[\psi](\mathbf{x}_i)| \leq & \sum_{\mathbf{x}_l \in \chi \cap B_\delta(\mathbf{x}_i)} |\psi(\mathbf{x}_l) - \psi(\mathbf{x}_i) - \phi_m(\mathbf{x}_l) + \phi_m(\mathbf{x}_i)||\omega_{i,l}^{(k)}| \\ = & \sum_{\mathbf{x}_l \in \chi \cap B_\delta(\mathbf{x}_i)} |\psi(\mathbf{x}_l) - \phi_m(\mathbf{x}_l)||\omega_{i,l}^{(k)}| \\ \leq & C||\psi||_{C^{m+1}} \sum_{\mathbf{x}_l \in \chi \cap B_\delta(\mathbf{x}_i)} |\mathbf{x}_l - \mathbf{x}_i|^{m+1}|\omega_{i,l}^{(k)}| \\ \leq & C||\psi||_{C^{m+1}} \Delta x^{m+1} \sum_{\mathbf{x}_l \in \chi \cap B_\delta(\mathbf{x}_i)} |\omega_{i,l}^{(k)}| \leq C\Delta x^{m+1}.\end{aligned}$$

Here, the last inequality can be proved following the argument in Levin (1998, Theorem 5): for each fixed k and i, the coefficient $\omega_{i,l}^{(k)}$ is a continuous function of $\mathbf{x}_l$. Moreover, the size of $\chi \cap B_\delta(\mathbf{x})$ is bounded. Thus, for a fixed Δx it follows $\sum_{\mathbf{x}_l \in \chi \cap B_\delta(\mathbf{x}_i)} |\omega_{i,l}^{(k)}| \leq C$ where C is independent of Δx, k, and $\mathbf{x}_i$. And therefore we obtain $|I[\psi](\mathbf{x}_i) - I_{\Delta x}[\psi](\mathbf{x}_i)| \leq C\Delta x^{m+1}$ and finish the proof.

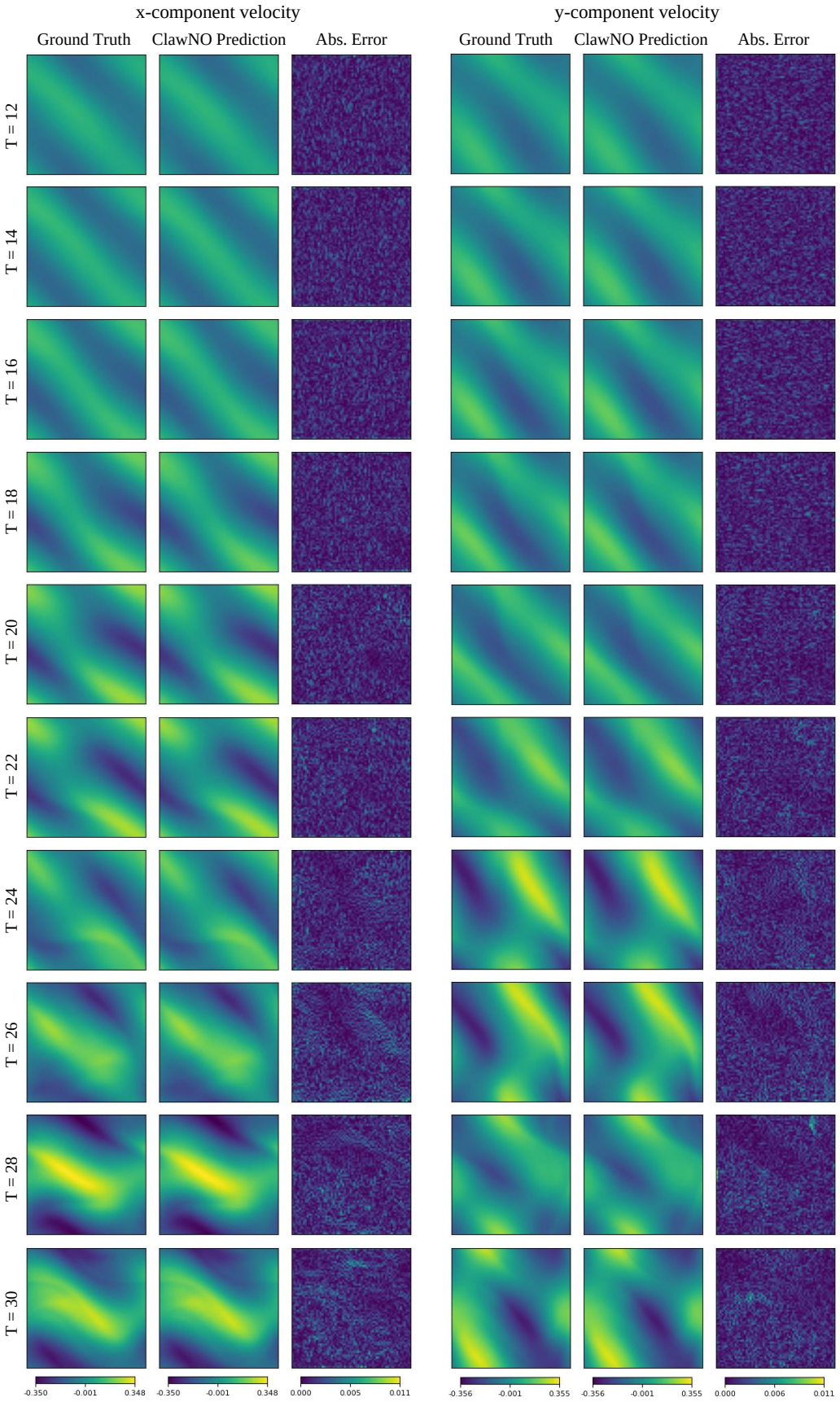


Figure 6: Rollout of incompressible Navier-Stokes case 1.

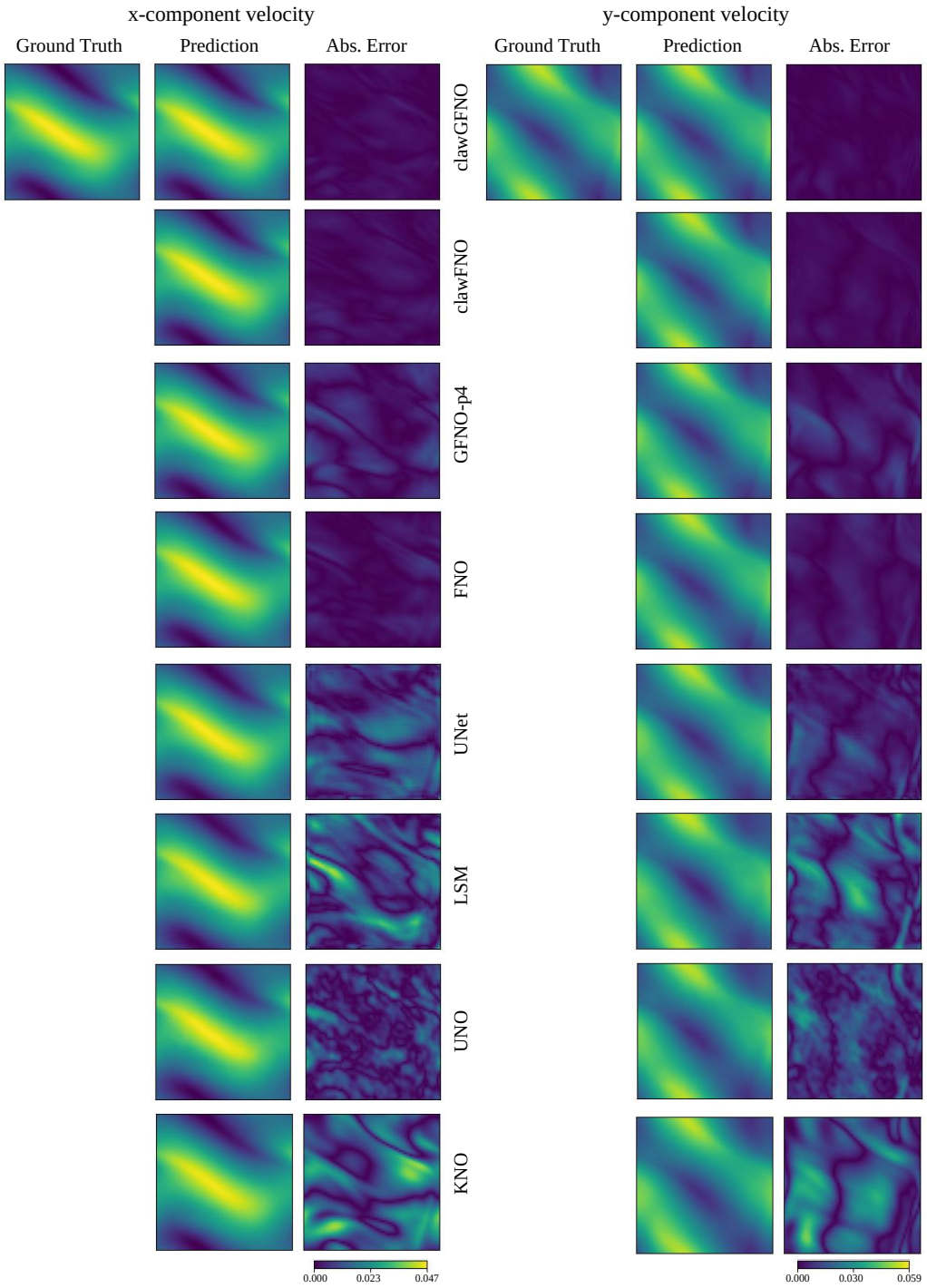


Figure 7: Last-step prediction comparison across models in incompressible Navier-Stokes case 1.

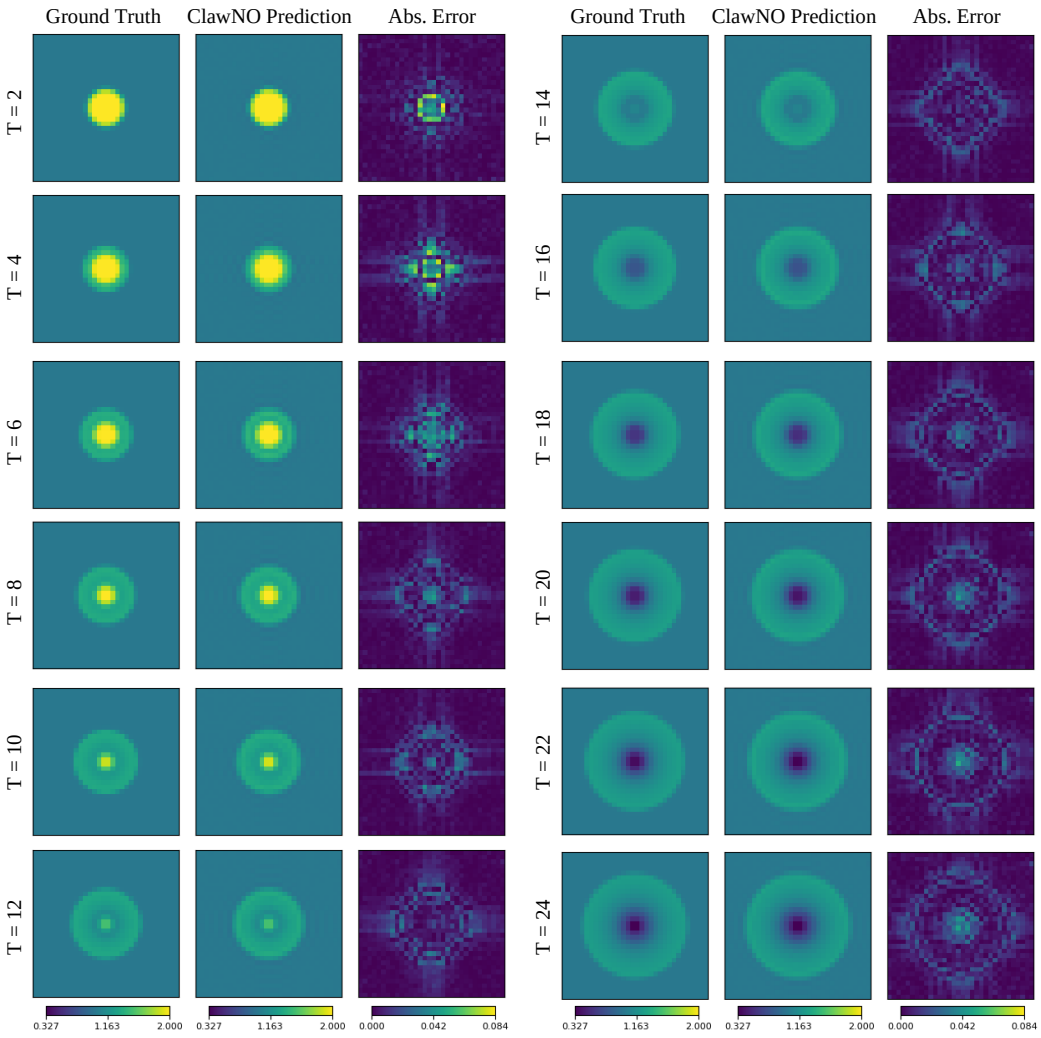


Figure 8: Rollout of water depth in radial dam break modeling.

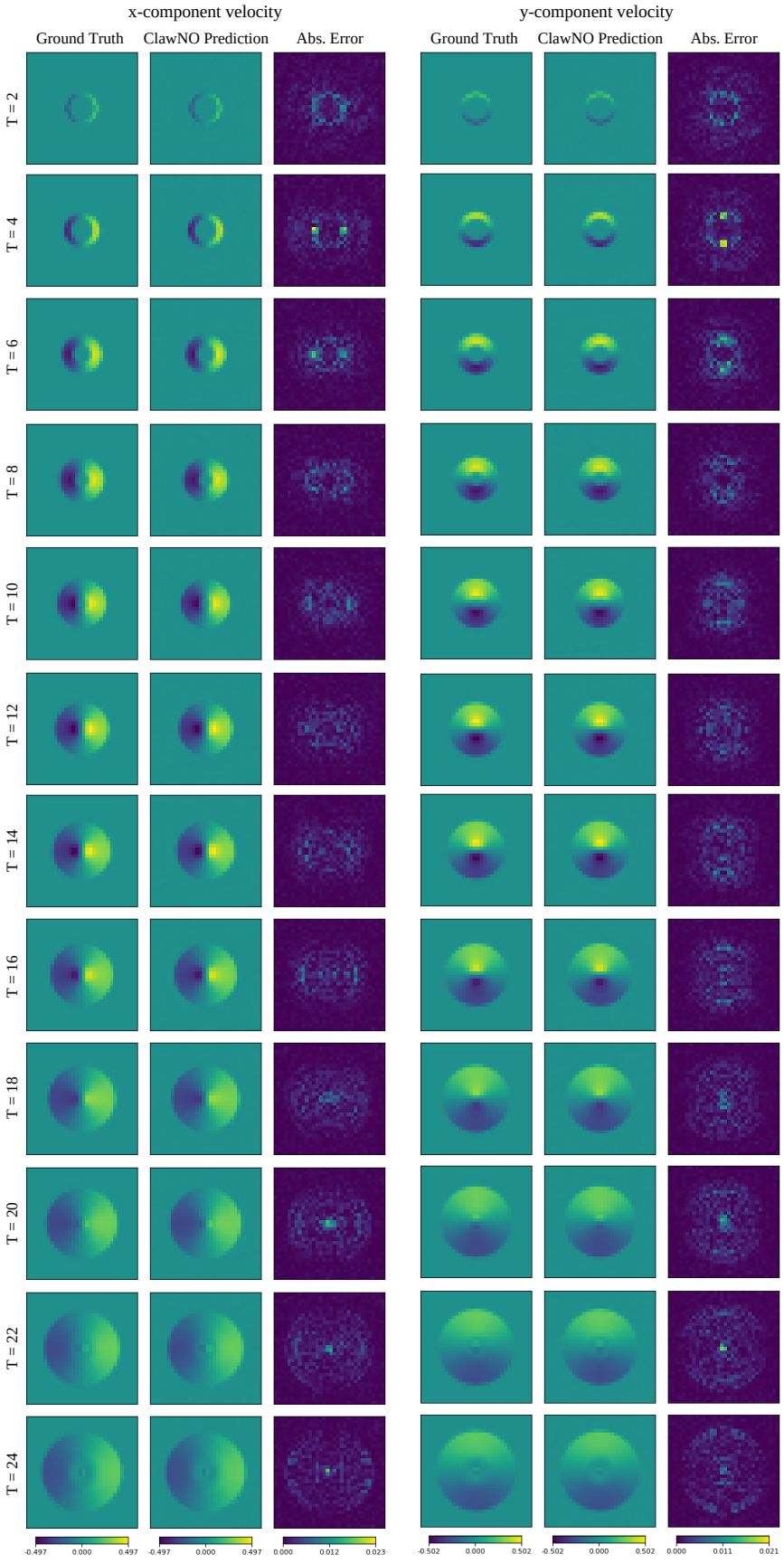


Figure 9: Rollout of velocity in radial dam break modeling.

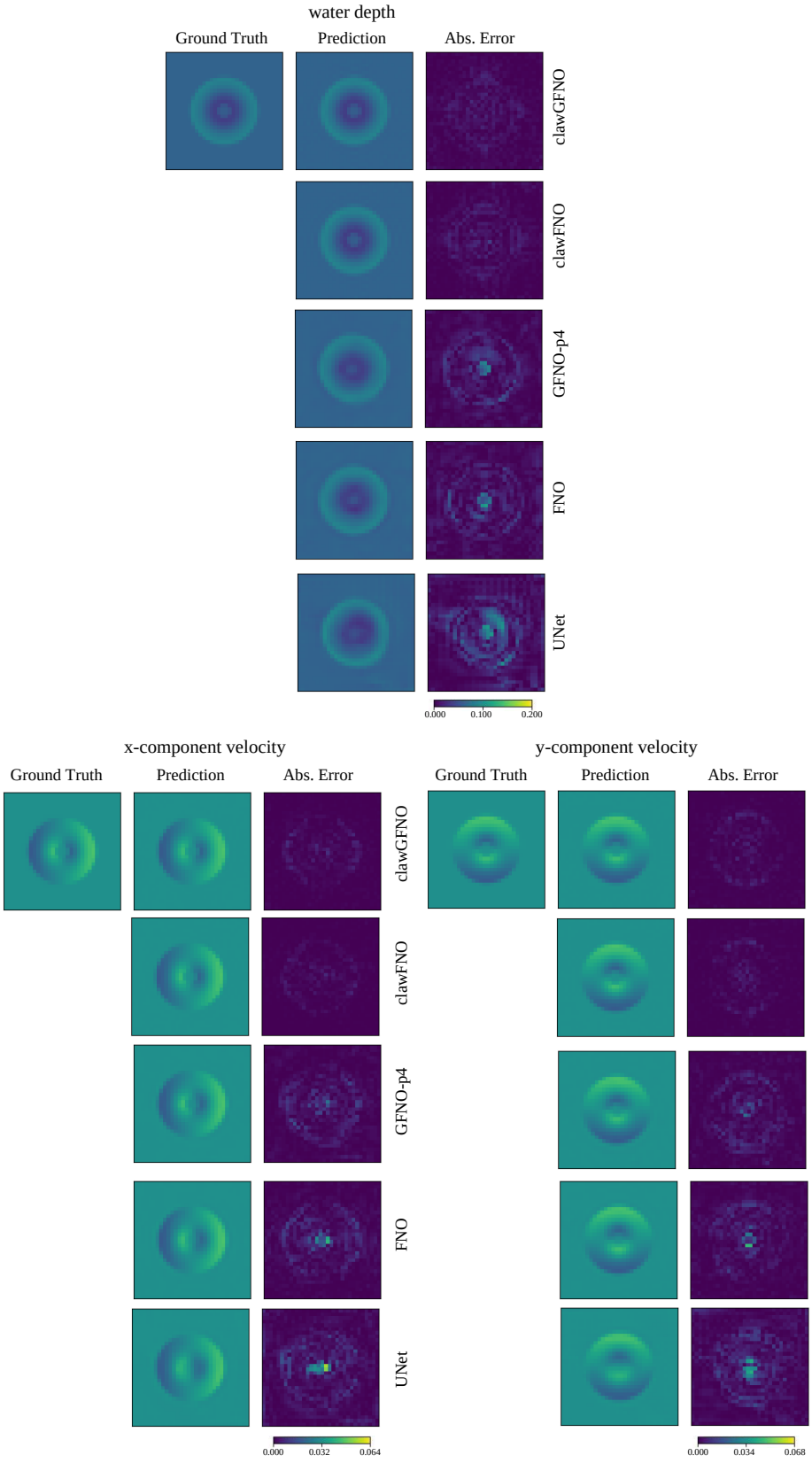


Figure 10: Last-step prediction comparison across models trained with ntrain=10 samples in radial dam break dataset.

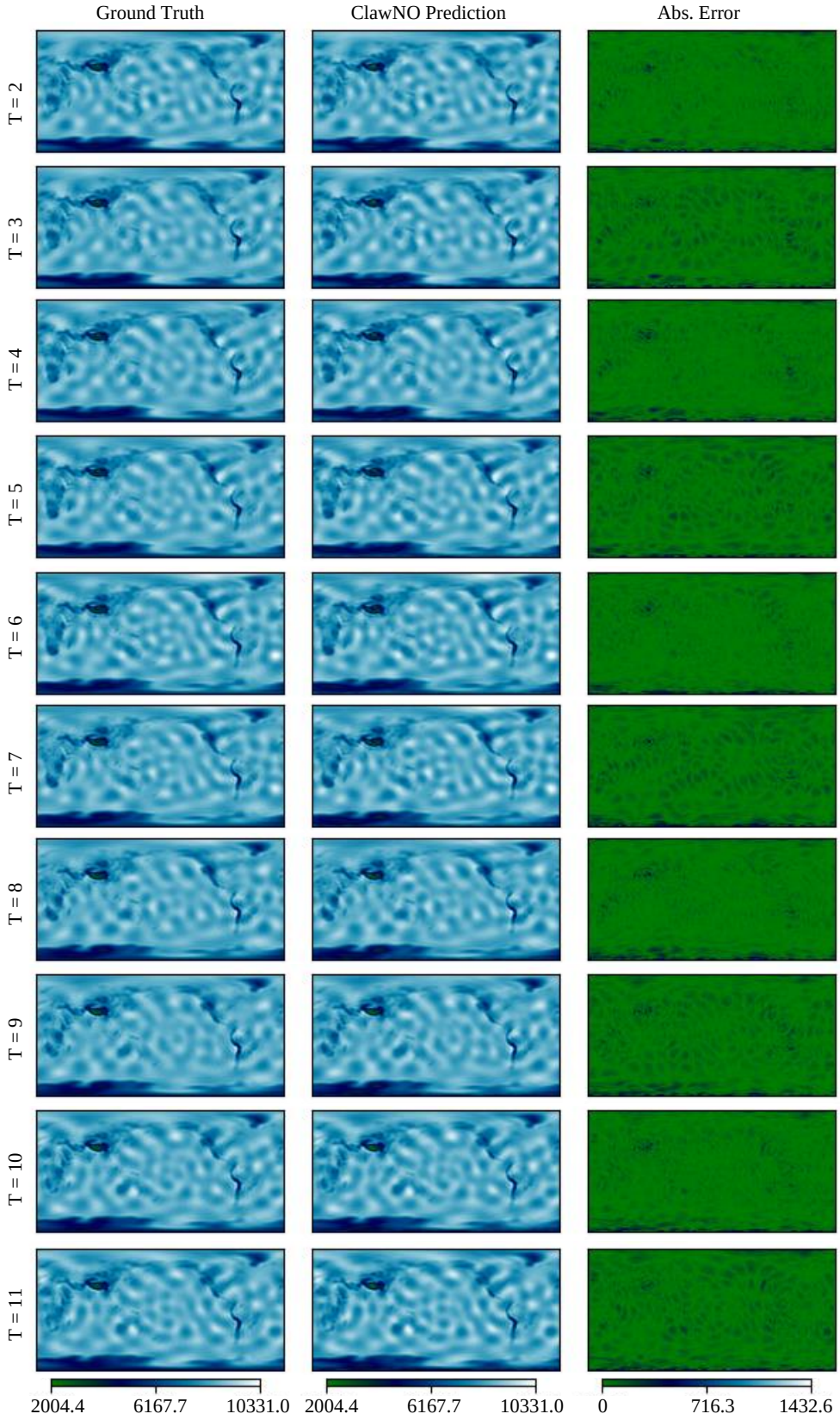


Figure 11: Rollout of layer thickness in atmospheric modeling.

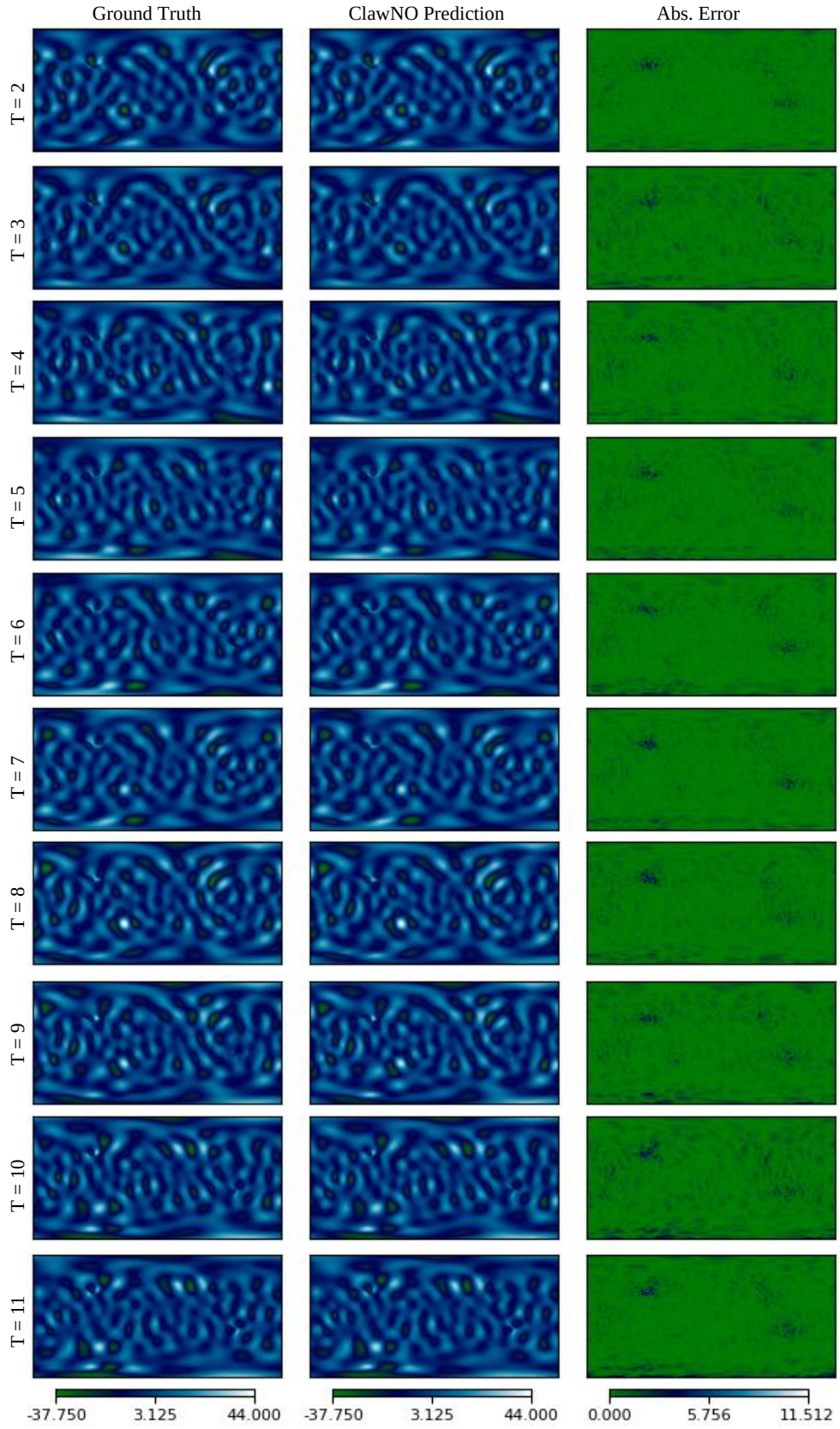


Figure 12: Rollout of zonal wind velocity in atmospheric modeling.

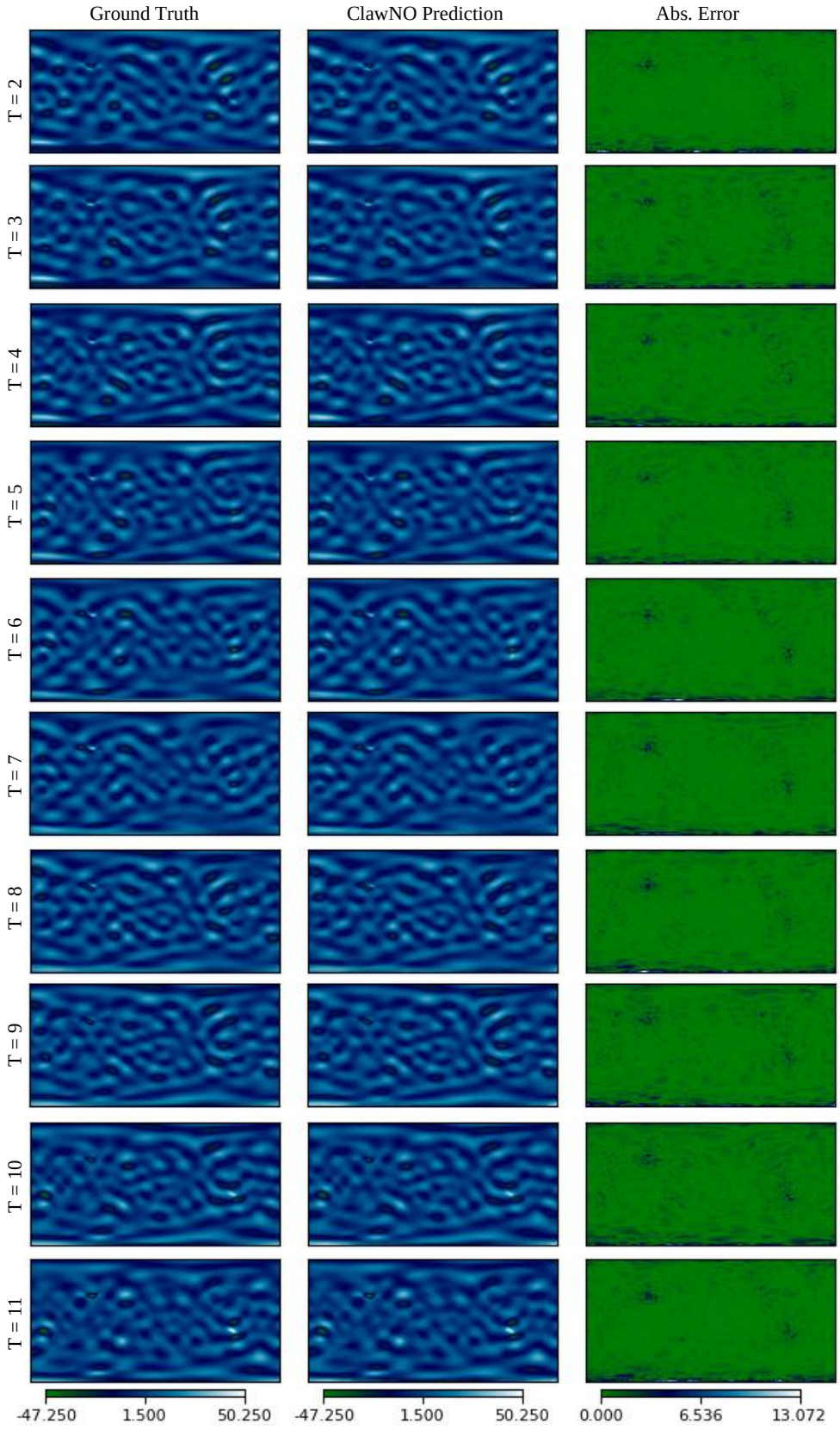


Figure 13: Rollout of meridional wind velocity in atmospheric modeling.

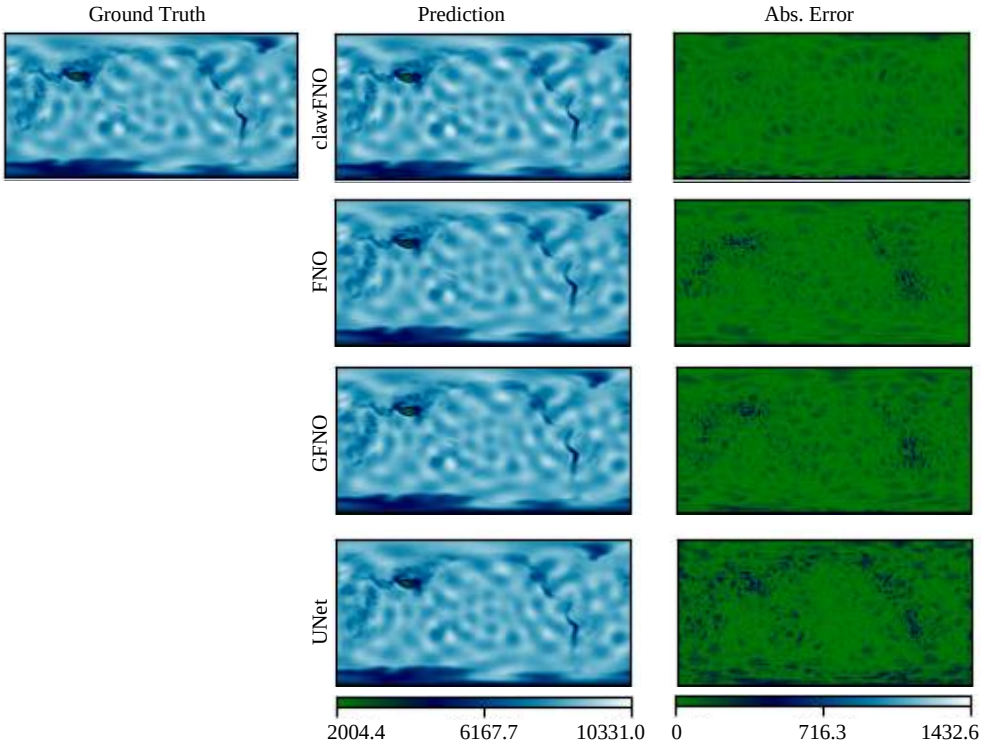


Figure 14: Last-step prediction comparison across models in atmospheric modeling dataset.

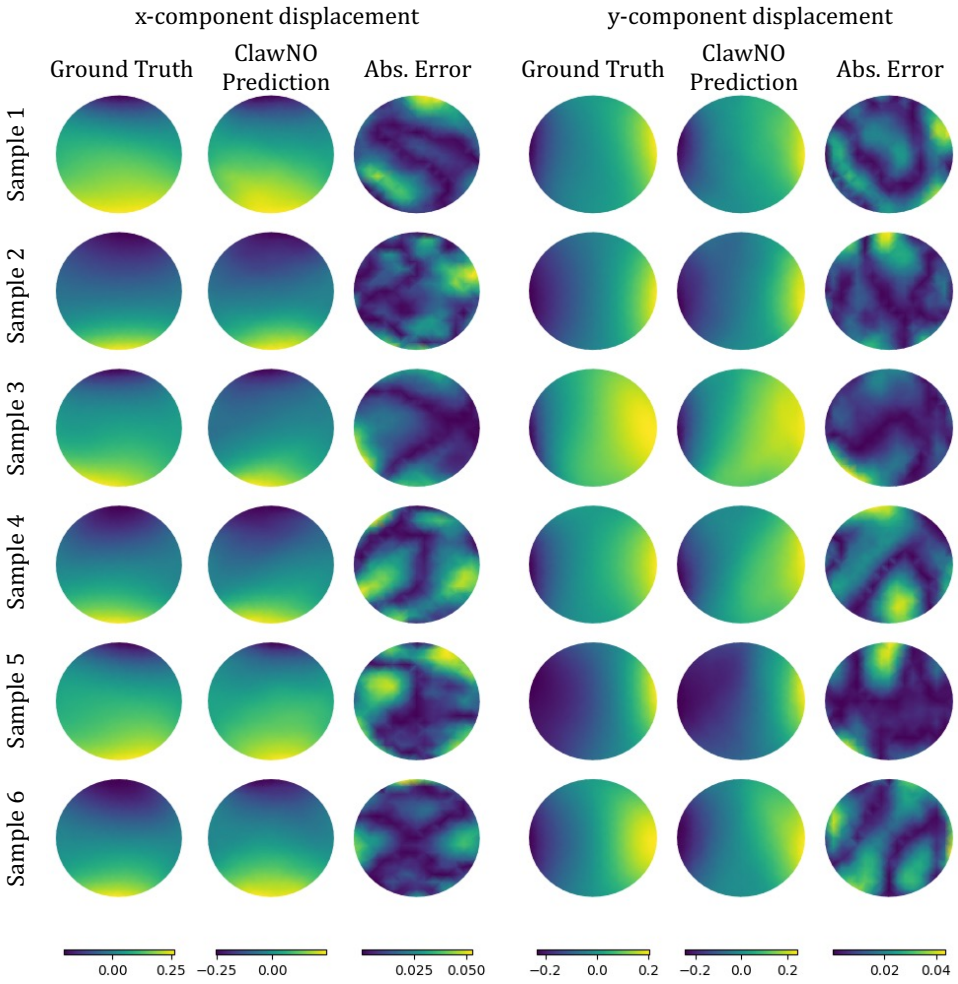


Figure 15: Demonstration of the constitutive modeling of material deformation.