Lyapunov Regularized Forecaster

Rong Zheng
University of California San Diego
r2zheng@ucsd.edu

Sicun Gao
University of California San Diego
sicung@ucsd.edu

Rose Yu*
University of California San Diego
roseyu@ucsd.edu

Abstract

Turbulent flow prediction plays a crucial role in climate change prediction. Especially, the long-term prediction of turbulent flow is a primary and promising goal for its future development and attracts more attention from researchers. However, because the Navier-Stokes equations on which turbulent flow relies are chaotic systems, imperceptible initial differences can lead to large differences in future states, making the long-term prediction extremely difficult, even for the state-of-the-art turbulence prediction model Turbulent-Flow Net (TF-Net) that introduces a trainable bispectral decomposition and combines the temporal properties of turbulence with spatial modeling. Realizing that the error propagation leads to severe instability over time in long-term prediction, we propose a time-based Lyapunov regularizer[2] to the loss function of TF-Net to avoid training error propagation and improve the trained long-term prediction. The comparison experiment shows that our Lyapunov-regularized forecaster does have more stable long-term predictions.

1 Introduction

Computational Fluid Dynamics (CFD) is a cross-cutting science in which fluid mechanics and computer science are interwoven, and it utilizes physical rules and statistics on time series and spatial distributions to compute and thus obtain approximate solutions to fluid control equations. CFD is widely used with great success in various fields and industries. CFD is also at the heart of climate simulation and is directly helpful in understanding and predicting climate change, in which the prediction of turbulent flow under the control of Navier-Stokes equations plays an important role in climate prediction. Predicting turbulent flow is one of the most challenging problems in science and engineering. Navier-Stokes equations belong to a chaotic system. That is, as a strongly nonlinear high-dimensional system, it brings huge degrees of freedom and multi-scale coupling. It is classified as a statistical physics problem far from equilibrium, and the sensitivity to small perturbations and errors makes the accuracy of long-term predictions very difficult to achieve.

Current deep learning utilizes a large amount of statistical data with known physical laws for turbulence to improve the long-term prediction of turbulence. For example, Turbulent-Flow Net (TF-Net) [4], a data-driven deep learning model, introduces a trainable bispectral decomposition using the popular and reliable Reynolds-averaged Navier-Stokes (RANS) and Large Eddy Simulation (LES) nonlinear coupled models and obtains good prediction results by combining the temporal properties of turbulence with spatial modeling. However, turbulence prediction is extremely error-sensitive and each step of the prediction depends on a large amount of data derived from the velocity field of the

*Corresponding author: Rose Yu (roseyu@ucsd.edu)

[2]Our code and models are available on Github

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

historical steps. The error propagation leads to severe instability over time in long-term prediction. Some models achieve robust stability by adding training noise, space-based loss constraints, and dataset augmentation [3]. Differently, we plan to add a time-based regularizer to constrain the model during training to further improve the stability of its long-term prediction.

We assume turbulence prediction as a continuous-time dynamical system, and the Lyapunov function is a direct tool to determine the stability of this dynamical system. In this paper, we use TF-Net as a turbulence prediction model and model the time evolution of turbulence prediction by adding a Lyapunov regularizer that exploits the properties of temporal dynamical systems to further enhance its long-term prediction effect. We observe that with the addition of the Lyapunov regularizer, we obtain more accurate long-time series predictions with an outstanding improvement in the test RMSE compared to the unregularized TF-Net.

2 Methodology

We develop the Lyapunov regularized turbulent flow forecaster training scheme by using TF-Net as the prediction model and adding our Lyapunov regularizer to the loss function. The purpose of our regularizer is to maintain the stability of the long-term prediction accuracy. We want the long-term prediction to be an asymptotically stable autonomous system. That is, the difference of $\hat{\mathbf{v}}_t$ and $\mathbf{v}_t$ decreases toward 0 as time t increases, where $\hat{\mathbf{v}}_t$ is the predicted two-dimensional velocity field at time t and $\mathbf{v}_t$ is the corresponding ground truth. Based on the theorem on asymptotic stability of Lyapunov functions, we want

$$\dot{V}(\mathbf{x}(t)) = \frac{d}{dt}V(\mathbf{x}(t)) = \frac{d}{dt}V(\hat{\mathbf{v}}_t - \mathbf{v}_t) \begin{cases} < 0 & \text{for all } \hat{\mathbf{v}}_t \neq \mathbf{v}_t \\ = 0 & \text{only if } \hat{\mathbf{v}}_t = \mathbf{v}_t \end{cases} \tag{1}$$

2.1 Lyapunov Function

We assume the Mean Square Error (MSE) of the prediction and ground truth at time t as the Lyapunov function,

$$V(\mathbf{x}(t)) = ||\mathbf{x}(t)||_2^2 = ||\hat{\mathbf{v}}_t - \mathbf{v}_t||_2^2 \tag{2}$$

where $\mathbf{x}(t) = \hat{\mathbf{v}}_t - \mathbf{v}_t$. Also, V satisfies the conditions of Lyapunov function that $V(\mathbf{x}) > 0$ for all $\mathbf{x} \neq \mathbf{0}$ and $V(\mathbf{0}) = 0$.

Then, we use the finite difference of $V(\mathbf{x}(t+1)) - V(\mathbf{x}(t))$ to approximate $\dot{V}$, which is used as a part of the regularizer,

$$\dot{V}(\mathbf{x}(t)) \approx V(\mathbf{x}(t+1)) - V(\mathbf{x}(t)) = ||\hat{\mathbf{v}}_{t+1} - \mathbf{v}_{t+1}||_2^2 - ||\hat{\mathbf{v}}_t - \mathbf{v}_t||_2^2 \tag{3}$$

While training, the Lyapunov regularizer reduces the difference in prediction error over long-term time steps, which constrains the cumulative error caused by the long-term prediction to increase slowly over time steps.

2.2 Wrappers

As our goal is to treat $\dot{V}$ as a convex constraint by maintaining its negativity, we wrap $\dot{V}$ with the log barrier function,

$$\phi_{log}(\mathbf{x}(t)) = -\frac{1}{\alpha}\log(-\dot{V}(\mathbf{x}(t))) \tag{4}$$

where α is a large positive number that increases the optimization accuracy and avoids the unnecessary minimization of $\dot{V}$.

Wrapping $\dot{V}$ with ReLU is another solution that helps maintain the negativity,

$$\phi_{relu}(\mathbf{x}(t)) = ReLU(\dot{V}(\mathbf{x}(t))) \tag{5}$$

It keeps the gradient of $\dot{V}$ for back-propagation only if the derivative is positive, which guides $\dot{V}$ to decrease toward negative.

2.3 Weight Scheduler

Without an appropriate weight scheduler, the regularizer may increase the prediction error in short-term prediction steps. Because at early stages, the prediction errors are low but the increment slope is steep, which may cause the weight of the Lyapunov regularizer to be higher than the prediction loss. Thus, we add a scheduler based on the prediction MSE to balance the weight of the regularizer,

$$w(\mathbf{x}(t)) = \frac{1}{1 + e^{-s \cdot (||\mathbf{x}(t)||_2^2 - m)}} \tag{6}$$

which is an adjusted sigmoid function. As s is a big positive constant, the weight approaches to 0 when the prediction MSE $||\mathbf{x}(t)||_2^2$ is less than the threshold m and approaches to 1 when the MSE is greater than m. Since the prediction MSE is low at early steps and high at later steps due to the teacher-forcing training method, the regularization weight at early steps will be approximately zero but approximately one at later steps with an appropriate m.

2.4 Lyapunov Regularizer

Utilizing Equation (5) as the constant weight and Equation (4) as the constraint, we can get the Lyapunov regularizer

$$\mathcal{L}_{lya} = \sum_{i}^{N} \sum_{t}^{T} w(\mathbf{x}(t)) \phi(\mathbf{x}(t)) = \sum_{i}^{N} \sum_{t}^{T} w(\hat{\mathbf{v}}(t) - \mathbf{v}(t)) \phi(\hat{\mathbf{v}}(t) - \mathbf{v}(t)) \tag{7}$$

where N is the batch size and T is the prediction length.

3 Experiments

3.1 Data

Our experimental data are the turbulent-flow velocity fields in two-dimensional space simulated by the Lattice Boltzmann Method [1]. The relevant physical parameters of the simulation are properly set, including Prandtl number $= 0.71$, Rayleigh number $= 2.5 \times 10^8$, and the maximum Mach number $= 0.1$. The simulation provides 2000 frames of two-dimensional velocity fields with the spatial resolution 1792×256. We divide each 1792×256 frame into seven 256×256 sub-regions and downsample them into 64×64 images. Then, we use a sliding window with a length of 100 to generate 9,870 samples, and we select 6,000 for training, 1,700 for validation, and 2,170 for testing. The data is normalized for training, validation, and testing.

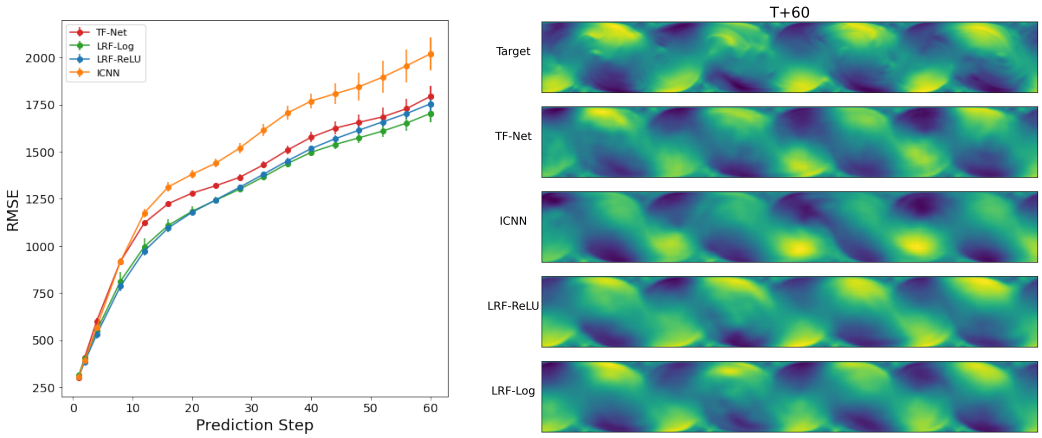
3.2 Results

In the comparison test, TF-Net is used as the prediction model for the turbulent flow prediction task. We compare our results with TF-Net without our regularizer and a state-of-the-art Lyapunov regularization baseline [2].

The baseline model contains a trainable Lyapunov function, which consists of an input-convex neural network (ICNN). We connect ICNN with the input and output of TF-Net and apply the loss function of the baseline while training. The baseline model is tuned based on the averages of validation results of the turbulent-flow data.

For our regularizer, we implement two versions of $\phi(\mathbf{x}(t))$ in $\mathcal{L}_{lya}$: one uses log barrier ϕ_{log} in Equation (4), and another uses ReLU ϕ_{relu} in Equation (5). We set $\alpha = 10^3$ in ϕ_{log}. We fix $s = 300$ in weight scheduler $w(\mathbf{x}(t))$ and tune m within the range of 0.5 and 0.8 for each of the two ϕ's. While training, we select multiple samples of turbulent flow sequences as one batch, and train sequences with one-step prediction and teacher-forcing method. We add the Lyapunov regularizer $\mathcal{L}_{lya}$ to the original loss function of TF-Net to obtain the regularized forecaster. All models are fixed at 100 training epochs. Every two models are trained using one Nvidia A6000, and the training takes approximately 1.5 hours to complete.

We utilize RMSE as the evaluation metric for quantifying the prediction performance of the following 60 frames. All test results are averaged over four runs with random initialization. The comparison



(a) RMSE (b) Prediction Examples

Figure 1: (a) RMSE of Turbulent Flow Forecasting, where vertical lines are the standard deviations; (b) Ground Truth (Target) and the predicted velocity fields generated by various methods, including TF-Net, ICNN and the proposed LRF at prediction step of 60.

results are shown in Figure 1(a), from which we can observe that our Lyapunov regularizers can provide distinct performance improvements.

With Equation (3) we can see that the role of the regularizer is to make the long-term prediction error curve smoother. Fortunately, when we directly add the regularizer described in Equation (3) combined with the wrapper in Equation (4) to the loss function of the prediction model, the long-term RMSE decreases instead of simply raising the short-term RMSE to make the curve smoother. We further added the weight scheduler so that the short-time RMSE is not negatively affected, resulting in the curve in Figure 1(a). The visualization of the ground-truth and predicted velocity fields are in Figure 1(b) and Figure 2 in Appendix.

4 Limitations

In fact, we observed that if we train TF-Net with our Lyapunov regularizer but without the weight scheduler, the RMSEs after time steps 20 are smoother and lower than the one with the weight scheduler. However, the short-term prediction RMSEs are high to further improve the overall smoothness to cooperate with the constraint of $\dot{V}$. Therefore, although the scheduler w can reduce the short-term prediction RMSE, it may have some negative impacts on the long-term prediction steps. In the future, we will develop a more efficient pair of weight scheduler and Lyapunov functions to thoroughly exploit the advantages of Lyapunov regularization in the long-term prediction part.

5 Conclusion

In this work, we use TF-Net, which combines the properties of turbulence physics and big data, as a prediction model and add to it a Lyapunov regularizer that exploits the properties of temporal dynamical systems to improve the accuracy of long-term predictions. We choose the prediction error of the velocity field at a time point t as the Lyapunov function and approximate the derivative of it with respect to time to serve as a regularizer for the long-time prediction. To make it act as a constraint function, we wrap it using a barrier function or a rectifier. We further add a weight scheduler to attenuate its negative effect on short-term prediction steps. Through comparison experiments we observe an effective long-time prediction boosting effect. After further enhancements to our Lyapunov regularizer, this regularization might be helpful for the long-term prediction of climate change.

Broader Impact

Currently, the application of Lyapunov Regularize Forecaster is limited to climate change prediction. We discovered that our Lyapunov regularizer does not have any targeted prerequisite on the prediction model and prediction data type. Therefore, we will enhance the regularizer from a generalization perspective and test its effectiveness in other rule-bounded prediction domains. For example, it may improve the accuracy of long-term fire ignition simulation on grassland, which helps wildfire prevention and control. It may also promote the reliability and safety of autonomous drone for industrial or agricultural sites.

Acknowledgement

We thank Professor Sicun Gao for providing helpful suggestions regarding the further enhancement of the Lyapunov model. We also thank Ray Wang for advice on preprocessing the dataset.

References

[1] D. B. Chirila. *Towards lattice Boltzmann models for climate sciences: The GeLB programming language with applications*. PhD thesis, Universität Bremen, 2018.

[2] J. Z. Kolter and G. Manek. Learning stable deep dynamics models. *Advances in neural information processing systems*, 32, 2019.

[3] K. Stachenfeld, D. B. Fielding, D. Kochkov, M. Cranmer, T. Pfaff, J. Godwin, C. Cui, S. Ho, P. Battaglia, and A. Sanchez-Gonzalez. Learned coarse models for efficient turbulence simulation. *arXiv preprint arXiv:2112.15275*, 2021.

[4] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1457–1466, 2020.

Checklist

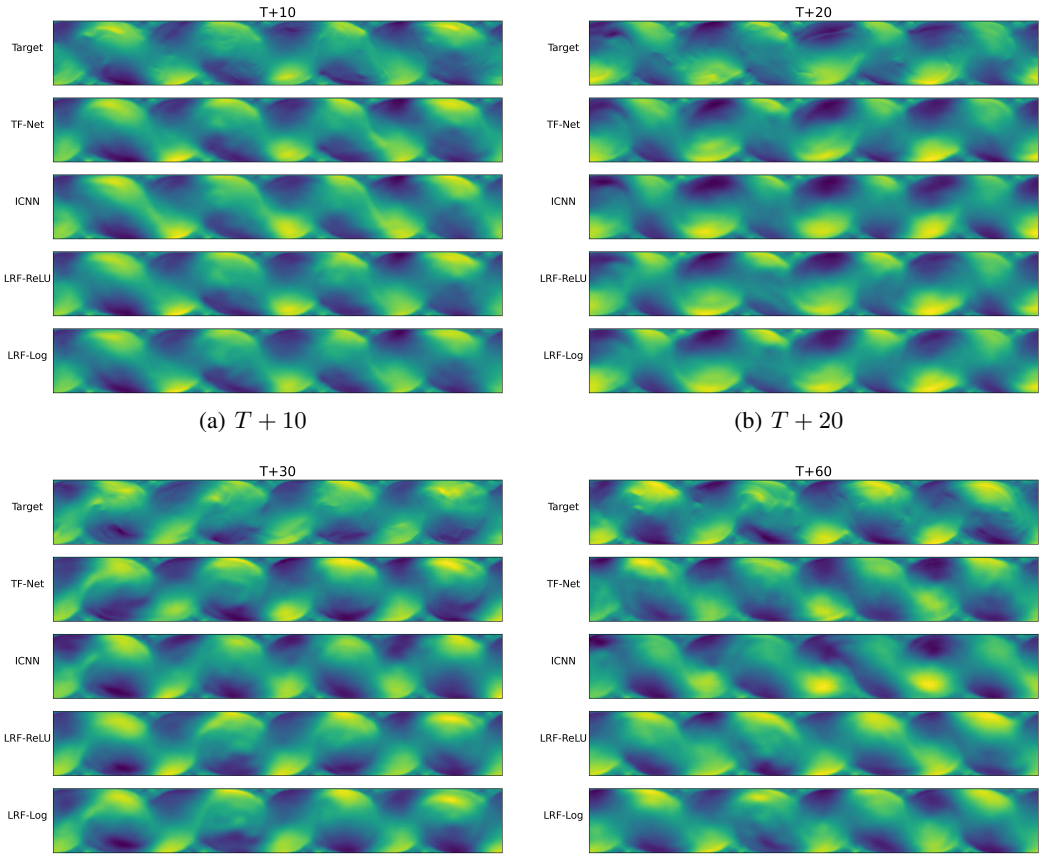
1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 4
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] All authors have read and confirmed
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We will make the source code publicly available later
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 3
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Figure 1(a)
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 3.2
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] All existing code, data, models are listed in References and cited in text
(b) Did you mention the license of the assets? [N/A]
(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] The data is publicly available
(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Visualization



(a) $T + 10$ (b) $T + 20$

(c) $T + 30$ (d) $T + 60$

Figure 2: Ground Truth (Target), and the predicted velocity fields $\mathbf{v}$ of TF-Net, Baseline (ICNN), and Regularizers (LRF-ReLU, LRF-Log); at time $T + 10$, $T + 20$, $T + 30$, and $T + 60$, where input sequence has length T