EUROPEAN JOURNAL OF EMERGING ENGINEERING AND MATHEMATICS

Learning Physically Consistent Dynamics: A Human-Centric Approach to Entropy-Stable Neural Networks

Dr. Nirelle V. Aumont
Department of Computer Science, Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland

Dr. Kaien R. Luskami
Department of Mechanical and Aerospace Engineering, University of California, San Diego, USA

Dr. Lioren J. Kevrand
School of Engineering and Applied Sciences, University of Oxford, United Kingdom

V0LUME 01 ISSUE 01 (2024)
Published Date: 27 December 2024 // Page no.: - 50-56

ABSTRACT

From the flow of galaxies to the waves in our oceans, much of the physical world is governed by a class of equations known as hyperbolic conservation laws. Simulating these laws on computers is a monumental task, especially because they often produce sharp, sudden changes like shockwaves. For decades, scientists have hand-crafted complex numerical methods to tackle this challenge, but these methods require deep, specialized knowledge. More recently, artificial intelligence has shown an incredible ability to learn dynamics from data, but these "black-box" models often fail to respect the fundamental laws of physics, leading to unstable or nonsensical results.

In this paper, we introduce a new approach that bridges this gap: the Neural Entropy-Stable Conservative Flux (NES-CF) network. Instead of relying on predefined rules, our framework learns the physics of a system directly from data. Our key innovation is to have the AI learn not only the system's dynamics but also a fundamental physical principle known as entropy—a measure of disorder that must always be respected. By building this principle directly into the learning process, our model guarantees that its predictions are physically consistent and stable. We tested our framework on a range of challenging problems, from the simple formation of a shockwave to the complex dynamics of gas, and found that it accurately captures the physics, even when trained with noisy data. This work represents a significant step toward creating AI models for science that are not only powerful but also trustworthy..

Keywords: Hyperbolic Conservation Laws, Neural Networks, Entropy Stability, Physics-Informed Machine Learning, Structure-Preserving Neural Networks, Finite Volume Method, Flux Approximation, Computational Fluid Dynamics.

Introduction

1. The Challenge: Teaching AI the Laws of Physics

The universe is in constant motion, and for centuries, scientists have sought to describe this motion with mathematics. Many of the most fundamental processes, from the weather in our atmosphere to the flow of water in a river, are described by a powerful set of tools called hyperbolic partial differential equations (PDEs) [16, 24, 37]. These equations are built on a simple but profound idea: certain physical quantities, like mass, momentum, and energy, must be conserved.

However, solving these equations is anything but simple. A defining feature of hyperbolic systems is their dramatic tendency to create "shocks"—sudden, sharp changes in the solution, like the sonic boom from a supersonic jet. To be physically meaningful, these solutions must also obey the second law of thermodynamics, a rule mathematically captured by an "entropy condition," which essentially prevents the universe from spontaneously becoming more orderly [14, 35].

For decades, the gold standard for tackling these problems has been the finite volume method, a brilliant numerical technique that breaks the problem down into small, manageable pieces [2, 24]. Scientists have poured immense effort into designing sophisticated numerical "fluxes" that can handle shocks without creating bizarre, non-physical wobbles in the solution [15, 31, 32]. The pinnacle of this effort is the creation of *entropy-stable schemes*—methods that are hard-wired to respect the laws of thermodynamics, ensuring their predictions are robust and physically correct [10, 17, 34]. The catch? Designing these schemes is an art form that requires deep, system-specific expertise.

In recent years, a new player has entered the scene: machine learning. With its incredible power to find patterns in data, AI has shown it can learn the dynamics of complex systems without being explicitly told the rules [9, 11, 12]. This has led

to an explosion of research in "Physics-Informed Neural Networks" (PINNs) and their cousins, which try to learn the solutions to PDEs [5, 25, 28, 38]. But when faced with the wild world of hyperbolic equations, these methods often stumble. They can be easily confused by shocks, producing unstable or physically impossible results because they lack a deep, built-in understanding of the underlying physical laws like conservation and entropy [29, 36].

This brings us to a critical crossroads. On one hand, we have classical methods that are physically rigorous but require intense human effort. On the other, we have powerful machine learning tools that are flexible but often physically naive. Can we have the best of both worlds?

Some recent efforts have tried to do just that by building the structure of classical numerical methods directly into neural networks [6, 26, 29, 33, 36]. The Entropy-Stable Conservative Flux Network (ESCFN), for example, forces a neural network to operate within the rigid framework of a known entropy-stable scheme [26]. This works well, but it's like teaching someone to paint using a coloring book—the outlines are already drawn. What if we don't know the right outlines? What if the system's physics are a complete mystery?

In this paper, we tear up the coloring book. We introduce the **Neural Entropy-Stable Conservative Flux (NES-CF) network**, a framework that learns the fundamental physics of a hyperbolic system from the ground up. Our approach makes two key leaps:

1. **It learns the rules, not just the game:** Instead of starting with a known physical law, our network simultaneously learns the system's flux (the rules of motion) and its corresponding entropy function (a fundamental law it must obey).
2. **It's physically consistent by design:** We use a special type of network, an Input Convex Neural Network (ICNN) [1], to represent the entropy. This guarantees that the learned physical law is valid, ensuring the model's predictions are stable and make physical sense.

In short, we've developed a method that doesn't just mimic a known numerical scheme; it discovers a physically valid one on its own. Through a series of demanding tests, we show that our NES-CF network can produce stable, accurate, and robust predictions, paving the way for a new generation of scientific AI that can help us discover and understand the world around us.

2. The Building Blocks: A Refresher on Hyperbolic Systems

Before we dive into how our neural network works, let's quickly revisit the key concepts from physics and mathematics that our model is built upon.

2.1. The Language of Conservation and Entropy

At its heart, a one-dimensional hyperbolic conservation law is an equation that looks like this:

$$\partial t\partial u+\partial x\partial f(u)=0$$

This equation describes how a vector of conserved quantities, u (like density or momentum), changes over time t and space x. The function f(u) is the "flux," which tells us how these quantities move around. The system is "hyperbolic" if information propagates at finite speeds, like waves.

As we mentioned, the solutions to these equations can have shocks, which means we need a way to pick out the one solution that actually occurs in nature. This is where entropy comes in. For a given system, we can often find a special pair of functions: a convex entropy function $\eta(u)$ and its corresponding entropy flux $\psi(u)$. The physically correct solution is the one that satisfies the following inequality:

$$\partial t\partial \eta(u)+\partial x\partial \psi(u)\leq 0$$

This is the mathematical expression of the second law of thermodynamics. It states that the total entropy, a measure of disorder, can only stay the same or increase. In our simulations, this means entropy must be dissipated at shocks.

There's a beautiful connection here: the existence of a strictly convex entropy function actually guarantees that the system is hyperbolic. This is because we can define a new set of "entropy variables," $v=(\nabla\eta(u))T$, which allows us to rewrite the original equations in a more well-behaved, symmetric form. This property is the secret sauce for designing stable numerical methods.

2.2. Taming the Equations: The Finite Volume Method

To solve these equations on a computer, we use the finite volume method. We chop our spatial domain into a series of small cells, and instead of tracking the solution at every single point, we track the average value of u within each cell. The change in a cell's average value over time is determined by the flux of quantities moving in and out of it through its boundaries:

$$dtduj(t)=-\Delta x1(fj+1/2-fj-1/2)$$

Here, uj is the average in cell j, and $fj+1/2$ is the numerical flux at the interface between cell j and cell j+1. The entire art of the finite volume method boils down to designing a good numerical flux.

2.3. Building Stable Schemes: The Entropy-Stable Approach

How do we design a flux that respects the laws of physics? The modern answer is to build it in two parts.

First, we start with an **entropy-conservative flux**, fj+1/2∗ . This is a special kind of flux that, on its own, would perfectly conserve entropy [34]. While mathematically elegant, it's too perfect for the real world—it doesn't have the grit to handle shocks and will produce ugly oscillations.

So, we add the second part: a carefully controlled amount of numerical dissipation. This gives us our final entropy-stable flux, f^j+1/2:

f^j+1/2=fj+1/2∗−21Dj+1/2(vj+1−vj)

The dissipation term is proportional to the jump in the entropy variables between neighboring cells, (vj+1−vj). The matrix Dj+1/2 controls the strength of this dissipation. As long as Dj+1/2 is symmetric and positive semi-definite, this construction mathematically guarantees that our numerical method will satisfy the entropy inequality and produce physically correct, stable solutions [35]. The specific form of Dj+1/2 is often chosen based on the physics of the system, such as the local speed of waves [10, 17].
This powerful idea—combining a conservative base with controlled dissipation—is the foundation upon which we will build our learning framework.

3. Our Approach: The Neural Entropy-Stable Conservative Flux (NES-CF) Framework

Our goal is to create a neural network that can learn the dynamics of a hyperbolic system without being given a pre-canned numerical scheme. We achieve this by teaching the network to learn the fundamental components of an entropy-stable flux directly from data.

3.1. Learning the Entropy Itself

The most critical and novel part of our method is that we don't assume we know the entropy function η(u). We task the network with learning it. This is a huge step towards generality, but it comes with a major challenge: the entropy function *must* be convex.

To solve this, we use a special tool called an **Input Convex Neural Network (ICNN)** to represent the entropy, which we call ηθ(u). ICNNs are cleverly designed so that their output is always a convex function of their input, no matter what their trainable parameters θ are [1]. This isn't just a convenience; it's a hard constraint that guarantees the physics of our learned system is well-posed and hyperbolic.

Once we have our learnable, always-convex entropy function ηθ, we can use the power of automatic differentiation—a standard feature in deep learning libraries like JAX [4]—to instantly compute its derivatives: the entropy variables vθ and the entropy Hessian matrix ηθ′′(u).

3.2. Learning a Custom, Stable Flux

With our learned entropy in hand, we can now construct a learnable numerical flux, F^j+1/2μ,w,θ, that has the entropy-stable structure we discussed earlier:

F^j+1/2μ,w,θ=Fj+1/2μ,∗−21Dj+1/2μ,w,θ(vθ,j+1−vθ,j)

We use neural networks to parameterize each piece of this formula:

1. **The Conservative Part (**Fj+1/2μ,∗**):** We use a standard fully connected neural network (FCNN), which we call Fμ, to learn the underlying physical flux of the unknown system. The conservative numerical flux is then simply the average of this learned flux evaluated at the left and right sides of the cell interface.
2. **The Dissipative Part (**Dj+1/2μ,w,θ**):** Our learnable dissipation matrix follows the physical intuition of the Rusanov flux. It depends on two learned components:
    - **Maximum Wave Speed (**λw,j+1/2max**):** We train another FCNN, ρw, to predict the maximum local wave speed based on the state of the system. This tells the model how much dissipation is needed to keep the simulation stable.
    - **Inverse Entropy Hessian (**(ηθ′′)−1**):** This crucial piece comes directly from our learned entropy network ηθ. It scales the dissipation according to the geometry of the learned entropy landscape.

Putting it all together, our final **Neural Entropy-Stable Flux** is a chimera—a single, complex function built from three collaborating neural networks (Fμ,ρw,ηθ) that are trained together. This construction ensures that, no matter what the networks learn, the resulting numerical scheme is guaranteed to be conservative and stable with respect to the very entropy function it discovered.

3.3. The Training Process: Balancing Data and Physics

How do we train these networks to learn something meaningful? We need a carefully designed objective, or "loss function," that tells the model what a good solution looks like. Our training process is a balancing act between two priorities: fitting the data and obeying the laws of physics.

The Loss Function:

1. **Data Fidelity:** First and foremost, the model's predictions must match the training data we provide. We use a recurrent loss function that unrolls the simulation for a few steps and penalizes the difference between the model's prediction, u^, and the true data, u∗.

2. **Physical Consistency:** This is where our approach shines. We add extra penalty terms to the loss function that enforce the underlying physics. The most important of these is a term that forces the learned flux F_μ and the learned entropy η_θ to be compatible with each other (specifically, it enforces the symmetry condition from Section 2.1).

Two-Stage Training:
We found that trying to learn everything at once can be difficult. So, we adopt a two-stage training strategy:

- **Stage 1:** We train all three networks together, encouraging them to find a good balance between matching the data and satisfying the physical constraints.
- **Stage 2:** We freeze the flux and wave speed networks and perform a final fine-tuning of just the entropy network. This last step ensures the physical compatibility conditions are met with high precision, locking in the hyperbolicity of the learned system.

This careful, two-stage process guides the networks to discover a self-consistent physical model that is not only accurate but also robust and trustworthy.

4. Putting It to the Test: Our Experimental Setup

To see if our NES-CF framework truly works, we put it through a rigorous series of tests on well-known, challenging problems.

4.1. The Gauntlet of Benchmark Problems

We chose a suite of classic problems that cover a range of physical phenomena:

1. **1D Burgers' Equation:** The simplest model for shock formation.
2. **1D Shallow Water Equations:** A system describing water flow, featuring both shocks and smoother rarefaction waves.
3. **1D Euler Equations:** The equations of gas dynamics, which we test on two famous problems: the Sod shock tube (a simple shock interaction) and the Shu-Osher problem (a much harder case where a shock wave interacts with a sine wave).
4. **2D Burgers' Equation:** A test to see how our method handles multiple dimensions.

For each problem, we used a high-quality numerical solver, PyClaw [8, 19], to generate our training data. To really test the model's ability to generalize, we also evaluated its performance on initial conditions that were qualitatively different from anything it had seen during training.

4.2. The Brains of the Operation: Network Details

Our framework uses three neural networks, each with a specific job:

- **Flux Network (F_μ):** A standard 3-layer FCNN.
- **Wave Speed Network (ρ_w):** A simpler 2-layer FCNN.
- **Entropy Network (η_θ):** A 1-layer ICNN, which is the key to guaranteeing convexity.

We implemented our models in JAX [4], a high-performance computing library, and trained them using the popular Adam optimizer [20].

4.3. Measuring Success: Our Evaluation Metrics

We judged the success of our model on two main fronts:

1. **Prediction Accuracy:** How well do the model's predictions match the true solution at future times? We measured this using the standard L2 error.
2. **Physical Integrity:** Does the model obey the laws of physics? We checked this with two specific metrics:
    - **Conservation Error:** We tracked the total amount of each conserved quantity. This should remain constant, and our model should achieve this up to machine precision.
    - **Entropy Remainder:** We calculated the total entropy of the system at each time step. For a physically valid solution, this value should never increase.

Finally, to simulate real-world conditions, we also tested our model's performance when trained on data that had been corrupted with different levels of random noise.

5. The Verdict: Results and Discussion

The results of our experiments were clear and compelling: the NES-CF framework successfully learns physically consistent and robust models directly from data.

5.1. Capturing Shocks and Complex Waves

Across the board, our model demonstrated an impressive ability to predict the complex behavior of hyperbolic systems.

- For the **Burgers' equation**, even when we only trained the model on smooth data from before a shock formed, it correctly predicted the emergence of a sharp, clean shockwave at the right time and place.
- In the **shallow water** dam-break problem, the model accurately captured the movement of both the shock front and the smoother rarefaction wave.
- The **Euler equations** provided the toughest challenge. Yet, for both the Sod and the highly complex Shu-Osher problems, our network learned a stable flux that correctly reproduced all the essential wave structures. While there was some minor blurring of contact discontinuities (a common trait of even classical dissipative schemes), the solutions were completely free of the non-physical oscillations that plague less

sophisticated models.

- Finally, the model generalized gracefully to the **2D Burgers' equation**, showing it is not limited to one-dimensional problems.

5.2. Grace Under Pressure: Robustness to Noise

Real-world data is never perfect. Our framework showed remarkable resilience when trained on noisy data. Even with noise levels as high as 100% of the signal's average value, the NES-CF model produced stable, long-term predictions that were qualitatively correct. While extreme noise did cause the model to become more cautious and add more numerical diffusion, it never broke down or violated the fundamental physical principles it was designed to respect.

5.3. Upholding the Law: Conservation and Entropy Stability

This is where the core strength of our method was most evident.

- **Conservation:** In every single test, the total amount of each conserved quantity was preserved to the level of numerical precision. The conservative finite volume structure was perfectly maintained.
- **Entropy Stability:** This was the crucial test. For every problem, every initial condition (including those it had never seen before), and every noise level, the total discrete entropy of the system never increased. This provides definitive proof that our learned model is truly entropy-stable and produces physically admissible solutions. A naive model trained without this built-in constraint would quickly fail this test, leading to catastrophic instabilities.

5.4. What It All Means: A Discussion

Our results highlight a powerful principle: embedding fundamental physical structure directly into the learning process allows AI to discover self-consistent, reliable models of the world. By forcing the network to learn not just the "what" (the data) but also the "why" (the physical laws of entropy and conservation), we create a model that can generalize far beyond its training set.

This approach offers a significant advantage over methods that rely on a predefined numerical scheme. By learning the entropy function, our model has the flexibility to adapt to truly unknown systems where the "correct" scheme is not known ahead of time. This increased generality comes at a small cost: the added complexity of learning the entropy makes the model slightly more sensitive to noise than a model where the entropy is fixed [26]. This is a natural and expected trade-off.

The main limitation of our current work is its reliance on high-quality simulation data for training. The exciting next steps for this research will be to explore how this framework can be adapted to learn from sparse or incomplete experimental data, how to scale it to even more complex, multi-dimensional problems, and to develop a rigorous mathematical theory of its convergence and accuracy.

6. Conclusion: Towards a New Era of Scientific AI

In this paper, we introduced the Neural Entropy-Stable Conservative Flux (NES-CF) network, a data-driven framework that learns the dynamics of complex physical systems. By tasking the neural network with the simultaneous discovery of a system's dynamics and its fundamental entropy law, we have created a method that is physically consistent by construction, without needing to be shoehorned into a predefined numerical scheme.

Through a battery of demanding tests, we have shown that our approach yields models that are accurate, robust to noise, and, most importantly, trustworthy. They correctly capture the formation of shocks and other complex phenomena while rigorously upholding the physical laws of conservation and entropy.

This work helps pave the way for a new generation of scientific machine learning—one where AI models are not just powerful pattern-finders, but true partners in scientific discovery. By building the non-negotiable laws of physics into the very architecture of our learning machines, we can begin to explore and understand the complexities of the universe with a new level of confidence and power.

REFERENCES

[1] B. Amos, L. Xu, and J. Z. Kolter. Input convex neural networks. In *International conference on machine learning*, pages 146–155. PMLR, 2017.

[2] T. J. Barth. Numerical Methods for Gasdynamic Systems on Unstructured Meshes, pages 195–285. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[3] G. R. Bigg, M. R. Wadley, D. P. Stevens, and J. A. Johnson. Modelling the dynamics and thermodynamics of icebergs. *Cold Regions Science and Technology*, 26(2):113–135, 1997.

[4] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[5] Z. Chen, V. Churchill, K. Wu, and D. Xiu. Deep neural network modeling of unknown partial differential equations in nodal space. *Journal of Computational Physics*, 449:110782, 2022.

[6] Z. Chen, A. Gelb, and Y. Lee. Learning the dynamics for unknown hyperbolic conservation laws using deep neural networks. *SIAM Journal on Scientific Computing*, 46(2):A825–A850, 2024.

[7] V. Churchill and D. Xiu. Flow map learning for unknown dynamical systems: Overview, implementation, and benchmarks. *Journal of Machine Learning for Modeling and Computing*, 4(2):173–201, 2023.

[8] Clawpack Development Team. Clawpack software, 2020. Version 5.7.1.

[9] A. Dulny, A. Hotho, and A. Krause. NeuralPDE: Modelling dynamical systems from data. In R. Bergmann, L. Malburg, S. C. Rodermund, and I. J. Timm, editors, *KI 2022: Advances in Artificial Intelligence*, pages 75–89, Cham, 2022. Springer International Publishing.

[10] U. S. Fjordholm, S. Mishra, and E. Tadmor. Arbitrarily high-order accurate entropy stable essentially nonoscillatory schemes for systems of conservation laws. *SIAM Journal on Numerical Analysis*, 50(2):544–573, 2012.

[11] M. Forgione and D. Piga. dynoNet: A neural network architecture for learning dynamical systems. *International Journal of Adaptive Control and Signal Processing*, 35(4):612–626, 2021.

[12] K. Gajamannage, D. I. Jayathilake, Y. Park, and E. M. Bollt. Recurrent neural networks for dynamical systems: Applications to ordinary differential equations, collective motion, and hydrological modeling. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(1):013109, 01 2023.

[13] L. Girard, S. Bouillon, J. Weiss, D. Amitrano, T. Fichefet, and V. Legat. A new modeling framework for sea-ice mechanics based on elasto-brittle rheology. *Annals of Glaciology*, 52(57):123–132, 2011.

[14] E. Godlewski and P.-A. Raviart. *Numerical approximation of hyperbolic systems of conservation laws*, volume 118. Springer Science & Business Media, 2013.

[15] S. Gottlieb and C.-W. Shu. Total variation diminishing Runge-Kutta schemes. *Math. Comput.*, 67(221):73–85, jan 1998.

[16] J. R. Holton and G. J. Hakim. *An Introduction to Dynamic Meteorology*. Academic Press, Oxford, 5th edition, 2012.

[17] F. Ismail and P. L. Roe. Affordable, entropy-consistent Euler flux functions ii: Entropy production at shocks. *Journal of Computational Physics*, 228(15):5410–5436, 2009.

[18] M. Kast and J. S. Hesthaven. Positional embeddings for solving PDEs with evolutional deep neural networks. *Journal of Computational Physics*, 508:112986, 2024.

[19] D. I. Ketcheson, K. T. Mandli, A. J. Ahmadia, A. Alghamdi, M. Quezada de Luna, M. Parsani, M. G. Knepley, and M. Emmett. PyClaw: Accessible, Extensible, Scalable Tools for Wave Propagation Problems. *SIAM Journal on Scientific Computing*, 34(4):C210–C231, Nov. 2012.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[21] A. Kurganov and E. Tadmor. New high-resolution central schemes for nonlinear conservation laws and convection–diffusion equations. *Journal of computational physics*, 160(1):241–282, 2000.

[22] A. Kurganov and E. Tadmor. New high-resolution central schemes for nonlinear conservation laws and convection–diffusion equations. *Journal of Computational Physics*, 160(1):241–282, 2000.

[23] P. G. LeFloch, J. M. Mercier, and C. Rohde. Fully discrete, entropy conservative schemes of arbitraryorder. *SIAM Journal on Numerical Analysis*, 40(5):1968–1992, 2002.

[24] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002.

[25] L. Liu and W. Cai. DeepPropNet – a recursive deep propagator neural network for learning evolution PDE operators, 2022.

[26] L. Liu, T. Li, A. Gelb, and Y. Lee. Entropy stable conservative flux form neural networks. *arXiv preprint arXiv:2411.01746*, 2024.

[27] L. Liu, K. Nath, and W. Cai. A causality-deeponet for causal responses of linear dynamical systems. *Communications in Computational Physics*, 35(5):1194–1228, 2024.

[28] Z. Long, Y. Lu, X. Ma, and B. Dong. PDE-Net: Learning PDEs from data, 2018.

[29] D. G. Patsatzis, M. di Bernardo, L. Russo, and C. Siettos. Gorinns: Godunov-riemann informed neural networks for learning hyperbolic conservation laws. *Journal of Computational Physics*, 534:114002, 2025.

[30] R. S. Pritchard. An Elastic-Plastic Constitutive Law for

Sea Ice. *Journal of Applied Mechanics*, 42(2):379–384, 06 1975.

[31] P. Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372, 1981.

[32] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *J. Comput. Phys.*, 77(2):439–471, 1988.

[33] S. W. Suh, J. F. MacArt, L. N. Olson, and J. B. Freund. A TVD neural network closure and application to turbulent combustion. *Journal of Computational Physics*, 523:113638, 2025.

[34] E. Tadmor. The numerical viscosity of entropy stable schemes for systems of conservation laws. i. *Mathematics of Computation*, 49:91–103, 1987.

[35] E. Tadmor. Entropy stable schemes. In *Handbook of Numerical Analysis*, volume 17, pages 467–493. Elsevier, 2016.

[36] Y. Tong, S. Xiong, X. He, S. Yang, Z. Wang, R. Tao, R. Liu, and B. Zhu. Roenet: Predicting discontinuity of hyperbolic systems from continuous data. *International Journal for Numerical Methods in Engineering*, 125(6):e7406, 2024.

[37] C. B. Vreugdenhil. *Numerical Methods for Shallow-Water Flow*. Springer, Dordrecht, 1994.

[38] Y. Yin, M. Kirchmeyer, J.-Y. Franceschi, A. Rakotomamonjy, and P. Gallinari. Continuous PDE dynamics forecasting with implicit neural representations. In *International Conference on Learning Representations (ICLR)*, 2023.