Riemann Tensor Neural Networks: Learning Conservative Systems with Physics-Constrained Networks

Anas Jnini ✉ Lorenzo Breschi Flavio Vella

University of Trento, Italy

✉ Correspondence to: anas.jnini@unitn.it

Full Paper Github

The Challenge of Physics-Informed Machine Learning

- **Standard Approach (PINNs):** A neural network u_θ is trained to approximate a PDE solution by penalizing the PDE residual and boundary/initial conditions in a loss function.

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BC}} + \mathcal{L}_{\text{IC}}$$

- **The Problem:** This "soft constraint" method creates a highly challenging optimization problem. The differential operators in the loss lead to ill-conditioned Hessians and extremely slow convergence for standard optimizers.
- **Our Motivation:** Simplify the optimization landscape by designing a neural network architecture that satisfies key physical laws **by construction**.

Unifying Conservation Laws with DFSTs

Many fluid dynamics systems are governed by conservation of mass and momentum.

$$\partial_t \rho + \tilde{\nabla} \cdot (\rho \mathbf{u}) = 0$$
$$\partial_t (\rho \mathbf{u}) + \tilde{\nabla} \cdot (\rho \mathbf{u} \otimes \mathbf{u} + \sigma) = 0$$

These can be unified into a single mathematical object on a **Divergence-Free Symmetric Tensor (DFST)**, S:

$$\text{Div}_{t,x}(S) = 0$$

where S is the flux tensor:

$$S = \begin{pmatrix} \rho & m^\top \\ m & \dfrac{m \otimes m}{\rho} + \sigma \end{pmatrix}$$

The Riemann Tensor Neural Network (RTNN) Architecture

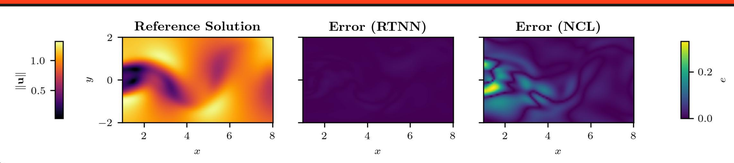
An RTNN constructs a DFST in three steps:

1. **Learn Coefficients:** An MLP, NN_θ, maps input coordinates x to a set of scalar coefficients $\{c_{ij}(x;\theta)\}$.
2. **Compute Hessians:** Use automatic differentiation to compute the Hessian components of these coefficients, $\partial^c \partial^d c_{ij}$.
3. **Construct the Tensor:** The final output is formed by contracting the Hessians with a fixed basis of constant tensors $\{T^{(i,j)}\}$ that satisfy the required symmetries:

$$\boxed{S_\theta(x)_{ab} = \sum_{1 \le i \le j \le m} T^{(i,j)}_{acbd} \partial^c \partial^d c_{ij}(x;\theta)}$$

This construction guarantees the output S_θ is always a symmetric, divergence-free tensor.

RTNN leads to Physically consistent solutions



Architectural Question

How can we design a neural network that outputs a tensor S_θ that is **guaranteed by construction** to be divergence-free and symmetric?

$$\nabla \cdot S_\theta = 0 \quad \text{and} \quad S_\theta = S_\theta^\top$$

Main Result I: Representation Theorem for DFSTs

Any sufficiently smooth DFST, S_{ab}, can be expressed as the double covariant derivative of a $(0,4)$-tensor potential, K_{acbd}, which satisfies the following symmetries:

1. **Antisymmetry within index pairs:**

$$K_{(ab)cd} = 0$$
$$K_{ab(cd)} = 0$$

2. **Symmetry between pairs:**

$$K_{abcd} = K_{cdab}$$

The resulting DFST is given by:

$$\boxed{S_{ab} = \nabla^c \nabla^d K_{acbd}}$$

Main Result II: Universal Approximation

Our RTNN architecture is expressive enough to represent any physical system that can be described by a sufficiently smooth DFST. For any such tensor S, and for any $\epsilon > 0$, there exists an RTNN S_θ such that:

$$\sup_{x \in \Omega} \|S(x) - S_\theta(x)\|_{\text{Fro}} < \varepsilon$$

Decomposition Theorem for the Potential Tensor K

Any tensor K with the required Riemann-like symmetries is isomorphic to a symmetric bilinear form on the space of 2-forms ($\text{Sym}^2(\Lambda^2 V^*)$). This allows us to construct a basis $\{T^{(i,j)}\}$ for the potential tensors from a basis of 2-forms $\{\omega_k\}$:

$$T^{(i,j)}_{abcd} = \omega_i(e^*_a \wedge e^*_b)\omega_j(e^*_c \wedge e^*_d) + \omega_j(e^*_a \wedge e^*_b)\omega_i(e^*_c \wedge e^*_d)$$

The potential tensor K can then be decomposed onto this fixed basis using a set of learnable scalar functions $\{c_{ij}(x)\}$:

$$K_{acbd}(x) = \sum_{1 \le i \le j \le m} c_{ij}(x) T^{(i,j)}_{acbd}$$

The network's task is to learn these scalar coefficients. The number of functions to learn, $\frac{m(m+1)}{2}$, depends only on the number of spacetime dimensions, n, where $m = \frac{n(n-1)}{2}$

RTNN Training Paradigm Loss Formulation

The RTNN outputs a single flux tensor S_θ. The physical fields are explicitly extracted from its components. Since mass and momentum conservation are satisfied by construction, the loss function only penalizes the remaining physics.

1. Field Extraction:

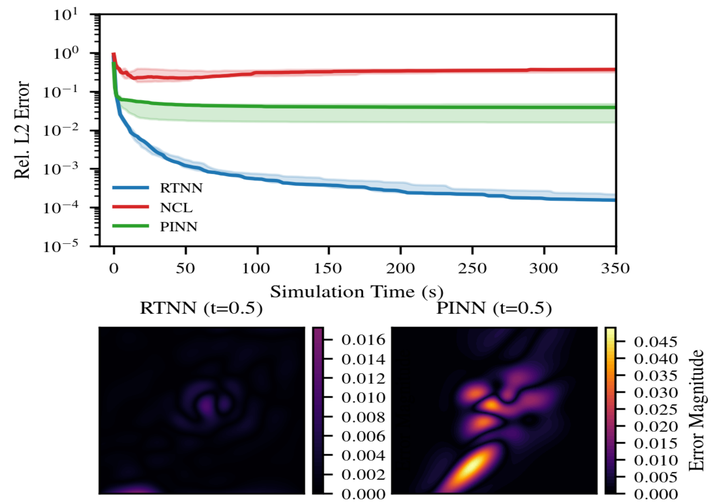
$$\rho_\theta = (S_\theta)_{00}, \quad m_\theta = (S_\theta)_{1:n,0}, \quad \sigma_\theta = (S_\theta)_{1:n,1:n} - \frac{m_\theta \otimes m_\theta}{\rho_\theta} - \mathcal{M}_\theta$$

(Note: Maxwell stress $\mathcal{M}_\theta = 0$ for pure fluid dynamics cases)

2. Loss Penalties:

Problem	Primary Loss Penalty Term
Euler (Inviscid)	Enforce zero shear stress: $\mathcal{L}_\sigma = \|\sigma_\theta - \frac{1}{n}\text{tr}(\sigma_\theta)\mathbf{I}\|^2$
Navier-Stokes	Enforce the viscous constitutive law: $\mathcal{L}_\sigma = \|(\sigma_\theta - \frac{1}{n}\text{tr}(\sigma_\theta)\mathbf{I}) - \nu(\nabla \mathbf{u}_\theta + (\nabla \mathbf{u}_\theta)^T)\|^2$
MHD	Enforce viscous law and magnetic induction: $\mathcal{L}_{\text{total}} = \mathcal{L}_\sigma + \|\mathcal{R}_{\text{induction}}\|^2$

RTNN improves training convergence and final accuracy



Quantitative Results: RTNN vs. Baselines

Experiment	RTNN Error	Baseline Error	Method	Gain
Euler Vortex	**9.9e-05**	3.8e-02	PINN	**385x**
NS Cylinder	**5.7e-03**	2.5e-02	NCL	**4.5x**
NS Airfoil	**1.4e-02**	1.5e-01	NCL	**10.6x**
NS Beltrami	**4.3e-04**	1.4e-03	PINN	**3.3x**
MHD (Velocity)	**2.3e-02**	1.6e-01	CurlSPINN	**6.9x**
MHD (B-Field)	**1.0e-01**	1.5e-01	CurlSPINN	**1.5x**