How To Design Stable Machine Learned Solvers For Scalar Hyperbolic PDEs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Machine learned partial differential equation (PDE) solvers trade the robustness of classical numerical methods for potential gains in accuracy and/or speed. A key challenge for machine learned PDE solvers is to maintain physical constraints that will improve robustness while still retaining the flexibility that allows these methods to be accurate. In this paper, we show how to design solvers for scalar hyperbolic PDEs in conservation form that are stable by construction. We call our technique 'global stabilization.' Unlike classical numerical methods, which guarantee stability by putting local constraints on the solver, global stabilization adjusts the time-derivative of the discrete solution to ensure that global invariants and stability conditions are satisfied. Although global stabilization can be used to ensure the stability of any scalar hyperbolic PDE solver that uses method of lines, we believe it is most useful as an error-correcting algorithm for machine learned solvers.

1 Introduction

Scientists and engineers are interested in solving partial differential equations (PDEs). Many PDEs cannot be solved analytically, and must be approximated using discrete numerical algorithms. We refer to these discrete numerical algorithms as PDE solvers. The fundamental challenge for PDE solvers is to balance between two competing objectives: first, to find an accurate approximation to the solution of the equation, and second, to do so with as few computational resources as possible.

In recent years, scientists and engineers have attempted to use machine learning (ML) to design new and better PDE solvers [36, 2, 40, 26, 16, 39, 3, 37]. On certain problems, machine learned PDE solvers have achieved high accuracy at low computational cost [20, 34, 21, 13, 23]. However, these high-performing machine learned PDE solvers suffer from at least two major problems. First, they struggle to generalize to conditions outside of the training data. Second, they tend to have no guarantees of stability as $t \to \infty$ and as a result often do not maintain stable solutions.

We consider scalar hyperbolic PDEs written in conservation form, given by

$$\frac{\partial u}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{f}(u) = 0. \tag{1}$$

If machine learned solvers for eq. (1) were somehow perfectly accurate, then stability would not be a concern because the solver would simply give the correct answer for all t. But, for a variety of reasons, machine learned solvers are not and will never be perfectly accurate. Some amount of error is inevitable, so the question becomes how to constrain the machine learned solver to give us the sorts of errors that we are willing to tolerate. Although the answer to this question is problem dependent, we take the view that a solution that blows up as $t \to \infty$ is an intolerable error. [2] and

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

[40] agree, writing that "figuring out how to guarantee stability" of machine learned PDE solvers is an "important topic for future work."

The purpose of this paper is to demonstrate how to design solvers for eq. (1) that are stable by construction. These solvers guarantee both mass conservation and stability as $t \to \infty$ for a subset of PDEs that are highly relevant in the physical sciences and engineering. We call our technique 'global stabilization'. We present the global stabilization technique in one and two dimensions for rectangular grids with periodic boundary conditions, though the method can easily be generalized to higher dimensional PDEs. We note that the method can also be used when the right hand side (RHS) of eq. (1) is nonzero by treating the right hand side terms using classical techniques and applying global stabilization only to the divergence term.

2 Sufficient Conditions for Stability

We consider solvers that use the method of lines (MOL). MOL converts a PDE to a system of ordinary differential equations (ODEs) by discretizing the spatial dimension(s). We restrict ourselves to rectangular, periodic grids with uniform cell size. Our results can easily be generalized to the case where the grid spacing is non-uniform.

Equation (1) is a statement of the conservation of $\int u \, d\boldsymbol{x}$, which we call 'conservation of mass'. In 1D, an integral makes this apparent: $\frac{d}{dt}\int_0^L u(x)\,dx = \int_0^L \frac{\partial u}{\partial t}\,dx = -\int_0^L \frac{\partial f}{\partial x}\,dx = f(0) - f(L) = 0$. In words: the total rate of change of u is equal to the flux through the boundaries; for a periodic system this equals zero. Scalar hyperbolic PDEs also satisfy an entropy inequality given by

$$\frac{\partial S(u)}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{F}(u) \geq 0 \tag{2}$$

for any $S''(u) \leq 0$ and $\boldsymbol{F}(u) = \int^u S'(u)\boldsymbol{f}'(u)\,du$ [9, 33]. Performing an integration over x in 1D shows that the total entropy S is non-decreasing: $\frac{d}{dt}\int S(u)\,dx \geq F(0) - F(L) = 0$. By choosing $S(u) = {-u^2}/{2}$, we can see that the discrete L2 norm is non-increasing: $\frac{d}{dt}\int \frac{u^2}{2}\,dx \leq 0$. We now list four conditions sufficient for ensuring that a solver of eq. (1) is stable:

1. The solver uses method of lines (MOL).
2. In the continuous-time limit, the integral of the solution (total mass) is conserved.
3. In the continuous-time limit, the discrete L2 norm of the solution is non-increasing.
4. The ODE integration is stable.

Condition 4 can usually be satisfied by using a strong stability preserving Runge Kutta ODE integration method [12, 11] and choosing the timestep to satisfy a CFL condition. We now demonstrate how to design MOL solvers that satisfy conditions 2 and 3 by construction.

3 Stability of Finite Volume (FV)

We focus on developing stable finite volume (FV) schemes. In 1D, the solution average in the jth cell is given by u_j, in 2D the solution average in the i, jth cell is given by $u_{i,j}$, etc. Integrating eq. (1) over a grid cell gives the standard equations for the time-derivative of a FV scheme in 1D and 2D:

$$\frac{\partial u_j}{\partial t} + \frac{f_{j+\frac{1}{2}} - f_{j-\frac{1}{2}}}{\Delta x} = 0 \quad \text{(3a)} \qquad \frac{\partial u_{i,j}}{\partial t} + \frac{f^x_{i+\frac{1}{2},j} - f^x_{i-\frac{1}{2},j}}{\Delta x} + \frac{f^y_{i,j+\frac{1}{2}} - f^y_{i,j-\frac{1}{2}}}{\Delta y} = 0. \quad \text{(3b)}$$

Equations (3a) and (3b) are exact, so long as the flux f at cell boundaries is exact. Therefore, the key challenge for a finite volume scheme is to accurately reconstruct the flux at cell boundaries.

Finite volume schemes conserve the discrete mass by construction, as we can see in 1D from $\frac{d}{dt}\sum_j u_j \Delta x = \Delta x \sum_j \frac{\partial u_j}{\partial t} = -\sum_j (f_{j+1/2} - f_{j-1/2}) = 0$. The rate of change of the discrete L2 norm in 1D is given by $\frac{d}{dt}\sum_j {u_j^2}/{2}\Delta x = \Delta x \sum_j u_j \frac{\partial u_j}{\partial t} = -\sum_j u_j (f_{j+1/2} - f_{j-1/2})$, which implies that a 1D FV scheme will be stable if

$$\sum_j u_j \left(f_{j+\frac{1}{2}} - f_{j-\frac{1}{2}}\right) \geq 0. \tag{4}$$

Because eqs. (3a) and (3b) are exact, a machine learned PDE solver that uses the finite volume formulation only needs to predict f at cell boundaries. Without constraints on $f_{j+1/2}$, however, eq. (4) will not be satisfied and the machine learned solver will not guarantee stability.

Classical FV PDE solvers ensure that eq. (4) is satisfied through careful choice of flux. For example, on the 1D advection equation $f = cu$ the centered flux $f_{j+1/2} = {}^{c(u_j+u_{j+1})}/_{2}$ conserves the discrete L2 norm exactly but introduces spurious oscillations around discontinuities, while the upwind flux can be written as *upwind* = *centered* + *diffusion* and decays the discrete L2 norm due to the numerical diffusion. For non-linear $f(u)$, Godunov flux [7] introduces numerical diffusion and decays the discrete L2 norm [24]. The second-order MUSCL scheme [38] is total variation diminishing (TVD) and introduces numerical diffusion at extremum and discontinuities. Each of these fluxes is *local*; i.e., $f_{j+1/2}$ is reconstructed from the solution at a few of the nearest points.

Classical techniques ensure stability either by using an entropy preserving centered scheme, or by satisfying a discrete analogue of the entropy inequality of eq. (2) locally, which introduces numerical diffusion everywhere, or by ensuring that the scheme is TVD, which involves reverting to a highly diffusive 1st-order method at extremum. While machine learned FV solvers are capable of ensuring stability via classical techniques [27, 25], we anticipate that such choices of flux will be too restrictive and too diffusive to outperform state-of-the-art numerical methods. Thus, in sections 4 and 6 we propose a different type of stability, one that allows a discrete analogue of the entropy inequality in eq. (2) to be violated locally but not globally.

4 Global Stabilization of Flux Predicting FV Schemes

We now introduce 'global stabilization,' a novel technique that guarantees the L2 stability of any FV scheme given by eq. (3a) or (3b). To derive this method in 1D, we begin with eq. (4) and perform summation by parts, giving $\sum_j u_j(f_{j+1/2} - f_{j-1/2}) = -\sum_j f_{j+1/2}\big(u_{j+1} - u_j\big)$. Thus, eq. (4) will be satified if the following transformation is made to $f_{j+1/2}$:

$$C \equiv -\sum_j f_{j+\frac{1}{2}}(u_{j+1} - u_j) \qquad f_{j+\frac{1}{2}} \Rightarrow f_{j+\frac{1}{2}} + \frac{(C - {}^{dS_{tot}}/_{dt})(u_{j+1} - u_j)^{\beta}}{\sum_k (u_{k+1} - u_k)^{\beta+1}} \tag{5}$$

for any scalars ${}^{dS_{tot}}/_{dt} \geq 0$ and $\beta \neq 0$. Note that ${}^{dS_{tot}}/_{dt}$ is a user-defined quantity that sets the rate that the discrete L2 norm decreases. A similar calculation in 2D reveals that the rate of change of the discrete L2 norm is given by

$$\frac{d}{dt}\sum_{i,j}\frac{u_{i,j}^2}{2}\Delta x \Delta y = \Delta y \sum_{i,j} f^x_{i+\frac{1}{2},j}(u_{i+1,j} - u_{i,j}) + \Delta x \sum_{i,j} f^y_{i,j+\frac{1}{2}}(u_{i,j+1} - u_{i,j}) \leq 0. \tag{6}$$

Equation (6) will be satisfied if the following transformations are made to $f^x_{i+\frac{1}{2},j}$ and $f^y_{i,j+\frac{1}{2}}$:

$$C^x \equiv \Delta y \sum_{i,j} f^x_{i+\frac{1}{2},j}\big(u_{i,j} - u_{i+1,j}\big) \quad f^x_{i+\frac{1}{2},j} \Rightarrow f^x_{i+\frac{1}{2},j} + \frac{(C^x - \frac{dS^x_{tot}}{dt})(u_{i+1,j} - u_{i,j})^{\beta^x}}{\Delta y \sum_{k,l}(u_{k+1,l} - u_{k,l})^{\beta^x+1}} \tag{7a}$$

$$C^y \equiv \Delta x \sum_{i,j} f^y_{i,j+\frac{1}{2}}\big(u_{i,j} - u_{i,j+1}\big) \quad f^y_{i,j+\frac{1}{2}} \Rightarrow f^y_{i,j+\frac{1}{2}} + \frac{(C^y - \frac{dS^y_{tot}}{dt})(u_{i,j+1} - u_{i,j})^{\beta^y}}{\Delta x \sum_{k,l}(u_{k,l+1} - u_{k,l})^{\beta^y+1}} \tag{7b}$$

for any scalars ${}^{dS^x_{tot}}/_{dt}$, ${}^{dS^y_{tot}}/_{dt}$, β^x, β^y where ${}^{dS^x_{tot}}/_{dt} + {}^{dS^y_{tot}}/_{dt} \geq 0$ and $\beta^x, \beta^y \neq 0$. Equations (5), (7a) and (7b) are the main results of section 4; for scalar conservation form PDEs in 1D and 2D they ensure that the discrete L2 norm will be non-increasing in the time-continuous limit. In all our experiments, we set $\beta = \beta^x = \beta^y = 1$. In section 5, we discuss how to set ${}^{dS_{tot}}/_{dt}$ for machine learned PDE solvers. Note that with global stabilization $f_{j+1/2}$ depends on the solution over the entire domain, which violates the physical property that in a hyperbolic PDE information propogates at finite speed.

Global stabilization can be used either to stabilize an unstable numerical method or to reduce or eliminate numerical damping from a stable numerical method.

Figure 1 demonstrates how global stabilization can stabilize an unstable scheme. On the inviscid Burgers equation the centered flux $f_{j+1/2} = {}^{(u_j^2+u_{j+1}^2)}/_{4}$, shown in red, is unstable. We apply global

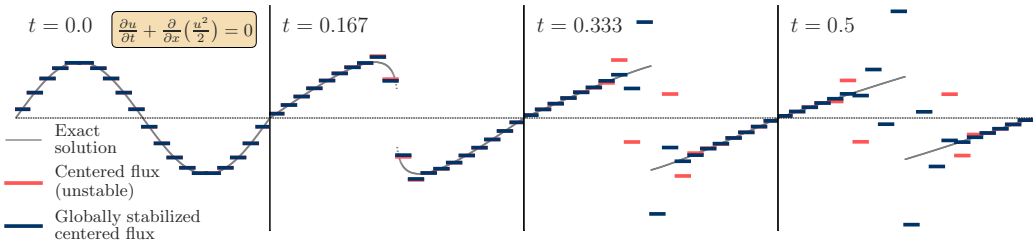
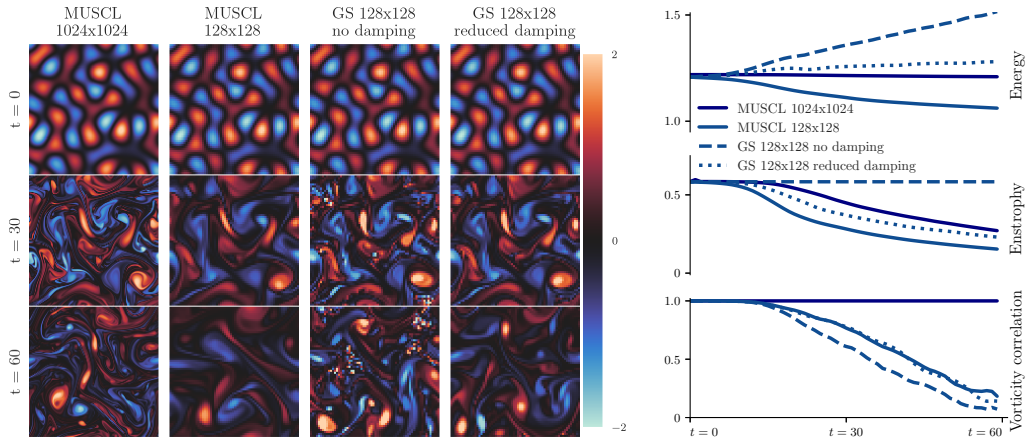


Figure 1: Global stabilization turns an unstable solver (red) into a stable solver (blue). While centered flux $f_{j+1/2} = (u_j^2 + u_{j+1}^2)/4$ is an unstable choice of flux on the inviscid Burgers equation, global stabilization ensures that the discrete L2 norm is conserved.



(a) (b)

Figure 2: Applying global stabilization to a stable FV scheme can reduce or eliminate numerical diffusion, but at the cost of introducing spurious high-k oscillations. (a) Images of the vorticity χ evolving under the incompressible Euler equations. The first and second columns show the baseline MUSCL scheme at high and low resolution. The third and fourth columns show the baseline MUSCL scheme with global stabilization (GS), either with no numerical damping or with numerical damping reduced by 75%. (b) Energy, enstrophy, and vorticity correlation over time. We use vorticity correlation as a benchmark measure of accuracy.

stabilization to the centered flux with $dS_{tot}/dt = 0$, shown in blue. This leads to exact conservation of the L2 norm and a stable numerical method. Our initial condition is $u(x) = \sin x$.

Note that the globally stabilized centered flux solution in fig. 1 conserves both the discrete mass and the discrete L2 norm, but does not maintain a discrete analogue of the total variation diminishing (TVD) property [7] of the scalar Burgers equation. As a result, the globally stable solution develops high-k oscillations; these spurious oscillations tend to be seen in L2 norm conserving schemes that do not have enough diffusion to damp high-k modes that develop near steep gradients [31].

Figure 2 demonstrates how global stabilization can reduce or eliminate numerical damping from a stable scheme. We solve the 2D incompressible Euler equations in vorticity form, given by

$$\frac{\partial \chi}{\partial t} + \boldsymbol{\nabla} \cdot (\boldsymbol{u}\chi) = 0, \qquad \boldsymbol{u} = \boldsymbol{\nabla}\psi \times \hat{e}_z, \qquad -\boldsymbol{\nabla}^2\psi = \chi. \tag{8}$$

Our baseline choice of flux is the second-order TVD MUSCL scheme with monotonized central (MC) slope limiters [35, 38]. We use a linear finite element (FE) solver for the poisson equation [1] and a strong stability preserving RK3 ODE integrator [12]. Note that eq. (8) exactly conserves both the energy $\frac{1}{2}\int \boldsymbol{u}^2\, dx\, dy$ and the enstrophy $\int \chi^2\, dx\, dy$ [32].

In each column of fig. 2a, we see the time-evolution of the vorticity χ according to one of four schemes. The baseline MUSCL schemes in the first and second columns decay the discrete L2 norm, while the

MUSCL schemes with global stabilization (GS) in the third and fourth columns either exactly conserve L2 norm by setting $dS^x_{tot}/dt = dS^y_{tot}/dt = 0$ (no damping) or reduce the rate of numerical diffusion by 75% by setting $dS^x_{tot}/dt = C^x/4$ and $dS^y_{tot}/dt = C^y/4$ (reduced damping). In the third column, we again find that globally ensuring L2 norm conservation introduces spurious high-k oscillations.

In fig. 2b, bottom row, we plot the vorticity correlation between the high resolution baseline scheme and each of the four schemes shown in figure 2a. Vorticity correlation has been used previously as a benchmark measure of accuracy for these equations [20]. We find that applying global stabilization with no damping underperforms relative to the baseline scheme at the same resolution, while global stabilization with reduced damping performs similarly to the baseline scheme at the same resolution. In fig. 2b, middle and top rows, we plot the discrete enstrophy $\sum_{i,j} \int \chi^2_{i,j} \Delta x \Delta y$ and discrete energy $\frac{1}{2} \sum_{i,j} \int (\boldsymbol{u}_{ij})^2 \Delta x \Delta y$ for each of the four schemes shown in figure 2a. The baseline schemes decay energy and enstrophy, while the globally stabilized scheme with no damping exactly conserves enstrophy and increases energy. The globally stabilized scheme with reduced damping decays enstrophy and displays a non-monotonic increase in energy.

5 Enstrophy and Coarse Graining

The incompressible Euler equations in vorticity form exactly conserve the L2 norm of the solution, called the enstrophy. Naively, we might expect that a good PDE solver would conserve enstrophy as well. Yet algorithms that exactly conserve enstrophy on eq. (8), such as the centered flux (not shown) or the global stabilization flux with no damping (shown in fig. 2), perform significantly worse than schemes that allow enstrophy to decay.

To understand this puzzling result, we consider the relationship between the continuous enstrophy $\int \chi^2 \, dx \, dy$ and the discrete enstrophy $\sum_{i,j} \int \chi^2_{i,j} \Delta x \Delta y$. Suppose that $\chi^{\text{exact}} = \chi(x, y, t)$ is the exact solution to eq. (8). As we know, χ^{exact} has constant enstrophy. Now suppose that we coarse grain χ^{exact}, such that $\chi^{\text{exact}}_{i,j} = \int_{i,j} \chi^{\text{exact}} \, dx \, dy$. It turns out that, with very high probability, the discrete enstrophy of $\chi^{\text{exact}}_{i,j}$ will decay in time [10]. This happens because χ^{exact} tends to develop structures on a scale smaller than the grid size. These structures cannot be represented by $\chi^{\text{exact}}_{i,j}$ and are replaced via coarse graining by a low-dimensional representation of the solution with lower enstrophy. The conclusion is as follows:

Although the continuous equations for χ^{exact} maintain constant enstrophy, the discrete equations for $\chi^{\text{exact}}_{i,j}$ decay enstropy.

Because machine learned PDE solvers solve discrete equations that are designed to approximate $\chi^{\text{exact}}_{i,j}$, then machine learned PDE solvers should inherit the properties of the discrete equations for $\chi^{\text{exact}}_{i,j}$, not the properties of the continuous equations for χ^{exact}. This means that solvers of eq. (1) should guarantee that enstrophy is non-increasing, not that enstrophy is constant.

Recall that in section 4, we found that we could ensure a stable FV scheme in 1D by transforming $f_{j+1/2}$ according to eq. (5) where dS_{tot}/dt is a non-negative free parameter that controls the rate of decay of the global L2 norm. In algorithm 1, we propose a practical method for choosing dS_{tot}/dt when applying global stabilization to PDE solvers that use ML to predict the flux f at cell boundaries. Algorithm 1 gives the machine learned PDE solver the freedom to decay the discrete L2 norm if it chooses to do so, but sets $dS_{tot}/dt = 0$ if the solver tries to increase the L2 norm. Intuitively, algorithm 1 is an error correcting algorithm that adjusts the machine learned output if that output moves the solution towards instability. As a result, the method is stable according to the criterion in section 2. Algorithm 1 is the main result of section 5.

6 Global Stabilization of Arbitrary FV Schemes

In section 4, we considered schemes that predict the flux f at cell boundaries. Using summation by parts, we found that we could adjust the flux prediction to ensure global stability. However, some machine learned PDE solvers may use an alternative form for the time-derivative which does not involve predicting the flux at cell boundaries. Thus, we now consider the more general problem of how to stabilize FV solvers for eq. (1) with arbitrary time-derivative. Suppose that the rate of change of the cell average u_j in 1D or $u_{i,j}$ in 2D is given by

Algorithm 1 A stable machine learned FV flux-predicting PDE solver in 1D

Inputs: Initial condition $\{u_j(t_0)\}_{j=1}^{N_x}$, ODE integrator, machine learned flux predictor
Time integration:
while $t < T_f$ **do**
  Choose Δt, compute flux $\{f_{j+\frac{1}{2}}\}_{j=1}^{N_x}$ using machine learned predictor
  if $C = -\sum_j f_{j+\frac{1}{2}}(u_{j+1} - u_j) < 0$ **then**
    Transform $\{f_{j+\frac{1}{2}}\}_{j=1}^{N_x}$ according to eq. (5) with $\frac{dS_{tot}}{dt} = 0$
  end if
  Advance time t by Δt and state $\{u_{j+\frac{1}{2}}\}_{j=1}^{N_x}$ according to ODE integrator
end while
Output: $\{u_j(T_f)\}_{j=1}^{N_x}$

$$\frac{\partial u_j}{\partial t} = N_j(\{u_j\}) \tag{9a}$$

$$\frac{\partial u_{i,j}}{\partial t} = N_{i,j}(\{u_{i,j}\}) \tag{9b}$$

where N_j and $N_{i,j}$ could be any arbitrary functions. Note that eqs. (9a) and (9b) do not guarantee mass conservation by construction. Ensuring stability therefore requires modifying N_j and $N_{i,j}$ to ensure both mass conservation and a non-increasing discrete L2 norm. In 1D, this can be written as $\Delta x \sum_j \frac{\partial u_j}{\partial t} = \Delta x \sum_j N_j = 0$ and $\Delta x \sum_j u_j \frac{\partial u_j}{\partial t} = \Delta x \sum_j u_j N_j \leq 0$ which implies that the discrete mass will be conserved and the discrete L2 norm will decay if

$$\sum_j N_j = \langle \boldsymbol{N} | \boldsymbol{1} \rangle = 0 \qquad \sum_j u_j N_j = \langle \boldsymbol{u} | \boldsymbol{N} \rangle \leq 0. \tag{10}$$

These conditions will be satisfied if the following transformation is applied to N_j:

$$U_j = u_j - \frac{\langle \boldsymbol{1} | \boldsymbol{u} \rangle}{\langle \boldsymbol{1} | \boldsymbol{1} \rangle} \qquad M_j = N_j - \frac{\langle \boldsymbol{1} | \boldsymbol{N} \rangle}{\langle \boldsymbol{1} | \boldsymbol{1} \rangle} \qquad N_j \Rightarrow \left(1 - \frac{\frac{dS_{tot}}{dt}}{\langle \boldsymbol{U} | \boldsymbol{M} \rangle}\right) M_j - \frac{\langle \boldsymbol{U} | \boldsymbol{M} \rangle}{\langle \boldsymbol{U} | \boldsymbol{U} \rangle} U_j \tag{11}$$

for any $dS_{tot}/dt \geq 0$. Although eq. (11) ensures stability by construction, it does not tell us how to set dS_{tot}/dt. In algorithm 2, we propose a practical method for choosing dS_{tot}/dt when applying global stabilization to machine learned PDE solvers. At each timestep, algorithm 2 first transforms N_j to ensure that mass is conserved by construction. If the resulting N_j increases the L2 norm, then we transform it again to ensure that the L2 norm is exactly conserved. Intuitively, algorithm 2 is an error correcting algorithm that adjusts the machine learned output to ensure that the errors in the output conserve mass and ensure stability. Algorithm 2 and eq. (11) are the main results of section 6.

Algorithm 2 A stable machine learned FV PDE solver in 1D with arbitrary time-derivative

Inputs: Initial condition $\{u_j(t_0)\}_{j=1}^{N_x}$, ODE integrator, machine learned flux predictor
Time integration:
while $t < T_f$ **do**
  Choose Δt, compute $\{N_j\}_{j=1}^{N_x}$ using machine learned predictor
  if $\langle \boldsymbol{M} | \boldsymbol{u} \rangle \leq 0$ **then**
    Set $\boldsymbol{N} = \boldsymbol{M}$
  else
    Transform $\{N_j\}_{j=1}^{N_x}$ according to eq. (11) with $\frac{dS_{tot}}{dt} = 0$
  end if
  Advance time t by Δt and state $\{u_{j+\frac{1}{2}}\}_{j=1}^{N_x}$ according to ODE integrator
end while
Output: $\{u_j(T_f)\}_{j=1}^{N_x}$

7 Related Work

LES models and backscattering: The objective of large eddy simulation (LES) is identical to that of many machine learned numerical solvers: both attempt to find an accurate approximation

to the solution of the PDE with fewer computational resources than classical numerical methods. Both also attempt to do so without resolving the smallest scales of the problem, relying on either an explicit or implicit subgrid model to do so [2, 40, 20, 34, 37, 22, 29, 14, 15, 25, 3, 39, 27]. Of particular relevance to the stability of subgrid models (both in LES and ML) are the concepts of 'forward-scatter' and 'backscatter'. In 2D LES turbulence, forward-scattering involves the transfer of enstrophy from resolved to unresolved scales, while backscattering involves the transfer of enstrophy from unresolved to resolved scales. Analysis across a wide range of flows demonstrates two important facts [30]. First, to be accurate a subgrid model must allow both forward-scatter and backscatter. In the context of scalar hyperbolic PDEs, this means that to be accurate a subgrid model must allow a discrete analogue of the entropy inequality in eq. (2) to be locally violated. Second, averaged over the entire domain there is always more forward-scatter than backscatter. If on average there were more backscatter than forward-scatter, then the subgrid model would be unstable [14]. Global stabilization can thus be interpreted as a way of constraining a subgrid model to ensure that on average there is always at least as much forward-scatter as backscatter.

Machine learned finite volume solvers: [2, 40, 20] use machine learned finite volume solvers to solve a variety of 1D and 2D PDEs. Almost all of these PDEs can be written in conservation form with added diffusion and forcing terms. These 'hybrid' solvers conserve mass by construction but not the discrete L2 norm and therefore do not guarantee stability; instead, they promote stability by unrolling the loss function over multiple timesteps. [20] trains a hybrid solver for the 2D incompressible Euler equations that "remains stable during long simulations." This impressive result is likely facilitated by the addition of physical diffusion to the PDE, which decays the L2 norm at each timestep.

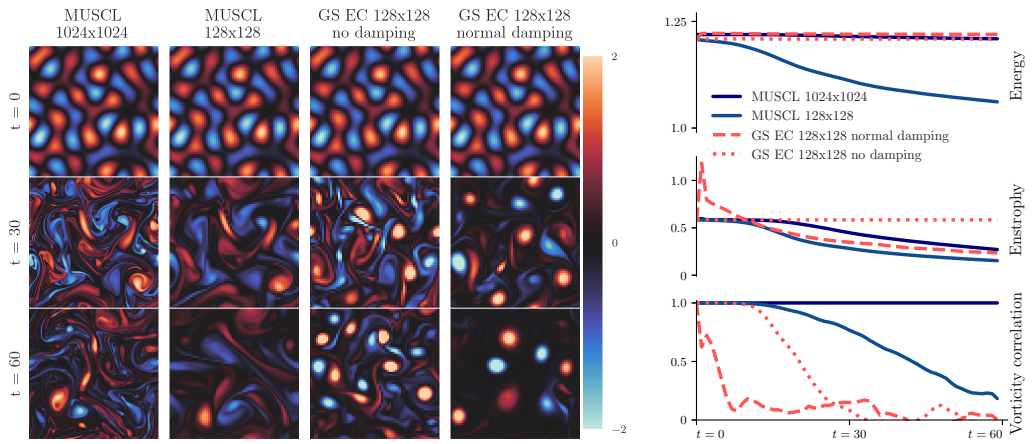
Other machine learned solvers: [22, 37, 29, 20] use convolutional neural networks to correct errors in low-resolution simulations; these hybrid solvers promote stability and improve accuracy by unrolling the loss function over multiple timesteps. [34] solves 1D, 2D and 3D hyperbolic PDEs using the 'fully learned' update equation $u_{i,j}(t+\Delta t) = u_{i,j}(t) + N_{i,j}(u_{i,j}(t), \Delta t))$ where $N_{i,j}$ is a the output of a convolutional neural network; this update equation is similar to eq. (9b) except with a discrete-time update instead of continuous-time ODE integration. [34] attempts to ensure stability by adding noise to the training distribution and by using very large timesteps. For scalar conservation form PDEs, this fully learned update equation will be stable if $\langle \boldsymbol{N} | \mathbf{1} \rangle = 0$ and $\langle \boldsymbol{u} | \boldsymbol{N} \rangle + \langle \boldsymbol{N} | \boldsymbol{N} \rangle \leq 0$. [4] argues that instability in machine learned iterative numerical algorithms arises due to a *distribution shift* where the distribution of training data differs from the outputs of the solver during inference due to small errors that accumulate over time. [4] uses the update equation $u_{i,j}(t+\ell\Delta t) = u_{i,j}(t) + \ell \Delta t N^{\ell}_{i,j}$ for $1 \leq \ell \leq K$ where $N^{\ell}_{i,j}$ is the output of a message passing graph neural network that predicts the next K timesteps. [4] attempts to ensure stability by modifying the loss function, adding random noise, and by predicting multiple timesteps into the future. A variety of papers have attempted to promote stability of dynamical systems that result from data-driven reduced order models, including by adding sparsity-promoting priors to a loss function [19, 8] and by constraining the eigenvalues of a learned Koopman operator [28].

KEP schemes: Kinetic energy preserving (KEP) and entropy preserving (EP) schemes are widely used in the numerical study of hyperbolic equations. Like global stabilization, KEP and EP schemes rely on the energy method for stability analysis and use summation by parts [17, 18, 6]. Unlike global stabilization, these schemes construct locally conservative algorithms which add just enough numerical damping at shocks to eliminate spurious oscillations.

8 Limitations

There are six main limitations of our work. First, our method violates causality. This happens because the time-derivative is adjusted in a way that depends on the solution across the entire domain. Second, our method works in the continuous-time limit. This means that the timestep must be chosen to satisfy a CFL condition and be small enough to ensure accuracy of the ODE integration.

Third, we restrict ourselves to rectangular grids with periodic boundary conditions. Fourth, we only consider *scalar* hyperbolic PDEs in conservation form, and do not consider *systems* of hyperbolic PDEs. Although many physically relevant equations can be written as scalar hyperbolic PDEs – including Hamiltonian systems, the incompressible Euler equations, and the Vlasov-Poisson equation – many more are systems of hyperbolic PDEs – including the compressible Euler equations, the magnetohydrodynamic (MHD) equations, the Einstein field equations, the shallow-water equations,



(a) (b)

Figure 3: Global stabilization can be modified to enforce energy conservation as well as stability for the incompressible Euler equations. (a) Images of the vorticity χ at three different times. The first and second columns show the baseline MUSCL scheme at high and low resolution. The third and fourth columns show the baseline MUSCL scheme with energy conserving global stabilization (GS EC), either with no numerical damping or with the normal (baseline) rate of damping. (b) Energy, enstrophy, and vorticity correlation over time. For a brief period soon after $t = 0$ the CFL condition is violated for the 'normal damping' solution, causing a transient increase in enstrophy and energy.

the Navier-Stokes equations, and the Vlasov-Maxwell equations. It may be possible to develop new methods that globally stabilize solvers for certain systems of hyperbolic PDEs, but we do not attempt to do so in this paper. We do, however, anticipate that our method could be extended to non-periodic boundary conditions, irregular domains, and many physically relevant systems of hyperbolic PDEs in a manner similar to KEP and EP schemes [17, 18].

Fifth, while global stabilization does solve the problem of finding *stable* solvers for eq. (1), it does not solve the problem of finding *accurate* machine learned solvers for eq. (1). Algorithm 1 prevents instability by adjusting the time-derivative if the solver makes an L2 norm increasing violation, but a solver which frequently commits such violations is likely to perform poorly. Alternatively, a solver could make no L2 norm increasing violations but decay the L2 norm too quickly. Or, it could decay the L2 norm at the correct rate but give inaccurate results. Building accurate, fast, and robust machine learned PDE solvers will require not only well-designed numerical methods but also well-engineered learning systems.

Sixth, for some scalar hyperbolic PDEs a solver might be stable according to the definition in section 2 but not result in a physically meaningful solution as $t \to \infty$ unless additional physical constraints are satisfied. For example, consider the Boltzmann equation $\frac{\partial f}{\partial t} + \frac{\boldsymbol{p}}{m} \cdot \boldsymbol{\nabla} f = \left(\frac{\partial f}{\partial t}\right)_{\text{coll}}$ from kinetic physics which describes the evolution of the particle distribution function f in phase space $(\boldsymbol{x}, \boldsymbol{p})$ due to collisions with other particles [5]. Because f conserves particles $\int f \, d\boldsymbol{x} \, d\boldsymbol{p}$, momentum $\int f \boldsymbol{p} \, d\boldsymbol{x} \, d\boldsymbol{p}$, and energy $\frac{1}{2m} \int f \, \boldsymbol{p}^2 \, d\boldsymbol{x} \, d\boldsymbol{p}$ while maintaining $f \geq 0$ and increasing the entropy $-\int f \log f \, d\boldsymbol{x} \, d\boldsymbol{p}$, then as $t \to \infty$ f must evolve towards a Gaussian distribution. Yet if global stabilization were applied naively to a Boltzmann equation solver without preserving the right combination of these invariants, then f could evolve towards a flat distribution function or some other physically incorrect state. Thus, while global stabilization may be useful, it is not by itself always going to be sufficient to ensure that the solution evolves to the correct state as $t \to \infty$.

To illustrate how this sixth limitation might be addressed, we again consider the incompressible euler equations in eq. (8) but now attempt to enforce an additional conservation law: conservation of energy. Recall that in fig. 2b, energy was not conserved by any of the schemes considered. Energy will be conserved if $\int \boldsymbol{u} \cdot \frac{\partial \boldsymbol{u}}{\partial t} \, dx \, dy = \int \psi \frac{\partial \chi}{\partial t} \, dx \, dy = \sum_{i,j} \bar{\psi}_{i,j} N_{i,j} \Delta x \Delta y = 0$ and is expressed most simply as $\langle \bar{\boldsymbol{\psi}} | \boldsymbol{N} \rangle = 0$ where $\bar{\psi}_{i,j}$ is the cell average of ψ and $N_{i,j}$ is the time-derivative in the ith, jth cell. Conservation of mass, conservation of energy, and L2 stability will therefore all be

guaranteed for eq. (8) if the following transformation is applied to $\boldsymbol{N}$:

$$\boldsymbol{U} = \boldsymbol{\chi} - \frac{\langle \boldsymbol{\chi} | \mathbf{1} \rangle}{\langle \mathbf{1} | \mathbf{1} \rangle} \mathbf{1} \qquad \boldsymbol{M} = \boldsymbol{N} - \frac{\langle \boldsymbol{N} | \mathbf{1} \rangle}{\langle \mathbf{1} | \mathbf{1} \rangle} \mathbf{1} \qquad \bar{\boldsymbol{\phi}} = \bar{\boldsymbol{\psi}} - \frac{\langle \bar{\boldsymbol{\psi}} | \mathbf{1} \rangle}{\langle \mathbf{1} | \mathbf{1} \rangle} \mathbf{1}$$

$$\boldsymbol{W} = \boldsymbol{U} - \frac{\langle \boldsymbol{U} | \bar{\boldsymbol{\phi}} \rangle}{\langle \bar{\boldsymbol{\phi}} | \bar{\boldsymbol{\phi}} \rangle} \bar{\boldsymbol{\phi}} \qquad \boldsymbol{P} = \boldsymbol{M} - \frac{\langle \boldsymbol{M} | \bar{\boldsymbol{\phi}} \rangle}{\langle \bar{\boldsymbol{\phi}} | \bar{\boldsymbol{\phi}} \rangle} \bar{\boldsymbol{\phi}}$$

$$\boldsymbol{N} \Rightarrow \left(1 - \frac{dS_{tot}/dt}{\langle \boldsymbol{W} | \boldsymbol{P} \rangle} \right) \boldsymbol{P} - \frac{\langle \boldsymbol{W} | \boldsymbol{P} \rangle}{\langle \boldsymbol{W} | \boldsymbol{W} \rangle} \boldsymbol{W} \tag{12}$$

for any $dS_{tot}/dt \geq 0$. Figure 3 demonstrates how this energy conserving global stabilization (GS EC) scheme affects the baseline MUSCL scheme. The third column and fourth columns of fig. 3a set $dS_{tot}/dt = 0$ (no damping) and $dS_{tot}/dt = -\langle \boldsymbol{\chi} | \boldsymbol{N} \rangle$ (normal damping). At small t the 'normal damping' solution has an large time-derivative, violating the CFL condition. This transient CFL violation causes an initial spike in enstrophy and energy in fig. 3b; afterwards the enstrophy decays and the energy is conserved as eq. (12) would predict in the time-continuous limit. Equation (12) and fig. 3 demonstrate that for some equations it may be possible to develop global stabilization schemes that enforce additional conservation laws.

9 Conclusion

Stability is a very desirable property of a PDE solver. Classical numerical methods almost always guarantee stability by construction. Machine learned PDE solvers have tried a variety of techniques to encourage stability (see section 7). To some extent, these techniques have been successful, as high-performing solvers have demonstrated the ability to give stable and accurate predictions for hundreds or thousands of timesteps. However, none of these techniques are capable of *guaranteeing* stability.

In this paper, we show how to design machine learned PDE solvers for scalar hyperbolic PDEs that are stable by construction. The main result of our paper is the 'global stabilization' technique. This can be used as an error-correcting algorithm to guarantee both mass conservation and L2 norm stability of machine learned FV PDE solvers (see algorithm 1), even when the time-derivative is an arbitrary function (see algorithm 2).

We also demonstrate that it is possible to design machine learned PDE solvers that by construction satisfy additional physical invariants, such as energy conservation (eq. (12)). Some machine learned solvers for hyperbolic PDEs guarantee preservation of one global invariant (mass conservation) using the finite-volume method [2, 40, 20]. We are not only the first to show how to design machine learned solvers for hyperbolic PDEs that guarantee preservation of *two* global invariants, but also the first to show how to preserve *three* global invariants (mass conservation, enstrophy decay, energy conservation).

Finally, we would like to reemphasize a subtle theoretical issue relevant to the design of accurate machine-learned PDE solvers. Although the exact solution $u^{\text{exact}}(x, t)$ to eq. (1) satisfies the entropy inequality in eq. (2) everywhere, the coarse-grained exact solution $u_j^{\text{exact}} = \int_j u^{\text{exact}} \, dx$ allows a discrete analogue of eq. (2) to be violated locally (see discussion of backscattering in section 7). This means that for a machine learned PDE solver to accurately predict u_j^{exact} even at coarse resolution, it must allow a discrete entropy inequality to be violated locally.

We believe that for machine learned PDE solvers to have real-world impact, they must be sufficiently robust and reliable to be trusted. Stability is one critical feature of robust and reliable PDE solvers. We therefore believe that global stabilization is a step towards the development of machine learned PDE solvers which could have real-world impact.

References

[1] L. Agbezuge. Finite element solution of the poisson equation with dirichlet boundary conditions in a rectangular domain. *Rochester Institute of Technology, Rochester, NY*, 2006.

[2] Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31): 15344–15349, 2019. doi: 10.1073/pnas.1814058116. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1814058116`.

[3] A. Beck, D. Flad, and C.-D. Munz. Deep neural networks for data-driven les closure models. *Journal of Computational Physics*, 398:108910, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2019.108910. URL https://www.sciencedirect.com/science/article/pii/S0021999119306151.

[4] J. Brandstetter, D. Worrall, and M. Welling. Message passing neural pde solvers, 2022. URL https://arxiv.org/abs/2202.03376.

[5] C. Cercignani. The boltzmann equation. In *The Boltzmann equation and its applications*, pages 40–103. Springer, 1988.

[6] P. Chandrashekar. Kinetic energy preserving and entropy stable finite volume schemes for compressible euler and navier-stokes equations. *Communications in Computational Physics*, 14 (5):1252–1286, 2013.

[7] D. R. Durran. *Numerical methods for wave equations in geophysical fluid dynamics*. Texts in applied mathematics. Springer, New York, 1999. ISBN 0387983767.

[8] N. B. Erichson, M. Muehlebach, and M. W. Mahoney. Physics-informed autoencoders for lyapunov-stable fluid flow prediction, 2019. URL https://arxiv.org/abs/1905.10866.

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

[10] A. N. Gorban. Basic types of coarse-graining. In *Model Reduction and Coarse-Graining Approaches for Multiscale Phenomena*, pages 117–176. Springer, 2006.

[11] S. Gottlieb. On high order strong stability preserving runge-kutta and multi step time discretizations. *Journal of scientific computing*, 25(1):105–128, 2005.

[12] S. Gottlieb, C.-W. Shu, and E. Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Review*, 43(1):89–112, 2001. doi: 10.1137/S003614450036757X. URL https://doi.org/10.1137/S003614450036757X.

[13] D. Greenfeld, M. Galun, R. Basri, I. Yavneh, and R. Kimmel. Learning to optimize multigrid PDE solvers. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2415–2423. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/greenfeld19a.html.

[14] Y. Guan, A. Chattopadhyay, A. Subel, and P. Hassanzadeh. Stable a posteriori LES of 2d turbulence using convolutional neural networks: Backscattering analysis and generalization to higher re via transfer learning. *Journal of Computational Physics*, 458:111090, jun 2022. doi: 10.1016/j.jcp.2022.111090. URL https://doi.org/10.1016%2Fj.jcp.2022.111090.

[15] Y. Guan, A. Subel, A. Chattopadhyay, and P. Hassanzadeh. Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate les, 2022. URL https://arxiv.org/abs/2201.07347.

[16] J.-T. Hsieh, S. Zhao, S. Eismann, L. Mirabella, and S. Ermon. Learning neural pde solvers with convergence guarantees, 2019. URL https://arxiv.org/abs/1906.01200.

[17] A. Jameson. The construction of discretely conservative finite volume schemes that also globally conserve energy or entropy. *Journal of Scientific Computing*, 34(2):152–187, 2008.

[18] A. Jameson. Formulation of kinetic energy preserving conservative schemes for gas dynamics and direct numerical simulation of one-dimensional viscous compressible flow in a shock tube using entropy and kinetic energy preserving schemes. *Journal of Scientific Computing*, 34(2): 188–208, 2008.

[19] A. A. Kaptanoglu, J. L. Callaham, A. Aravkin, C. J. Hansen, and S. L. Brunton. Promoting global stability in data-driven models of quadratic nonlinear dynamics. *Physical Review Fluids*, 6(9), sep 2021. doi: 10.1103/physrevfluids.6.094401. URL https://doi.org/10.1103%2Fphysrevfluids.6.094401.

[20] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer. Machine learning accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021. doi: 10.1073/pnas.2101784118. URL https://www.pnas.org/doi/abs/10.1073/pnas.2101784118.

[21] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations, 2020. URL https://arxiv.org/abs/2010.08895.

[22] B. List, L.-W. Chen, and N. Thuerey. Learned turbulence modelling with differentiable fluid solvers, 2022. URL https://arxiv.org/abs/2202.06988.

[23] I. Luz, M. Galun, H. Maron, R. Basri, and I. Yavneh. Learning algebraic multigrid using graph neural networks, 2020. URL https://arxiv.org/abs/2003.05744.

[24] L. G. Margolin and N. M. Lloyd-Ronning. Artificial viscosity – then and now, 2022. URL https://arxiv.org/abs/2202.11084.

[25] R. Maulik, O. San, A. Rasheed, and P. Vedula. Subgrid modelling for two-dimensional turbulence using neural networks. *Journal of Fluid Mechanics*, 858:122–144, 2019. doi: 10.1017/jfm.2018.770.

[26] A. T. Mohan, N. Lubbers, D. Livescu, and M. Chertkov. Embedding hard physical constraints in neural network coarse-graining of 3d turbulence, 2020. URL https://arxiv.org/abs/2002.00021.

[27] N. Nguyen-Fotiadis, M. McKerns, and A. Sornborger. Machine learning changes the rules for flux limiters, 2021. URL https://arxiv.org/abs/2108.11864.

[28] S. Pan and K. Duraisamy. Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability. *SIAM Journal on Applied Dynamical Systems*, 19(1):480–509, jan 2020. doi: 10.1137/19m1267246. URL https://doi.org/10.1137%2F19m1267246.

[29] J. Pathak, M. Mustafa, K. Kashinath, E. Motheau, T. Kurth, and M. Day. Using machine learning to augment coarse-grid computational fluid dynamics simulations, 2020. URL https://arxiv.org/abs/2010.00072.

[30] U. Piomelli, W. H. Cabot, P. Moin, and S. Lee. Subgrid-scale backscatter in turbulent and transitional flows. *Physics of Fluids A: Fluid Dynamics*, 3(7):1766–1771, 1991. doi: 10.1063/1.857956. URL https://doi.org/10.1063/1.857956.

[31] S. Premasuthan, C. Liang, and A. Jameson. Computation of flows with shocks using the spectral difference method with artificial viscosity, i: Basic formulation and application. *Computers & Fluids*, 98:111–121, 2014. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2013.12.013. URL https://www.sciencedirect.com/science/article/pii/S0045793013004933.

[32] T. G. Shepherd. Symmetries, conservation laws, and hamiltonian structure in geophysical fluid dynamics. In *Advances in Geophysics*, volume 32, pages 287–338. Elsevier, 1990.

[33] C.-W. Shu. Discontinuous galerkin methods: general approach and stability. *Numerical solutions of partial differential equations*, 201, 2009.

[34] 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, 2021. URL https://arxiv.org/abs/2112.15275.

[35] P. K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, 21(5):995–1011, 1984. doi: 10.1137/0721062. URL https://doi.org/10.1137/0721062.

[36] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin. Accelerating eulerian fluid simulation with convolutional networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3424–3433. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/tompson17a.html`.

[37] K. Um, R. Brand, Y. Fei, P. Holl, and N. Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers, 2020. URL `https://arxiv.org/abs/2007.00016`.

[38] B. van Leer. Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method. *Journal of Computational Physics*, 32(1):101–136, 1979. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(79)90145-1. URL `https://www.sciencedirect.com/science/article/pii/0021999179901451`.

[39] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu. Towards physics-informed deep learning for turbulent flow prediction, 2019. URL `https://arxiv.org/abs/1911.08655`.

[40] J. Zhuang, D. Kochkov, Y. Bar-Sinai, M. P. Brenner, and S. Hoyer. Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. *Phys. Rev. Fluids*, 6:064605, Jun 2021. doi: 10.1103/PhysRevFluids.6.064605. URL `https://link.aps.org/doi/10.1103/PhysRevFluids.6.064605`.

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] We claim in the abstract and introduction that we demonstrate how to design PDE solvers for scalar hyperbolic PDEs that are stable by construction. This claim is accurate.
   (b) Did you describe the limitations of your work? [Yes] Yes, in Section 8 (Limitations) we discussed six limitations of our work.
   (c) Did you discuss any potential negative societal impacts of your work? [No] The NeurIPS Ethics Guidelines writes that "*As ML research and applications have increasing real-world impact, the likelihood of meaningful social benefit increases, as does the attendant risk of harm.*" With these guidelines in mind, we include a paragraph at the end of Section 9 (Conclusion) about the possible real-world impacts of our paper. We do not, however, speculate on the possible *negative* impacts of such solvers because we find it difficult to speculate on what such impacts could be.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] We believe we did so.
   (b) Did you include complete proofs of all theoretical results? [Yes] Yes, although there are a few places we do not list out every step in a derivation for the purpose of simplicity. This happens in equation (6), where we write "a similar calculation in 2D…" as well as in line 276, where we do not explicitly write out an integration by parts to compute the energy.
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 are including these in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We included a random seed in the initialization of figures 2a and 3a, but did not include error bars. We believe that error bars on figures 2b and 3b would be distracting and unnecessary, as the purpose of these figures is to demonstrate qualitative relationships between variables rather than quantitative results.

(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)? [No] We did not include the total amount of compute for our three simple experiments. The amount of compute required was quite small. Generating the 1024x1024 data takes an hour or two on a laptop CPU, but the other experiments run in a few minutes on a laptop CPU.

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? [N/A]

   (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? [N/A]

   (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]