Synthesizing Sound and Precise Abstract Transformers for Nonlinear Hyperbolic PDE Solvers

JACOB LAUREL, Georgia Institute of Technology, USA
IGNACIO LAGUNA, Lawrence Livermore National Laboratory, USA
JAN HÜCKELHEIM, Argonne National Laboratory, USA

Partial Differential Equations (PDEs) play a ubiquitous role in scientific computing and engineering. While numerical methods make solving PDEs tractable, these numerical solvers encounter several issues, particularly for hyperbolic PDEs. These issues arise from multiple sources including the PDE's physical model, which can lead to effects like shock wave formation, and the PDE solver's inherent approximations, which can introduce spurious numerical artifacts. These issues can cause the solver's program execution to crash (due to overflow) or return results with unacceptable levels of inaccuracy (due to spurious oscillations or dissipation). Moreover, these challenges are compounded by the nonlinear nature of many of these PDEs. In addition, PDE solvers must obey numerical invariants like the CFL condition. Hence there exists a critical need to apply program analysis to PDE solvers to certify such problems do not arise and that invariants are always satisfied.

As a solution, we develop Phocus, which is the first abstract interpretation of hyperbolic PDE solvers. Phocus can certify precise bounds on nonlinear PDE solutions and certify key invariants such as the CFL condition and a solution's total variation bound. Hence Phocus can verify the absence of shock formation, the stability of the solver, and bounds on the amount of spurious numerical effects. To enable effective abstract interpretation of hyperbolic PDE solvers, Phocus uses a novel optimization-based procedure to synthesize precise abstract transformers for multiple finite difference schemes. To evaluate Phocus, we develop a new set of PDE benchmark programs and use them to perform an extensive experimental evaluation which demonstrates Phocus's significant precision benefits and scalability to several thousand mesh points.

CCS Concepts: • **Mathematics of computing** → **Partial differential equations**; **Numerical differentiation**; • **Theory of computation** → **Program analysis**; **Program verification**.

Additional Key Words and Phrases: Finite Difference, Abstract Interpretation, Scientific Computing

ACM Reference Format:
Jacob Laurel, Ignacio Laguna, and Jan Hückelheim. 2025. Synthesizing Sound and Precise Abstract Transformers for Nonlinear Hyperbolic PDE Solvers. *Proc. ACM Program. Lang.* 9, OOPSLA2, Article 310 (October 2025), 29 pages. https://doi.org/10.1145/3763088

1 Introduction

Partial Differential Equations (PDEs) play a ubiquitous role in scientific computing and engineering due to their use in tasks as diverse as fluid simulation [9, 69], traffic modeling [58], and electromagnetics [36], among others. However, these equations often lack tractable analytical solutions. Hence one must resort to approximate numerical methods like finite difference (FD) schemes.

Yet even with tractable numerical methods, PDEs still raise unavoidable challenges. For example, PDEs may experience nonlinear instabilities or encounter issues like shock wave formation or finite time blowup [84]. Additionally, numerical methods like FD schemes are approximate and can

Authors' Contact Information: Jacob Laurel, Georgia Institute of Technology, Atlanta, USA, jlaurel6@gatech.edu; Ignacio Laguna, Lawrence Livermore National Laboratory, Livermore, USA, lagunaperalt1@llnl.gov; Jan Hückelheim, Argonne National Laboratory, Lemont, USA, jhueckelheim@anl.gov.

ACM 2475-1421/2025/10-ART310
https://doi.org/10.1145/3763088

introduce spurious errors into the PDE solution [34]. For instance, first-order accurate FD schemes can introduce spurious dissipation [54] while higher-order accurate FD schemes can introduce spurious local extrema [28, 75]. Further complicating matters, the instabilities in nonlinear PDEs often stem from algorithmic-level issues which are unrelated to floating-point imprecision [7, 80]. Thus traditional floating point analyses [82] are inapplicable. Indeed, prior work showed that certain instabilities of FD schemes are "independent of the [...] precision used in the [PDE] calculations" [7]. Hence even if one increases the floating-point precision, these instabilities persist.

Due to the possibility of algorithmic-level instabilities, shock waves, and spurious numerical artifacts, reasoning about the set of values a nonlinear PDE can attain remains difficult. How could one ensure that no shock waves arise or that the amount of spurious numerical effects (introduced by an FD scheme) lies below a tolerance? How does one certify the robustness of computed PDE solutions for a range of conditions? Scientists need answers to these questions to obtain formal assurances about their PDE simulations. Answers to these questions can even help engineers select a numeric datatype to avoid overflows when solving PDEs on hardware [31].

Hence, a major need exists for automated formal guarantees of numerical PDE solvers. Despite this need, and despite the surge of recent interest in applying formal methods to scientific computing problems [17, 26, 27, 39, 40, 70], automated formal methods for PDE solvers remain understudied.

Nonlinear Hyperbolic PDEs. While one might hope to automate the verification of all PDE solvers, given the large variety of PDEs (hyperbolic, parabolic, etc.) an exhaustive treatment lies beyond the realm of any prior method. For this reason, our work focuses on nonlinear hyperbolic PDEs. Hyperbolic PDEs are a general class that model wave-like flows through a domain [55] and include many popular physical models such as wave equations and fluid dynamics [9].

Why Program Analysis for Hyperbolic PDEs? Though hyperbolic PDEs abound in scientific computing, they remain severely understudied in the programming languages and formal methods communities. Despite this lack of study, hyperbolic PDEs and their numerical solvers represent appealing yet challenging targets for program analysis for several reasons.

- **Explicit Schemes** - Unlike the implicit schemes for parabolic PDEs, hyperbolic PDEs are often solved by *explicit* schemes which directly expose the primitive numerical operations at the source code level. Hence a static analyzer can directly analyze these operations.
- **Nonlinearity and Shocks** - While other PDEs are often linear or use linear stencils, hyperbolic PDEs may contain significant nonlinearity due to their flux functions. This nonlinearity can also cause shock formations which can in turn cause numerical blowup.
- **Spurious Numerical Effects** - Solvers for hyperbolic PDEs can suffer from spurious numerical effects such as artificial smoothing or oscillations. These issues are *independent* of the numerical precision used, meaning they do not come from floating point roundoff.
- **Complex Data and Control Flow** - In certain FD schemes such as upwinding, the underlying physics causes PDE solutions at future times to have crucial yet challenging data and control flow dependencies on the values of previous timesteps.
- **Key Numerical Invariants** - For a given discretization, the stability of a hyperbolic PDE solver requires invariants, like the CFL condition, be satisfied at every step. These program invariants capture the underlying physics of the simulation.

In light of these challenges, there exists a crucial need to develop program analyzers to reason about the computed solutions of hyperbolic PDEs and their numerical invariants. However, a unified, automated framework to answer these questions lies beyond the scope of any prior work.

Our Work. We present Phocus, the first abstract interpretation framework for hyperbolic PDE numerical solvers. The core idea of Phocus is to synthesize sound linear bounds for the Lax-Friedrichs, Leapfrog and Upwinding FD schemes used to solve nonlinear hyperbolic PDEs. Hence our work certifies precise bounds on numerical PDE solutions. Our work is also the first to formalize and verify properties pertaining to the total variation, which quantifies spurious numerical artifacts in a PDE solution, and the CFL condition, which quantifies stability of a PDE solver.

Contributions. In summary, this paper makes the following contributions:

(1) **Problem Formulation**. We formulate the first abstract interpretation of 3 different nonlinear hyperbolic PDE solvers to compute sound bounds on their solutions (Section 4). We also propose new formal properties for the PDE solutions, specifically an interval bound on the CFL condition and an interval bound on a solution's total variation (Section 4.7).

(2) **Technique**. We present Phocus, a novel static analyzer for PDE solvers that acheives significant precision by reducing a FD scheme's bound computation to tractable optimization problems (Sections 4.1-4.3). Hence we can synthesize precise abstract transformers specialized for PDE solvers. Furthermore, we prove the soundness of these abstractions (Section 4.4).

(3) **Evaluation**. We propose a novel set of program analysis benchmarks for nonlinear hyperbolic PDEs (Section 5) and then evaluate Phocus on these benchmarks (Section 6). Our evaluation compares against a standard zonotope baseline and Phocus shows significant precision improvements while scaling to thousands of meshpoints.

2 Example

We now illustrate Phocus's technique with an end-to-end example. Scientists often strive to understand how a PDE solution, $u(x,t)$, can differ if initial conditions changed. Hence the core application of Phocus is to statically analyze hyperbolic PDE solvers for a *range* of conditions. Our example PDE is a 1D hyperbolic conservation law with cubic flux $f(u) = u^3$ and periodic boundary conditions. A cubic flux is a common flux [42, 89] that leads to nonlinear wave effects. The PDE is shown in Eq. 1.

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(u^3) = 0 \tag{1}$$

To solve this PDE, we resort to numerical methods, specifically the Lax-Friedrichs finite difference (FD) scheme. For this PDE the FD scheme is the nonlinear stencil shown in Eq. 2:

$$u_i^{n+1} = \frac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \frac{\Delta t}{2\Delta x}\left((u_{i+1}^n)^3 - (u_{i-1}^n)^3\right) \tag{2}$$

This FD formula will be implemented in code as a function, `lax_friedrichs_stencil`:

```
def lax_friedrichs_stencil(u_i_plus_1,u_i_minus_1,K):
    return 0.5*(u_i_plus_1+u_i_minus_1)-K*((u_i_plus_1**3)-(u_i_minus_1**3))
```

Next, the `lax_friedrichs_stencil` is called by `lax_friedrichs_PDE_solver` (shown below) at each mesh point (x) for each time step (n) of the PDE simulation. Synthesizing an abstract transformer for `lax_friedrichs_stencil` enables precise abstract interpretation of the full solver.

```
def lax_friedrichs_PDE_solver(u0,Timesteps,dt,dx):
    L = 4           //four spatial mesh points used for the discretization
    u[0] = u0       //user provided initial condition
    K = (dt/dx)*0.5
    for n in range(0, Timesteps):
        for x in range(1, L-1):
            u[n+1][x] = lax_friedrichs_stencil(u[n][x+1],u[n][x-1],K)
        u[n+1][0] = lax_friedrichs_stencil(u[n][1],u[n][L-1],K)      //Periodic Boundary
        u[n+1][L-1] =  lax_friedrichs_stencil(u[n][0],u[n][L-2],K)   //Periodic Boundary
    return u
```

PDE Configuration. For simplicity we restrict the domain to $x \in [0,3]$. The notation u_i^n represents variable `u[n][i]`. The time step discretization is $\Delta t = .525$ and boundary conditions are periodic. We spatially discretize with $\mathsf{L} = 4$ mesh points, thus $\Delta x = 1$, $i \in \{0,1,2,3\}$ and $x_i = i\Delta x$.

```
1  u[0] = Input(...)                                        //Timestep 0: user provided initial condition range
2  ≀ u[0][0] = 0.00 + 0.075$\epsilon_1$, [-0.075, 0.075] ≀        //Every $\epsilon_j$ noise symbol has a range of $[-1, 1]$
3  ≀ u[0][1] = 0.21 + 0.075$\epsilon_2$, [0.135, 0.285] ≀
4  ≀ u[0][2] = 0.39 + 0.075$\epsilon_3$, [0.315, 0.465] ≀
5  ≀ u[0][3] = 0.54 + 0.075$\epsilon_4$, [0.465, 0.615] ≀
6  dt, dx = .525, 1.0                                       //Chosen discretization
7  K = dt/(2*dx)
8
9  u[1][1]= lax_friedrichs_stencil(u[0][2],u[0][0],K)       //Start of Timestep 1 computations
10 ≀ u[1][1] = 0.178 + 0.0375$\epsilon_1$ + 0.028$\epsilon_3$ + 0.001$\epsilon_5$, [0.111, 0.244] ≀   //Bounds shown in Fig. 2a (bottom)
11 u[1][2]= lax_friedrichs_stencil(u[0][3],u[0][1],K)
12 ≀ u[1][2] = 0.335 + 0.04$\epsilon_2$ + 0.02$\epsilon_4$ + 0.0018$\epsilon_6$, [0.274, 0.395] ≀
13 u[1][0]= lax_friedrichs_stencil(u[0][1],u[0][3],K)       //Handle Periodic Boundaries
14 ≀ u[1][0] = 0.414 + 0.0348$\epsilon_2$ + 0.055$\epsilon_4$ + 0.0018$\epsilon_7$, [0.325, 0.505] ≀
15 u[1][3]= lax_friedrichs_stencil(u[0][0],u[0][2],K)
16 ≀ u[1][3] = 0.211 + 0.0374$\epsilon_1$ + 0.0465$\epsilon_3$ + 0.001$\epsilon_8$, [0.128, 0.296] ≀
17
18 Assert($\bigwedge_{i=0}^{3}$abs(u[1][i])<=sqrt(dx/(3*dt)))  //Verify discretization still obeys CFL Condition
```

Fig. 1. Abstract Interpretation of a single timestep of `lax_friedrichs_PDE_solver`

Hence at time $t_n = n\Delta t$, we use `u[n][0]`, `u[n][1]`, `u[n][2]`, and `u[n][3]` as the PDE's values: $u(x_i, t_n)$. We model the wave's initial condition as $u(x, 0) = -0.015(x(x-15))$ with $x \in [0, 3]$.

Abstract Interpretation. While the `lax_friedrichs_PDE_solver` program solves the PDE, our ultimate goal is to interpret this code *abstractly* instead of concretely. While our example will run the solver for multiple timesteps, for illustration purposes, we now zoom into a single timestep's computation. The single timestep's unrolled code is shown in Fig. 1.

We abstractly interpret this code to certify PDE solution bounds for *ranges* of inputs. We want to enclose the set of PDE solutions when initial condition $u(x, 0)$ has uncertainty of ± 0.075. To formalize this idea, we represent a mesh point's range with the reduced product of the zonotope[23] and interval domains. The abstract state is shown in blue where ≀ `u[n][i]` $= a_0 + \sum_{j=1}^{n} a_j\epsilon_j$, $[l, u]$ ≀ means that at the n^{th} timestep, the i^{th} spatial mesh point (`u[n][i]`) is abstracted with an affine form $a_0 + \sum_{j=1}^{n} a_j\epsilon_j$ and refined interval bounds $[l, u]$. We show the abstract elements enclosing each mesh point's initial value in lines 2-5. Visual bounds are shown in the top subplot of Fig. 2a.

Formal Specification. Our verified bounds allow us to certify the FD scheme obeys the Courant-Friedrichs-Lewy (CFL) condition for any PDE whose initial values lie in this input range. The CFL specification to verify is shown in line 18 of the code and is formalized as $\forall i, n, |u_i^n| \leq \sqrt{\frac{\Delta x}{3\Delta t}}$.

Lax-Friedrichs Abstract Transformer. The `lax_friedrichs_stencil` is called in lines 9-15 for each mesh point and takes as input the previous timestep's values: `u[0][0]`,`u[0][1]`,`u[0][2]`, and `u[0][3]`. While one could use standard zonotope or interval abstract transformers to abstractly interpret the stencil's `-K*((u_i_plus_1**3)-(u_i_minus_1**3))` cubic flux terms, due to their nonlinearity, the imprecision would compound. Hence the contribution of Phocus is to synthesize a *single* abstract transformer for the *entire* Lax-Friedrichs stencil. For the computation of meshpoint `u[1][1]` on line 9, the synthesis first performs linear regression to find the best linear fit to the nonlinear stencil in the region of interest. This means finding $A, B, C \in \mathbb{R}$ such that

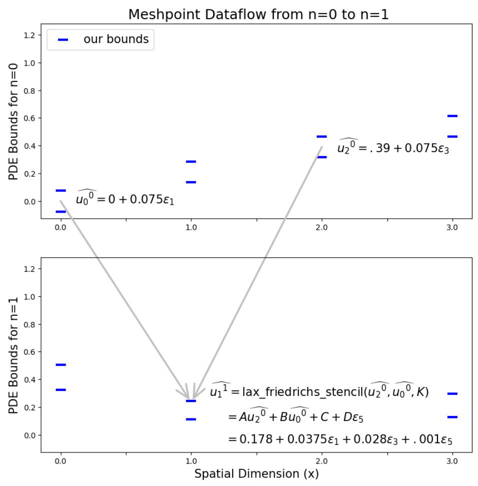
$$Au_{i+1}^n + Bu_{i-1}^n + C \approx \frac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \frac{\Delta t}{2\Delta x}\left((u_{i+1}^n)^3 - (u_{i-1}^n)^3\right)$$

Phocus performs the linear regression by sampling $\frac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \frac{\Delta t}{2\Delta x}\left((u_{i+1}^n)^3 - (u_{i-1}^n)^3\right)$ at concrete values of u_{i-1}^n and u_{i+1}^n that lie in the 2D box $[l_{u_{i-1}^n}, u_{u_{i-1}^n}] \times [l_{u_{i+1}^n}, u_{u_{i+1}^n}]$. When $n = 0$, $i = 1$, this box is $[-0.075, 0.075] \times [0.315, 0.465]$ and the regression finds $A = .379$, $B = .50$, and $C = .03$. To obtain a *sound* enclosure, we introduce a new symbol $\epsilon_5 \in [-1, 1]$ with coefficient $D \in \mathbb{R}$ that captures the deviation between the linear approximation and the stencil. To solve for D, Phocus computes:

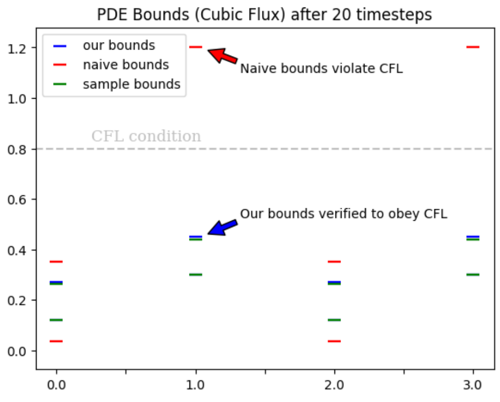
$$D = \max_{u_{i-1}^n \in [-.075,.075],\ u_{i+1}^n \in [0.315, 0.465]} \left| Au_{i+1}^n + Bu_{i-1}^n + C - \left(\frac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \frac{\Delta t}{2\Delta x}\left((u_{i+1}^n)^3 - (u_{i-1}^n)^3\right) \right) \right|$$

A key contribution of Phocus is proving that this problem can be reduced to simpler 1D subproblems. We denote $g = Au_{i+1}^n + Bu_{i-1}^n + C - (\frac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \frac{\Delta t}{2\Delta x}((u_{i+1}^n)^3 - (u_{i-1}^n)^3))$ and use the identity $|g| = \max(g, -g)$ to remove the absolute value. The core insight is that irrespective of the flux, the objective g (and $-g$) is *separable*. Hence to solve $\max g$, we can solve $\max_{u_{i+1}^n} Au_{i+1}^n - \frac{1}{2}u_{i+1}^n - \frac{\Delta t}{2\Delta x}(u_{i+1}^n)^3$ and $\max_{u_{i-1}^n} Bu_{i-1}^n - \frac{1}{2}u_{i-1}^n + \frac{\Delta t}{2\Delta x}(u_{i-1}^n)^3 + C$. After separating the 2D problem into these 1D subproblems, we use the fact that for each 1D subproblem the flux function $(f(u) = u^3)$ has a derivative that can be inverted: $f'^{-1}(u) = \pm\sqrt{\frac{u}{3}}$. Hence we can find critical points by solving $u^* = f'^{-1}(\frac{2\Delta x(A-\frac{1}{2})}{\Delta t})$ and $u^{**} = f'^{-1}(\frac{-2\Delta x(B-\frac{1}{2})}{\Delta t})$. Thus we only need to evaluate the 1D problem at $\{l_{u_{i+1}^n}, u_{u_{i+1}^n}\} \cup \{u^*\}$ and $\{l_{u_{i-1}^n}, u_{u_{i-1}^n}\} \cup \{u^{**}\}$ respectively to find the maximizing value $D = .001$. Furthermore, virtually the same idea can be used to solve the simpler optimization problems $\min\left(\frac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \frac{\Delta t}{2\Delta x}\left((u_{i+1}^n)^3 - (u_{i-1}^n)^3\right)\right)$ and $\max\left(\frac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \frac{\Delta t}{2\Delta x}\left((u_{i+1}^n)^3 - (u_{i-1}^n)^3\right)\right)$ to obtain direct interval bounds. This abstract transformer synthesis is applied to every meshpoint, hence this exact same procedure is computed for lines 11-15. The bounds computed for this first timestep (including the dataflow) are shown in the bottom subplot of Fig. 2a.

Results. While Fig. 2a shows bounds for $n = 0$ and $n = 1$ obtained by abstractly interpreting Fig. 1, we study how the bounds evolve with more iterations. Hence we abstractly interpret 20 iterations of the `lax_friedrichs_PDE_solver`. Our final bounds (shown in blue) can be seen in Fig. 2b. To compare the precision of Phocus, we also compute naive bounds using standard zonotope abstract transformers for the cubic functions (in red). We also compute unsound, sampled bounds (in green) by evaluating the program concretely on random inputs sampled from the initial condition range. The desired behavior is the tightest possible enclosure around the green bounds. As shown, Phocus's bounds (blue) are much tighter than naive zonotope baseline bounds (red).



(a) Meshpoint dataflow from timestep $n = 0$ to $n = 1$ and Phocus's corresponding bounds. Gray arrows show dataflow for computing u[1][1]



(b) Final results showing PDE solution bounds after $n = 20$ timesteps. Phocus's bounds are shown in blue and enclose the green empirical samples more tightly than the red naive zonotope bounds. Our bounds prove the discretization obeys the CFL condition.

Fig. 2. (a) The evolution of PDE bounds for the first timestep (b) The final bounds after 20 timesteps

Lastly, Phocus's tight bounds lie below the dashed gray line thus proving the discretization obeys the CFL condition at each step. In contrast, the naive zonotope abstract interpreter accumulates imprecision expanding beyond the gray line, which violates the CFL condition, as shown in Fig. 2b

3 Preliminaries

We now describe the mathematical preliminaries needed to understand both finite difference schemes for hyperbolic PDEs [54] as well as abstract interpretation [13].

3.1 Hyperbolic Partial Differential Equations

The class of PDEs we study are nonlinear hyperbolic PDEs which model wave-like phenomena. Hyperbolic PDEs are general and they can also experience notable behavior like shock formations.

3.1.1 PDE Conservation Laws. The nonlinear hyperbolic PDEs we study have the following form:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(f(u)) = 0 \tag{3}$$

These hyperbolic PDEs are also called conservation laws [45, 54]. The solution u is a function of (one) spatial dimension x and time dimension t, hence we write $u(x, t)$, or u for short. We assume no source terms exist, hence why the RHS of Eq. 3 is 0. While two of the FD schemes we study use Eq. 3, the upwinding scheme uses the *quasilinear* version of this equation which is:

$$\frac{\partial u}{\partial t} + f'(u) \cdot \frac{\partial u}{\partial x} = 0 \tag{4}$$

3.1.2 Nonlinear Flux Functions. The nonlinear function $f : \mathbb{R} \to \mathbb{R}$ in Eqs. 3 and 4 is the flux function. The flux determines the flow of the PDE solution u by capturing the underlying physics. Many flux functions exist, some are convex, while others are concave, and some are neither (e.g. Buckley-Leverett [8]). A core part of our contribution is to develop a technique to precisely bound the reachable set of PDE solutions in the face of nonlinear flux functions.

3.2 Numerical PDE Solutions and Finite Difference Schemes

We now describe the finite difference schemes that our work studies. These schemes are designed for 1D hyperbolic PDEs and represent explicit (instead of implicit) schemes that march forward in time. While other schemes like forward time-centered space (FTCS) exist, many such schemes are inapplicable or unconditionally unstable for hyperbolic problems [71].

3.2.1 PDE Computational Domain. In FD schemes, the PDE is solved along a computational domain that simulates the underlying physics [3, 54]. This domain includes both spatial and time dimensions. Since FD schemes solve *discretizations* of the PDE, both space and time are discretized. In our case, Δx represents the size of the spatial discretization and Δt represents the time discretization. The PDE is numerically solved along *mesh* points which for our purposes are equally spaced apart by Δx. To denote the *numerical* solution at the i^{th} mesh point and n^{th} time step, we write u_i^n, which numerically approximates the true solution $u(x_i, t_n)$ at $x_i = i\Delta x, t_n = n\Delta t$. The computational domain must also encode how the *boundary conditions* are defined. Our work considers both Dirichlet and Periodic boundary conditions. For Dirichlet boundaries, the values at the computational domain's end points are fixed as constants and for Periodic boundary conditions, the endpoints are "glued" together meaning the computational domain becomes a torus instead of a rod.

3.2.2 Lax-Friedrichs. The Lax-Friedrichs FD scheme is a first-order accurate finite difference scheme for hyperbolic PDEs [54]. The scheme is an explicit forward-time stepping scheme that uses centered differences for the spatial derivatives. The scheme is shown in Eq. 5.

$$u_i^{n+1} = \frac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \frac{\Delta t}{2\Delta x}\big(f(u_{i+1}^n) - f(u_{i-1}^n)\big) \tag{5}$$

One should note that Lax-Friedrichs schemes can introduce artificial smoothing or dissipation into the solution which means traveling waves may artificially lose some of their energy or wave height.

3.2.3 Leapfrog. The leapfrog FD scheme is a second-order accurate scheme [18, 87] and is shown in Eq. 6. Although the accuracy of the leapfrog scheme is better than that of first-order schemes, the leapfrog scheme can introduce spurious local minima and maxima and dispersive effects [7, 80]

$$u_i^{n+1} = u_i^{n-1} - \frac{\Delta t}{\Delta x}\big(f(u_{i+1}^n) - f(u_{i-1}^n)\big) \tag{6}$$

An important detail is that the Leapfrog scheme requires the previous *two* timesteps instead of just the previous one. To compute the first timestep, we will use the Lax-Friedrichs scheme.

3.2.4 Upwinding. The upwinding FD scheme [12, 65] dynamically selects a forward or backward FD approximation for the derivative at each step by adapting to the wave's propagation direction. The formulation is based on the quasilinear form (Eq. 4). For a nonlinear hyperbolic PDE with differentiable flux f, the (1D) upwinding finite difference scheme is defined as:

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x} f'(u_i^n) \cdot \begin{cases} u_i^n - u_{i-1}^n & \text{if } f'(u_i^n) \geq 0 \\ u_{i+1}^n - u_i^n & \text{else if } f'(u_i^n) < 0 \end{cases} \tag{7}$$

The control flow and dataflow of this scheme come from the underlying physics of hyperbolic PDEs. The solution's value at a point only depends on upwind values that physically flow *towards* that point. Downwind values correspond to waves that flow *away* from that point and will continue to move further away thus producing no effect. The wave direction comes from the sign of $f'(u_i^n)$. For example, if $f'(u_i^n) < 0$, then u_{i+1}^n is upwind and u_{i-1}^n is downwind and a forward FD is chosen.

Similar to ReLU [76] and geometric [90] patterns in DNN verification, we can construct an equivalent formulation of Eq. 7 using the min and max functions. This formulation is given in Eq. 8.

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}\Bigg(\max(0, f'(u_i^n)) \cdot (u_i^n - u_{i-1}^n) + \min(f'(u_i^n), 0) \cdot (u_{i+1}^n - u_i^n)\Bigg) \tag{8}$$

The key distinction between upwinding and the Lax-Friedrichs or Leapfrog scheme is that upwinding is not in conservative form and uses the derivative of the flux f' instead of the flux f. While the non-conservative property of this upwinding stencil sometimes discourages its use, this upwinding still finds many applications in practice [38, 56].

3.2.5 CFL Condition. Specific to hyperbolic PDEs is the Courant-Friedrich-Lewy (CFL) condition which governs the allowed time step Δt for a PDE discretization to behave stably [11]. The CFL condition must be obeyed since a discretization that is too coarse could erroneously miss a travelling wave. Hence the CFL condition is a numerical invariant. We now describe the CFL condition.

Definition 3.1. CFL Condition [6, 11]- For a mesh spatially discretized by Δx and temporally discretized by Δt where the PDE solution at the i^{th} mesh point and n^{th} timestep is u_i^n, a finite difference method for a 1D hyperbolic PDEs is stable at step n if the following condition is met:

$$\max_i |f'(u_i^n)| \frac{\Delta t}{\Delta x} \leq C_{\max} \tag{9}$$

For explicit finite difference schemes $C_{\max} = 1$. Intuitively, Eq. 9 relates how small Δt must be relative to the granularity of the spatial discretization, Δx, and the wave velocity, $f'(u_i^n)$. If this condition is violated, the numerical scheme can become unstable and suffer numerical blowup.

3.2.6 Oscillation and Total Variation. Besides the CFL condition, another key property that quantifies a FD scheme's behavior is the total variation. Due to the approximate nature of FD schemes, they can introduce spurious and non-physical numerical artifacts into the solution [34]. These spurious numerical artifacts can take the form of artificial oscillations in the computed PDE solution, especially when the wave modeled by the PDE experiences sharp fronts [10, 34]. Hence, the total variation serves as a useful measure of the amount of oscillation in the solution [21]. Similar to prior work [81, 88], we define the n^{th} timestep's total variation (TV) in Eq. 10 as:

$$TV_n = \int \left|\nabla_x u(x, t_n)\right| dx \approx \sum_i \left|u_i^n - u_{i-1}^n\right| \tag{10}$$

While the TV is defined as an integral of derivatives, for PDE discretizations, ones takes the (finite) sum of differences: $\sum_i |u_i^n - u_{i-1}^n|$. The leapfrog and non-conservative upwinding scheme we study can increase the TV while the Lax-Friedrichs scheme can lower the TV through dissipation.

Due to the TV property's significance, many works directly compute the TV of a solution at different timesteps [16, 81, 88] to quantify whether the amount of introduced numerical artifacts remains tolerable. Our contribution is a novel formalization of this property in the language of abstract interpretation so that we can provably track the TV with certified lower and upper bounds.

3.2.7 Shock Formation. Nonlinear hyperbolic PDEs may develop shock waves which can arise even for smooth initial conditions [53]. Computationally, this phenomenon can register as a numerical blowup in finite time. Additionally, shock waves stem from the underlying physics of the problem and are *not* an artifact of a particular numerical scheme or floating point roundoffs.

3.3 Numerical Abstract Interpretation

Within the realm of abstract interpretation, our technique focuses on numerical abstract domains [62]. We will represent a concrete program state as a tuple with all the real-valued variables $\sigma : (x_1, ..., x_m) \in \mathbb{R}$. Similarly, we represent the abstract program state with the notation $\sigma^\sharp$. Our construction requires the following:

(1) A numerical abstract domain $\mathcal{A}$ which symbolically represents linear bounds exactly, and its reduced product with the interval domain.
(2) A *concretization* function $\gamma : \mathcal{A} \to \mathcal{P}(\mathbb{R}^m)$
(3) A *range* function that computes the tightest interval enclosing an abstract element $\sigma^\sharp \in \mathcal{A}$.

Inverting Flux Derivatives. For a flux $f : \mathbb{R} \to \mathbb{R}$, we require that f be continuous and contain at most a measure zero set of non-differentiable points. For points where f is differentiable, we require a method to compute f'^{-1} to find critical points where $f' = 0$. Additionally for any critical points of f where f is not differentiable, we require a list of those points. Lastly, for our synthesized abstract transformers to support the upwinding scheme we will require a method to compute f''^{-1}.

Hence, we give a general recipe for synthesizing precise abstract transformers for nonlinear FD schemes where the required ingredients are just the PDE's flux function and an automated way to compute the flux's critical points. While not all flux functions have easy to compute critical points or easy to invert derivatives, many popular PDE flux functions do, including all our benchmarks.

Linear Abstract Domains. Since the core part of Phocus's abstract transformer synthesis procedure uses *linear regression* to capture relationships between variables, we require an abstract domain that represents these linear relationships exactly. While many numeric domains satisfy this property, Phocus is implemented where $\mathcal{A}$ is the reduced product of the zonotope domain [23] with the interval domain [62]. In the zonotope domain, variables are encoded as an affine forms, which represent symbolic first-order polynomials over noise symbols $\epsilon_j \in [-1, 1]$. For a variable x, its affine form over p noise symbols is denoted as $\widehat{x}$ where $\widehat{x} = a_0 + \sum_{j=1}^{p} a_j \epsilon_j$ and each $a_j \in \mathbb{R}$.

A zonotope $\mathcal{Z} \subseteq \mathbb{R}^m$ is a collection of affine forms which determines the joint range of all variables. Zonotopes express linear dependencies exactly. Additionally, due to the reduced product, our abstraction also keeps a separate interval bound using the interval abstract domain $\mathbb{IR}$. To retrieve from the abstract state $\sigma^\sharp$ the symbolic affine form $a_0 + \sum a_j \epsilon_j$ associated to a variable x, we will write $\sigma^\sharp[x].\widehat{x}$ and to retrieve a specific (symbolic) coefficient a_j, we may write $\sigma^\sharp[x].a_j$. Similarly to retrieve the corresponding interval bounds for x we can access $\sigma^\sharp[x].[l_x, u_x]$.

Hence for our instantiation of Phocus with the reduced product of zonotopes and intervals, the concretization function $\gamma : \mathcal{A} \to \mathcal{P}(\mathbb{R}^m)$ is defined as:

$$\gamma(\sigma^\sharp) = \{\sigma : \sigma[x_i] \in \sigma^\sharp[x_i].[l_{x_i}, u_{x_i}] \wedge \exists \epsilon_1, ..., \epsilon_n \in [\text{-}1, 1].\forall j \in \{1, ..., p\}, \sigma[x_i] = \sigma^\sharp[x_i].a_0 + \sum_{j=1}^{p} \sigma^\sharp[x_i].a_j \epsilon_j\}$$

An abstract state $\sigma^\sharp$ includes all concrete states σ whose variables x_i lie within their respective interval bounds $[l_{x_i}, u_{x_i}]$ *and* are such that there exists an assignment of the noise symbols $\epsilon_j \in [1, 1]$ so that the evaluation of $\widehat{x_i}$ with that assignment equals the program state's value of x_i, which comes from affine arithmetic's fundamental invariant [83]. As later shown in Section 4.5, our synthesis procedure remains sound for any abstract domain that encodes linear relationships exactly.

4 Synthesizing Precise Abstract Transformers for Finite Difference Schemes

We now present Phocus, our optimization-based synthesis of precise abstract transformers for FD methods used to solve nonlinear hyperbolic PDEs. Using a linear regression technique (inspired by [49]), Phocus synthesizes abstract transformers for the three FD schemes of Section 3.2.

Unlike DNNs, the value of a mesh point is only a function of a few of the previous timestep's mesh points. Hence the (nonlinear) stencils for FD methods are *sparse*. Phocus leverages this sparsity to keep the dimension of the multivariate optimization problems tractable.

Another key insight Phocus relies upon is that given a way to minimize and maximize a function $f(x)$, one can use that technique as a building block to optimize a linear shifted version of the function: $f(x) - (Ax + b)$. Hence we can use the same building blocks to achieve optimal interval domain abstract transformers *and* precise zonotope domain transformers.

As a final insight, Phocus leverages both additive separability and multilinearity to enable the clean reduction of multivariate optimization problems into tractable 1D subproblems.

4.1 Lax-Friedrichs Synthesized Abstract Transformer

We now detail our procedure for synthesizing an abstract transformer for the Lax-Friedrichs finite difference scheme of Eq. 5. The entire procedure is given in Algorithm 1.

4.1.1 Grid Sampling and Linear Regression. The abstract transformer's first step is to compute the joint range of u_{i+1}^n and u_{i-1}^n, which we perform with the *Range* function. Upon obtaining a 2D joint range, we sample uniformly spaced *grid* points in that range and evaluate the concrete Lax-Friedrichs stencil (Eq. 5) on those grid points to compute pts. We then use linear regression to find the best linear approximation of the Lax-Friedrichs stencil for this grid.

4.1.2 Optimization Problem. After using linear regression to synthesize the best fit coefficients A, B, and C, we optimize for the maximum deviation between the linear fit and the original Lax-Friedrichs stencil. This deviation, D, gives a sound enclosure around the linear bounds and is shown in Eq. 11.

Algorithm 1 Lax Friedrichs Abstract Transformer $T^{\sharp}_{LF}$

```
**Input:** Abstract state $\sigma^{\sharp}$ where $\widehat{u^n_{i+1}} = \sigma^{\sharp}[u^n_{i+1}].\widehat{u^n_{i+1}}$ and $\widehat{u^n_{i-1}} = \sigma^{\sharp}[u^n_{i-1}].\widehat{u^n_{i-1}}$
$l_1, u_1 \leftarrow Range(\sigma^{\sharp}[u^n_{i+1}])$
$l_2, u_2 \leftarrow Range(\sigma^{\sharp}[u^n_{i-1}])$
$grid \leftarrow GridSample([l_1, u_1] \times [l_2, u_2])$
$pts \leftarrow \{\frac{1}{2}(x+y) - \frac{\Delta t}{2\Delta x}(f(x) - f(y)) : (x,y) \in grid\}$
$A, B, C \leftarrow LinearRegression(grid, pts)$
$D \leftarrow \max_{u^n_{i+1} \in [l_1,u_1], u^n_{i-1} \in [l_2,u_2]} \left|\frac{1}{2}(u^n_{i+1} + u^n_{i-1}) - \frac{\Delta t}{2\Delta x}(f(u^n_{i+1}) - f(u^n_{i-1})) - (Au^n_{i+1} + Bu^n_{i-1} + C)\right|$   ▷Eq. 12
$lo \leftarrow \min_{u^n_{i+1} \in [l_1,u_1], u^n_{i-1} \in [l_2,u_2]} \frac{1}{2}(u^n_{i+1} + u^n_{i-1}) - \frac{\Delta t}{2\Delta x}(f(u^n_{i+1}) - f(u^n_{i-1}))$   ▷Eq. 15
$up \leftarrow \max_{u^n_{i+1} \in [l_1,u_1], u^n_{i-1} \in [l_2,u_2]} \frac{1}{2}(u^n_{i+1} + u^n_{i-1}) - \frac{\Delta t}{2\Delta x}(f(u^n_{i+1}) - f(u^n_{i-1}))$   ▷Eq. 16
**return** $A\widehat{u^n_{i+1}} + B\widehat{u^n_{i-1}} + C + D\epsilon_{new}$, $[lo, up]$
```

$$D = \max_{u^n_{i+1} \in [l_1,u_1], u^n_{i-1} \in [l_2,u_2]} \left|\frac{1}{2}(u^n_{i+1} + u^n_{i-1}) - \frac{\Delta t}{2\Delta x}\big(f(u^n_{i+1}) - f(u^n_{i-1})\big) - (Au^n_{i+1} + Bu^n_{i-1} + C)\right| \tag{11}$$

While this multivariate, nonconvex constrained maximization problem appears difficult, we reduce this problem to enumeration over a finite list of points by solving tractable 1D optimization subproblems. In general (and for other schemes like Lax-Wendroff) reducing a multi-dimensional optimization problem to independent 1D subproblems is not always possible, however by identifying this pattern in the Lax-Friedrichs scheme, we can leverage this idea. The set of points where a minimum or maximum could occur is given by the finite set S_{lin}:

$$S_{lin} = \Big(\{l_1, u_1\} \cup \{x \in [l_1, u_1] : (\frac{1}{2} - A) - \frac{\Delta t}{2\Delta x} f'(x) = 0 \vee f'(x) \text{ d.n.e}\}\Big) \times \Big(\{l_2, u_2\} \cup \{x \in [l_2, u_2] : (\frac{1}{2} - B) + \frac{\Delta t}{2\Delta x} f'(x) = 0 \vee f'(x) \text{ d.n.e}\}\Big)$$

Because we assumed a way to compute the preimage of f', we can compute f'^{-1} at the points $\frac{(\frac{1}{2}-A)2\Delta x}{\Delta t}$ and $\frac{-(\frac{1}{2}-B)2\Delta x}{\Delta t}$, which yields a key insight. Since the structure of the objective is the nonlinear flux f minus a linear expression (obtained by linear regression), when taking derivatives, the objective's derivative reduces to f' minus constants. Due to this fact, we can still use f'^{-1} to find critical points of the objective, we simply shift the argument over by those constants. While in general, this technique is not always possible, by leveraging the special structure of both the FD stencil and the choice of a linear numeric abstract domain, Phocus can use this insight. In addition, we include any points within the input intervals where the derivative f' does not exist, which we denote as d.n.e. Hence the actual optimization problem reduces to:

$$D = \max_{(u^n_{i+1}, u^n_{i-1}) \in S_{lin}} \left|\frac{1}{2}(u^n_{i+1} + u^n_{i-1}) - \frac{\Delta t}{2\Delta x}\big(f(u^n_{i+1}) - f(u^n_{i-1})\big) - (Au^n_{i+1} + Bu^n_{i-1} + C)\right| \tag{12}$$

A simplified form of this optimization problem can be solved to produce interval lower and upper bounds, giving an optimal interval domain abstract transformer.

$$lo = \min_{u^n_{i+1} \in [l_1,u_1], u^n_{i-1} \in [l_2,u_2]} \frac{1}{2}(u^n_{i+1} + u^n_{i-1}) - \frac{\Delta t}{2\Delta x}\big(f(u^n_{i+1}) - f(u^n_{i-1})\big) \tag{13}$$

$$up = \max_{u^n_{i+1} \in [l_1,u_1], u^n_{i-1} \in [l_2,u_2]} \frac{1}{2}(u^n_{i+1} + u^n_{i-1}) - \frac{\Delta t}{2\Delta x}\big(f(u^n_{i+1}) - f(u^n_{i-1})\big) \tag{14}$$

While Eq. 13 and Eq. 14 define two separate optimization (sub)problems, the critical points that one must examine remain the same for both equations. We denote the finite set of points where the minimum or maximum could be achieved as S_{int}. By enumerating over the finitely many points in this set, we can solve both Eq. 13 and Eq. 14 simultaneously.

$$S_{int} = \Big(\{l_1, u_1\} \cup \{x \in [l_1, u_1] \ : \ \frac{1}{2} - \frac{\Delta t}{2\Delta x} f'(x) = 0 \vee f'(x) \text{ d.n.e}\}\Big) \times \Big(\{l_2, u_2\} \cup \{x \in [l_2, u_2] \ : \ \frac{1}{2} + \frac{\Delta t}{2\Delta x} f'(x) = 0 \vee f'(x) \text{ d.n.e}\}\Big)$$

As before, since we have a way to compute the preimage of f', we can compute f'^{-1} at the point $\frac{\Delta x}{\Delta t}$ and also use the same points as before where the derivative does not exist. Hence we have reduced the original multivariate problem to the 1D problem of solving $\frac{1}{2} - \frac{\Delta t}{2\Delta x} f'(x) = 0$ Thus the optimization problems become:

$$lo = \min_{(u_{i+1}^n, u_{i-1}^n) \in S_{int}} \frac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \frac{\Delta t}{2\Delta x}\big(f(u_{i+1}^n) - f(u_{i-1}^n)\big) \tag{15}$$

$$up = \max_{(u_{i+1}^n, u_{i-1}^n) \in S_{int}} \frac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \frac{\Delta t}{2\Delta x}\big(f(u_{i+1}^n) - f(u_{i-1}^n)\big) \tag{16}$$

Thus the same technique of finding critical points of f by computing f'^{-1} produces sound abstract transformers for both a linear abstract domain (e.g., zonotopes or DeepPoly) and the interval domain. Further, we can use both of these abstract transformers in a reduced product to refine the results of the other. Having computed these bounds, we now state their optimality.

Theorem 4.1. *Optimality - The interval bounds computed in Equations 15 and 16 are optimal and thus comprise the most precise interval domain abstract transformer for the Lax-Friedrichs scheme.*

4.2 Leapfrog Synthesized Abstract Transformer

We now describe our procedure for synthesizing a precise abstract transformer for the Leapfrog FD scheme, which is shown in Algorithm 2. In contrast to the Lax-Friedrichs scheme, the Leapfrog scheme requires optimizing over 3 variables (since it is a 3-point stencil) to compute the abstraction of each timestep's mesh points. Since Leapfrog requires 2 timesteps, we use the Lax-Friedrichs abstract transformer, $T_{LF}^{\sharp}$ to abstract the initial timestep.

4.2.1 Grid Sampling and Linear Regression. As before, the first task is to compute the joint range of u_{i+1}^n, u_{i-1}^n, and u_i^{n-1} which we again perform with the *Range* function. Upon obtaining a 3D *grid*, we take uniformly spaced points in that grid and evaluate the concrete Leapfrog stencil (Eq. 6) on those grid points to compute *pts*. Similar to before, we then use linear regression to find the best linear approximation of the Leapfrog stencil for this grid.

4.2.2 Optimization Problem. After having solved a linear regression problem to synthesize coefficients A, B, C, D, we now solve for the optimal enclosure amount $E \in \mathbb{R}$, shown in Eq. 17.

$$E = \max_{u_{i+1}^n \in [l_1, u_1], u_{i-1}^n \in [l_2, u_2], u_i^{n-1} \in [l_3, u_3]} \left| u_i^{n-1} - \frac{\Delta t}{\Delta x}\big(f(u_{i+1}^n) - f(u_{i-1}^n)\big) - (Au_{i+1}^n + Bu_{i-1}^n + Cu_i^{n-1} + D) \right| \tag{17}$$

As in the case of Lax-Friedrichs, we can reduce the problem to enumeration over a finite set of points obtained by solving independent 1D subproblems. The set of points where a maximum or minimum could occur is given by the following finite set S_{lin}:

$$S_{lin} = \Big(\{l_1, u_1\} \cup \{x \in [l_1, u_1] : \text{-}A\text{-}\frac{\Delta t}{\Delta x} f'(x) = 0 \vee f'(x) \text{ d.n.e}\}\Big) \times \Big(\{l_2, u_2\} \cup \{x \in [l_2, u_2] : \text{-}B + \frac{\Delta t}{\Delta x} f'(x) = 0 \vee f'(x) \text{ d.n.e}\}\Big) \times \{l_3, u_3\}$$

Algorithm 2 Leapfrog Abstract Transformer: $T^{\sharp}_{leapfrog}$

```
Input: Abstract state $\sigma^{\sharp}$ where $\widehat{u^n_{i+1}} = \sigma^{\sharp}[u^n_{i+1}].\widehat{u^n_{i+1}}$, $\widehat{u^n_{i-1}} = \sigma^{\sharp}[u^n_{i-1}].\widehat{u^n_{i-1}}$, and $\widehat{u^{n-1}_i} = \sigma^{\sharp}[u^{n-1}_i].\widehat{u^{n-1}_i}$
$l_1, u_1 \leftarrow Range(\sigma^{\sharp}[u^n_{i+1}])$
$l_2, u_2 \leftarrow Range(\sigma^{\sharp}[u^n_{i-1}])$
$l_3, u_3 \leftarrow Range(\sigma^{\sharp}[u^{n-1}_i])$
$grid \leftarrow GridSample([l_1, u_1] \times [l_2, u_2] \times [l_3, u_3])$
$pts \leftarrow \{z - \frac{\Delta t}{\Delta x}(f(x) - f(y)) : (x, y, z) \in grid\}$
$A, B, C, D \leftarrow LinearRegression(grid, pts)$
$E \leftarrow \max_{u^n_{i+1} \in [l_1,u_1], u^n_{i-1} \in [l_2,u_2], u^{n-1}_i \in [l_3,u_3]} \left| u^{n-1}_i - \frac{\Delta t}{\Delta x}(f(u^n_{i+1}) - f(u^n_{i-1})) - (Au^n_{i+1} + Bu^n_{i-1} + Cu^{n-1}_i + D) \right|$
$lo \leftarrow \min_{u^n_{i+1} \in [l_1,u_1], u^n_{i-1} \in [l_2,u_2], u^{n-1}_i \in [l_3,u_3]} u^{n-1}_i - \frac{\Delta t}{\Delta x}(f(u^n_{i+1}) - f(u^n_{i-1}))$    ▷Eq. 21
$up \leftarrow \max_{u^n_{i+1} \in [l_1,u_1], u^n_{i-1} \in [l_2,u_2], u^{n-1}_i \in [l_3,u_3]} u^{n-1}_i - \frac{\Delta t}{\Delta x}(f(u^n_{i+1}) - f(u^n_{i-1}))$    ▷Eq. 22
return $A\widehat{u^n_{i+1}} + B\widehat{u^n_{i-1}} + C\widehat{u^{n-1}_i} + D + E\epsilon_{new}$, $[lo, up]$
```

As before, since we assume a way to compute f'^{-1}, the critical points along the u^n_{i+1} dimension can be found by solving $-A - \frac{\Delta t}{\Delta x} f'(x)$ and similarly the critical points along the u^n_{i-1} dimension can be found by solving $-B - \frac{\Delta t}{\Delta x} f'(x)$. Both are independent 1D problems that can be easily solved. Additionally, along the u^{n-1}_i dimension we only need to check the 2 corners $\{l_3, u_3\}$. Hence the original 3D optimization problem reduces to:

$$E = \max_{(u^n_{i+1}, u^n_{i-1}, u^{n-1}_i) \in S_{lin}} \left| u^{n-1}_i - \frac{\Delta t}{\Delta x}\big(f(u^n_{i+1}) - f(u^n_{i-1})\big) - (Au^n_{i+1} + Bu^n_{i-1} + Cu^{n-1}_i + D) \right| \tag{18}$$

Similarly, we can solve for direct lower and upper bounds by optimizing over a simplified objective function:

$$lo = \min_{u^n_{i+1} \in [l_1,u_1], u^n_{i-1} \in [l_2,u_2], u^{n-1}_i \in [l_3,u_3]} u^{n-1}_i - \frac{\Delta t}{\Delta x}\big(f(u^n_{i+1}) - f(u^n_{i-1})\big) \tag{19}$$

$$up = \max_{u^n_{i+1} \in [l_1,u_1], u^n_{i-1} \in [l_2,u_2], u^{n-1}_i \in [l_3,u_3]} u^{n-1}_i - \frac{\Delta t}{\Delta x}\big(f(u^n_{i+1}) - f(u^n_{i-1})\big) \tag{20}$$

As with the Lax Friedrichs bounds (Eqs. 15-16), both the lower bound (Eq. 19) and upper bound (Eq. 20) for the Leapfrog scheme share the same set of critical points. Hence we can solve these optimization problems by enumerating over the corner points and any critical point of f that is contained in $[l_1, u_1]$ or $[l_2, u_2]$. This set S is described below:

$$S = \Big(\{l_1, u_1\} \cup \{x \in [l_1, u_1] \ : \ f'(x) = 0 \vee f'(x) \text{ d.n.e}\}\Big) \times \Big(\{l_2, u_2\} \cup \{x \in [l_2, u_2] \ : \ f'(x) = 0 \vee f'(x) \text{ d.n.e}\}\Big) \times \{l_3, u_3\}$$

Thus the optimization problems for the direct interval lower and upper bounds reduce to:

$$lo = \min_{(u^n_{i+1}, u^n_{i-1}, u^{n-1}_i) \in S} u^{n-1}_i - \frac{\Delta t}{\Delta x}\big(f(u^n_{i+1}) - f(u^n_{i-1})\big) \tag{21}$$

$$up = \max_{(u_{i+1}^n, u_{i-1}^n, u_i^{n-1}) \in S} u_i^{n-1} - \frac{\Delta t}{\Delta x}\left(f(u_{i+1}^n) - f(u_{i-1}^n)\right) \tag{22}$$

We can now state the optimality of Phocus's bounds.

Theorem 4.2. *Optimality - The interval bounds computed in Equations 21 and 22 are optimal and thus comprise the most precise interval domain abstract transformer for the Leapfrog scheme.*

4.3 Upwinding Synthesized Abstract Transformer

A key reason why we encode the upwinding scheme with Eq. 8 instead of with Eq. 7 is because Eq. 8 *avoids* an explicit branch, thus making the static analysis easier. To statically analyze branches with linear abstract domains (like zonotopes) requires expensive and imprecise join operations [29, 79], something our encoding successfully avoids. Furthermore, our encoding (Eq. 8) allows us to leverage multilinear optimization techniques to synthesize precise abstract transformers.

4.3.1 Multilinearity. A novel contribution of our work is our generalization of the multilinearity ideas of [47] (which only handles the interval domain) to support linear abstract domains like zonotopes. We now formally define multilinearity.

Definition 4.3. Multilinear Functions [47]: A function $f(x_1, ..., x_n) : \mathbb{R}^n \rightarrow \mathbb{R}$ is a multilinear polynomial if f is both a polynomial and if for all $j \in \{1, ..., n\}$, the second derivative $\frac{\partial^2 f}{\partial x_j^2} = 0$.

In interval analysis, multilinear polynomials defined over hypercubes can be easily minimized and maximized by checking only the hypercube's corner points since no interior extrema exist [47]. We now generalize Definition 4.3 to support *piecewise* versions of these functions.

Definition 4.4. Piecewise Multilinear Polynomials: We say that a function $f(x_1, ..., x_n) : \mathbb{R}^n \rightarrow \mathbb{R}$ is a *piecewise* multilinear polynomial, if f can be represented as a piecewise function where each piece is a multilinear polynomial.

Piecewise multilinear polynomials are almost as easy to optimize as multilinear polynomials, the only caveat is that one must check boundaries where different pieces touch, which we now state:

Lemma 4.5. *A piecewise multilinear polynomial defined over a hypercube obtains both its minima and maxima at either a corner of the hypercube or at a boundary point where different pieces touch.*

Hence our work generalizes multilinearity by extending it to (a) linear abstract domains like zonotopes and (b) piecewise functions. These insights will ensure the soundness of Algorithm 3.

4.3.2 Computational Pattern. The synthesis of the abstract transformer for the Upwinding scheme comes in two parts. We first abstract the derivative of the flux function, f' and then abstract the pattern in Eq. 23 using our generalization of the theory of multilinear functions to affine forms.

$$M(x_1, x_2, x_3, x_4) = x_1 - \tfrac{\Delta t}{\Delta x}\left(\max(0, x_2)\cdot(x_1 - x_3) + \min(0, x_2)\cdot(x_4 - x_1)\right) \tag{23}$$

The key insight for the upwinding scheme is that for fixed Δt, Δx, the function $M(\cdot)$ is piecewise multilinear. We note that the computational pattern M has 4 nonlinearities: 2 multiplications and the $\max(0, x_2)$ and $\min(0, x_2)$ functions, which are equivalent to ReLU activations from DNNs. However, as the DNN verification community observed [76], ReLU nonlinearities can cause imprecision to accumulate, hence why we synthesize a precise abstract transformer using optimization.

Algorithm 3 Upwinding Abstract Transformer $T^{\sharp}_{Upwind}$

```
**Input:** Abstract state $\sigma^{\sharp}$ where $\widehat{u^n_{i+1}} = \sigma^{\sharp}[u^n_{i+1}].\widehat{u^n_{i+1}}$, $\widehat{u^n_{i-1}} = \sigma^{\sharp}[u^n_{i-1}].\widehat{u^n_{i-1}}$, and $\widehat{u^n_i} = \sigma^{\sharp}[u^n_i].\widehat{u^n_i}$
$l_1, u_1 \leftarrow Range(\sigma^{\sharp}[u^n_i])$
$grid \leftarrow GridSample([l_1, u_1])$
$pts \leftarrow \{f'(x) : x \in grid\}$
$A', B' \leftarrow LinearRegression(grid, pts)$
$C' \leftarrow \max_{u^n_i \in [l_1,u_1]} \left| f'(u^n_i) - (A'u^n_i + B') \right|$
$l_2 \leftarrow \min_{u^n_i \in [l_1,u_1]} f'(u^n_i)$
$u_2 \leftarrow \max_{u^n_i \in [l_1,u_1]} f'(u^n_i)$
$\widehat{x_2} \leftarrow A' \cdot \widehat{u^n_i} + B' + C'\epsilon_{new}, [l_2, u_2]$
$l_3, u_3 \leftarrow Range(\sigma^{\sharp}[u^n_{i-1}])$
$l_4, u_4 \leftarrow Range(\sigma^{\sharp}[u^n_{i+1}])$
$grid_2 \leftarrow GridSample([l_1, u_1] \times [l_2, u_2] \times l_3, u_3] \times [l_4, u_4])$
$pts_2 \leftarrow \{x_1 - \frac{\Delta t}{\Delta x}\Big(\max(0, x_2) \cdot (x_1 - x_3) + \min(0, x_2) \cdot (x_4 - x_1)\Big) : (x_1, x_2, x_3, x_4) \in grid_2\}$
$A, B, C, D, E \leftarrow LinearRegression(grid_2, pts_2)$
$F \leftarrow \max_{(x_1,x_2,x_3,x_4) \in S_{lin}} \left| x_1 - \frac{\Delta t}{\Delta x}(\max(0, x_2)(x_1\text{-}x_3) + \min(0, x_2)(x_4\text{-}x_1))\text{-}Ax_1\text{-}Bx_2\text{-}Cx_3\text{-}Dx_4\text{-}E \right|$
$lo \leftarrow \min_{(x_1,x_2,x_3,x_4) \in S_{lin}} (x_1 - \frac{\Delta t}{\Delta x}(\max(0, x_2)(x_1 - x_3) + \min(0, x_2)(x_4 - x_1)))$   ▷Eq. 29
$up \leftarrow \max_{(x_1,x_2,x_3,x_4) \in S_{lin}} (x_1 - \frac{\Delta t}{\Delta x}(\max(0, x_2)(x_1 - x_3) + \min(0, x_2)(x_4 - x_1)))$   ▷Eq. 30
**return** $A\widehat{u^n_i} + B\widehat{x_2} + C\widehat{u^n_{i-1}} + D\widehat{u^n_{i+1}} + E + F\epsilon_{new}, \ [lo, up]$
```

4.3.3 Optimization Problem. Unlike the Lax-Friedrichs and Leapfrog schemes which are abstracted in a single step, Algorithm 3 splits the abstract transformer for upwinding into the composition of two steps. First, we use linear regression to synthesize an abstract transformer for the flux function's derivative. The regression computes coefficients $A', B' \in \mathbb{R}$ that provide a good linear approximation of f'. We then solve for the approximation's enclosure term $C' \in \mathbb{R}$ as:

$$C' = \max_{u^n_i \in [l_{u^n_i}, u_{u^n_i}]} |f'(u^n_i) - (A'u^n_i + B')| \tag{24}$$

Similarly to how the Lax-Friedrichs and Leapfrog abstract transformers assumed a way to compute f'^{-1}, for the upwinding abstract transformer, we require a method to compute f''^{-1} to solve for the critical points that maximize Eq. 24. The points to check are the set $S' \subset [l_{u^n_i}, u_{u^n_i}]$:

$$S' = \{l_{u^n_i}, u_{u^n_i}\} \cup \{x \in [l_{u^n_i}, u_{u^n_i}] : f''(x) - A' = 0\}$$

We compute $f''^{-1}(A')$ to find the critical points. We also directly solve for interval bounds l_2, u_2:

$$l_2 = \min_{u^n_i \in [l_{u^n_i}, u_{u^n_i}]} f'(u^n_i) \qquad u_2 = \max_{u^n_i \in [l_{u^n_i}, u_{u^n_i}]} f'(u^n_i)$$

To solve for l_2 and u_2, the set of critical points we must consider is $S'_{int} \subset [l_{u^n_i}, u_{u^n_i}]$:

$$S'_{int} = \{l_{u^n_i}, u_{u^n_i}\} \cup \{x \in [l_{u^n_i}, u_{u^n_i}] : f''(x) = 0\}$$

We finally take $\widehat{x_2}$ as the affine form of the intermediate result of propagating the mesh value through the derivative of the flux function.

$$\widehat{x_2} = A'\widehat{u_i^n} + B' + C'\epsilon_{new}, [l_2, u_2]$$

After synthesizing an abstract transformer for the intermediate result of applying the (nonlinear) flux derivative to the solution $f'(u)$, we then compose this intermediate result with a synthesized abstract transformer for the piecewise multilinear pattern of Eq. 23. To synthesize the abstract transformer for the piecewise multilinear pattern, we first perform another linear regression to obtain constants $A, B, C, D, E \in \mathbb{R}$. To compute the enclosure amount $F \in \mathbb{R}$, we solve Eq. 25:

$$\max_{x_1 \in [l_1,u_1], x_2 \in [l_2,u_2], x_3 \in [l_3,u_3], x_4 \in [l_4,u_4]} \left| x_1 - \tfrac{\Delta t}{\Delta x}\big(\max(0,x_2)(x_1 - x_3) + \min(0,x_2)(x_4 - x_1)\big) - (Ax_1 + Bx_2 + Cx_3 + Dx_4 + E) \right| \tag{25}$$

The set of critical points to examine is given as $S_{lin} \subset [l_1, u_1] \times [l_2, u_2] \times [l_3, u_3] \times [l_4, u_4]$ where:

$$S_{lin} = \{l_1, u_1\} \times \{x_2 \in \{l_2, 0, u_2\} : l_2 \le x_2 \le u_2\} \times \{l_3, u_3\} \times \{l_4, u_4\} \tag{26}$$

Intuitively, if the range $[l_2, u_2]$ includes 0, then 0 is included in the second set. The reason is that the $\max(0, x_2)$ and $\min(0, x_2)$ functions are piecewise and 0 is the nondifferentiable point where two pieces touch, hence that point must be accounted for. We likewise solve for direct interval lower and upper bounds of the piecewise multilinear expression respectively in Eqs. 27 and 28.

$$lo = \min_{x_1 \in [l_1,u_1], x_2 \in [l_2,u_2], x_3 \in [l_3,u_3], x_4 \in [l_4,u_4]} x_1 - \tfrac{\Delta t}{\Delta x}\big(\max(0,x_2)(x_1 - x_3) + \min(0,x_2)(x_4 - x_1)\big) \tag{27}$$

$$up = \max_{x_1 \in [l_1,u_1], x_2 \in [l_2,u_2], x_3 \in [l_3,u_3], x_4 \in [l_4,u_4]} x_1 - \tfrac{\Delta t}{\Delta x}\big(\max(0,x_2)(x_1 - x_3) + \min(0,x_2)(x_4 - x_1)\big) \tag{28}$$

Unlike the Lax-Friedrichs and Leapfrog schemes, due to upwinding scheme's piecewise multilinearity, we can check the *same set* of critical points, S_{lin} (Eq. 26) to solve the direct interval lower (Eq. 27) and upper (Eq. 28) bound optimization problems, ultimately reducing the problems to:

$$lo = \min_{(x_1,x_2,x_3,x_4) \in S_{lin}} x_1 - \tfrac{\Delta t}{\Delta x}\big(\max(0,x_2)(x_1 - x_3) + \min(0,x_2)(x_4 - x_1)\big) \tag{29}$$

$$up = \max_{(x_1,x_2,x_3,x_4) \in S_{lin}} x_1 - \tfrac{\Delta t}{\Delta x}\big(\max(0,x_2)(x_1 - x_3) + \min(0,x_2)(x_4 - x_1)\big) \tag{30}$$

4.4 Soundness

We now prove the soundness of the abstract transformers that Phocus synthesizes for the Lax-Friedrichs, Leapfrog, and Upwinding FD schemes. We first state a useful lemma about optimizing the absolute value of additively separable functions:

Lemma 4.6. *Let $g(x_1, ..., x_n) : \mathbb{R}^n \to \mathbb{R}$ where $g(x_1, ..., x_n) = \sum_{i=1}^n g_i(x_i)$. Then for any $S \subseteq \mathbb{R}^n$*

$$\max_S\{|g(x_1, .., x_n)|\} = \max\{\sum \max_{S_{x_i}} g_i(x_i), -\sum \min_{S_{x_i}} g_i(x_i)\}$$

where each S_{x_i} is the projection of S onto the x_i^{th} variable.

Proof. By the definition of the absolute value function $|a| = \max(a, -a)$ we have:

$$\max_S\{|g(x_1, ..., x_n)|\} = \max_S\{\max(g(x_1, ..., x_n), -g(x_1, ..., x_n)\}$$

Further, we can interchange the order of the max functions and rewrite max $-g$ using $-\min g$.

$$= \max\{\max_S g(x_1,...,x_n), \max_S -g(x_1,...,x_n) = \max\{\max_S g(x_1,...,x_n), -\min_S g(x_1,...,x_n)\}\}$$

$g(x_1,...,x_n) = \sum_{i=1}^n g_i(x_i)$, g is additively separable, hence $\max_S g(x_1,...,x_n) = \sum_{i=1}^n \max_S g_i(x)$ and $\min_S g(x_1,..,x_n) = \sum_{i=1}^n \min_S g_i(x_i)$, giving:

$$= \max\{\sum_{i=1}^n \max_S g_i(x_i), -\sum_{i=1}^n \min_S g_i(x_i)\}$$

Since each $g_i(x_i)$ does not depend on x_j for $j \neq i$ we can take projections to give:

$$= \max\{\sum_{i=1}^n \max_{S_{x_i}} g_i(x_i), -\sum_{i=1}^n \min_{S_{x_i}} g_i(x_i)\}$$

□

Corollary 4.7. *Let $g(x_1,...x_n) = \sum_i g_i(x_i)$ and $S \subseteq \mathbb{R}^n$ be a convex set. Then* $\max_S\{|g(x_1,...x_n)|\}$ *is realized at a point in the following set (where d.n.e means "does not exist"):*

$$\bigtimes_{i=1}^n \left(\{l_{x_i}, u_{x_i}\} \cup \{x_i \in [l_{x_i}, u_{x_i}] : g_i'(x_i) = 0 \lor g_i'(x_i) \text{ d.n.e}\}\right)$$

Proof. Since S is convex, the set projections onto each x_i are closed compact intervals, hence each $S_{x_i} = [l_{x_i}, u_{x_i}]$. Additionally by Fermat's stationary point theorem, the maximizer of each $\max_{S_{x_i}} g_i(x_i)$ will occur at a point in S_{x_i} that is either a boundary point, a non-differentiable point of g_i or a point where $g_i'(x_i) = 0$. Since Fermat's theorem covers both minimums and maximums, that same set of points is also guaranteed to contain the minimizer each $\min_{S_{x_i}} g_i(x_i)$.

The boundary points for each S_{x_i} are just the pair $\{l_{x_i}, u_{x_i}\}$. Similarly the set of non differentiable points and vanishing derivative points in each S_{x_i} are $\{x_i \in [l_{x_i}, u_{x_i}] : g_i'(x_i) = 0 \lor g_i'(x_i) \text{ d.n.e}\}$

Hence the maximizer of each $\max_{S_{x_i}} g_i(x_i)$ and minimizer of each $\min_{S_{x_i}} g_i(x_i)$ will occur in the set $\{l_{x_i}, u_{x_i}\} \cup \{x_i \in [l_{x_i}, u_{x_i}] : g_i'(x_i) = 0 \lor g_i'(x_i) \text{ d.n.e}\}$. Since each i^{th} dimension is independent due to the additive separability of g, we can take the Cartesian product along all n dimensions. □

The insights of Lemma 4.6 and Corollary 4.7 are that a multivariate, constrained, possibly nonconvex objective that may also contain points of nondifferentiability can be optimized by only examining a small finite set of points for a collection of tractable 1D optimization subproblems. In addition, because these 1D problems are independent, they can theoretically be parallelized.

We now state one additional lemma relating the soundness of linear bounds to optimization:

Lemma 4.8. *Let $f(x_1,...,x_m) : \mathbb{R}^m \to \mathbb{R}$, let $a_0,...,a_m \in \mathbb{R}$ and let $B = [l_{x_1}, u_{x_1}] \times ... \times [l_{x_m}, u_{x_m}] \subset \mathbb{R}^m$. Then the following implication holds*

$$C = \max_B \left| f(x_1,...,x_m)\text{-}(a_0 + \sum_{j=1}^m a_j x_j) \right| \Rightarrow \forall (x_1,...,x_m) \in B : a_0\text{-}C + \sum_{j=1}^m a_j x_j \leq f(x_1,...,x_m) \leq a_0 + C + \sum_{j=1}^m a_j x_j$$

Having now stated these preliminary results, we can prove the soundness of Phocus.

Theorem 4.9. *The Lax-Friedrichs synthesized abstract transformer $T_{LF}^{\sharp}$ for any $f : \mathbb{R} \to \mathbb{R}$ satisfying the properties of Section 3.3 is sound*

Proof. We first prove the soundness of the linear bounds before proving the soundness of the direct interval bounds. Let $f(x) : \mathbb{R} \to \mathbb{R}$ and A, B, C be the coefficients inferred by linear regression.

For the symbolic linear bounds to be sound we will need find a $D \in \mathbb{R}$ such that the following inequalities are always satisfied:

$$Au_{i+1}^n + Bu_{i-1}^n + (C-D) \leq \frac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \frac{\Delta t}{2\Delta x}\big(f(u_{i+1}^n) - f(u_{i-1}^n)\big) \leq Au_{i+1}^n + Bu_{i-1}^n + (C+D)$$

Thus to compute the offset adjustment $D \in \mathbb{R}$ we must solve for the maximum deviation between the linear approximation and the Lax-Friedrichs stencil, as such a D would imply the above inequality by Lemma 4.8:

$$D = \max_{u_{i+1}^n \in [l_1,u_1], u_{i-1}^n \in [l_2,u_2]} \left| \tfrac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \tfrac{\Delta t}{2\Delta x}\big(f(u_{i+1}^n) - f(u_{i-1}^n)\big) - (Au_{i+1}^n + Bu_{i-1}^n + C) \right|$$

However our claim is that the maximum value in the 2D box $[l_1, u_1] \times [l_2, u_2]$ will actually lie in a much smaller, finite set called S_{lin} which we define as follows:

$$\Big(\{l_1,u_1\} \cup \{x_1 \in [l_1,u_1] : (\frac{1}{2}\text{-}A)\text{-}\frac{\Delta t}{2\Delta x}f'(x_1) = 0 \vee f'(x_1) \text{ d.n.e}\}\Big) \times \Big(\{l_2,u_2\} \cup \{x_2 \in [l_2,u_2] : (\frac{1}{2}\text{-}B)\text{+}\frac{\Delta t}{2\Delta x}f'(x_2) = 0 \vee f'(x_2) \text{ d.n.e}\}\Big)$$

The core idea is that because the Lax-Friedrichs stencil, and the entire objective (including the linear regression coefficients) are additively separable, we can apply Lemma 4.6 and Corollary 4.7. We let $g_1(x_1) = (\frac{1}{2} - A)x_1 - \frac{\Delta t}{2\Delta x}f(x_1)$ and $g_2(x_2) = (\frac{1}{2} - B)x_2 + \frac{\Delta t}{2\Delta x}f(x_2) - C$.

We also note that for any $C_1, C_2, C_3 \in \mathbb{R}_{\neq 0}$ $f'(x_i)$ d.n.e $\iff$ $C_1 f'(x_i) + C_2 x + C_3$ d.n.e, hence $\{x_i \in [l_{x_i}, u_{x_i}] : f'(x_i) \text{ d.n.e}\} = \{x_i \in [l_{x_i}, u_{x_i}] : C_1 f'(x_i) + C_2 x + C_3 \text{ d.n.e}\}$

□

THEOREM 4.10. *The Leapfrog synthesized abstract transformer $T^\sharp_{Leapfrog}$ for any $f : \mathbb{R} \to \mathbb{R}$ satisfying the properties of Section 3.3 is sound.*

PROOF. As before, we first prove the soundness of the linear bounds before proving the soundness of the direct interval bounds. Let $f(x) : \mathbb{R} \to \mathbb{R}$ and A, B, C, D be the coefficients inferred by linear regression. For the linear bounds to be sound we need to find an $E \in \mathbb{R}$ such that the following inequalities are always satisfied:

$$Au_{i+1}^n + Bu_{i-1}^n + Cu_i^{n-1} + D - E \leq u_i^{n-1} - \tfrac{\Delta t}{\Delta x}\big(f(u_{i+1}^n) - f(u_{i-1}^n)\big) \leq Au_{i+1}^n + Bu_{i-1}^n + Cu_i^{n-1} + D + E$$

By lemma 4.8, an $E \in \mathbb{R}$ that satisfies the following equation would imply the above inequalities:

$$E = \max_{u_{i+1}^n \in [l_1,u_1], u_{i-1}^n \in [l_2,u_2], u_i^{n-1} \in [l_3,u_3]} \left| u_i^{n-1} - \frac{\Delta t}{\Delta x}\big(f(u_{i+1}^n) - f(u_{i-1}^n)\big) - (Au_{i+1}^n + Bu_{i-1}^n + Cu_i^{n-1} + D) \right|$$

The key observation is that this optimization problem involves maximizing the absolute value of an additively separable function, thus Lemma 4.6 and Corollary 4.7 apply equally well to this scheme as they did to Lax-Friedrichs.

We let $g_1(x_1) = -Ax_1 - \frac{\Delta t}{\Delta x}f(x_1)$, $g_2(x_2) = -Bx_2 + \frac{\Delta t}{\Delta x}f(x_2)$, and $g_3(x_3) = (1 - C)x_3 - D$. □

While we cannot leverage additive separability for the upwinding scheme the way we could for the Lax-Friedrichs and Leapfrog schemes, we can use multilinearity to prove the soundness. We state this result below.

THEOREM 4.11. *The Upwinding synthesized abstract transformer $T^\sharp_{Upwind}$ for any $f' : \mathbb{R} \to \mathbb{R}$ satisfying the properties of Section 3.3 is sound.*

Proof. We first prove the soundness of the synthesized abstract transformer for f', and then show the soundness of the synthesized abstract transformer for the multilinear pattern. For the linear bounds enclosing f' to be sound (by lemma 4.8) we must find a $C' \in \mathbb{R}$ such that:

$$C' = \max_{u_i^n \in [l_{u_i^n}, u_{u_i^n}]} |f'(u_i^n) - (A' u_i^n + B')|$$

By Fermat's theorem the optimizer will lie in the smaller set:

$$S_{lin} = \{l_{u_i^n}, u_{u_i^n}\} \cup \{x_1 \in [l_{u_i^n}, u_{u_i^n}] \ : \ f''(x_1) - A' = 0 \lor f''(x_1) \text{ d.n.e}\}$$

For the zonotope abstract domain $\widehat{x_2} = A'\widehat{u_i^n} + B' + C'$ is a sound affine form enclosing $f'(u_i^n)$. We then synthesize linear bounds for $x_1 - \frac{\Delta t}{\Delta x}(\max(0, x_2) \cdot (x_1 - x_3) + \min(0, x_2) \cdot (x_4 - x_1))$. To do so we must find $F \in \mathbb{R}$ such that:

$$Ax_1 + Bx_2 + Cx_3 + Dx_4 + E\text{-}F \le x_1\text{-}\tfrac{\Delta t}{\Delta x}(\max(0, x_2)(x_1\text{-}x_3) + \min(0, x_2)(x_4\text{-}x_1)) \le Ax_1 + Bx_2 + Cx_3 + Dx_4 + E + F$$

Hence by Lemma 4.8, we can solve the following optimization problem to compute F:

$$\max_{x_1 \in [l_1,u_1], x_2 \in [l_2,u_2], x_3 \in [l_3,u_3], x_4 \in [l_4,u_4]} \left| x_1\text{-}\tfrac{\Delta t}{\Delta x}\big(\max(0, x_2)(x_1\text{-}x_3) + \min(0, x_2)(x_4\text{-}x_1)\big)\text{-}(Ax_1 + Bx_2 + Cx_3 + Dx_4 + E) \right|$$

Using $\max(|a|) = \max(\max(a, -a))$ and interchanging the order of the max functions, we can split this optimization into the following problem, where F now equals:

$$\max\Big(\max_{x_1 \in [l_1,u_1], x_2 \in [l_2,u_2], x_3 \in [l_3,u_3], x_4 \in [l_4,u_4]} x_1\text{-}\tfrac{\Delta t}{\Delta x}\big(\max(0, x_2)(x_1\text{-}x_3) + \min(0, x_2)(x_4\text{-}x_1)\big)\text{-}(Ax_1 + Bx_2 + Cx_3 + Dx_4 + E),$$
$$\max_{x_1 \in [l_1,u_1], x_2 \in [l_2,u_2], x_3 \in [l_3,u_3], x_4 \in [l_4,u_4]} \text{-}\big(x_1\text{-}\tfrac{\Delta t}{\Delta x}\big(\max(0, x_2)(x_1\text{-}x_3) + \min(0, x_2)(x_4\text{-}x_1)\big)\text{-}(Ax_1 + Bx_2 + Cx_3 + Dx_4 + E)\big)\Big)$$

Since both inner maximization subproblems are over piecewise multilinear objectives, both inner subproblems will attain their respective maxima at either a corner point of the hypercube or a boundary point where different pieces touch (Lemma 4.5). This set of points is exactly S_{lin}:

$$S_{lin} = \{l_1, u_1\} \times \{x_2 \in \{l_2, 0, u_2\} : l_2 \le x_2 \le u_2\} \times \{l_3, u_3\} \times \{l_4, u_4\}$$

□

4.5 Generality

While this work uses the reduced product of zonotopes and intervals, our optimization-based procedure to find a precise linear approximation can be used with any abstract domain that represents linear relationships exactly. Hence Phocus's synthesis procedure yields linear bounds that could also be used with the DeepPoly [77] and polynomial zonotope [46] abstract domains.

4.6 Abstraction Space Complexity

A core benefit of synthesizing an abstract transformer for the entire PDE stencil instead of abstracting the individual operations is that when using the zonotope domain, one obtains large memory savings. The reason is that Phocus's synthesized abstract transformers introduce only a single noise symbol per application of the Lax-Friedrichs and Leapfrog stencils and only two symbols for upwinding. Such memory savings are critical for long running computations like multiple timesteps of PDEs because the number of noise symbols grows over the duration of a computation [1].

For the Lax-Friedrichs and Leapfrog scheme, if $OPS(f)$ is the number of nonlinear operations used to compute flux f, then for a naive application of zonotope abstract transformers the number of noise symbols (ϵ_j) introduced for an application of those stencils is $2 \cdot OPS(f)$. For an application of the upwinding stencil, the number introduced is up to $OPS(f') + 4$ new symbols. In contrast, Phocus introduces either 1 or 2 noise symbols per stencil application. Hence a naive use of zonotopes on our benchmarks introduces between 2-8× more noise symbols and memory overhead. Furthermore, since Phocus consumes less memory, in many cases, Phocus is both more precise *and* faster.

4.7 Formal Properties for PDEs

We can now encode the total variation, CFL condition, and the PDE solution's range as formal properties to statically analyze and verify. Unlike lower-level implementation concerns (e.g., floating point roundoff) our properties capture higher-level *algorithmic* effects. The reason for this focus is that higher-level numerical properties of PDE solutions like the total variation caused by spurious oscillations stem from algorithmic details that are often independent of the floating-point precision used [7]. Hence higher-level properties require a novel analysis. Thus by moving beyond floating-point issues, our work opens the door to a broad new class of formal verification problems.

4.7.1 Reachable Solution Set. When statically analyzing FD schemes, the abstract interpretation encloses a reachable set of PDE solutions. For hyperbolic PDEs, these bounds can certify physically meaningful information. For instance bounds that remain tight and never explode can certify the solution avoids any shock waves. For an abstractly computed program state $\sigma^\sharp$, the absence-of-shocks property can be formulated as:

$$\forall \sigma \in \gamma(\sigma^\sharp),\ \forall u_i^n \in \sigma : L \leq \sigma[u_i^n] \leq U \tag{31}$$

For global bounds $L, U \in \mathbb{R}$ where L and U are user-specified thresholds that empirically quantify what constitutes a shock wave.

4.7.2 Abstract CFL Condition. Our abstraction of the CFL condition is given in Definition 4.12:

Definition 4.12. Abstract CFL Condition - For a mesh spatially discretized by Δx and temporally discretized by Δt where the abstract PDE solution at the i^{th} mesh point at timestep n is given as $\sigma^\sharp[u_i^n]$, the abstract CFL condition is a numerical invariant that is satisfied if Eq. 32 holds:

$$\max_i \max\left(\left\{|f'(v)|\frac{\Delta t}{\Delta x} : v \in \gamma(\sigma^\sharp[u_i^n])\right\}\right) \leq C_{\max} \tag{32}$$

Eq. 32 formalizes the notion that *every* possible solution value in the abstract state must obey the original CFL condition from Definition 3.1. One can always use abstract automatic differentiation (as in [50–52, 63]) to bound f', however through algebraic manipulation, one can often do better. For example in our benchmarks (and in Section 2), for monotonic f'^{-1}, one can prove $f'^{-1}(\frac{-C_{\max}\Delta x}{\Delta t}) \leq l_{u_i^n} \wedge u_{u_i^n} \leq f'^{-1}(\frac{C_{\max}\Delta x}{\Delta t})$ or when f' is bounded, show that $f'(v) \leq C_{\max}\frac{\Delta x}{\Delta t}$ for any $v \in \mathbb{R}$.

4.7.3 Abstract Total Variation. Similarly, our interval abstraction of the total variation is given as:

$$TV_n^\sharp = \sum_i^{\sharp} |range(\widehat{u_i^n} - \widehat{u_{i-1}^n})|^\sharp \tag{33}$$

By computing a sound over-approximation of the total variation, Phocus formally certifies that the spurious numerical artifacts (e.g., oscillations) introduced by the FD scheme are bounded below a threshold. Hence our work is the first to certify properties of this nature for hyperbolic PDEs.

5 Methodology

We now discuss the methodology for our evaluation of Phocus across multiple different PDEs.

Implementation. We implement Phocus in Python and build upon the Affapy library [35] and other standard libraries [32, 67]. Phocus's implementation assumes ideal real arithmetic and thus is not floating-point sound though there exist several techniques to add floating-point soundness at the cost of additional runtime [61]. The experiments were run on a Dell desktop with 6 i5 CPUs and 7.6GB RAM. All timing measurements come from the Python `timeit` library.

PDE Benchmarks. We now describe the nonlinear hyperbolic PDE benchmarks for our evaluation. Since no prior hyperbolic PDE verification benchmark suite exists, we created our own based on well-known PDEs. All our PDE benchmarks follow the form of Eq. 3, the only difference that distinguishes each benchmark is the flux function f. The flux functions were chosen for their diverse properties (e.g., convex and nonconvex) since a different flux leads to different behavior of the PDE solution. The PDEs and their corresponding flux function we consider are:

(1) Burgers: $f(u) = \frac{1}{2}u^2$ [9]. This PDE and its convex flux function model nonlinear fluid flow.
(2) Flood Wave Height: $f(u) = cu\sqrt{u}$ [57, 60]. This flux function comes from Chezy's law, and the PDE models the height of a moving flood wave.
(3) Buckley-Leverett: $f(u) = \frac{u^2}{u^2+\frac{1}{4}(1-u)^2}$ [8]. This PDE models the flow of mixture of liquid and gas through a porous material. The nonlinear flux is neither concave nor convex.
(4) LWR Traffic Flow: $f(u) = c \cdot u(1-u)$ [58]. This PDE models congested vehicle traffic flow.
(5) Cubic: $f(u) = u^3$ [42] This nonconvex flux is used to study nonlinear wave propagation and is the same as in Section 2. Cubic fluxes are also used in Scientific ML [89].
(6) Trilinear: $f(u) = -ReLU(-u) + b \cdot ReLU(u-1)$ [33]. This nonconvex flux is called trilinear since the flux is a piecewise linear function comprised of 3 linear parts. Additionally, this flux contains two points of nondifferentiability.

Boundary Values, Initial Conditions and Computational Domain. The benchmark suite also specifies the PDE's initial and boundary conditions and the chosen discretization. We consider both Dirichlet and periodic boundary conditions. We use D for Dirichlet and P for periodic boundary conditions in the first column of Table 1. The initial condition for all benchmarks (except Buckley-Leverett and Flood Height) is $u(x,0) = (0 < x < 15.01)$? -0.015$(x(x-15))$: 0. For Flood Height, we shift $u(x,0)$ up by 1. For Buckley-Leverett, the initial condition is $u(x,0) = 100\sin(0.0001x)$. The computational domain contains one spatial dimension (x) with 20 mesh points spaced apart by Δx. Both Δx and Δt are specified per benchmark in Table 1.

Program Size and Analysis Scalability. PDE solver programs are concise (as shown in Section 2) since the whole program can be represented as a few loops applying a FD stencil to an array of meshpoints. Hence, a better way to understand program size is not raw lines of code, but rather the number of program variables computed and stored in the program state. This number can be large since it is the number of spatial mesh points multiplied by the number of timesteps, which in our biggest benchmark is >5000 total meshpoints. Our experimental timeout limit was 45 minutes.

Baselines. As a baseline, we compare against (1) unsound point-based sampling and (2) the combined zonotope-interval domain using its standard abstract transformers.

The unsound sampling of scalar input points is used as an *approximate* ground truth. We sample 10000 points and take the smallest and largest value encountered at each mesh point as our (unsound) sampled "interval". While one could sample more points to improve the estimate, we found there were diminishing returns as the sampled bounds did not change much when taking more than 10000 samples. We then see how tightly the sound methods (Phocus and the naive zonotope baseline) enclose the samples. For the sound naive zonotope baseline (2) we use standard abstract transformers. As an example, for the upwinding scheme, the $\min(0,v)$ and $\max(0,v)$ nonlinear operations use the DeepZ ReLU abstract transformers [76]. Similarly, the multiplication and division operations use the zonotope abstract transformers defined in [15].

6 Evaluation

6.1 Research Questions

Our evaluation seeks to answer the following research questions.

(1) Compared to a random sampling-based empirical (unsound) estimate of the ground truth ranges for each mesh point and time step, how tight are the bounds of Phocus?
(2) Can Phocus obtain significant precision improvements compared to a naive zonotope baseline implementation of the FD schemes?
(3) With respect to the precision difference between Phocus and the naive baseline, what trends emerge at different timesteps as the PDE solution evolves?
(4) How do the runtime and scalability of Phocus compare with the naive zonotope implementation that uses standard abstract transformers?

6.2 Experimental Results

Table 1 presents the main results of our experiments. The leftmost column describes which benchmark and which FD scheme (Lax-Friedrichs, Leapfrog or Upwinding) was applied to that benchmark. The BC column specifies the boundary condition, the $\pm\epsilon$ column shows the interval width of the initial condition and the next column contains the Δx, Δt parameters and number of timesteps n.

6.2.1 RQ 1: Comparison with Sampling. Because Phocus and the naive zonotope abstract transformers are sound, the empirical range obtained by first sampling random initial conditions within the given input intervals and then solving the PDE for that sampled input will always be contained in respective intervals of Phocus and the naive baseline. Hence one can compute the ratio: $\frac{u-l}{u_{smp}-l_{smp}}$ where l and u are *certified* lower and upper bounds while l_{smp} and u_{smp} are the empirically sampled bounds. The Phocus ratio and Naive ratio columns of Table 1 show this ratio for both our method and the naive zonotope baseline respectively. The width ratio is the geomean across all mesh points over all time steps. While taking the smallest (l_{smp}) and largest (u_{smp}) values obtained by solving the PDE concretely on 10000 randomly sampled conditions is not sound and not a ground truth, it serves as a useful estimate for how "close" the certified bounds are. Hence for these two columns, lower values closer to 1 are better. In all cases, Phocus's ratio was small (which is desirable), and at worst (in Buckley-Leverett Leapfrog) only 3.11x wider than the sampled range. Generally the Lax-Friedrichs scheme attains tighter ratios, while Upwind and Leapfrog produce wider ranges.

6.2.2 RQ 2: Precision vs Naive Zonotope. The % Fully Enclosed column shows the percentage of mesh points (over all timesteps) where Phocus's interval is *strictly contained* in the naive zonotope method's respective interval. Thus closer to 100% is better. An X signifies that the naive method failed, meaning its intervals blew up to infinity or an overflow occurred. In many cases like Burgers Leapfrog or LWR Lax-Friedrichs, all (100%) of Phocus's bounds are contained in the naive bounds.

Total Variation Bound. The Phocus and Naive *TV* columns show the respective certified bounds on the abstract total variation of Eq. 33. For these columns lower is better. In all benchmarks, the TV bound certified by Phocus is more precise (smaller) than the naive zonotope baseline's TV bound. The cases where the Naive TV bound is disproportionately large (e.g., Burgers Leapfrog) stem from the Naive baseline's interval exploding, whereas Phocus's bounds consistently remain tight.

CFL Condition Invariant. The Phocus and Naive *CFL* columns indicate whether the CFL condition (Eq. 32) was certified at every mesh point and every timestep before the last step (Safe) or if the CFL property was ever violated, at which timestep ($< n$) the violation occurred. In several cases (e.g., Cubic), the naive zonotope baseline violated the CFL condition whereas Phocus did not.

Phocus to Naive Ratio. Though we show ratios of Phocus and the naive zonotope's bounds relative to sampled ranges, one can compute the ratio of Phocus bounds relative to naive zonotope bounds

Table 1. PDE experimental configurations and summary results for each benchmark and each FD scheme. An X denotes failure of the standard zonotope abstract interpreter that we compare against as a baseline.

Benchmark	BC	$\pm\epsilon$	$\Delta x, \Delta t, n$	% Fully Enclosed	Phocus Ratio	Naive Ratio	Phocus *TV*	Naive *TV*	Phocus *CFL*	Naive *CFL*	Phocus *t* (s)	Naive *t* (s)
Burgers Lax-Fried.	D	0.05	2.0,1.0,25	100.0	1.2	1.37	2.15	2.77	Safe	Safe	1.95	16.46
Burgers Lax-Fried.	P	0.05	2.0,1.0,25	100.0	1.29	1.54	2.33	3.31	Safe	Safe	2.45	22.27
Burgers Leapfrog	D	0.05	2.0,1.0,25	100.0	1.26	1.74	11.33	301.76	Safe	21	3.06	11.95
Burgers Leapfrog	P	0.05	2.0,1.0,25	100.0	1.27	1.78	11.42	302.59	Safe	21	4.37	21.09
Burgers Upwind	D	0.05	2.0,0.475,25	66.35	1.33	1.58	4.74	46.45	Safe	24	20.51	57.82
Burgers Upwind	P	0.05	2.0,0.475,25	66.73	1.38	1.63	4.68	46.19	Safe	24	25.02	95.33
Burgers80 Lax-Fried.	D	0.05	2.0,1.0,80	100.0	1.22	2.69	1.44	2483.63	Safe	75	36.83	621.7
Burgers280 Lax-Fried.	D	0.05	2.0,1.0,280	X	1.11	X	0.83	X	Safe	75	1747.97	X
Flood Height Lax-Fried.	P	0.075	2.0,0.5,40	X	1.29	X	3.17	X	Safe	X	8.01	X
Flood Height Leapfrog	P	0.075	2.0,0.5,40	X	1.86	X	25.61	X	Safe	X	14.23	X
Buckley Lev. Lax-Fried.	P	0.05	2.0,0.35,30	X	1.3	X	2.33	X	Safe	X	4.03	X
Buckley Lev. Leapfrog	P	0.05	2.0,0.35,30	X	3.11	X	47.3	X	Safe	X	7.84	X
LWR Lax-Fried.	P	0.07	2.0,0.275,25	100.0	1.28	1.45	3.02	3.87	Safe	Safe	2.39	22.09
LWR Leapfrog	P	0.07	2.0,0.275,25	100.0	1.21	1.41	7.22	10.46	Safe	Safe	5.6	22.58
Cubic Lax-Fried	P	0.1	2.0,0.625,25	100.0	1.29	2.53	4.22	5519.47	Safe	19	2.44	96.28
Cubic Leapfrog	P	0.1	2.0,0.25,20	100.0	1.15	1.42	10.04	135.72	Safe	15	2.44	42.5
Cubic Upwind	P	0.1	2.0,0.185,15	75.0	1.28	1.25	8.82	9.23	Safe	Safe	15.23	38.25
Trilinear Lax-Fried.	P	0.1	2.0,0.5,35	99.31	1.35	2.05	4.05	9.32	Safe	Safe	5.83	22.23
Trilinear Leapfrog	P	0.1	2.0,0.5,35	61.53	2.93	3.15	89.21	231.32	Safe	Safe	11.42	37.15

using Table 1. One takes the Phocus Ratio entry and divides by the respective Naive Ratio entry. Quotients less than 1 quantify how much tighter Phocus's bounds are on average.

6.2.3 RQ 3: Evolution of the Bounds over Time. To study how the bounds evolve over time for Phocus, the naive zonotopes, and the sampled baseline, we take a snapshot of the abstract state every $\frac{n}{4}$ timesteps, where n is the total timesteps (specified in Table 1) a given benchmark was run for. We show these snapshots for select benchmarks while the remaining benchmarks' plots are provided in the Appendix. By visualizing different snapshots in time, we can qualitatively and quantitatively observe changes in bound precision. The plots showing 4 snapshots in time for Burgers PDE for the Lax-Friedrichs, Leapfrog, and Upwinding schemes are shown in Fig. 3. The plots showing 4 snapshots for the trilinear flux PDE for Lax-Friedrichs and Leapfrog are shown in Fig. 4. Plots showing 4 snapshots for the Cubic flux PDE with the Leapfrog scheme are shown in Fig. 5. For all plots, the x-axis represents the spatial dimension (x) and the y-axis represents the value of the solution $u(x, t_n)$ for that particular t_n (e.g.,0,8,16,25). In all plots, the bounds of Phocus are in blue, the bounds of the naive zonotope baseline are in red, and the (unsound) sampled bounds are in green. Since Phocus and the naive zonotope baseline are sound, they always enclose the green sampled bounds. The desired behavior is to *as tightly as possible* enclose the green bounds. Upper bounds and lower bounds are respectively denoted by upward and downward facing semicircles.

Burgers PDE. While we numerically solve Burgers PDE for multiple configurations (all shown in Table 1), here we select 3 for visualization. These three configurations correspond to the Table 1 rows where we run Burgers PDE for 25 timesteps with $\Delta x = 2$, $\pm\epsilon = 0.05$ and periodic boundary conditions. The Lax-Friedrichs and Leapfrog schemes use $\Delta t = 1$ while Upwinding uses $\Delta t = 0.475$. *Lax-Friedrichs.* The results of the Burgers PDE with the Lax-Friedrichs scheme are shown in Fig 3a. For the first eight timesteps, Phocus's bounds, naive zonotope bounds, and sampled bounds remain nearly identical. However with more timesteps, a greater separation between Phocus and the naive zonotope emerges. By 25 timesteps, clear separation exists between the blue bounds of Phocus and the red bounds of the baseline and our blue bounds enclose the green sampled bounds more tightly.

One trend is that the wave's peak moves rightward, since the initial peak is centered at $x \approx 7$, whereas after 25 timesteps the peak is at $x \approx 16$, which is expected given Burger's flux. Another trend is that the wave *dissipates* from an initial height of slightly above 0.8 down to a peak height below 0.4 after 25 timesteps. This trend occurs for all 3 colors since it is an artifact of Lax-Friedrichs being a *dissipative*, 1st-order accurate TVD scheme. Both Phocus and the naive baseline provably obey the CFL condition and the total variation is certified to be small for all methods.

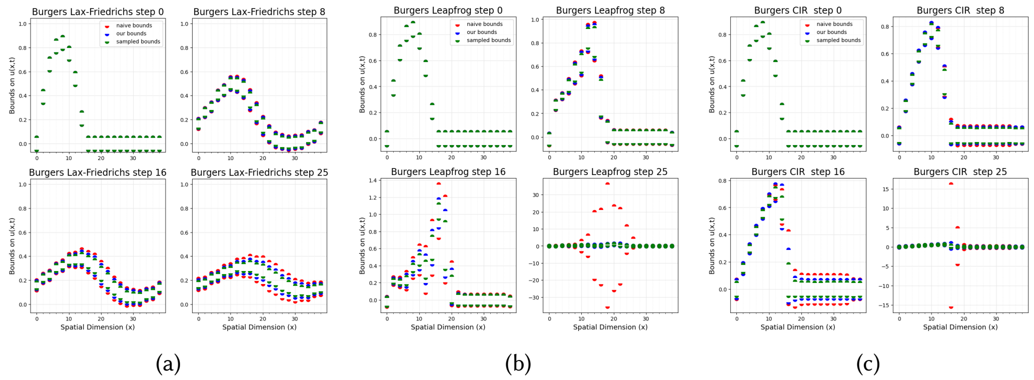


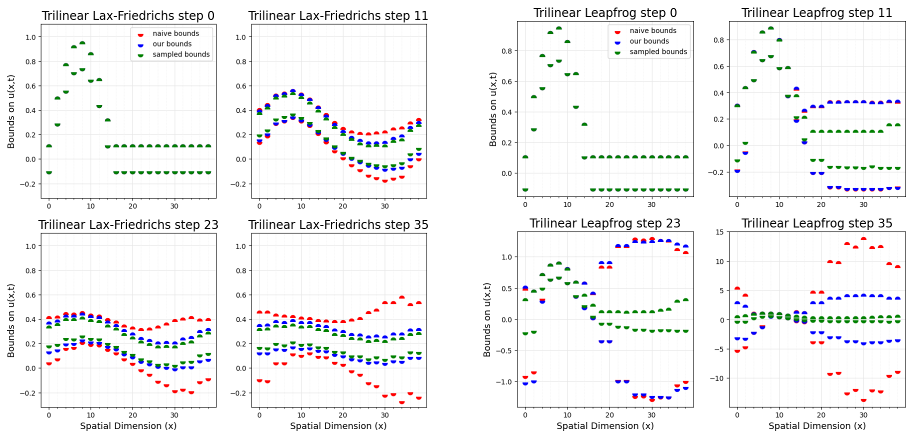
Fig. 3. Burgers PDE bounds for (a) Lax-Friedrichs (b) Leapfrog and (c) Upwinding FD schemes.

Leapfrog. The leapfrog scheme bounds are shown in Fig. 3b. After the first 8 timesteps, the bounds are all nearly identical, however the naive zonotope bounds explode by timestep 25. In contrast, the bounds of Phocus still tightly enclose the green sampled bounds. Unlike the naive baseline, since Phocus's bounds remain tight, we can verify that no shocks formed (as described in Section 4.7.1).

The explosion of the naive method's bounds is confirmed by its violated CFL condition at timestep 21. In contrast, Phocus's bounds verify the CFL property over all mesh points and timesteps.
Upwinding. The upwinding scheme bounds are shown in Fig. 3c. Because the upwinding scheme is non-conservative, we use a smaller $\Delta t = .475$ for all methods. Through the first 8 timesteps all bounds are similar, but Phocus's bounds (blue) become noticeably more precise at timestep 16 and significantly more precise by timestep 25. Since the wave's peak moves with faster velocity than the wave's lower part, a sharp front appears. This front causes the naive (red) bounds to loosen early, whereas Phocus maintains precision. This fact also explains why the naive baseline violates the CFL condition at timestep 24. Lastly, Phocus computes bounds faster than the naive baseline.
Trends. While the 3 schemes solve the same PDE, each scheme has a different level of accuracy hence why the 3 subplots of Fig. 3 are different, even for the concrete sampled points. However in all cases, the waves move rightward due to Burger's flux. The Lax-Friedrichs scheme is first-order accurate and introduces artificial dissipation, hence why the wave flattens out and the bounds actually compress. In contrast, Leapfrog is 2nd-order accurate, but introduces spurious oscillations. For instance, timestep 16 of Fig 3b contains some degree of oscillation hence why the total variation for both Phocus and the zonotope baseline are greater when using the leapfrog FD scheme instead of Lax-Friedrichs. Since the Leapfrog FD is *not* dissipative, the peak of the waves *can grow* over time (even for concrete executions), a finding echoed in [80].

Trilinear Flux PDE. The evolution of the bounds for both the Lax-Friedrichs and Leapfrog schemes applied to the trilinear flux PDE are shown in Fig. 4. The upwinding scheme is inapplicable to this benchmark hence it is not included.
Lax-Friedrichs. For the Lax-Friedrichs scheme in Fig. 4a, modest separation between Phocus (blue) and the naive zonotopes (red) emerges by timestep 11. As time progresses, the imprecision of the naive baseline accumulates, whereas Phocus bounds remain precise, even after 35 timesteps. For 99.3% of mesh points, Phocus's bounds are strictly within the naive zonotope baseline bounds.
Leapfrog. The Leapfrog bounds are shown in Fig. 4b. The Phocus and naive zonotope bounds are nearly identical through timestep 23, with the naive bounds occasionally being more precise. Hence the % Fully enclosed is only 61.53%. However by timestep 35, Phocus's bounds are *all* more precise.

Cubic Flux PDE. The Cubic flux PDE bounds with the Leapfrog scheme are shown in Fig. 5.
Leapfrog. While modest separation between Phocus and the naive zonotopes emerges at timestep



(a) (b)

Fig. 4. Trilinear Flux PDE bounds for (a) Lax-Friedrichs FD scheme and (b) Leapfrog FD scheme.

13, significant separation emerges by timestep 20. Phocus's final bounds are significantly more precise which is confirmed by the Naive zonotope baseline violating the CFL condition at step 15. Phocus bounds are also tight enough to verify no shocks formed. Since the cubic flux contains more nonlinearity than the quadratic Burgers flux, more imprecision from the naive baseline is expected, which is indeed true (and also true for the Section 2 example).

6.2.4 RQ 4: Runtime and Scalability. The runtimes for both Phocus and the naive zonotope abstract transformers can be seen in the final two columns of Table 1. In many cases, Phocus's more precise, synthesized abstract transformers are actually *faster* than existing zonotope abstract transformers. While it may seem counterintuitive that Phocus's more complex procedure can be faster, the savings have to do with the memory complexity described in Section 4.6. For instance in the Burgers Lax-Friedrichs benchmark with $n = 80$ timesteps, Phocus successfully certifies bounds in 36.83 seconds while the naive zonotope method takes over 621 seconds.

Scaling to Large Programs As a test of the scalability of Phocus, we run one benchmark: Burgers Lax-Friedrichs for 280 timesteps to see if tight bounds that satisfy the CFL condition are obtainable. This benchmark's snapshots are shown in Fig. 6. Our PDE domain has 20 mesh points per timestep, but the first and last points are fixed (Dirichlet boundaries) so each timestep computes the inner 18 points. Since we run for 280 timesteps, we compute bounds for $280 \cdot 18 = 5040$ total mesh points.

Each mesh point invokes the nonlinear Lax-Friedrichs stencil, hence our method introduces over 5000 noise symbols into the symbolic affine expression. The naive baseline is not shown because even though it did not time out, its bounds overflowed, raising an exception. The naive method's failure started when it violated the CFL condition at timestep 75, whereas Phocus obeys the CFL condition for all 280 timesteps and finished within the time limit. Similar to Section 6.2.3, Phocus's bounds and the empirical sampled bounds compress over time (but not for the naive zonotopes) due to the dissipation of the Lax-Friedrichs stencil.

7 Related Work

To the best of our knowledge, no prior work has abstractly interpreted hyperbolic PDE solvers.

Formal Analysis of PDEs. Despite the ubiquity of PDEs, automated formal reasoning about their properties has lagged. While naive interval analyses have been proposed [59, 74] these approaches are imprecise and limited in scope. Further, while interval arithmetics for finite element

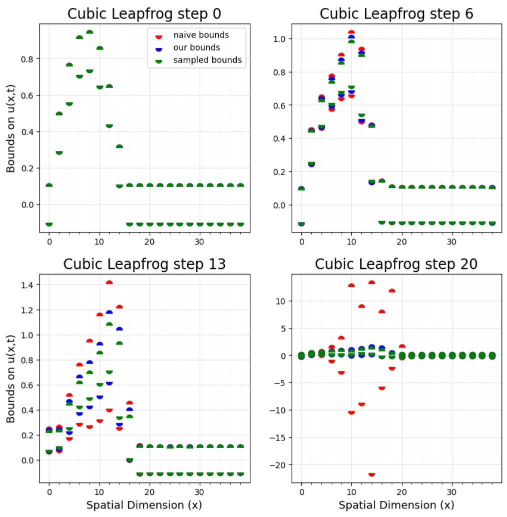


Fig. 5. Cubic Flux PDE run for 20 steps with the Leapfrog FD scheme.

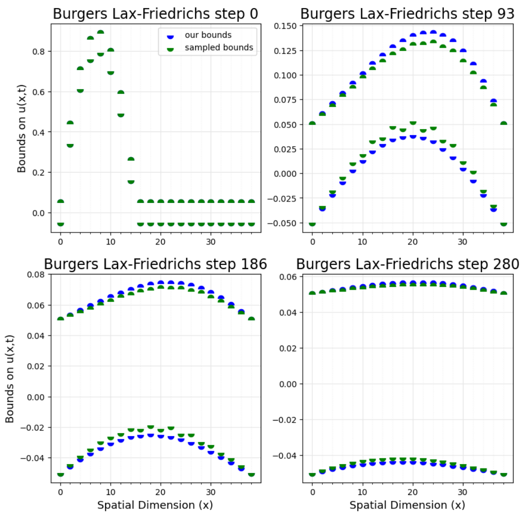


Fig. 6. Burgers PDE run for 280 timesteps with the Lax-Friedrichs FD scheme.

PDE solvers exist [64, 68], our work focuses on finite *difference* schemes for hyperbolic PDEs which are less studied. A few works move beyond naive interval analysis of PDE solutions, however these works still suffer significant limitations that Phocus avoids. For instance, while [24] uses zonotopes, they restrict to parabolic heat PDEs and linear stencils, whereas our PDEs are hyperbolic and nonlinear. Similarly, while [2, 5, 86] study reachability of PDEs, they also restrict to parabolic heat equations and linear stencils. Since these techniques neither target hyperbolic PDEs nor nonlinear stencils, they all remain inapplicable to our setting.

While a few formal analyses for hyperbolic PDEs exist, they either only perform (unsound) statistical analysis [22], are purely dynamic (with large overheads) [73], only analyze numerical error [4] instead of formally verified reachability or only support interval arithmetic [59]. Further, [59] does not formalize or verify other properties like the total variation studied in our work.

Correctness in Scientific Computing. Beyond analyzing PDEs there exists a broader push for correctness in scientific computing [17, 26, 27, 70]. Prior work focused on floating point issues [14, 30], roundoff error analysis [44], correctness of SciML [19, 20], ODEs [41], or iterative solvers [85] which are all orthogonal to our work.

Precise Abstract Interpretation. We draw upon ideas from abstract interpretation of multilinear polynomials [47] and from works that synthesize abstract transformers [25, 37, 43, 49, 66, 72, 78]. Of these, Pasado [49] is closest in spirit to Phocus. However Pasado analyzes AD code instead of finite differences and uses different proof techniques. In addition, while [47] restricts multilinearity to the interval domain, we generalize multilinearity to linear abstract domains and piecewise functions.

8 Conclusion

We presented Phocus, the first abstract interpretation of finite difference solvers for nonlinear hyperbolic PDEs. Phocus also verifies key program invariants of PDE solvers like the CFL condition or a solution's total variation. Using a novel optimization-based abstract transformer synthesis procedure, Phocus achieves significant precision while simultaneously scaling to thousands of mesh points. By formalizing rigorous PDE analysis as a tractable verification problem, our work opens the door to verifying a broad new class of scientific computing programs.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback. Argonne National Laboratory's contribution was supported by the U.S. Department of Energy Assistant Secretary for Environmental Management, Office of Science, under contract DE-AC02-06CH11357. Part of this work was also performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory (LLNL) under contract DE-AC52-07NA27344 (LLNL-CONF-2003978). Additionally, the portion of the work done by LLNL was funded by the DOE ASCR X-Stack Comport project (SCW1743).

Data Availability Statement

Our artifact including the code and documentation is publicly available on Zenodo [48]. In addition, the code is also available at https://github.com/jsl1994/Phocus.

References

[1] Aws Albarghouthi. 2021. Introduction to neural network verification. *Foundations and Trends® in Programming Languages* 7, 1–2 (2021), 1–157.

[2] Stanley Bak and Parasara Sridhar Duggirala. 2017. Simulation-equivalent reachability of large linear systems with inputs. In *International Conference on Computer Aided Verification*. Springer, 401–420.

[3] Dinshaw S Balsara. 2013. Numerical PDE Techniques for Scientists and Engineers. (2013).

[4] JW Banks, JAF Hittinger, JM Connors, and CS Woodward. 2012. Numerical error estimation for nonlinear hyperbolic PDEs via nonlinear error transport. *Computer methods in applied mechanics and engineering* 213 (2012), 1–15.

[5] Tianshu Bao. 2023. *Modelling Physics-based Dynamic System using Machine Learning*. Ph. D. Dissertation.

[6] Michael Breuß. 2004. The correct use of the Lax–Friedrichs method. *ESAIM: Mathematical Modelling and Numerical Analysis* 38, 3 (2004), 519–540.

[7] William L Briggs, Alan C Newell, and Talib Sarie. 1983. Focusing: A mechanism for instability of nonlinear finite difference equations. *J. Comput. Phys.* 51 (1983). doi:10.1016/0021-9991(83)90082-7

[8] S.E. Buckley and M.C. Leverett. 1942. Mechanism of fluid displacement in sands. *Transactions of the AIME* 146, 01 (1942), 107–116.

[9] Johannes Martinus Burgers. 1948. A mathematical model illustrating the theory of turbulence. *Advances in applied mechanics* 1 (1948), 171–199.

[10] Bernardo Cockburn, Claes Johnson, Chi-Wang Shu, and Eitan Tadmor. 2006. *Advanced numerical approximation of nonlinear hyperbolic equations: lectures given at the 2nd session of the Centro Internazionale Matematico Estivo (CIME) held in Cetraro, Italy, June 23-28, 1997*. Springer.

[11] Richard Courant, Kurt Friedrichs, and Hans Lewy. 1967. On the partial difference equations of mathematical physics. *IBM journal of Research and Development* 11, 2 (1967), 215–234.

[12] Richard Courant, Eugene Isaacson, and Mina Rees. 1952. On the solution of nonlinear hyperbolic differential equations by finite differences. *Communications on pure and applied mathematics* 5, 3 (1952), 243–255.

[13] Patrick Cousot and Radhia Cousot. 1977. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. 238–252.

[14] Arnab Das, Ian Briggs, Ganesh Gopalakrishnan, Sriram Krishnamoorthy, and Pavel Panchekha. 2020. Scalable yet rigorous floating-point error analysis. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–14. doi:10.1109/SC41405.2020.00055

[15] Luiz Henrique De Figueiredo and Jorge Stolfi. 2004. Affine arithmetic: concepts and applications. *Numerical algorithms* 37 (2004), 147–158.

[16] Mehdi Dehghan and Rooholah Jazlanian. 2011. On the total variation of a third-order semi-discrete central scheme for 1D conservation laws. *Journal of Vibration and Control* 17, 9 (2011), 1348–1358.

[17] Venkata Dhavala, Jan Hückelheim, Paul D Hovland, and Stephen F Siegel. 2025. Verifying PETSc Vector Components Using CIVL. In *International Conference on Computer Aided Verification*. 148–161.

[18] Harvey Diamond. 2011. Leapfrog method for hyperbolic pdes. https://math.wvu.edu/~hdiamond/Math522S11/leapfrog.pdf

[19] Francisco Eiras, Adel Bibi, Rudy R Bunel, Krishnamurthy Dj Dvijotham, Philip Torr, and M Pawan Kumar. 2024. Efficient Error Certification for Physics-Informed Neural Networks. In *Forty-first International Conference on Machine Learning*.

[20] Petros Ellinas, Rahul Nellikath, Ignasi Ventura, Jochen Stiasny, and Spyros Chatzivasileiadis. 2024. Correctness Verification of Neural Networks Approximating Differential Equations. *arXiv preprint arXiv:2402.07621* (2024).

[21] Charbel Farhat. [n. d.]. The Finite Difference Method. http://web.stanford.edu/group/frg/course_work/AA214/CA-AA214-Ch6.pdf

[22] Camilla Fiorini. 2023. Uncertainty propagation of the shock position for hyperbolic PDEs using a sensitivity equation method. In *International Conference on Finite Volumes for Complex Applications*. Springer, 131–139.

[23] Khalil Ghorbal, Eric Goubault, and Sylvie Putot. 2009. The zonotope abstract domain taylor1+. In *Computer Aided Verification: 21st International Conference, CAV 2009, Grenoble, France, June 26-July 2, 2009. Proceedings 21*. 627–633.

[24] Antoine Girard and Colas Le Guernic. 2008. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 215–228.

[25] Shaurya Gomber, Debangshu Banerjee, and Gagandeep Singh. 2025. Universal Synthesis of Differentiably Tunable Numerical Abstract Transformers. *arXiv preprint arXiv:2507.11827* (2025).

[26] Ganesh Gopalakrishnan, Paul D Hovland, Costin Iancu, Sriram Krishnamoorthy, Ignacio Laguna, Richard A Lethin, Koushik Sen, Stephen F Siegel, and Armando Solar-Lezama. 2017. *Report of the HPC Correctness Summit, January 25-26, 2017, Washington, DC*. Technical Report. USDOE Office of Science (SC), Washington, DC (United States).

[27] Ganesh L Gopalakrishnan and Robert M Kirby. 2010. Top ten ways to make formal methods for HPC practical. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*. 137–142.

[28] Sigal Gottlieb, Jae-Hun Jung, and Saeja Kim. 2011. A review of David Gottlieb's work on the resolution of the Gibbs phenomenon. *Communications in Computational Physics* 9, 3 (2011), 497–519.

[29] Eric Goubault, Tristan Le Gall, and Sylvie Putot. 2012. An accurate join for zonotopes, preserving affine input/output relations. *Electronic Notes in Theoretical Computer Science* 287 (2012), 65–76.

[30] Hui Guo and Cindy Rubio-González. 2020. Efficient generation of error-inducing floating-point inputs via symbolic execution. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 1261–1272.

[31] Cong Hao. 2025. Exploring and Exploiting Runtime Reconfigurable Floating Point Precision in Scientific Computing: a Case Study for Solving PDEs *(ASPDAC '25)*. doi:10.1145/3658617.3697699

[32] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. doi:10.1038/s41586-020-2649-2

[33] Brian T Hayes and Michael Shearer. 2001. A nonconvex scalar conservation law with trilinear flux. *Quart. Appl. Math.* (2001), 615–635.

[34] Michael T Heath. 2018. *Scientific computing: an introductory survey, revised second edition*. SIAM.

[35] Thibault Helaire et al. 2021. affapy library. (2021).

[36] Ralf Hiptmair. 2002. Finite elements in computational electromagnetism. *Acta Numerica* 11 (2002), 237–339.

[37] Zixin Huang, Jacob Laurel, Saikat Dutta, and Sasa Misailovic. 2025. AURA: Precise Abstract Interpretation of Probabilistic Programs with Interval Data Uncertainty. In *International Static Analysis Symposium*. Springer.

[38] Jan Hückelheim, Navjot Kukreja, Sri Hari Krishna Narayanan, Fabio Luporini, Gerard Gorman, and Paul Hovland. 2019. Automatic differentiation for adjoint stencil loops. In *Proceedings of the 48th International Conference on Parallel Processing*. 1–10. doi:10.1145/3337821.3337906

[39] Jan Hückelheim, Ziqing Luo, Sri Hari Krishna Narayanan, Stephen Siegel, and Paul D Hovland. 2018. Verifying properties of differentiable programs. In *International Static Analysis Symposium*. Springer, 205–222.

[40] Geoffrey Hulette, John Bender, Samuel Pollard, Heidi Thornquist, and Ariel Kellison. 2021. *Formal Methods-based Certification Frameworks for Scientific Computing Applications*. Technical Report. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

[41] Fabian Immler. 2018. A verified ODE solver and the Lorenz attractor. *Journal of automated reasoning* 61, 1 (2018), 73–111.

[42] Doug Jacobs, Bill McKinney, and Michael Shearer. 1995. Traveling wave solutions of the modified Korteweg-deVries-Burgers equation. *Journal of Differential Equations* 116, 2 (1995), 448–467.

[43] Pankaj Kumar Kalita, Sujit Kumar Muduli, Loris D'Antoni, Thomas Reps, and Subhajit Roy. 2022. Synthesizing abstract transformers. *Proceedings of the ACM on Programming Languages* 6, OOPSLA2 (2022), 1291–1319. doi:10.1145/3563334

[44] Ariel E Kellison and Justin Hsu. 2024. Numerical Fuzz: A Type System for Rounding Error Analysis. *Proceedings of the ACM on Programming Languages* 8, PLDI (2024), 1954–1978.

[45] Andreas Kloeckner. 2022. Numerical Methods for Partial Differential Equations. https://andreask.cs.illinois.edu/cs555-s22/notes.pdf

[46] Niklas Kochdumper, Christian Schilling, Matthias Althoff, and Stanley Bak. 2023. Open-and closed-loop neural network verification using polynomial zonotopes. In *NASA Formal Methods Symposium*. Springer, 16–36.

[47] Cosimo Laneve, Tudor A Lascu, and Vania Sordoni. 2010. The interval analysis of multilinear expressions. *Electronic Notes in Theoretical Computer Science* 267, 2 (2010), 43–53.
[48] Jacob Laurel, Ignacio Laguna, and Jan Hückelheim. 2025. Reproduction Artifact for 'Synthesizing Sound and Precise Abstract Transformers for Nonlinear Hyperbolic PDE Solvers'. Zenodo. doi:10.5281/zenodo.16921568
[49] Jacob Laurel, Siyuan Brant Qian, Gagandeep Singh, and Sasa Misailovic. 2023. Synthesizing precise static analyzers for automatic differentiation. *Proceedings of the ACM on Programming Languages* 7, OOPSLA2 (2023), 1964–1992. doi:10.1145/3622867
[50] Jacob Laurel, Rem Yang, Gagandeep Singh, and Sasa Misailovic. 2022. A dual number abstraction for static analysis of Clarke Jacobians. *Proceedings of the ACM on Programming Languages* 6, POPL (2022), 1–30. doi:10.1145/3498718
[51] Jacob Laurel, Rem Yang, Shubham Ugare, Robert Nagel, Gagandeep Singh, and Sasa Misailovic. 2022. A general construction for abstract interpretation of higher-order automatic differentiation. *Proceedings of the ACM on Programming Languages* 6, OOPSLA2 (2022), 1007–1035. doi:10.1145/3563324
[52] Jacob Scott Laurel. 2024. *Static analysis of differentiable programs.* Ph. D. Dissertation. University of Illinois at Urbana-Champaign.
[53] Peter D Lax. 1973. *Hyperbolic systems of conservation laws and the mathematical theory of shock waves.* SIAM.
[54] Randall J LeVeque. 1992. *Numerical methods for conservation laws.* Vol. 214. Springer.
[55] Randall J LeVeque. 2002. *Finite volume methods for hyperbolic problems.* Vol. 31. Cambridge university press.
[56] Zhilin Li, Zhonghua Qiao, and Tao Tang. 2017. *Numerical solution of differential equations: introduction to finite difference and finite element methods.* Cambridge University Press.
[57] Michael James Lighthill and G Be Whitham. 1955. On kinematic waves I. Flood movement in long rivers. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 229, 1178 (1955), 281–316.
[58] Michael James Lighthill and Gerald Beresford Whitham. 1955. On kinematic waves II. A theory of traffic flow on long crowded roads. *Proceedings of the royal society of london. series a. mathematical and physical sciences* 229, 1178 (1955), 317–345.
[59] Qun Lin and Lung-an Ying. 2009. Interval Inclusion Computation for the Solutions of the Burgers Equation. *SIAM J. Numer. Anal.* 47, 4 (2009), 2496–2517. doi:10.1137/080722011
[60] J David Logan. 2014. *Applied partial differential equations.* Springer.
[61] Antoine Miné. 2004. Relational abstract domains for the detection of floating-point run-time errors. In *European Symposium on Programming*. Springer, 3–17.
[62] Antoine Miné et al. 2017. Tutorial on static inference of numeric invariants by abstract interpretation. *Foundations and Trends® in Programming Languages* 4, 3-4 (2017), 120–372. doi:10.1561/2500000034
[63] Ashitabh Misra, Jacob Laurel, and Sasa Misailovic. 2023. Vix: analysis-driven compiler for efficient low-precision variational inference. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1–6. doi:10.23919/DATE56975.2023.10137324
[64] Rafi L Muhanna and Shahrokh Shahi. 2020. Uncertainty in boundary conditions—an interval finite element approach. *Decision Making under Constraints* (2020), 157–167.
[65] Suhas Patankar. 2018. *Numerical heat transfer and fluid flow.* CRC press.
[66] Brandon Paulsen and Chao Wang. 2022. Example guided synthesis of linear approximations for neural network verification. In *International Conference on Computer Aided Verification*. Springer, 149–170.
[67] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.
[68] Sebastião C Pereira, Ulisses T Mello, Nelson FF Ebecken, and Rafi L Muhanna. 2006. Uncertainty in thermal basin modeling: An interval finite element approach. *Reliable computing* 12, 6 (2006), 451–470.
[69] Richard H Pletcher, John C Tannehill, and Dale Anderson. 2012. *Computational fluid mechanics and heat transfer.* CRC press. doi:10.1201/b12884
[70] Samuel Douglas Pollard, Jon M Aytac, Ariel Eileen Kellison, Ignacio Laguna, Srinivas Nedunuri, Sabrina Arianne Reis, Matthew J Sottile, and Heidi Krista Thornquist. 2024. The First Tri-Lab Workshop on Formal Verification: Capabilities, Challenges, Research Opportunities, and Exemplars. (2024).
[71] Luciano Rezzolla. 2011. Numerical methods for the solution of partial differential equations. *Lecture Notes for the COMPSTAR School on Computational Astrophysics* (2011), 8–13.
[72] Wonryong Ryou, Jiayu Chen, Mislav Balunovic, Gagandeep Singh, Andrei Dan, and Martin Vechev. 2021. Scalable polyhedral verification of recurrent neural networks. In *Computer Aided Verification: 33rd International Conference, CAV 2021, Virtual Event, July 20–23, 2021, Proceedings, Part I 33*. Springer, 225–248.
[73] Philipp Samfass, Tobias Weinzierl, Anne Reinarz, and Michael Bader. 2021. Doubt and redundancy kill soft errors—Towards detection and correction of silent data corruption in task-based numerical software. In *2021 IEEE/ACM 11th Workshop on Fault Tolerance for HPC at eXtreme Scale (FTXS)*. IEEE, 1–10.

[74] Hartmut Schwandt. 1987. An interval arithmetic method for the solution of nonlinear systems of equations on a vector computer. *Parallel Comput.* 4, 3 (1987), 323–337.

[75] Tapan K Sengupta, G Ganerwal, and Anurag Dipankar. 2004. High accuracy compact schemes and Gibbs' phenomenon. *Journal of Scientific Computing* 21, 3 (2004), 253–268.

[76] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. 2018. Fast and effective robustness certification. *Advances in neural information processing systems* 31 (2018).

[77] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 2019. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 1–30.

[78] Gagandeep Singh, Jacob Laurel, Sasa Misailovic, Debangshu Banerjee, Avaljot Singh, Changming Xu, Shubham Ugare, and Huan Zhang. 2025. Safety and Trust in Artificial Intelligence with Abstract Interpretation. *Foundations and Trends® in Programming Languages* (2025), 250–408. doi:10.1561/2500000062

[79] Gagandeep Singh, Markus Püschel, and Martin Vechev. 2017. Fast polyhedra abstract domain. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*. 46–59. doi:10.1145/3093333.3009885

[80] DM Sloan and AR Mitchell. 1986. On nonlinear instabilities in leap-frog finite difference schemes. *J. Comput. Phys.* 67, 2 (1986), 372–395.

[81] Alexey Smirnov. 2024. Discrete total variation in multiple spatial dimensions and its applications. (2024).

[82] Alexey Solovyev, Marek S Baranowski, Ian Briggs, Charles Jacobsen, Zvonimir Rakamarić, and Ganesh Gopalakrishnan. 2018. Rigorous estimation of floating-point round-off errors with symbolic taylor expansions. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 41, 1 (2018), 1–39. doi:10.1145/3230733

[83] Jorge Stolfi and Luiz Henrique De Figueiredo. 1997. Self-validated numerical methods and applications. In *Monograph for 21st Brazilian Mathematics Colloquium, IMPA, Rio de Janeiro. Citeseer*, Vol. 5. Citeseer, 122.

[84] Terence Tao. 2016. Finite time blowup for Lagrangian modifications of the three-dimensional Euler equation. *Annals of PDE* 2 (2016), 1–79.

[85] Mohit Tekriwal, Andrew W Appel, Ariel E Kellison, David Bindel, and Jean-Baptiste Jeannin. 2023. Verified correctness, accuracy, and convergence of a stationary iterative linear solver: Jacobi method. In *International Conference on Intelligent Computer Mathematics*. Springer, 206–221.

[86] Hoang-Dung Tran, Weiming Xiang, Stanley Bak, and Taylor T Johnson. 2018. Reachability analysis for one dimensional linear parabolic equations. *IFAC-PapersOnLine* 51, 16 (2018), 133–138.

[87] John Arthur Trangenstein. 2009. *Numerical Solution of Hyperbolic Partial Differential Equations*.

[88] Sulin Wang and Zhengfu Xu. 2019. Total variation bounded flux limiters for high order finite difference schemes solving one-dimensional scalar conservation laws. *Math. Comp.* 88, 316 (2019), 691–716.

[89] Liu Yang and Stanley J. Osher. 2024. PDE generalization of in-context operator networks: A study on 1D scalar nonlinear conservation laws. *J. Comput. Phys.* 519 (2024). https://www.sciencedirect.com/science/article/pii/S0021999124006272

[90] Rem Yang, Jacob Laurel, Sasa Misailovic, and Gagandeep Singh. 2023. Provable Defense Against Geometric Transformations. In *11th International Conference on Learning Representations*. doi:10.48550/arXiv.2207.11177

Received 2025-03-25; accepted 2025-08-12