Differentiable Physics for Improving the Accuracy of Iterative PDE-Solvers with Neural Networks

Kiwon Um, Robert Brand, Yun (Raymond) Fei, Philipp Holl, Nils Thuerey

Phiflow solver

Correction code

NEURAL INFORMATION PROCESSING SYSTEMS

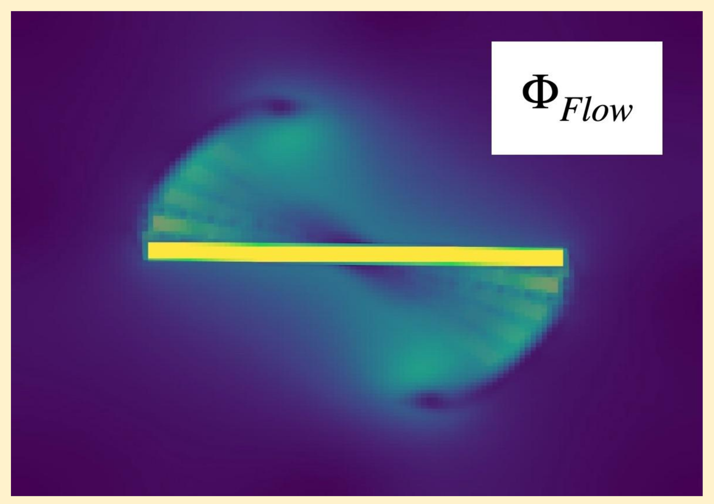
ML4PS Workshop

Differentiable Physics with ΦFlow (*phiflow*)

Differentiable physics are required whenever an optimizer like a neural network interacts with a simulation. A differentiable simulation can compute the temporal gradient with respect to any parameter it is affected by.

ΦFlow is a framework for using and implementing differentiable simulations. Any simulation written in ΦFlow can automatically leverage the automatic differentiation utilities from PyTorch and TensorFlow.

It comes with a number of built-in simulations like fluids. A basic simulation can be set up in few lines of code, and the provided demos showcase what can be done out of the box. The right image shows an example of a rotating obstacle in a fluid flow.

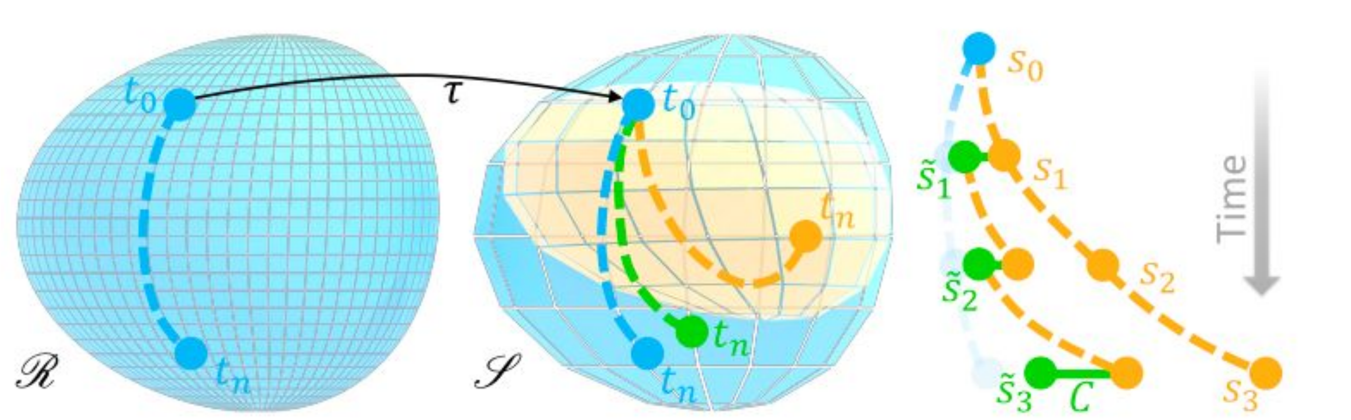


Applications of Differentiable Physics Solvers

Learning to Reduce Numerical Errors:

- Consider two different discrete versions of the same PDE P:
- P_R denoting a more accurate discretization with solutions $r \in R$ from the reference manifold, and an approximate version P_S with solutions $s \in S$ from the source manifold.
- r and s represent phase space points and evolutions over time given by a trajectory in each solution manifold.
- A mapping operator T that transforms from one to the other manifold
- E.g., for the initial conditions of the sequences above, we typically choose s = T r. Due to the inherently different numerical approximations, P_S (T r_t) = T r_t+1 for the vast majority of states.
- We use an L2-norm to quantify the deviations: L(s_t, T r_t) = |s_t − T r_t|_2.

Central learning objective: learn a correction operator C(s) such that a solution to which the correction is applied has a lower error than an unmodified solution: L(P_s(C(T r_t0)), T r_t1) < L(P_s(T r_t0), T r_t1). The correction function C(s|θ) is represented as a deep neural network.



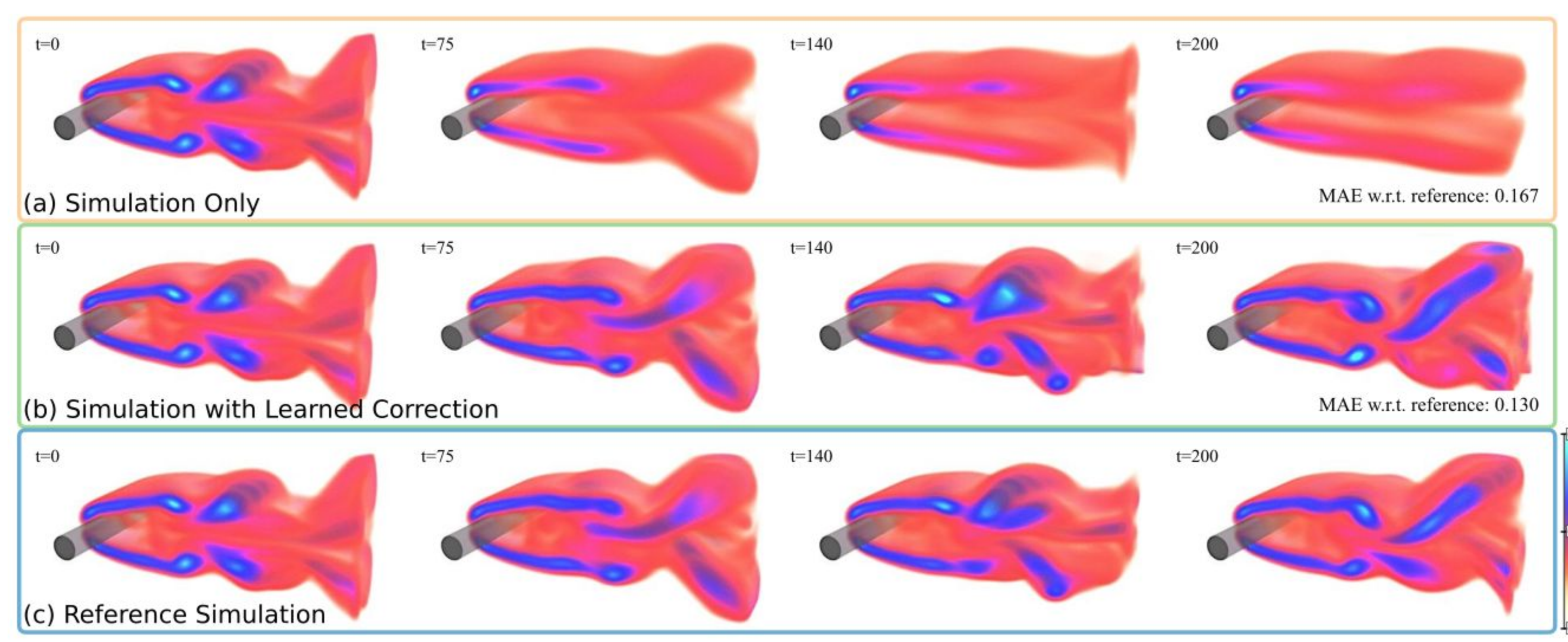
Transformed solutions of the reference sequence computed on R (blue) differ from solutions computed on the source manifold S (orange). A correction function C (green) updates the state after each iteration to more closely match the projected reference trajectory on S.

Experiments

Comparing three methodologies for DNN-PDE interactions:

- **Non-interacting (NON)**: Purely uses the unaltered PDE trajectories.
- **Pre-computed interaction (PRE)**: Can make use of phase space states that are altered by the pre-computed or analytic correction.
- **Solver-in-the-loop (SOL)**: By integrating the model into training with differentiable physics solver, the corrections can interact with the physical system, alter the states, and receive gradients about the future performance.

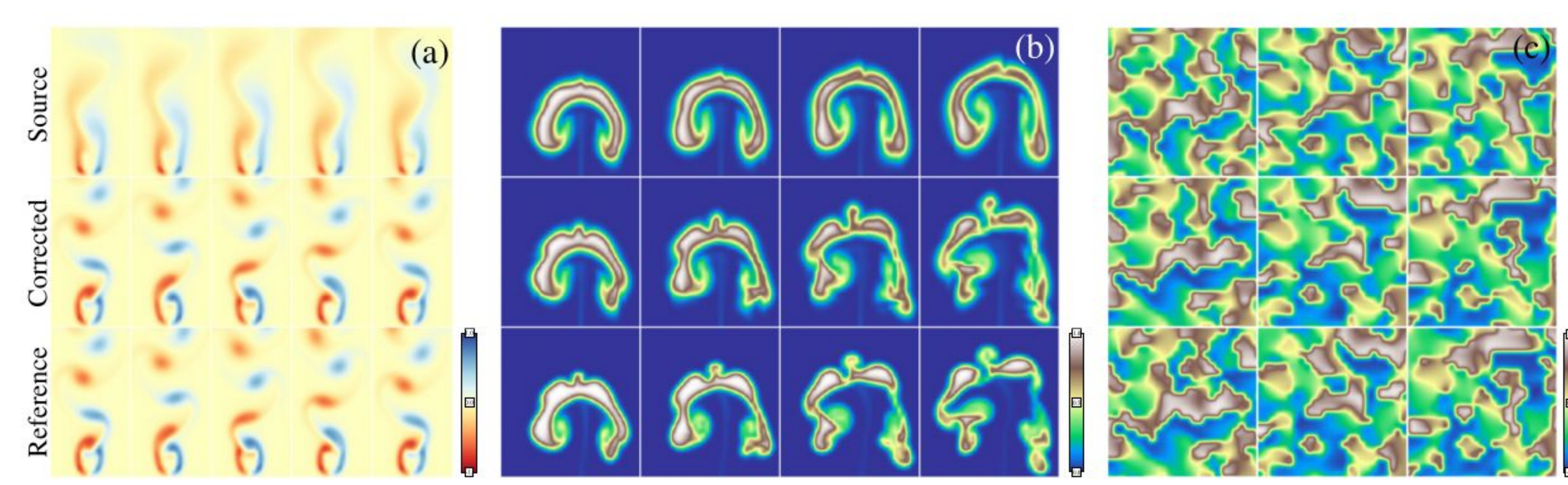
Highlight: Our SOL model for a 3D Fluid Problem



Shown in terms of vorticity

Results

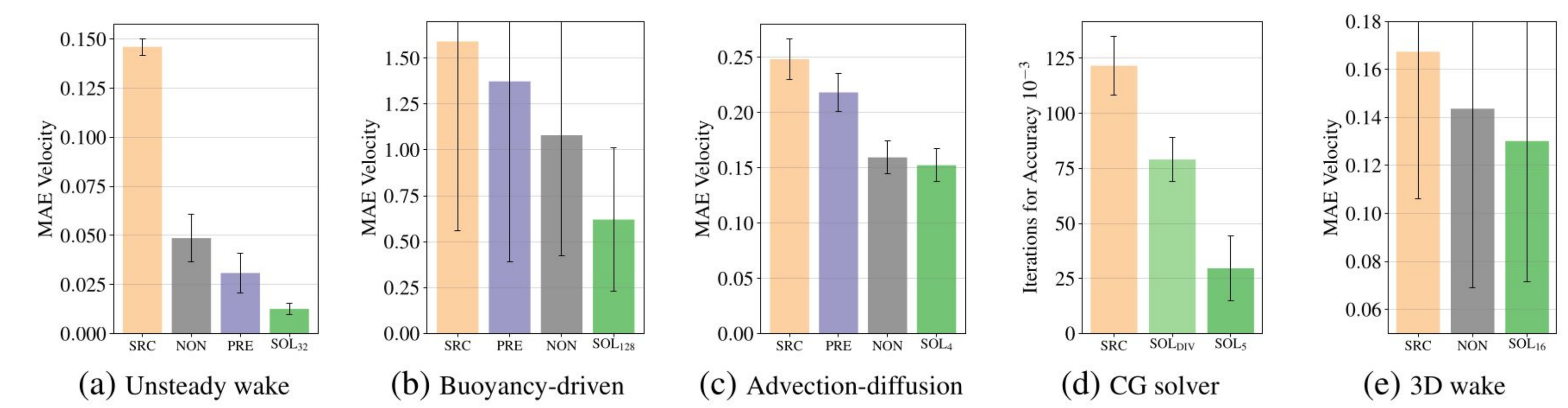
Our experiments show that learned correction functions can achieve substantial gains in accuracy over a regular simulation. When training the correction functions with differentiable physics, this additionally yields further improvements of more than 70% over supervised and pre-computed approaches from previous work. Our examples include the different types of PDE interactions: Unsteady Wake Flow, Buoyancy-driven Flow, Forced Advection-Diffusion, Conjugate Gradient Solver, and Three-dimensional Fluid Flow.



Our PDE scenarios cover a wide range of behavior including (a) vortex shedding, (b) complex buoyancy effects, and (c) advection-diffusion systems. Shown are different time steps (l.t.r.) in terms of vorticity for (a), transported density for (b), and angle of velocity direction for (c).

Analysis

The models trained with differentiable physics and look-ahead achieve significant gains over the other models.



Numerical approximation error with respect to reference solution for unaltered simulations (SRC) and with learned corrections.

For systems with deterministic behavior, long rollouts via differentiable physics at training time yield significant improvements. Our tests consistently show that, without changing the number of weights or the architecture of a network, the gradients provided by the longer rollout times allow the network to anticipate the behavior of the physical system better and react to it.

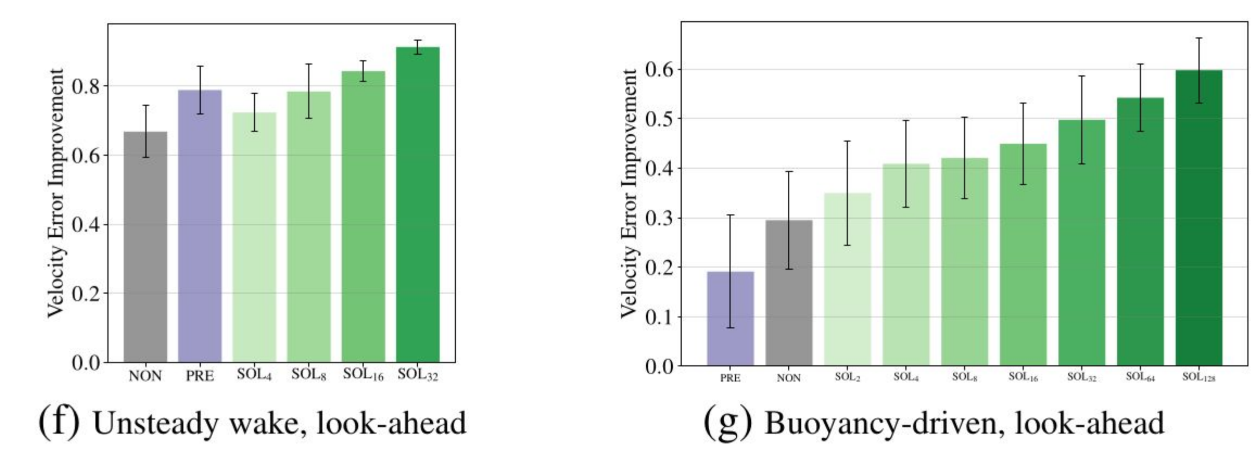


Table: A summary of the quantitative evaluation for the five PDE scenarios. SOL_s denotes a variant with shorter look-ahead compared to SOL. (∗For the CG solver scenario, iterations to reach an accuracy of 0.001 are given. Here, SOL_s denotes the physics-based loss version.)

Exp.	Mean absolute error of velocity					Rel. improvement			
	SRC	PRE	NON	SOL_s	SOL	PRE	NON	SOL_s	SOL
Wake Flow	0.146±0.004	0.031±0.010	0.049±0.012	0.041±0.009	0.013±0.003	79%	67%	72%	91%
Buoyancy	1.590±1.033	1.373±0.985	1.080±0.658	0.944±0.614	0.620±0.390	19%	29%	41%	60%
Adv.-diff.	0.248±0.019	0.218±0.017	0.159±0.015	0.152±0.015	0.158±0.017	12%	36%	39%	36%
*CG Solver	121.6±13.44	-	-	79.03±10.02	29.59±14.83	-	-	35%	76%
3D Wake	0.167±0.061	-	0.144±0.074	-	0.130±0.058	-	14%	-	22%

Further information

- phiflow: Research-oriented differentiable simulation framework https://github.com/tum-pbs/PhiFlow
- Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers, NeurIPS'20. https://ge.in.tum.de/publications/
- Learning to Control PDEs with Differentiable Physics, ICLR'20. https://ge.in.tum.de/publications/2020-iclr-holl/