Differentiable Physics with \( \Phi \text{Flow} \) (\texttt{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.

\( \Phi \text{Flow} \) is a framework for using and implementing differentiable simulations. Any simulation written in \( \Phi \text{Flow} \) can automatically leverage the automatic differentiation utilities from \texttt{PyTorch} and \texttt{TensorFlow}.

It comes with a number of built-in simulations like fluids. A basic simulation can be set up in a 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.
  - Choose \( s = T r \). Due to the inherently different numerical approximations, \( P_S(T r_{t+1}) \neq T r_{t+1} \) for the vast majority of states.
- We use an L2-norm to quantify the deviations: \( L(s_t, T r_t) = |s_t - T r_t|_2 \). While \( T r_t \) is a trajectory in the reference manifold, and an approximate version \( P_S \) with solutions \( s \), the reference manifold, and an approximate version \( P_S \) with solutions \( s \), the reference manifold, and an approximate version \( P_S \) with solutions \( s \), the reference manifold, and an approximate version \( P_S \) with solutions \( s \), the reference manifold, and an approximate version \( P_S \) with solutions \( s \).
- Before step \( t+1 \), we transform \( r_t \) into \( s_t \) and learn a correction operator \( C(s) \) such that a transformed solution of the reference sequence computed on \( R \) (blue) differs from solutions computed on the source manifold.
- The correction term \( L(P_s(C(T r_t0)), T r_t1) < L(P_s(T r_t0), T r_t1) \). The 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 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 solution to which the correction is applied has a lower error than an unmodified solution: \( L(P_s(C(T r_t0)), T r_t1) < L(P_s(T r_t0), T r_t1) \).

Comparison and analysis of differentiable physics solvers:

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

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 experiments include the different types of PDE interactions: Unsteady Wake Flow, Buoyancy-driven Flow, Forced Advection-Diffusion, Conjugate Gradient Solver, and Three-dimensional Fluid Flow.

Highlight: Our SOL model for a 3D Fluid Problem

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.